INFORMATIQUE :: CONCEPTS FONDAMENTAUX

ECCG Aimée-Stitelmann - SIMCE
Antoine MELO et al.

2025–2026

Historique

Dernière mise à jour : 4 octobre 2025

Version 0.1_β.bc jan. 2023	Mise à jour : – Corrections collègues – Couleur language – Instructions aux enseignants Ajouts : — Plan d'étude ECG — Compléments modules
Version 0.1_β.a nov. 2023	Distribution : — 1e, 2e et 3e (fichier Programmation.pdf sous //Distribution) — Collègues
Version 0.1_β.c fév. 2024	Mise à jour : aucune prévue Ajouts : — Exercices 2e — Exercices Turtle
Version 0.2_α.a août 2024	Révision du document avant nouvelle année scolaire
Version 0.2_β.a déc. 2024	Compléments autres modules + programme 3e
Version 0.3_α.a août 2025	Création de la platforme informatic.li — Programmation (Python, HTML, SQL) — Echange de documents (Markdown) Backup disponible sous parapente.world

Feuille de route

Version 0.3_β oct. 2025	Mise à jour : relecture + à compléter Distribution : version 0.3_β aux collègues Ajouts : — Exercices listes et fichiers — Exercices web — Exercices BD

eduge.ch informatic.li simce.swiss

Sommaire

INTRODUCTION

Avant-propos

L'évolution rapide des ordinateurs a fait exploser la production et l'échange de données, impliquant dans son sillage une amélioration quasi exponentielle des techniques de communication [1]. Face aux défis mondiaux que cela implique, tous les pays et plus particulièrement l'Europe se doit de possèder une éducation moderne et une constance recherche de qualité, voir d'excellence. Les compétences de base enseignées ne peuvent pas, dans ce contexte, se passer de l'informatique, essentielle dans notre vie quotidienne et professionnelle.

Preface

Présentation

Je suis convaincu que l'enseignement des concepts fondamentaux d'informatique contribue de manière essentielle au développement intellectuel des jeunes et des adolescents. Les sciences du numériques devrait probablement être enseignée beacoup plus tôt, sans nécessairement utiliser un language de programmaton, mais dans le but d'éveiller, chez les apprenants, un intérêt et une passion pour la pensée logique et technique.

Concernant le contenu de ce polycopié, il existe beaucoup de ressources disponibles, mais je n'en ai trouvé aucune qui soit à la fois suffisament synthétique, détaillée, mais pas trop ;), libre et écrite à l'attention de jeunes et adolescents. À travers ce matériel d'enseignement, je cherche donc à réduire les obstacles susceptibles de limiter les premiers pas, en particulier, vers la programmation. Son contenu prend racine dans des situations concrètes et les connaissances acquises dans le monde des entreprises et devraient, j'en suis convaincu, s'appliquer plus largement dans des diverses situations de la vie courante qui font intervenir des notions d'informatique.

Concernant plus spécifiquement la programmation, bien que le langage Python ne soit plus tout jeune, ce n'est que récemment qu'il a fait son entrée dans les écoles jusqu'à constituer actuellement une véritable tendance dans de nombreuses institutions. Cela tient probablement au fait que Python, est d'un apprentissage relativement aisé mais également du fait qu'il ne

1. Lire en ce sens l'excellent rapport [Jouët, 1991]

requière que très peu de ressources machines (il tourne très bien sur des micro-systèmes, comme les Rasberry).

Programmation Python, de l'algorithmique et des modules, pourquoi ?

En combinant la programmation avec l'étude des système informatiques, des algorithmes et des bases de données, les apprenants acquièrent des bases solides en informatique, tout en ayant, comme mentionné précédement, les outils nécessaires pour appliquer ces connaissances dans des projets pratiques et diversifiés.

L'utilisation de Python 3, combiné à l'algorithmique, me semble particulièrement pertinent pour plusieurs raisons :

— La syntaxe est clair, Python 3 est en effet conçu pour être facile à lire et à écrire, même pour des personnes (et peut-être surtout) pour des débutants

— Il existe une large communauté de développeurs et de ressources, tutoriels, vidéos, forums et plus largement, divers supports de cours

— L'apprentissage des algorithmes est crucial pour comprendre comment résoudre efficacement des problèmes (à l'aide de la programmation ou non)

— Python possède un large éventail de modules et de bibliothèques qui élargissent les capacités du langage de base et permet aux apprenants d'explorer divers domaines tels que l'analyse de données, l'intelligence artificielle, le développement web, etc.

— Python 3 est maintenant facilement portable et compatible avec la plupart des systèmes d'exploitation

— Afin, également, de bénéficier d'une source fiable et commune d'enseignement

Ces différents points éclairent, je l'espère, les raisons qui m'ont amenés à rédiger, sur la base de quelques éléments, comme les supports de cours sur l'algorithmique [Solnon, 2007], enseignante à l'Institut National des Sciences Appliquées (INSA) de Lyon, sur Python [Bonjour, 2021][2], enseignant et responsable informatique à la Faculté de l'environnement naturel, architectural et construit (ENAC) de l'École polytechnique fédérale de Lausanne (EPFL), du manuel interactif en ligne [Arnold *et al.*, 2022], développé par J. Arnold de la Haute École Pédagogique de Berne, T. Kohn à l'époque de sa thèse à l'École polytechnique fédérale de Zürich (ETHZ), en colaboration avec A. Plüss, enseignant d'informatique à l'Université de Berne. J'ai également souvent utilisé chatGPT, afin de me fournir, rapidement, soit des simplification / vulgarisation de supports, soit pour me permettre de rédiger plus rapidement ce support à l'aide de LaTeX.

2. Pour la petite histoire, J.D. Bonjour m'a fait découvrir les joies de l'informatique, à l'EPFL, quand, j'avais un vingtaine d'années

Instructions aux enseignants

Le matériel du présent ouvrage couvre environ trois ans de cours de base d'informatique. Suivant le niveau de la classe et la dotation horaire du cours, il est possible de ne suivre que certains chapitres et d'en laisser tomber d'autres. Puisque les graphiques avec la tortue ou les algorithme constituent tous les deux des approches pertinantes pour l'introduction des notions fondamentales, les enseigants choisiront probablement d'introduire les notions de programmation par l'un ou l'autre de ces chapitres. Nous suggérons les variantes suivantes :

1. Débuts avec le chapitre sur les graphiques avec Turtle et éventuellement un projet (à définir), le tout, soutenu par le chapitre sur les concepts généraux pour la rédaction de scripts en Python pour un cours à l'attention de 1e année, doté de une à deux heures par semaine sur une année
2. Exercices choisis sur les graphiques avec Turtle, soutenu par celui sur quelques concepts généraux pour la rédaction de scripts en Python pour un cours d'introduction sur un semestre
3. Débuts avec les notions d'algorithmique et de résolution de problèmes, puis, quelqeus exercices sur les graphiques avec Turtle et la programmation de jeux, soutenu par le chapitre portant sur les concepts généraux pour la rédaction de scripts en Python pour un cours à l'attention de 2e et/ou 3e année, doté de une à deux heures par semaine sur l'année

Les éléments liés à l'alogorithmique sont présentés en français et ceux de programmation en anglais. Le passage de l'un vers l'autre ne devrait pas poser de problème et celui d'un outil de programmation à un autre devrait en conséquence être facilité. Les autres sujets, sur les réseaux, systèmes informatiques ou l'introduction aux bases de données peuvent être enseignées de manière séparée ou indépendante des notions précédentes.

Plan de formation

École de Culture Générale ECG

2. Notions d'algorithmique

Niveau	Contenus et ressources	Périodes	Théorie
Utiliser les structures algorithmiques de base et les présenter pour résoudre un problème.			Exercices A.2.1, p. 133
1e	— Algorithmique sous forme de schémas ou de problèmes débranchés — Instructions sélectives sans et avec alternatives · si condition alors instruc1 sinon instruc2 · imbrication	4	p. 27 p. 28
Implémenter des algorithmes classiques et en mesurer l'efficacité.			Exercices A.2.2, p. 135 Exercices A.3.3 / TP04, p. 140
2 et 3e	— Recherche séquentielle et dichotomique dans un tableau ou une liste — Mise en oeuvre d'un tri	2 x 6	p. 32 p. 46

3.1. Généralités : les variables

Niveau	Contenus et ressources	Périodes	Théorie
Créer des variables avec le type d'information adéquat.			Exercices A.3.1 / TP01, p. 136
1e	— Affectation d'une valeur à une variable · Boolean · Entiers et réels · Caractères — Opérations de base (+, -, etc.)	6	p. 41 et 72

3.1. Généralités : entrées-sorties et validation

Niveau	Contenus et ressources	Périodes	Théorie
Programmer un dialogue (textes).		Exercices A.3.1 / TP02, p. 136	
1e	— Input — Print	3	p. 41
Effectuer une validation de programme simple.		Exercices A.3.2 / TP04, p. 137	
	Messages d'erreurs et données de vérification	2	p. 73
Programmer un dialogue avec des fenêtres graphiques.		Projet A.3.4, p. 141	
2e	— Input et Print — Utiliser un outil de développement intégré	6	p. 41
Traiter les erreurs d'exécution du code [a].			
3e	— Division par zéro — Dépassement de capacité — Erreur d'entrée/sortie	2	p. 58

a. Illustré ici dans des exemples de manipulation de fichiers

3.2. Containers : structure des données

Niveau	Contenus et ressources	Périodes	Théorie
Choisir de manière appropriée les structures de données.		Exercices A.3.2 / TP03, p. 137	
1e	— Tableaux — Chaînes de caractères	4	p. 46
2e	Tableaux à plusieurs dimensions et dictionnaires	6	p. 49
3e	— Tableaux d'articles — Objets avec attributs, méthodes et héritages — Surcharge et le polymorphisme des objets	8	p. 79

3.3. Contrôles et 4.3. Turtle : maîtriser son code

Niveau	Contenus et ressources	Périodes	Théorie
Comprendre la syntaxe d'un langage (Python).		Exercices A.3.3 / TP01, p. 140	
1e	Indentation du code	1	p. 50
Lire et comprendre le code d'un programme simple.		Exercices A.3.3 / TP02, p. 140	
	— Boucles · compteur de début à fin faire instruction · répéter instruction jusqu'à condition ou tant que condition faire instruction Exemples d'applications [a] — Repère de chemin dans un labyrinthe — Traçage de formes géométriques — Recherche d'une donnée dans un tableau	4	p. 51 et 70

a. Utilisation de modules

3.4. Fonctions : sous-programmes et paramètres

Niveau	Contenus et ressources	Périodes	Théorie
Coder en découpant le problème à résoudre en sous-problèmes.		Exercices A.3.3 / TP03, p. 140	
1e	— Procédures et fonctions — Paramètres — Retour de fonction	4	p. 54 et 71
2e	Analyse montante/descendante	6	p. 26

Autres sujets de 3e année

— 3.5 Manipulation de fichiers, p. 58

— 4.1 Module Matplotlib et 4.2 Random (librairies et réutilisation du code), p. 66

— Balises, sites web et base de données à venir.

Chapitre 1

LES SYSTÈMES INFORMATIQUES

Le saviez-vous ? Le mot *informatique* est la concaténation de *information* et *automatique*. Tout système informatique a besoin, pour fonctionner, d'électricité. Au début du XIXe siècle, grâce à la découverte de cette dernière (on parle ici d'électricité et des améliorations techniques faites pour la capturer / la transmettre), on a pu utiliser le réseau électrique pour envoyer des messages. En 1832, naît ainsi le code Morse, qui s'impose rapidement comme un standard universel de communication. Si on y regarde de plus près, on constate que les signaux utilisés pour représenter les lettres, en code Morse, ne suivent pas vraiment la logique de l'ordre alphabetique, les lettres les plus fréquentes étant celles avec les codes les plus courts.
Source : Wikipédia

E	**T**	**I**	**A**	**N**	**M**	**S**	**U**	**R**	**W**	**D**	**K**	**G**
.	–	. .	. –	– .	– –	. . .	. . –	. – .	. – –	– . .	– . –	– – .
O	**V**	**L**	**F**	**P**	**J**	**B**	**X**	**C**	**Y**	**Z**	**Q**	**H**
– – –	. . . –	. – . .	. . – .	. – – .	. – – –	– . . .	– . . –	– . – .	– . – –	– – . .	– – . –	. . . .

FIGURE 1.1 – Code Morse

Dans cette partie introductive du cours d'informatique, nous traiterons plus particulièrement des interfaces qui permettent les interactions de l'être humain avec les machines et du fonctionnement de ces systèmes, permettant ainsi l'encodage et la sauvegarde des données.

1.1 Softwares (logiciels)

Étudions tout d'abord ce qui fait tourner la plupart des machines, un logiciel qui se présente sous la forme d'une interface graphique et qu'on appelle "système d'exploitation".

1.1.1 Systèmes d'exploitation

Exercices p. 120

Interface graphique

En anglais *GUI*, pour *Graphical User Interface*, ce type d'interface s'oppose à l'interface en ligne de commande. Les parties les plus typiques en sont le pointeur de souris, les fenêtres, le bureau, les icônes. Les contrôles graphiques sont utilisés pour interagir avec l'utilisateur : icônes, boutons, menus et barres de défilement sont les plus fréquemment utilisés.

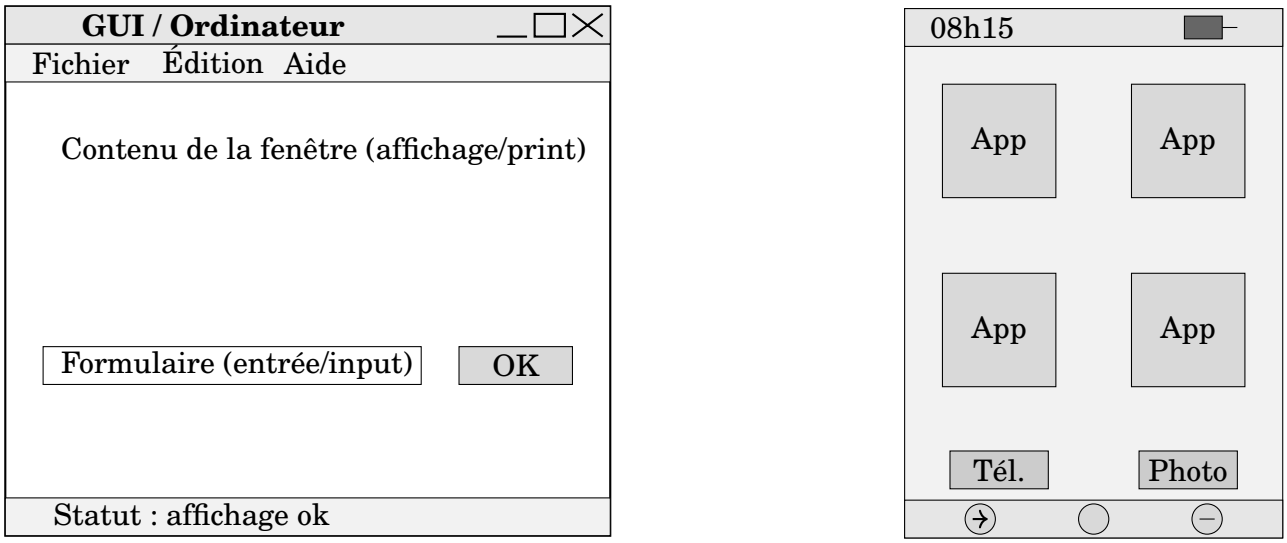


FIGURE 1.2 – Interface graphique d'un ordinateurs de bureau et d'un portable

Progiciel système

Un progiciel (de programme et logiciel) est un ensemble de programmes conçus pour être fournis à différents utilisateurs en vue d'une même application ou d'une même fonction (*définiton Larousse*).

Un système d'exploitation (en anglais *Operating System* ou OS) est un ensemble de programmes responsables de la liaison entre les ressources matérielles d'un ordinateur et les applications informatiques de l'utilisateur (traitement de texte, jeux vidéo, etc.).

Systèmes d'exploitation les plus connus

— **Windows**, préinstallés sur la plupart des ordinateurs personnels (69% des ordinateurs de bureau et portables)

— **Linux**, système d'exploitation libre, est installé sur environ 3% du parc informatique mondial (fait tourner bon nombre de serveurs, de sites Internet et de supercalculateurs), mais c'est une version Linux mobile qui supplentes toutes les autres...

— **MacOS et iOS**, basés sur un noyau Linux, préinstallés sur les appareils vendus par Apple, représentant 24% du marché des OS

— **Android**, système d'exploitation open-source créé par le consortium Open Handset Alliance, sponsorisé par Alphabet (dont fait partie Google) sur un noyau Linux destiné aux

tablettes et smartphones, détient, en effet, 42% du marché

(*Origine des pourcentages : Wikipédia*)

Zoom sur Windows

Windows est un système d'exploitation multitâche, il offre en effet la possibilité de travailler simultanément avec de nombreuses applications ouvertes. À l'école, Windows démarre automatiquement dès que vous mettez l'ordinateur sous tension.

Sur le bord inférieur de l'écran se trouve la barre des tâches dont l'élément le plus important est le bouton *Démarrer*, en bas à gauche.

Sur l'espace de bureau, vous trouverez plusieurs icônes :

— **Poste de travail**, pour visualiser les composants de l'ordinateur, notamment les lecteurs (locaux ou disponibles sur le réseau)

— **Corbeille**, lieu de stockage temporaire des fichiers supprimés, mais qui, initialement, se trouvaient sur le disque dur de votre ordinateur

— **Rendu**, qui ouvre un navigateur web qui vous permet de rendre vos oeuvres aux enseignants.

— **Espace personnel**, libellé à votre nom dans le poste de travail, c'est l'équivalent, dans notre école, de *Mes documents*, c'est un espace qui vous est réservé.

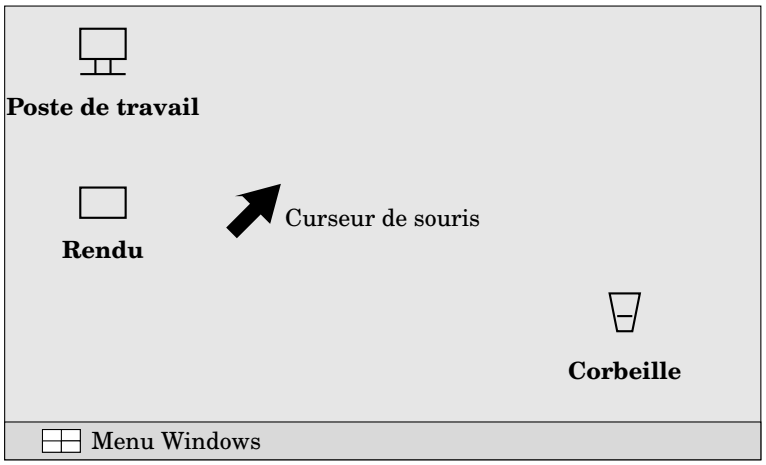


FIGURE 1.3 – Bureau MS Windows

1.1.2 Logiciels métiers et logiciels importants pour le cours

Nous avons étudié ci-dessus les différents systèmes d'exploitation (ou progiciels systèmes). Ajoutons à cette description d'autres outils essentiels pour nous, tant à l'école que dans un cadre professionel.

Le saviez-vous ? Les logiciels métiers ne sont pas des progiciels, mais des programmes conçus pour répondre aux besoins spécifiques d'une profession ou d'un secteur particulier, comme un

logiciel de gestion de cabinet médical, ou un logiciel de dessin assistée par ordinateur (DAO). Les progiciels, eux, sont des systèmes qui répondent à des besoins généraux de gestion d'entreprise, comme les systèmes d'exploitation, souvent utliser pour gérer des processus transversaux dans une organisation, ce qui inclut par exemple les ERP (Enterprise Resource Planning) ou les CRM (Customer Relationship Management). Les logiciels comme MS Word ou Excel, sont regroupés dans une catégorie plus large, nommée générique : ils ne sont en effet pas spécifiquement adaptés à un secteur ou une profession.

Explorateur Windows

C'est un peu comme la carte du contenu de votre ordinateur : il permet de naviguer à travers les fichiers et dossiers stockés sur votre machine ou des disques réseaux. Imaginez votre ordinateur comme une grande armoire avec différents tiroirs et classeurs. Chaque classeur (dossier) peut contenir d'autres classeurs ou des feuilles individuelles (fichiers). L'Explorateur vous aide à organiser, trouver et ouvrir ces classeurs et feuilles.

Plusieurs moyens sont possibles pour **ouvrir l'Explorateur** :

1. **Barre de tâches** Icône à la forme d'un dossier jaune en bas de l'écran.
2. **Démarrer** Sélectionnez le menu Windows du clavier ou cliquez sur la fenêtre Windows en bas à gauche ; tapez "explorateur" et appuyez sur Entrée.
3. **Raccourci clavier** Appuyez simultanément sur les touches Windows + E.
4. **Commande d'exécution** Appuyez sur les touches Windows + R, une petite fenêtre apparaît ; tapez `explorer` et appuyez sur Entrée.

Voyons maintenant comment créer des fichiers, des dossiers avec en prime quelques petits trucs et astuces. Pour **créer un dossier** :

1. **Ouvrir l'Explorateur**
2. **Sélectionner la destination** Naviguer dans l'arborescence jusqu'à être là où vous voulez travailler
3. **Cliquez-Droit** Placer le curseur de souris dans une zone sans dossier ni fichier, réaliser un clic droit et choisir "Nouveau dossier".
4. **Nommez-le** : Votre nouveau dossier apparaîtra avec un nom par défaut comme "Nouveau dossier". Tapez le nom que vous souhaitez et appuyez sur Entrée.

On procédera de la même façon pour créer un nouveau fichier. Avant d'étudier plus avant d'autres logiciels, ci-après quelques trucs et astuces...

Trucs et astuces

Copier / coller L'Explorateur rend très facile le déplacement de fichiers d'un dossier à un autre. Vous pouvez copier, couper et coller (Ctrl + C, puis Ctrl + V) dans les répertoires de tra-

vail désiré. Attention toutefois aux droits d'utilisation, contrairement à votre espace personel, ceraines zones sont réservées : vous ne pourez ni y renommer, ni y coller vos fichiers / répertoires.

Attention avec la touche "Suppr"! Si un fichier ou un dossier situé sur un disque réseau, le supprimer l'effacera définitivement... il se trouvera dans la corbeille uniquement si c'était un fichier stocké en local sur votre ordinateur.

Recherche Si vous ne savez plus où se trouve un fichier ou un répertoire, vous pouvez utiliser le caractère générique `*` pour effectuer votre recherche. Par exemple, si vous recherchez tous les travaux pratiques (fichiers ou répertoires) qui contiennent l'acronyme "TP", vous pouvez écrire `*TP*`.

De la même manière, si vous ne vous souvenez que de l'extension d'un fichier, vous pouvez rechercher tous les fichiers avec cette extension en utilisant `*.pdf` au début. Cela affichera tous les fichiers PDF dans un répertoire donné.

Téléchargements Lorsque vous naviguer sur le web, les fichiers téléchargés sont enregistrés dans le dossier de téléchargement...

Encore un petit conseil, **utilisez les touches de raccourcis**...

Actualiser affichage	F5	Zoom avant / arrière	Ctrl + + / + -
Annuler	Ctrl + Z	Aide	F1
Copier / Couper / Coller	Ctrl + C / X / V	Créer à nouveau	Ctrl + Y
Enregistrer	Ctrl + S	Fermer	Alt + F4
Fermer sous-fenêtre	Ctrl + W	Imprimer	Ctrl + P
Espace insécable	Ctrl + Maj + Espace	Interrompre	Esc
Taille d'affichage	Ctrl + Souris	Naviguer entre fenêtres	Alt + Tab
Nouveau	Ctrl + N	Ouvrir	Ctrl + O
Ouvrir explorateur	Win + E	Page suivante	Alt + →
Rechercher / Remplacer	Ctrl + F / H	Rechercher document	Win + F
Revenir en arrière	Ctrl + Z	Réduire les fenêtres	Win + M
Renommer	F2	Retrait à gauche	Ctrl + Shift + Tab
Sauvegarder sous	F12	Saut de page	Ctrl + Enter
Sélection continue	Shift	Sélectionner tout	Ctrl + A
Sélectionner ligne	Shift + ↓ / ↑	Sélections séparées	Ctrl

TABLE 1.1 – Touches de raccourcis clavier

Outils génériques

Microsoft Word est un logiciel de traitement de texte qui permet de créer, modifier et mettre en forme des documents tels que des lettres, des rapports, des brochures, etc. Il offre de nombreuses fonctionnalités comme la vérification orthographique, l'insertion d'images, de tableaux et des mises en page avancées. Voir le paragraphe suivant pour connaître les extensions utilisées par ce logiciel.

Excel est un tableur utilisé pour organiser, analyser et visualiser des données sous forme de tableaux et de graphiques. Il est particulièrement utile pour effectuer des calculs, créer des budgets, suivre des finances et bien plus encore grâce à des fonctions intégrées. Voir le paragraphe suivant pour connaître l'extension utilisée par ce logiciel.

Navigateurs web

L'école met à disposition trois outils de navigatin web, relativement similaires : Microsoft Edge, Google Chrome et Mozilla Firefox.

— **Chrome** est un navigateur web développé par Google, réputé pour sa rapidité et son intégration avec les services Google.

— **Firefox**, développé par Mozilla, est connu pour sa protection fort de la vie privée et sa flexibilité ; contrairement à Chrome, Firefox est open-source et met l'accent sur l'utilisateur plutôt que sur les services connectés

Les moteurs de recherche et les IA génératives sont tous deux basés sur des logiciels complexes, mais généralement intégrés aux navigateurs web, ce qui justifie leur intégration ici. Les moteurs de recherche, tels que Google Search et Microsoft Bing, permettent aux utilisateurs de trouver des informations sur le web en entrant des mots-clés ou des requêtes, basé sur le profil des utilisateurs, utilisant en parrallèle des intéligences artificielles afin d'améliorer l'expérience de ses utilisateurs.

Fonctionnant de manière intégrée ou en parallèle, les IA (notamment génératives) sont des systèmes conçus pour créer du contenu qui semble vraisemblable à l'utilisateur, comme du texte, des images, ou de la musique, en se basant sur des données existantes et en utilisant des modèles statistiques. Des IA comme ChatGPT ou DALL-E génèrent des réponses ou des créations qui imitent le langage ou les styles humains.

ATTENTION L'utilisation d'une IA, tout comme celle de sources d'information, doit être systématiquement mentionnée et citée.

1.1.3 Formats de fichiers

Exercices p. 122

Traitement de texte

Comme expliqué à la page précédente, les outils de traitements de texte permettent d'éditer et structurer des documents avec des options de mise en forme avancées. Ils facilitent grande-

ment le partage et l'impression de contenu.

— **.docx** C'est le format standard pour Microsoft Word, idéal pour des documents avec beaucoup de mise en forme ; Google docs permet aussi de travailler sur ce format

— **.pdf** Parfait pour partager parce que le format ne change pas, peu importe l'appareil ou le logiciel que vous utilisez (au détriment parfois de la taille du document)

— **.txt** Juste du texte, rien d'autre, pas d'images, pas de mise en forme, rien que les mots (et donc très petit) !

— **.odt** Utilisé par OpenOffice Writer, une alternative gratuite à Word

— **.pages** Spécifique aux produits Apple sur Macs et iPads

Tableurs

Les tableurs permettent de manipuler, organiser et analyser des données. Ils offrent en outre des fonctionnalités pour le calcul et créer des graphiques.

— **.xlsx** Utilisé par Microsoft Excel, c'est votre format de prédilection pour les tableaux, les graphiques et les calculs ; Google Sheets permet aussi de travailler sur ce format

— **.csv** Comma-Separated Values ; c'est un fichier texte qui peut aussi être lu par des tableurs (il est super pour les données simples) !

— **.ods** Utilisé par OpenOffice Calc, encore une fois, une alternative gratuite

— **.numbers** Également spécifique à Apple

Publication assistée par ordinateur (PAO)

La PAO permet de créer des documents (présentations, magazines, brochures ou affiches) visuellement complexes.

— **.psd** Utilisé par Adobe Photoshop ; si vous êtes un artiste en herbe, c'est ce que vous utiliserez probablement

— **.indd** Adobe InDesign utilise ce format ; parfait pour le design de magazines, brochures, etc.

— **.pptx** PowerPoint, aussi utilisé pour la PAO, même si on restera ici plutôt basique

— **.odp** OpenOffice Impress, l'alternative gratuite

— **.key** Pour les utilisateurs Apple, c'est le format de Keynote

Images

— **.jpg** Très commun, qualité suffisante pour la plupart des usages

— **.png** Excellente qualité, supporte la transparence

— **.gif** Vous savez, ces images animées drôles que tout le monde partage...

— **.bmp** Gros fichiers, rarement utilisé en dehors de Windows

Vidéos

— **.mp4** Le format le plus commun pour les vidéos, compatible avec presque tous les appareils

— **.avi** Un peu plus ancien, mais encore utilisé, généralement plus gros en taille

— **.mov** Spécifique à Apple, mais de haute qualité

— **.wmv** Petit en taille, bon pour le streaming, mais surtout sous Windows

Sons

— **.mp3** Très répandu, bonne qualité sonore, fichiers de petite taille

— **.wav** Qualité sonore sans perte, idéal pour l'édition

— **.m4a** Qualité sonore sans perte, surtout sur produits Apple

1.1.4 À retenir...

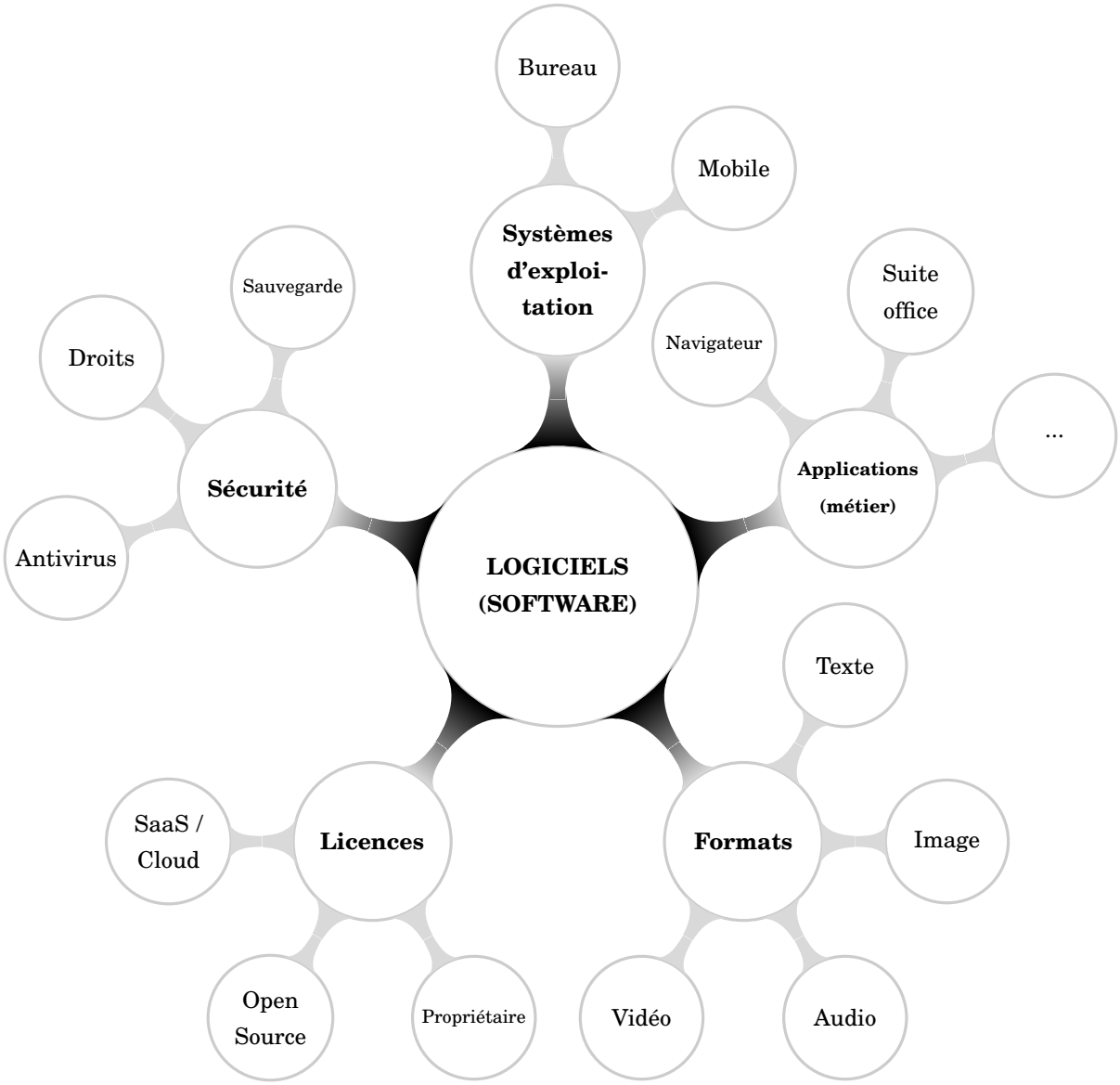
Points clés abordés

— Systèmes d'exploitation et interfaces
— Logiciels et classifications
— Fichiers et formats
— Raccourcis clavier
— Recherche sous windows
— Importance des périphériques
— Stockage numérique

Questions récapitulatives

1. Comment les inventeurs du code Morse ont-ils optimisé la représentation des lettres ? Fournissez un exemple précis.
2. Quelle est la différence entre le système d'exploitation et les applications ?
3. Quels sont les avantages d'une interface graphique (GUI) par rapport à une interface en ligne de commande (CLI) ? Donnez deux exemples d'objets que l'on retrouve dans les GUI.
4. Dans quel contexte utilise-t-on la fonction de recherche avec un caractère générique $\star$? Donnez un exemple d'utilisation.
5. Qu'est-ce qu'un logiciel métier, et en quoi diffère-t-il d'un progiciel ou d'un logiciel générique ? Donnez un exemple de chaque type de logiciel.

SYNTHÈSE DES SAVOIRS ET SAVOIRS-FAIRE



Rappel : exercices et TP pages 120 et suivantes.

Page vide

1.2 Hardware (la machine)

Exercices p. 126

Les bases logiciel étant maintenant acquises, étudions de plus près ce qui se cache sous le capot des ordinateurs.

1.2.1 Ordinateur

Dans les faits, un ordinateur est un ensemble de circuits électroniques permettant de manipuler des données sous forme binaire. Toute machine capable de manipuler des informations binaires peut être ainsi qualifiée d'ordinateur.

Signaux binaires

La plupart des civilisations utilisent le système décimal. Pourquoi ? Tout simplement parce que nous avons 10 doigts ! L'ordinateur, lui, n'a pas de doigts mais utilise l'électricité. Par conséquent, il ne connaît que deux types d'informations : allumé, éteint. On dit qu'il travaille dans un système binaire, ou en base deux, constitué de deux valeurs représentées par 0 ou 1, selon que le signal électrique est transmis ou non (aucun voltage détecté) par le transistor qui compose les circuits électriques (intégré à la carte mère).

Le saviez-vous ? On aurait pu choisir un code possédant plus de deux signaux différents. Par exemple, avec trois signaux, on pourrait coder trois valeurs avec un courant faible, un courant moyen, un courant fort, ou encore mieux : une tension négative, une tension nulle et une tension positive. On appelle cette dernière proposition le *ternaire balancé*. Il est cependant plus simple de concevoir des circuits électroniques qui ne doivent traiter que deux valeurs.

Codage de l'information

Les informations qui sont utilisées par les ordinateurs se font par groupe de 8 octets (*bytes* en anglais, à ne pas confondre avec la notion de bits aussi utilisés pour représenter des vitesses de transmission, par exemple lors de transferts de données). Le bit vient de la terminologie anglo-saxonne de *binary digit* : un ensemble de 8 bits est appelé un octet. Par exemple, la lettre A est composée des huit chiffres suivants : 01000001 (à ne pas confondre avec les codes ASII qui ne compte que sept codes binaires).

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
01000001	01000010	01000011	01000100	01000101	01000110	01000111	01001000
I	**J**	**K**	**L**	**M**	**N**	**O**	**P**
01001001	01001010	01001011	01001100	01001101	01001110	01001111	01010000
Q	**R**	**S**	**T**	**U**	**V**	**W**	**X**
01010001	01010010	01010011	01010100	01010101	01010110	01010111	01011000
Y	**Z**						
01011001	01011010						

FIGURE 1.4 – Code binaire

Le saviez-vous ? Le 4 juin 1996, le premier vol de la fusée Ariane 5 a explosé 40 secondes après l'allumage. La fusée et son chargement avaient coûté 500 millions de dollars. La commission d'enquête a rendu son rapport au bout de deux semaines. Il s'agissait d'une erreur de programmation dans le système inertiel de référence. À un moment donné, un nombre codé en virgule flottante sur 64 bits (qui représentait la vitesse horizontale de la fusée par rapport à la plate-forme de tir) était converti en un entier sur 16 bits. Malheureusement, le nombre en question était plus grand que 32'767 (le plus grand entier que l'on peut coder en tant qu'entier signé sur 16 bits) ; la conversion était donc incorrecte, induisant ainsi un changement de trajectoire fatal ! (*Source : J.-L. Lions, Rapport de la Commission d'enquête Ariane 501*)

Systèmes logiques

Aujourd'hui, les ordinateurs contiennent des dizaines de milliards d'interrupteurs (transistors). Ces minuscules composants sont organisés en systèmes logiques, conçus pour avoir une ou plusieurs entrées qui produisent, in fine, une donnée en sortie. Par exemple, une porte logique de type *ET* possède deux entrées, X et Y, ainsi qu'une sortie, Z. La sortie Z vaut 1 si, et seulement si, X et Y valent tous deux 1. À l'inverse, le résultat d'une opération logique *OU* vaut 1 dès qu'au moins une des entrées vaut 1. Sans entrer davantage dans les détails, on démontre qu'il est possible de produire de nouvelles informations à partir d'un ensemble de données existant. On retiendra cependant que les systèmes logiques sont de taille microscopique [1] et qu'ils composent les transistors intégrés sur les cartes mères, permettant d'effectuer les calculs nécessaires au fonctionnement des ordinateurs.

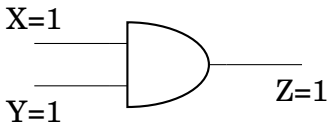


FIGURE 1.5 – Système logique ET

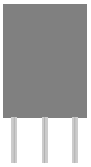


FIGURE 1.6 – Transistor (schéma)

Entrée / Sortie

D'un point de vue matériel, on distingue donc plusieurs éléments : l'alimentation, bien sûr, la carte mère, le processeur, la mémoire vive / cache et les disques durs, mémoire morte ou persistente. On peut également citer les cartes graphiques, sons, réseau, USB etc. qui ne sont pas absolument indispensables au bon fonctionnement d'un ordinateur mais facilitent quand même grandement son utilisation.

Un ordinateur récupèrera donc les instructions données via les ports d'entrée et redistribura l'information, après traitement, via les ports de sortie définis. L'ensemble de cet environnement d'entrées-sorties constitue ce que l'on nomme les périphériques : clavier, écran, enceintes

1. À titre d'exemple, une puce Apple M4 intègre environ 28 milliards de transistors, ce qui lui permet d'exécuter jusqu'à 38 000 milliards d'opérations par seconde.

audio ou casque, imprimante, souris ou pad, disques externes, microphone, réseau Ethernet ou wifi, etc. Certains périphériques sont par nature destinés uniquement à l'entrée de données (claviers et souris, microphones), tandis que d'autres s'occupent avant tout de la sortie (imprimantes, écrans non-tactiles) ; d'autres permettent, à la fois, l'entrée et la sortie (disques durs, clés USB, etc.).

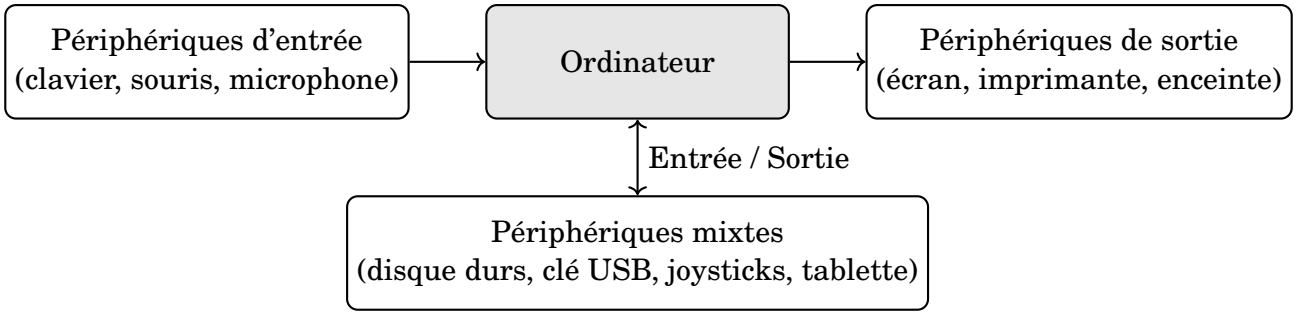


FIGURE 1.7 – Architecture générale d'un ordinateur

1.2.2 Stockage

On sait maintenant comment les informations sont codées et transmises... se pose alors la question du stockage (pour traiter des informations sur la durée et non uniquement dans l'immédiat). Dans les ordinateurs, il existe plusieurs types de mémoire, généralement classés en deux catégories. La mémoire volatile (aussi appelée vive ou cache) s'efface quand la machine s'éteint (cas de la RAM, *random-access memory*, par exemple), contrairement aux bandes magnétiques, disques durs ou mémoire de masse, comme le SSD, *solid-state drive*, ou la ROM, *read-only memory*, qui sont ds mémoires persistantes. La mémoire vive est beaucoup plus coûteuse à produire, mais elle est également beaucoup, beaucoup plus rapide. Pour stocker les informations temporairement, les systèmes logiques utilisent des bascules (*flip-flops*), qui sont des systèmes logiques capables de maintenir la valeur d'un bit (et donc d'une information) jusqu'à ce qu'une nouvelle donnée soit fournie. Un ensemble de bascules forme ce qu'on appelle les registres.

Concrètement ?

Le stockage des données informatiques s'effectue par des mécanismes différents, selon le support utilisé.

Les bandes magnétiques, support de stockage surtout utilisé pour l'archivage et les sauvegardes à long terme, sont constituées d'une fine couche d'oxyde de fer (matériau aux propriétés magnétique) appliquée sur un ruban en plastique. Pour stocké une donnée, un courant électrique crée un champ magnétique qui aligne les particules d'oxyde de fer dans une direction (représentant un 1) ou dans une autre (représentant un 0).

Les disques durs utilisent également des propriétés magnétiques pour stocker les données, mais ils le font sur des disques rigides appelés plateaux. Ces plateaux tournent relativement

vite (de l'ordre de 6-7000 tours / minute), et des têtes de lecture / écriture montées sur des bras mobiles se déplacent au-dessus des plateaux pour lire ou écrire des données.

La mémoire vive est constituée de millions de petites cellules logique [2], chacune capable de stocker un bit d'information. Rappelons que chaque cellule est composée d'un transistor et d'un condensateur. Le condensateur peut, comme on la vu plus haut, être chargé (représentant un 1) ou déchargé (représentant un 0). L'unité centrale de calcul, ou processeur, peut ainsi, grâce aux registres (illustration 1.8 ci-dessous), lire ou écrire directement dans n'importe quelle cellule de mémoire, sans avoir à parcourir l'ensemble du contenu.

1.2.3 Unité centrale de calcul

L'unité centrale de calcul (*central processing unit*, CPU ou processeur) est un composant qui exécute les instructions machine des programmes informatiques en utilisant des éléments appelés registres (qui sont composés de bascules, directement intégrées au processeur) et des bus, afin de permettre aux informations de circuler entre la mémoire et les autres composants d'un ordinateur. Plus un système possède de processeurs, plus il sera gourmand en énergie.

Illustrations

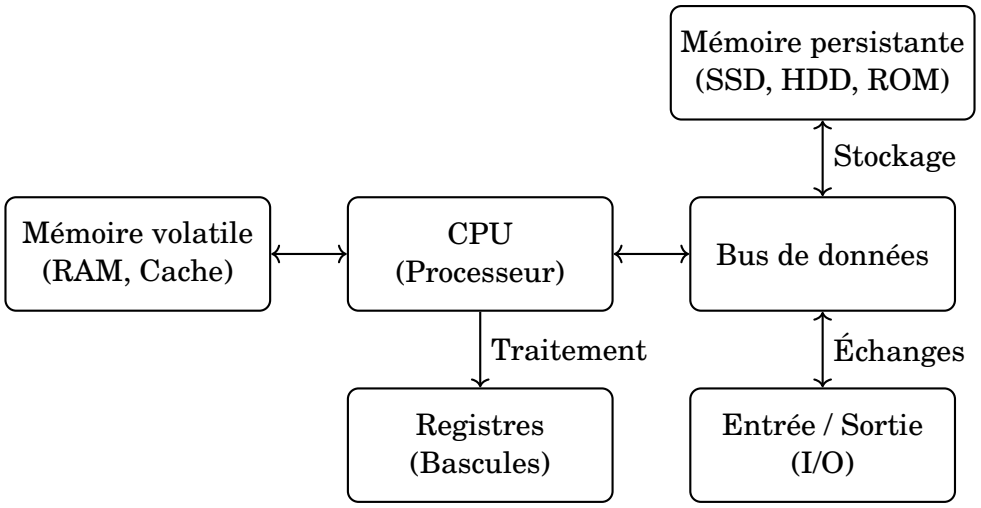


FIGURE 1.8 – Fonctionnement de l'unité centrale de calcul CPU

Nous avons montré que les ordinateurs fonctionnent de manière relativement basique : ils utilisent des portes logiques pour traiter des 0 et des 1. Il est difficile toutefois de se rendre compte à quel point ces traitements sont rapides. Pour l'illustrer, imaginons que le processeur écrive toutes ses opérations sur un ruban de papier.
Prenons l'exemple d'un processeur fonctionnant à une fréquence de 3 GHz, soit une opération toutes les 3 nanosecondes. Supposons qu'une opération écrive 64 bits (ce qui correspond à 8 caractères ou environ 5 cm de ruban). La vitesse de défilement du ruban serait alors de 150'000 km/s, soit à peu de chose près, la moitié de la vitesse de la lumière. Cela signifie que

2. Voir le chapitre précédent pour plus d'information

si un microprocesseur transcrivait toutes ses actions sur un ruban de papier, celui-ci devrait se déplacer à la moitié de la vitesse de la lumière, soit environ quatre fois le tour de la Terre par seconde !

Informatique quantique

Les ordinateurs quantiques se distinguent des ordinateurs classiques par l'utilisation de qubits qui, contrairement aux systèmes binaires, peuvent être simultanément dans l'état 0 et 1. Cette propriété permet d'effectuer des calculs beaucoup plus rapidement. Comme un programme classique qui doit tester chaque chemin d'un labyrinthe, un système quantique peut explorer plusieurs chemins en même temps !

Le saviez-vous ? il serait plus approprié de parler de "processeur quantique" [3] car seule une petite partie d'un calcul est réalisée par ce dernier, le reste étant toujours orchestré par un processeur classique. Bien que de petits ordinateurs quantiques aient été construits depuis les années 1990, les processeurs quantiques d'aujourd'hui en sont encore au stade expérimental, occupant des dispositifs encombrants de refroidissement et nécessitant des conditions de vide poussé pour fonctionner (pression similaire à celle qui règne à la surface de la lune). Le contrôle des qubits repose également sur des dispositifs sophistiqués, à des températures proches de -273°C, le zéro absolu.

1.2.4 À retenir...

Points clés abordés

- Signaux binaires et codage de l'information
- Systèmes logiques
- Stockage
- Architecture d'un ordinateur
- Unité centrale de calcul
- Rôles des périphériques

Questions récapitulatives

1. Pourquoi le système binaire est-il utilisé dans les ordinateurs et donnez un exemple de ce type de codage.
2. Quelle est la différence entre la mémoire vive et la mémoire morte ? Donnez un exemple de chaque.
3. Représentez à l'aide d'un schéma toutes les composantes d'un ordinateur.
4. Citez les différentes catégories de mémoire d'un ordinateur, et pourquoi la mémoire vive est-elle généralement plus rapide que la mémoire morte ?
5. Donnez des exemples de périphériques d'entrée, de sortie et d'entrée-sortie.

3. À noter que des transistors expérimentaux ont été réalisés à l'échelle d'un seul atome (de carbone) : soit environ 0,3 nanomètre.

Page vide

1.3 Networks (réseaux)

Exercices p. 129

Les équipements de réseau sont des appareils utilisés pour la communication d'informations entre plusieurs composants informatiques. Les communications peuvent se faire par câble, par onde radio, par satellite, ou par fibre optique. On notera qu'Internet est l'ensemble de l'infrastructure qui en fait, aujourd'hui, le réseau le plus utilisé, notamment avec les données du Web.

1.3.1 Le web ?

Vous connaissez certainement le nom Web, mais où a-t-il été créé et comment fonctionne-t-il ? Cette exemple, probablement connu par la plupart d'entre vous, illustre cependant bien les notions clés liées aux réseaux.

Création du web

Le Web, ou World Wide Web (WWW), a été créé en 1989 par Tim Berners-Lee, un informaticien britannique. Il travaillait alors au CERN (Centre européenne pour la recherche nucléaire), situé à Meyrin, près de Genève. Berners-Lee a inventé un système pour partager des informations entre chercheurs, en permettant la connexion et la navigation entre différents documents référencés par des liens hypertextes.

Fonctionnement

Le Web fonctionne grâce à une infrastructure composée de clients et de serveurs, connectés via Internet. Les ordinateurs connectés ensemble peuvent être classés en deux catégories distinct : les clients et les serveurs.

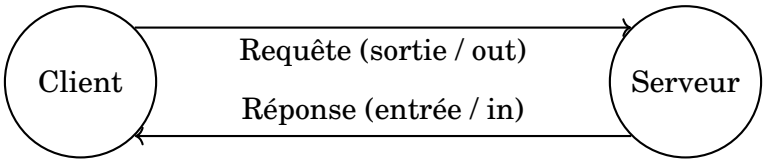


FIGURE 1.9 – Modèle de fonctionnement client / serveur (vision client)

Les clients sont les appareils utilisés par vous et moi pour se connecter au web ; les logiciels utilisés pour y accéder sont appelés des navigateurs web (voir p. 6 sur le sujet).

Les serveurs sont des ordinateurs qui hébergent des pages web, des sites ou des applications. Lorsque nous voulons accéder à une page web depuis un appareil client, une copie de la page est envoyée du serveur vers notre appareil, par le biais de réseaux informatique, contenu qui s'affiche alors dans le navigateur utilisé.

1.3.2 Composants clés des réseaux

Routeurs

Les routeurs, appareils de filtrage d'informations utilisés dans les réseaux informatiques, sont la base de tout réseau : ils sont équipés de deux connecteurs. Une entrée qui reçoit et analyse les informations reçues, qui, en fonction de la table de routage, décide s'il est nécessaire de les retransmettre plus loin (sortie).

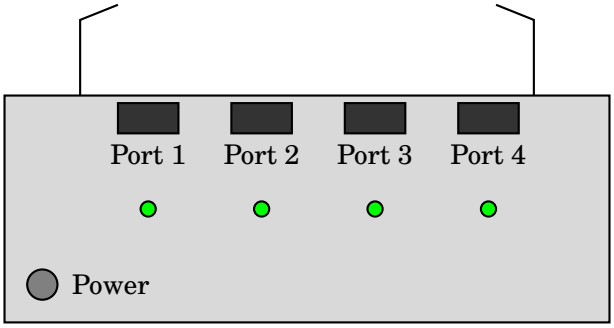


FIGURE 1.10 – Routeur, quatre ports Ethernet et antennes WiFi

Hub

Un hub est aussi un appareil relais. Il est équipé de plusieurs connecteurs, chaque information reçue par un des connecteurs est retransmise sur tous les autres.

Switch

Un switch est, tout comme les hubs, équipé de plusieurs connecteurs d'entrée, mais chaque information reçue par un des connecteurs est analysée et transmise sur le connecteur unique (souvent en fibre optique) vers un seul destinataire qui renvoi les données demandées en retour.

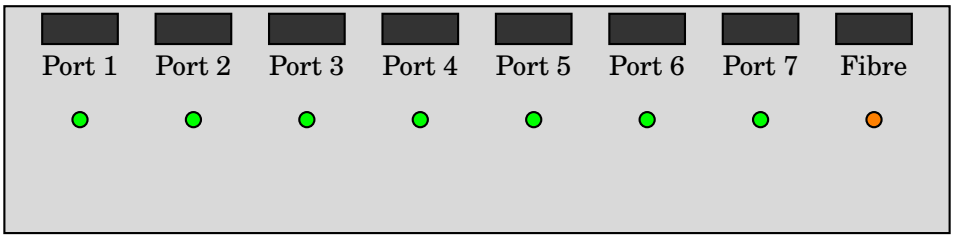


FIGURE 1.11 – Switch, sept ports Ethernet, une connection fibre

Firewall

Un firewall (ou pare-feu) est un système de sécurité qui filtre les communications entre un réseau interne (ordinateur, entreprise) et l'extérieur (Internet). Son rôle est de bloquer les connexions suspectes ou non autorisées, tout en laissant passer celles qui sont légitimes. Pour

cela, il applique des règles définies à l'avance, par exemple en fonction d'une adresse IP, d'un port ou d'un protocole utilisé. On distingue plusieurs formes de firewalls : les logiciels, installés directement sur un ordinateur ou un serveur (comme Windows Defender Firewall), et les matériels, sous forme de boîtiers ou intégrés dans des routeurs. Dans les grandes entreprises, les firewalls matériels peuvent analyser des millions de paquets par seconde et incluent des fonctions avancées (détection d'intrusion, filtrage applicatif). À la maison, c'est souvent la box Internet qui joue ce rôle, complété par le pare-feu logiciel intégré au système d'exploitation.

1.3.3 Types de réseau

Les résaeux sont souvent classés en fonction de leurs utilisations et des services qu'ils offrent. Ainsi, pour les réseaux utilisant les technologies Internet (famille des protocoles TCP/IP), la nomenclature est la suivante :

— **Intranet** : réseau interne

— **Extranet** : réseau externe

— **Internet** : réseau de réseaux externes, interconnectés à l'échelle de la planète

La classification ci-dessus s'applique à une famille de protocoles donnée. De manière générale, il existe plusieurs types de classifications différentes, en fonction de l'organisation (architecture) du réseau ou du mode de communication, chacune étant divisée en sous-catégories selon le type de classification.

Classification selon l'organisation du réseau

Un réseau client / serveur se caractérise par un ou plusieurs ordinateurs (PC clients) qui se connectent à un ou plusieurs serveurs pour accéder à des ressources, des services, ou des applications. Ces serveurs peuvent être, par exemple, des serveurs de fichiers, des serveurs de messagerie ou, comme dans le cas 1.12, des serveurs web. La communication dans un réseau client-serveur se fait la plupart du temps via des protocoles comme HTTP, FTP, ou TCP/IP, mais pas que. Ce modèle d'organisation permet d'avoir une vision centralisée des ressources et de son administration.

Une architecture Peer-to-peer (P2P) est, elle, décentralisée : chaque ordinateur, appelé "pair" (ou "peer"), fonctionne à la fois comme client et serveur. Il n'y a donc pas d'architecture préétabli : chaque pair peut demander et fournir des ressources directement aux autres pairs du réseau. Les ressources, telles que les fichiers, la bande passante ou la puissance de calcul, sont partagées entre les pairs, souvent via des protocoles spécifiques au P2P comme BitTorrent. Ce mode de fonctionnement a été grandement popularisé par des applications de communication comme Skype ou WhatsApp, mais aussi par les blockchains qui sous-tendent le fonctionnement de Bitcoin et autres cryptomonnaies. Bien qu'extrêmement pratique, ce type de réseau peut soulever des questions en termes de sécurité, de gestion des ressources ou de protection de la vie privée, car les données sont distribuées sur plusieurs machines qui ne sont pas nécessairement contrôlées.

Classification selon le mode de communication

Les réseaux terrestres (par câbles) permettent une communication rapide entre les ordinateurs, une utilisation rationnelle des données partagées et une stratégie plus facile à maîtriser dans le domaine de la sécurité (contrôle et sauvegarde des données). Cependant, un tel type de réseau a pour inconvénient sa complexité (nécessitant souvent le recours à un personnel spécialisé en cas de problème). La configuration en étoile est la situation la plus souvent rencontrée en entreprise. Selon la taille du réseau, on parlera de PAN (Personal Area Network), LAN (Local Area Network) ou, à l'échelle d'une région, d'un pays ou d'un continent, de WAN (Wide Area Network).

Les réseaux sans fil (ou *wireless network*) permettent, de leurs côtés, à des appareils de communiquer entre eux sans utiliser de câbles, grâce à des ondes radio ou infrarouges. Les utilisateurs peuvent donc rester connectés tout en se déplaçant (dans une zone donnée relativement reteinte). De la même façon que pour les réseaux terrestre, selon la taille du réseau sans fil, on parlera de WPAN (Wireless Personal Area Network pour les connexions type Bluetooth ou infrarouge), WLAN (Wireless Local Area Network pour le WiFi par exemple) ou, à l'échelle d'une région ou d'un pays, de WWAN (Wireless Wide Area Network de le cas par exemple de réseaux téléphoniques).
Le saviez-vous ? Les ondes électromagnétiques sont constituées de deux champs : électrique et magnétique. La qualité du signal diminue inévitablement à mesure que l'on s'éloigne de la source, en raison des perturbations environnantes. Plus la fréquence est élevée, plus le signal est sensible aux perturbations, mais en contrepartie, la transmission des données est plus rapide. Débit qui dépend aussi de la modulation de l'onde, laquelle est déterminée par la fréquence et le type de modulateur utilisé.

1.3.4 Protocoles de communication

Un protocole est un ensemble de règles mises en place pour le bon fonctionnement d'un processus.

TCP/IP

Comme vu précédemment, Internet utilise le modèle TCP/IP, une architecture de réseau en quatre couches : la couche d'accès au réseau physique (Ethernet ou WiFi), la couche responsable des adresses et du routages des données appelée le protocole Internet (IP), celle qui assure le contrôle des transmissions (TCP) et enfin, celle qui gère les échanges entre les applications : de nombreux protocoles existent, les plus connus sont le transfert de lien hypertexte (HTTP), le transfert de fichiers (FTP), de simple message (SMTP) ou encore le système de nom de domaine (DNS).

Concrètement, lorsque vous saisissez une adresse (URL) dans votre navigateur, ce dernier demande au serveur DNS situé chez l'hébergeur du site, l'adresse IP réelle du site web. Il en-

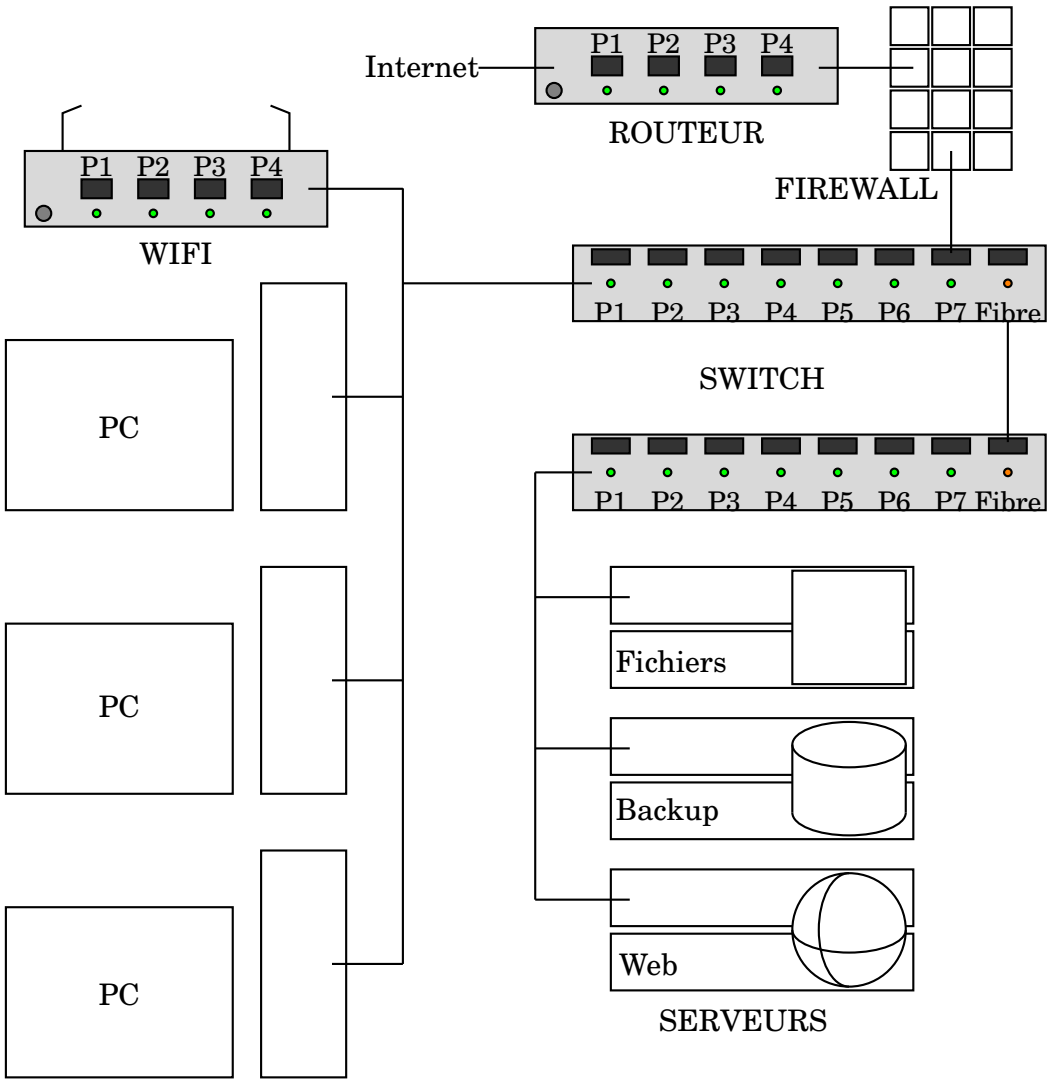


FIGURE 1.12 – Réseau de type client / serveur (en étoile)

voie une requête HTTP pour obtenir une copie du site, un peu comme si vous passiez une commande dans un restaurant. Cette communication se fait via une connexion internet en utilisant le protocole de cryptage TCP/IP. Le navigateur assemble ensuite les différents paquets reçus et affiche ainsi le site sur votre écran.

Zoom sur les adresses IP

Adresses privées Chaque machine connectée à un réseau est associée une adresse IP unique, permettant de communiquer avec les autres machines. Cette adresse IP est fixée par l'administrateur du réseau ou attribuée automatiquement au démarrage grâce au protocole DHCP (*Dynamic Host Configuration Protocol*). Une adresse IP standard (IPv4) est formée de 4 octets (32 bits), compris entre 0 et 255, séparés par des points. Souvent l'administrateur d'un réseau donnera un intervalle pour les adresses locales (de 192.168.0.0 à 192.168.255.255). L'adresse est également associée à un masque qui fournit le nombre de bits de l'adresse, sur une réseau local, le masque sera souvent 255.255.255.0.

Ouverture des réseaux privés sur Internet Lorsqu'un réseau local est relié à Internet, l'ordinateur avec la connexion Internet sert de relais pour les autres ordinateurs du réseau local. Cet ordinateur possédera ainsi deux adresses IP : une pour le réseau local et une pour la connexion Internet.

1.3.5 Sécurité et accès distant

1.3.6 VPN (Virtual Private Network) et SSH (Secure Shell)

Un **VPN** fait référence à l'utilisation d'une technologie qui permet de créer une connexion sécurisée et cryptée entre deux réseaux via Internet. L'outil VPN, très utilisé dans toutes les entreprises, masque l'adresse IP de l'utilisateur et chiffre toutes les données échangées. De cette manière, les informations restent confidentielles et protégées des interceptions éventuelles par des tiers.

Les informaticiens et administrateurs réseaux, utiliseront souvent des **SSH**, protocoles de communication sécurisés utilisé également pour accéder de manière sécurisée à des systèmes informatiques. Une fois l'authentification réussie, ce protocole permet de gérer un serveur à distance, transférer des fichiers (Secure File Transfer Protocol SFTP), exécuter des commandes, etc.

Ce mode de connexion est souvent utilisé via une session terminal lancée par `ssh nomutilisateur@monsite.ch`. À noter que, contrairement au VPN qui gère la connexion au niveau des réseaux, la connexion SSH intervient au niveau des applications, rendant donc son utilisation beaucoup plus spécifique [4].

Erreurs HTTP

Pour mémoire, les erreurs HTTP sont des messages que les serveurs renvoient aux navigateurs pour indiquer des problèmes lorsqu'ils traitent les requêtes des utilisateurs. Parmi les erreurs les plus fréquentes, on retrouve les codes 403 et 404.

L'erreur **403** indique que l'accès à la ressource demandée est interdit. Cela se produit généralement lorsque le serveur comprend la requête, mais refuse de l'exécuter en raison de permissions insuffisantes.

L'erreur **404** se produit lorsque le serveur ne trouve pas la ressource demandée. Cela signifie que le fichier ou la page a été déplacé, supprimé ou que l'URL saisie est incorrecte.

DoS (Denial of Service)

Une attaque **DoS** vise à rendre un service ou un serveur indisponible. Le principe est le suivant : de nombreux appareils compromis (appelés botnets) envoient un grand nombre de re-

4. SSH sert, en effet, à sécuriser une connexion précise, par exemple pour gérer un serveur ou transférer des fichiers (SFTP). Un VPN, lui, crée un tunnel chiffré qui protège tout le trafic réseau de l'ordinateur (web, mails, applis, etc.)

quêtes vers un serveur cible, le surchargeant jusqu'à ce qu'il ne puisse plus répondre aux requêtes légitimes. Divers mesures permettent, au niveau d'une entreprise de mieux se protéger : l'utilisation de pare-feux et listes noires d'adresses IP reconnues comme corrompues ou la mise en place d'un répartisseur de charge du trafique (Content Delivery Networks).

Le saviez-vous ? En 2016, une attaque massive a frappé le fournisseur de DNS Dyn, rendant inaccessibles de nombreux sites comme Twitter (maintenant appelé X), Netflix et Reddit. L'attaque a utilisé des millions d'appareils IoT (Internet of Things ou, en français, Internet des objets) compromis pour lancer cette attaque de manière coordonnée.

1.3.7 À retenir...

Points clés abordés

- Fonctionnement des réseaux
- Le Web et son histoire
- Modèle client-serveur
- Équipements de réseau
- Types de réseaux (classification)
- Réseaux câblés et sans fil
- Protocoles de communication (TCP/IP)

Questions récapitulatives

1. Quelle est la différence entre le modèle client-serveur et peer-to-peer (P2P) en termes d'architecture réseau et gestion des ressources ?
2. Pourquoi le protocole TCP/IP est-il si important pour le fonctionnement d'Internet ? Expliquez les rôles spécifiques des couches TCP et IP.
3. Quels sont les avantages et inconvénients d'un réseau sans fil par rapport à un réseau câblé dans une entreprise ?
4. Expliquez en quoi un routeur diffère-t-il d'un switch.
5. Donnez des exemples d'adressage IP dans un réseau local. Quelle est l'utilité du protocole DHCP pour la gestion de ces adresses ?
6. Pourquoi les réseaux de type LAN (Local Area Network) sont-ils souvent utilisés ?
7. Que signifie concrètement le S dans HTTPS ?
8. Quels rôles jouent les DNS dans une connexion web ?
9. Comment un réseau local peut-il accéder à Internet de manière sécurisée à l'aide d'une adresse IP publique ? Illustrer par un schéma.
10. Comment les ondes électromagnétiques sont-elles utilisées dans les réseaux sans fil pour transmettre les informations ?

Page vide

Chapitre 2

NOTIONS D'ALGORITHMIQUE

2.1 Introduction à l'algorithmique

L'algorithmique est une composante essentielle de la science informatique. Elle se situe au coeur de la façon dont nous communiquons avec les ordinateurs et leur indiquons quoi faire. À travers cet apprentissage, vous découvrirez les principes fondamentaux qui permettent de créer des programmes informatiques.

2.1.1 Notion d'algorithmique

Définition et rôle des algorithmes

Un **algorithme** est une méthode permettant de résoudre un problème de manière systématique. Cette définition montre bien que les algorithmes ne sont pas spécifiques à l'informatique ; on peut par exemple citer :

- La résolution de problèmes
- Les recette de cuisine
- Déterminer un itinéraire
- Assembler un meuble

Ce qui distingue un algorithme d'une autre méthode de résolution de problèmes, c'est le caractère systématique de son exécution. En d'autres termes, un algorithme ne doit demander aucune initiative à celui ou celle qui l'exécute. Ceci explique l'importance de l'algorithmique en informatique : un ordinateur n'étant capable d'aucune initiative, il ne peut exécuter une tâche que si on lui fournit strictement un algorithme pour le faire. Tout programme informatique a donc, par définition, la structure d'un algorithme.

Dans cette partie du cours nous n'aborderons pas les spécificités d'un langage en particulier, les notions présentées restent ainsi valables dans la majorité des cas pour

1. Décrire un problème
2. Diriger sa résolution
3. Traiter les données efficacement et de manière logique pour, finalement,
4. Résoudre des problèmes complexes (souvent séquencé en étapes plus simples)

2.2 Démarche pour la résolution d'un problème

2.2.1 Analyse de problèmes et décomposition

Exercices p. 133

Imaginons que vous avez un problème compliqué et que vous voulez utiliser un ordinateur pour le résoudre. Pour y arriver, nous vous proposons de le décomposer en six étapes :

Étapes de base

1. Comprendre le problème Parfois, on commence avec une idée un peu floue de ce qu'on veut faire. On doit transformer cette idée en quelque chose de très clair et précis pour vous afin que l'ordinateur puisse l'interpréter correctement.

2. Rédiger les "spécifications" Les spécifications (ou "spéc.") sont des représentations plus ou moins détaillées qui expliquent ce que doit faire le programme, sans toutefois dire comment le faire. On y décrit les informations que le programme va recevoir, ce qu'il doit faire avec, et ce qu'il doit donner à la fin (voir modélisation page suivante).

Développements

3. Créer votre algorithme Après avoir une spécification claire, on va crée un algorithme. On commence par diviser le gros problème en plusieurs petits problèmes plus simples, et trouver des solutions à notre portée.

4. Vérifier l'algorithme L'algorithme doit etre testé pour que l'on soit sûr qu'il fonctionne correctement. On vérifie qu'il donne les bonnes réponses à différentes questions, qu'il ne commet pas d'erreurs et qu'il ne tourne donc pas en rond sans fin.

5. Écrire le programme correspondant Souvent, on utilisera un langage de programmation pour écrire le programme. C'est là qu'on transforme l'algorithme en quelque chose que l'ordinateur peut vraiment comprendre et exécuter (c'est le sujet du prochain chapitre ;).

Contrôles et exécution

Enfin, sixième étape, le programme que vous avez écrit est transformé en "exécutable" (pour réaliser ce qui est demandé) que vous devrez bien souvent encore le tester (ou faire tester) !

En résumé, pour résoudre un problème, vous devrez définir les...

1. **Consignes** Quelles sont (avec vos propres mots) les informations données pertinentes ?
2. **Contraintes à respecter** Exemple, les outils à utiliser ou le temps de réalisation.
3. **Attentes** En d'autres termes, quel est le rendu souhaité et que me demande-t-on ?
4. **Conditions de réalisation** Comme par exemple la date de reddition (ce qui est différent du temps de réalisation mentionné au point 2).

2.2.2 Modélisation

En algorithmique, quand on parle de "problème" ou de "modélisation", c'est simplement une question ou une tâche que l'on doit effectuer ou qu'on veut résoudre de manière organisée et efficace. Cependant...

— Trouver la racine carrée de 25

— Alice a dix bonbons et en donne deux à Bob... Combien en reste-t-il pour Alice ?

— Trouver le trajet le plus court pour aller, à 17h, de l'école à chez moi

... ne sont, en soit, pas vraiment utiles (sur le moyen ou long terme) parce qu'ils ont une réponse unique à une question spécifique. Ce qui est plus intéressant, c'est de trouver des solutions utilisables à plusieurs reprises, et qui fonctionnent, encore mieux, pour plein de situations différentes, comme par exemple :

— Trouver la racine carrée d'un nombre

— Si Alice a un certain nombre de bonbons et en donne, combien lui en reste-t-il ?

— Trouver le trajet le plus court entre deux endroits, peut importe l'heure choisie

Spécifier les données du problème Exercices p. 135

Pour trouver la meilleure solution (ou, pour le moins, une solution acceptable), il faut d'abord modéliser le problème. Pour cela, on procède, généralement, de la manière suivante :

1. **Paramètres d'entrée** Les informations de départ, par exemple le nombre de bonbons qu'Alice possède
2. Définition des **contraintes** Règles (appelée souvent pré-contraintes) que les paramètres doivent respecter, par exemple on ne peut pas avoir un nombre négatif de bonbons
3. Identification du ou des **paramètres de sortie** Résultat attendu à la fin, par exemple le nombre de bonbons restants
4. Clarification les **post-conditions** Règles que la réponse finale doit respecter pour être validée, par exemple le résultat doit être un entier positif (pas un nombre réel)

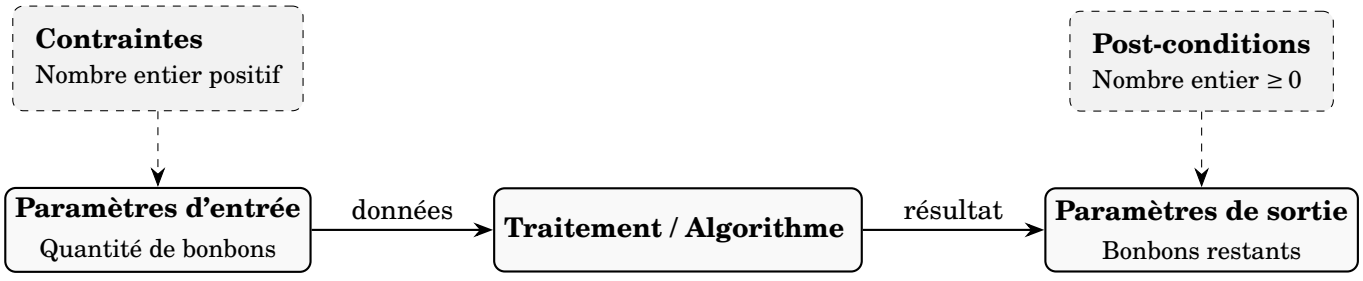


FIGURE 2.1 – Modélisation d'un problème

2.2.3 Application de l'algorithmique à la programmations (résumé)

Affectation de variables

En algorithmique, "affecter" une valeur à une ou des variables signifie enregistrer cette valeur dans une "boîte" que l'on a étiquetée avec un nom spécifique. Exemple :

```
a <-- 5
```

Tant que / While

La boucle `tant que` ou `while` (en anglais) nous permet de répéter des instructions tant qu'une certaine condition est vraie. Exemple :

```
tant que a est différent de... faire
```

Si / If

La structure `si` ou `if` (en anglais) nous permet de prendre des décisions dans notre algorithme. Exemple :

```
si a est impair alors...
```

2.2.4 Représentations algorithmiques

Résolution d'un problème de manière séquencée et méthodique

Appliquons ce que nous avons appris à un problème concret. Imaginons que vous deviez multiplier deux nombres, mais qu'au lieu d'utiliser la manière classique de procéder, vous devez innover en appliquant la **multiplication russe**. Voici les principes qui vous sont imposés :

A Diviser et Doubler

- Prenez le premier nombre (multiplicateur) et divisez-le par deux à chaque étape, en ne retenant que la partie entière du résultat (quotient). Continuez jusqu'à obtenir 1.
- En parallèle, prenez le deuxième nombre (multiplicande) et doublez-le à chaque étape.

B Sélectionner et Additionner

- À chaque ligne (ou étape), si le nombre de gauche (multiplicateur divisé) est impair, alors conservez le nombre de droite (multiplicande doublé).
- Additionnez tous les nombres ainsi retenus : la somme obtenue correspond au produit recherché !

Démarche

1. Analyser et décompoer le problème à l'aide des six étapes clés vues auparavant, soit ici, utiliser une méthode alternative pour effectuer la multiplication entre deux grandeurs donnée
2. Rédiger des spéc.

Entrées : deux nombres, a et b, choisi par un utilisateur
Sorties : résultat r de la multiplication a par b
Pré-contraintes : a et b sont des entiers non-négatifs
Post-conditions : résultat r affiché à l'écran

3. Créer l'algorithme et écrire le programme (étapes de bases 3 à 5), illustration...

Illustrons cette méthode avec 12 x 25. On pose les deux chiffres de part et d'autre d'un trait :

12	25

On commence la division de la colonne de gauche par deux, arrondi à l'entier (soit 12 / 2 = 6, 6 / 2 = 3, etc.), puis on multiplie par deux la colonne de droite (25 x 2 = 50, 50 x 2 = 100, etc.).

6	50
3	100
1	200
0	400

Deux possibilités, soit, on barre les lignes ayant un reste nul et additionne les nombres de la colonne de droite, divisé par deux, soit on additionne les nombres correspondant aux valeurs impairs.

Méthode A

~~pair / 6~~	~~50~~
impair / 3	100 -> Σ = 100
impair / 1	200 -> Σ = 300
~~pair / 0~~	~~400 -> Σ = 300~~

Méthode B

~~r 0 / entier 6~~	~~50~~
~~r 0 / entier 3~~	~~100~~
r + / entier 1	200 -> Σ = 200
r + / entier 0	400 -> Σ = 600

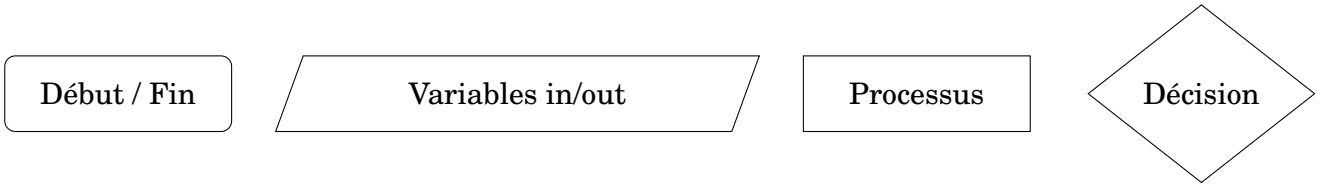
Le résultat de la multiplication trouvé est donc de 300 [1]. À noter que la séquence des restes (méthode B), du bas vers le haut, donne le code binaire du chiffre décimal (ici 12 s'écrit 1100). L'on se réferrera aux annexes, pour une résolution à l'aide d'un tableur.

Pourquoi ça fonctionne et le résultat est-il acceptable ?

Le résultat est acceptable, je peux le confirmer par exemple en effectuant le calcul de manière classique (10 x 25 + 2 x 25 = 250 + 50). La méthode, en fait, fonctionne, et cela nous permet de comprendre le raisonnement, car elle décompose (divise) le premier nombre par des multiples de 2 et impose, dans le même temps, de multiplir le deuxième nombre par le même chiffre (12 x 25 = 6 x 50 = 3 x 100 = 1 x 300). C'est une technique très ancienne, plutôt intelligente et qui montre qu'il y a plusieurs façons de résoudre un même problème !

Représentation de processus algorithmiques sous forme d'ordinogramme Exercices p. 134

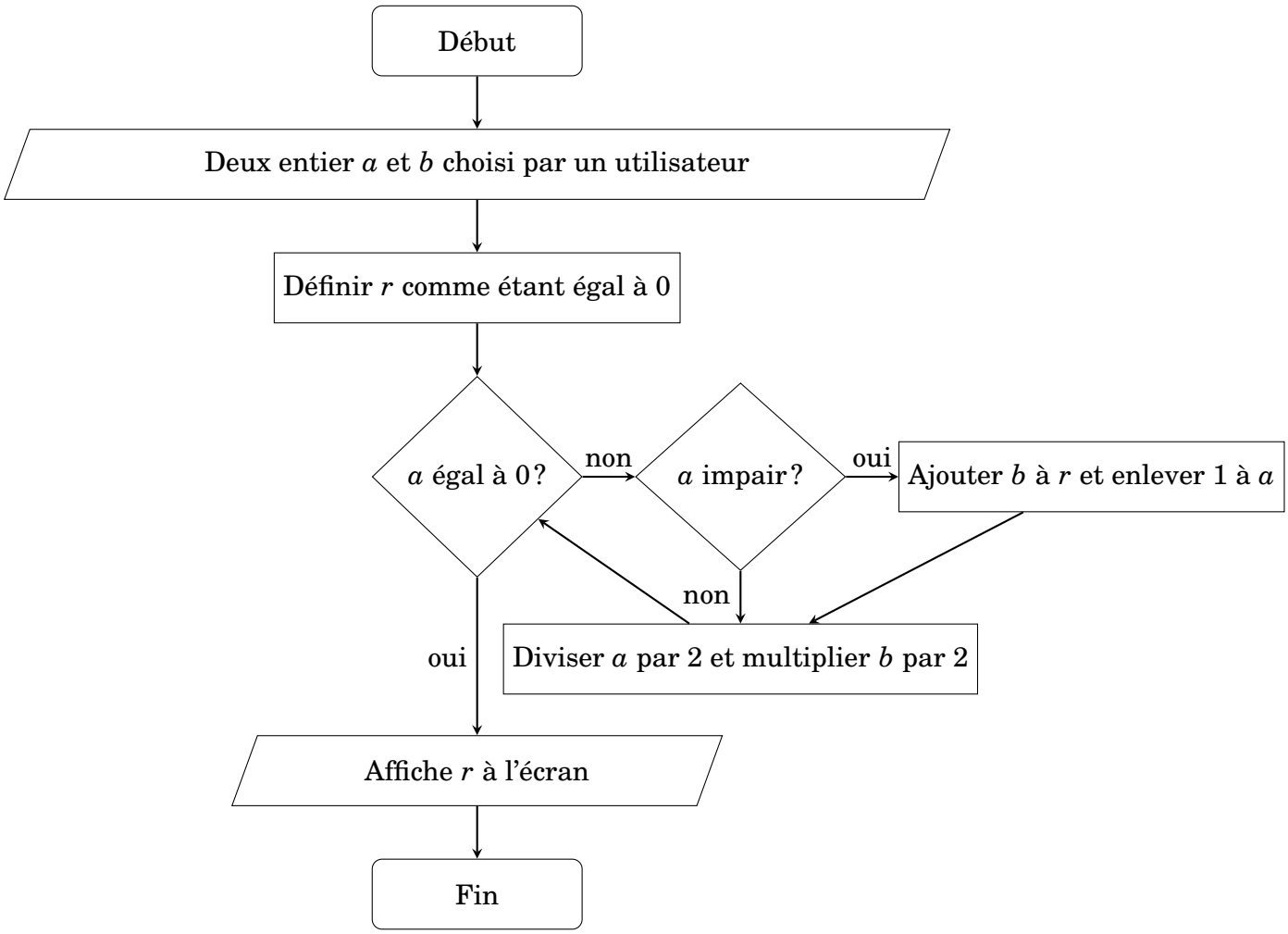
En utilisant les formes suivant ...



... on décrira le fonctionnement (processus) d'un agorithme de manière rigoureuse, ce qui, tout comme la compréhension même du problème, n'est pas toujours facile !

1. (100 + 200) avec la méthode A ou (200 + 400) / 2 avec la B... ce qui revient du point de vu du résultat, finalement, au même ;)

Dans le cas du processus de multiplication russe, à chaque itération, si a est impair, on ajoute la valeur de b à r et quelle que soit la situation, on divise a par deux (division entière) et on double b. Le processus se répète tant que a est différent de zéro.



Représentation linéaire

Le même algorithme peut être écrit sous forme linéaire, en langage séquenciel (pseudo-code).

```
1  début
2  |  r ← 0
3  |  tant que a est différent de 0 faire
4  |  |  si a est impair alors
5  |  |  |  r ← r + b
6  |  |  |  a ← a − 1
7  |  |  fin
8  |  |  a ← a/2
9  |  |  b ← bx2
10 |  fin
11 |  affiche r à l'écran
12 fin
```

2.2.5 Exemple complet d'algorithmique : recherche de dossiers

Imaginons qu'on ait une pile de dossiers sur une table. Chaque dossier a un nom écrit dessus et ils sont rangés dans l'ordre alphabétique (comme dans un dictionnaire). Notre mission, c'est de vérifier si une personne, disons M. X, a un dossier à son nom dans cette pile.

Comment aborder le problème ?

Avant de plonger tête la première, on doit réfléchir et planifier comment on va chercher. Voici ce qu'on doit définir :

Entrées : une pile de dossiers P, un nom X
Sorties : une réponse R, oui ou non, pour dire si le dossier X est dans la pile
Pré-contraintes : les dossiers P sont rangés par ordre alphabétique
Post-conditions : notre réponse doit être adéquate (M. X et non pas Mme Y) en d'autres termes,
si P contient un dossier au nom de X alors R doit être égal à oui
sinon R doit être égal à non

Recherche pas à pas

Maintenant, on peut commencer à chercher. Une première idée, c'est de prendre chaque dossier depuis le haut de la pile, un par un, et de voir si c'est celui de M. X. Si on trouve un dossier avec un nom après M. X dans l'alphabet, on peut arrêter parce qu'on saura que le dossier de M. X n'est pas là. Soit de manière plus séquentielle :

```
1  début
2  |  tant que pile de dossiers P n'est pas vide et nom du dossier au sommet de P est
   |   inférieur à X faire
3  |  |  Prendre le dossier au sommet de P
4  |  |  Le poser à coté de P
5  |  fin
6  |  si P est vide ou le nom du dossier au sommet de P est différent de X alors
7  |  |  R ← non
8  |  sinon
9  |  |  R ← oui
10 |  fin
11 fin
```

Dans cet algorithme, nous supposons que la personne est capable d'effectuer les opérations suivantes :

— Comparer (selon l'ordre alphabétique) le nom du dossier au sommet de la pile avec X

— Prendre le dossier au sommet de la pile

— Poser un dossier à côté de la pile

— Regarder si la pile de dossiers est vide

On observera en outre que cet algorithme introduit trois types d'enchaînement différents ; des enchaînements :

1. **Séquenciel** : qui consiste à effectuer les opérations les unes à la suite des autres ; par exemple, les lignes 6 et 7 s'enchaînent de manière séquenciel et, par conséquent, l'opération élémentaire de la ligne 6 n'est commencée qu'une fois que celle de la ligne 5 est terminée !
2. **Alternatif** : qui consiste à effectuer soit une première suite d'opérations, soit une autre, en fonction d'une condition ; par exemple, en fonction de la condition de la ligne 9, on effectuera alternativement l'opération élémentaire de la ligne 11, ou celle de la ligne 13...
3. **Répétitif** : qui consiste à effectuer plusieurs fois une même suite d'opérations tant qu'une condition donnée est vérifiée ; par exemple, les opérations des lignes 6 et 7 sont répétées tant que les deux conditions des lignes 3 et 4 sont vérifiées ;)

On remarquera que cet algorithme termine, c'est-à-dire qu'il s'exécute en un temps fini. En effet, si la pile contient n dossiers, les opérations des lignes 4 et 5 seront exécutées au plus n fois : à chaque fois, un dossier est enlevé de la pile ; au bout de n fois, la pile est vide et la condition de la ligne 2 et 3 n'est plus vérifiée.

L'algorithme de recherche séquentielle permet effectivement de résoudre notre problème. Cependant, il est relativement inefficace : si la pile contient 1000 dossiers, il faudra en moyenne regarder le nom de 500 dossiers avant de pouvoir résoudre le problème, et dans le pire des cas, il faudra en regarder 1000.

Cette première méthode est-elle vraiment la meilleure ?

N'y a-t-il pas une autre (meilleure) méthode ? On pourrait par exemple couper la pile en deux, regarder le nom sur le dossier au milieu, et décider de continuer à chercher dans la moitié supérieure ou inférieure en fonction du nom obtenu. On continue à couper la pile en deux jusqu'à trouver le dossier de M. X ou jusqu'à ce qu'il n'y ait plus de dossiers.

```
   Entrées : pile de dossiers P, nom X
   Sorties : réponse R
   Pré-contraintes : dossiers P rangés par ordre alphabétique
 1 début
 2     tant que pile de dossiers P n'est pas vide et nom du dossier au sommet de P est
         différent de X faire
 3         Couper la pile de dossiers P en 2 parties
 4         (On appellera P_sup la moitié sup., et P_inf la moitié inf.)
 5         si nom du dossier au sommet de P_inf = X alors
 6             Ne garder que le premier dossier de P_inf dans P;
 7         sinon
 8             si le nom du dossier au sommet de P_inf > X alors
 9                 Enlever de P tous les dossiers de P_inf;
10             sinon
11                 Enlever de P le premier dossier de P_inf;
12                 Enlever de P tous les dossiers de P_sup;
13             fin
14         fin
15     fin
16     si la pile P est vide alors
17         R ← non
18     sinon
19         R ← oui
20     fin
21 fin
```

Ce deuxième algorithme est plus complexe à comprendre (et à mettre en oeuvre) que le premier, mais il est beaucoup plus efficace. Pour 1000 dossiers : on va diviser la pile en deux parties égales, soit deux piles de 500. Ensuite, on prend une de ces piles et on la divise encore en deux. On continue comme cela jusqu'à ce qu'on trouve le dossier recherché ou que la pile soit vide... ce qui réduit à rechercher au maximum 10 dossiers en tout pour trouver celui qu'on cherche ou pour réaliser qu'il n'est pas là. C'est beaucoup moins que les 500 dossiers qu'on aurait dû consulter en moyenne avec la première méthode de recherche séquentielle !

2.3 Quelques bonnes pratiques de programmation

On peut facilement faire l'analogie entre l'écriture du code et l'écriture d'un texte. Lorsque le texte est mal écrit, qu'il contient des erreurs, que les phrases sont mal structurées et les idées pas organisées, le texte est difficile à lire et donc à comprendre. Il en va de même pour le code : brouillon, il est difficile et fatiguant à comprendre... les bugs s'y cachent facilement.

Il est donc très important de respecter certaines bonnes pratiques de programmation (qui s'appliquent à la rédaction d'algorithmes) pour rendre le plus agréable possible la lecture de code. Ce partie n'a pas vocation à vous enseigner toutes les bonnes pratiques de programmation... il y'a cependant quelques bonnes habitudes que vous devez prendre dès le début.

2.3.1 Interface et algorithmique

En algorithmique, un contrat fournit les éléments sur ce que nous les humains d'une part, et l'ordinateur qui ne comprend que les mathématiques d'autre part, attendons. Il décrit également l'interface à décelopper pour rendre utilisable les entrées données, les sorties produites, sous certaines pré-condition au fonctionnement et post-condition sur le résultat.

Par conséquent, lorsqu'on lit un contrat, i.e. que l'on consulte l'interface prévue, on est renseigné sur ce à quoi on peut s'attendre, ainsi que sur les limites de l'algorithme que l'on va utiliser, sans pour autant avoir besoin de comprendre comment l'algorithme fonctionne ou est implémenté. Par exemple, quand vous utilisez print("Hello world !") en Python, vous savez que vous allez obtenir, mais vous ne savez pas comment Python procède réellement pour afficher le message.

2.3.2 Caractéristique d'un code "bien écrit"

Un algorithme, ou code "bien écrit" doit avoir les propriétés suivantes :

1. Facile à lire, par soi-même mais aussi par les autres
2. Organiser de manière logique et la plus évidente possible
3. Explicite
4. Soigné et robuste au temps qui passe

Facile à lire

Pour que le code soit facile à lire, il faut d'une part qu'il soit bien structuré et bien présenté, et d'autre part, que les noms des variables et des fonctions soient choisis avec soin.

Pour ce qui est de la structure et de la présentation, Python nous aide beaucoup car le langage impose beaucoup de choses qui nous "forcent" à bien présenter notre code. Par exemple, les blocs d'instructions du même niveau doivent être précédés du même nombre d'espaces, ce qui nous conduit naturellement à bien indenter notre code.

Dans d'autres langages, plus de libertés de présentation sont offertes au développeur, par conséquent, il convient de se forcer à respecter des conventions pour écrire le code le plus propre possible. Par exemple, regardez le code ci-dessous. Il est évident que le manque d'indentation ne facilite pas la lecture et la compréhension du code, n'est-ce pas ?

>>>>

```
int i, j; int n = 10; int p[n][1]; int k = 0, m = 0, o = 0;
for(i = 0; i < n; i++){
p[i][0] = i; m = i; printf("----");
for(j = 0; j < n; j++){printf("|");}}
return;
```

Organisation logique et évidente

Ce point est délicat car nous avons souvent des solutions différentes pour résoudre le même problème. Il est donc normal qu'un code logique pour quelqu'un semble "tordu" à son voisin.

Étant conscient de cela, il faut vous efforcer de trouver des solutions logiques aux problèmes que vous devez résoudre et d'éviter d'emprunter des chemins plus compliqués qui ne feraient que semer la confusion. Par exemple, si l'on vous demande d'afficher tous les nombres de 1 à 10, il suffit de faire une boucle qui fait évoluer un compteur entre 1 et 10 et qui affiche, à chaque tour, la valeur de ce compteur. La solution qui consisterait à faire une boucle qui fait évoluer un compteur de 9 à 0 fonctionne aussi, mais est moins "élégante".

Code explicite

Lorsque l'on écrit des algorithmes ou que l'on développe des programmes, on est parfois tenté de prendre des raccourcis car "on sait" que telle méthode permet de faire telle ou telle chose...

Il n'est pas interdit de prendre ces raccourcis, mais il faut toujours prendre le soin d'expliquer, au moins à travers des commentaires, pourquoi on fait cela. C'est important à la fois pour permettre aux autres de comprendre pourquoi votre solution est astucieuse... mais aussi pour vous, au cas où vous ne vous souveniez plus de "pourquoi vous avez fait ça".

"Clean" et robuste

Lorsque l'on écrit du code, on a la fâcheuse tendance à s'arrêter dès que celui-ci fonctionne. C'est un tort ! Le code doit être entretenu. Cela signifie qu'il faut relire son code après l'avoir terminé, vérifié que l'on a bien supprimé les éléments obsolètes, vérifier que les commentaires sont à jour et cohérents avec le code conservé, etc.
Cette opération de "maintenance" du code est cruciale, mais elle est pourtant souvent négligée par beaucoup, ce qui peut poser des problèmes, notamment lorsque vous rencontrez un bug.

Chapitre 3

CONCEPTS GÉNÉRAUX POUR LA RÉDACTION DE SCRIPTS

3.1 Généralités

Nous apprendrons ici, en et avec Python, les bases de la programmation. On va utiliser pas mal d'exemples et on suppose que vous savez déjà un peu programmer (notamment au cycle d'orientation).

Pourquoi Python 3 ?
Nous utilisons Python 3 parce qu'il est moderne et très utilisé.

3.1.1 Petit historique

Python a été créé en 1991 par Guido van Rossum. Depuis, il a beaucoup évolué et en 2023, on utilise la version 3.12. Python est connu pour être un langage de programmation qui a beaucoup de fonctions et est facile à apprendre.

3.1.2 Pourquoi Python est cool ?

Il a plein de fonctions pour faire, en programmation, presque tout et il est ...

— Rapide et interactif

— Gratuit et fonctionne sur tous les ordinateurs

— Facile à apprendre et à utiliser

Python est utilisé par beaucoup d'écoles et d'universités pour enseigner la programmation. Il est aussi utilisé par les scientifiques comme alternative à MATLAB, surtout pour les calculs et l'analyse de données.

3.1.3 Fondamentaux pour acquérir des bases solides

Un script Python est un fichier texte qui finit par ".py". Il contient des instructions Python et commence souvent par quelques lignes spéciales appelées en-têtes.

L'interpréteur et interface utilisateur Aide-mémoire p. 175

L'interpréteur est le programme qui lit et exécute votre code. La première ligne du script, appelée "she-bang", dit à l'ordinateur quel interpréteur utiliser. Cette ligne aide à s'assurer que le script fonctionne peu importe où Python est installé sur l'ordinateur. Voici un exemple de script Python basique...

```
#!/usr/bin/env python
# Prenom NOM / Programme-Version
print("Hello world")
```

CODE 3.1 – PYTHON En-têtes

Si vous voulez exécuter ce script directement depuis un shell, cette première ligne est nécessaire. Sinon, vous pouvez également lancer le script en utilisant un des logiciels disponible (interface utilisateur graphique d'un environnement de développement intégré IDE / GUI) sur les ordinateurs : InformaticLi, Thonny ou TigerJython [Arnold *et al.*, 2022]. Pour plus d'informations sur la rédaction de scripts, voir le chapitre correspondant, p. 57.

Instructions sur plusieurs lignes et commentaires

Une ligne de code ne devrait pas être trop longue (environ 80 caractères). Si vous avez besoin de plus de place, vous pouvez continuer sur la ligne suivante avec un back-slash (\). Mais si vous utilisez des parenthèses, des crochets, ou des accolades, ou si vous écrivez des textes longs (avec ' ' ' ou " " "), vous pouvez écrire sur plusieurs lignes sans le back-slash.

Commentaires

Les commentaires sont des parties de votre code qui ne sont pas exécutées. Ils sont utiles pour expliquer ce que fait votre code.

— Tout texte après le caractère # jusqu'à la fin de la ligne est un commentaire

— Vous pouvez aussi créer des commentaires qui s'étendent sur plusieurs lignes

```
# Commentaire une ligne
print("x = 2") # Instruction et commentaire

'''Ceci est un commentaire
   sur plusieurs lignes...'''

"""Et ceci est un autre
   commentaire multi-ligne"""
```

CODE 3.2 – Commentaires

Instructions sur plusieurs lignes

En Python, vous pouvez écrire du code sur plusieurs lignes de différentes manières :
- Si vous utilisez des parenthèses ( ), des crochets [ ], ou des accolades , vous pouvez répartir votre code sur plusieurs lignes sans utiliser le back-slash (\). - Pour les textes longs, vous pouvez toujours, comme mentionné plus haut, utiliser des triples apostrophes `'''` ou des triples guillemets `"""`.

```
# Utilisation du back-slash \ pour continuer sur la ligne suivante
print("Affichage d'une longue ligne \
       de texte. Blabla blabla blabla \
       blabla blabla blabla blabla blabla.")

# Liste repartie sur plusieurs lignes
liste = ["un", "deux", "trois",
         "quatre", "cinq", "six"]

# Chaine de caracteres multi-lignes
chaine = """Une chaine de caracteres
            multi-lignes"""
```

Variables et mots-clés réservés

Les noms des variables en Python doivent commencer par une lettre minuscule ou un souligné, suivi de lettres, chiffres ou soulignés. Attention, Python fait la différence entre majuscules et minuscules ! Pour plus de détails, on se référera au chapitre suivant sur les conventions pour nommer fonctions, variable ou fichiers. À noter qu'il y a des mots spéciaux appelés "mots-clés réservés" que vous ne devez pas utiliser comme noms de variables. Notamment...

```
>>>>
False, None, True, and, as, assert, break, class,                    #1
continue, def, del, elif, else, except, finally,                     #2
for, from, global, if, import, in, is, lambda,                       #3
nonlocal, not, or, pass, raise, return, try, while,                  #4
with, yield                                                          #5
```

Comprendre le typage

Python : un typage dynamique fort

Python est un langage où vous n'avez pas à dire de quel type est une variable (comme nombre, texte, etc.) avant de l'utiliser. Le type de la variable peut changer pendant que vous utilisez votre programme. Si vous mélangez des types qui ne vont pas ensemble (comme ajouter un nombre à du texte), Python vous dira qu'il y a une erreur.

Vérifier le type d'une variable

Pour savoir de quel type est une variable, utilisez la fonction `type()`.

Les types simples en python

Python a quatre types simples de base...

- `bool` : pour vrai ou faux
- `int` : pour les nombres entiers
- `float` : pour les nombres avec des décimales
- `complex` : pour les nombres complexes

On verra plus loin que ces quatres types sont immutables (contrairement par exemple aux listes et dictionnaires).

Convertir et arrondir des nombres

On dispose par ailleurs de fonctions d'arrondi et de conversion, notamment :

- Pour changer un nombre à décimales en entier : `int()`
- Pour arrondir un nombre : `round()`
- Pour convertir un entier en nombre à décimales : `float()`
- Pour avoir la valeur absolue (sans signe - ou +) d'un nombre : `abs()`.

```
vlog = True      # vlog est vrai
not vlog         # => False

a = 2            # int (entier) par defaut
b = 2**80        # => 1208925819614629174706176 (2 puissance 80)

# Nombre decimaaux
c = -3.3
abs(c)           # => 3.3

# Arrondi et conversion
int(3.67)        # => 3 (int)
round(3.67)      # => 4 [equivalent: round(3.67, 0) ou int(3.67)]
round(3.67,1)    # => 3.7 (float)
round(133,-1)    # => 130 (int)
round(133,-2)    # => 100 (int)

# Nombres complexes
d = 3.4 + 2.1j
d.real           # => 3.4
d.imag           # => 2.1
```

CODE 3.3 – Formats de nombres

Assignation simple et multiple Exercices p. 136, aide-mémoire p. 175

Assignation simple

On vient de voir plus haut que l'assignation de variables s'effectue classiquement avec =. Pour enlever une variable ou un objet de la mémoire, on utilisera `del`.

```
x = 3  # x prend la valeur 3
del x  # x n'existe plus
```

Assignation multiple

Python permet de faire aussi de l'assignation multiple, c'est-à-dire : donner la même valeur à plusieurs variables en même temps ou assigner plusieurs valeurs en même temps à différentes variables

```
x1 = x2 = x3 = 3       # x1, x2, x3 prennent tous la valeur 3
y1, y2, y3 = 4, 5, 6   # y1 prend 4, y2 prend 5, y3 prend 6
y1, y2 = y2, y1        # Maintenant y1 vaut 5 et y2 vaut 4
```

CODE 3.4 – Variables (Assigner une valeur aux ...)

Interagir avec l'utilisateur (entréer - sortie) Exercices p. 136, aide-mémoire p. 175

Affichage dans la console (sortie / print)

Pour afficher des informations à l'utilisateur, on utilise la fonction `print`.

```
# Cinq resultats equivalents
print("Hello Croquignol")                       # 1. Version de base
print("Hello ", end=""); print("Croquignol")    # 2. Message sur une ligne

# Avec utilisation de variables
msg = "Hello Croquignol"; print(msg)            # 3. Une seule variable
prenom = "Croquignol"; print("Hello", prenom)   # 4. Mix base et variable
msg = f"Hello {prenom}"; print("Hello", msg)    # 5. Format avec f

# Formats avances de diverses variables
x, y, age, montant = 2, 3, 20, 50
print("Le produit de", x, "et", y, "est", x*y)
print("J'ai {age} ans")                         # Python adaptera le format
phrase = "Je m'appelle {} et j'ai {} ans.".format(prenom, age)
phrase = "Je m'appelle {p} et j'ai {a} ans.".format(p=prenom, a=age)

# Autres methodes de formatage
print("%s = %.2f CHF" %("Depenses",montant))    # %String = %Float
```

CODE 3.5 – Fonction print (afficher dans...)

Lecture au clavier (entrée / input)

On utilise pour cela la fonction `input(prompt)` qui affiche le `prompt` puis retourne ce que l'utilisateur a frappé **systématiquement** sous forme de chaîne.

```
nom   =       input("Votre nom ? ")      # Chaine de caractere
poids = float(input("Votre poids ? "))   # Erreur si mauvaise saisie
annee = int   (input("Votre annee de naissance ? "))
```

CODE 3.6 – Fonction input (demander à...)

Tout ce que l'utilisateur tape est considéré comme du texte. Si vous avez besoin d'un nombre, vous devez convertir ce texte en nombre (avec `int` pour les entiers ou `float` pour les nombres à décimales).

Opérateurs simples Aide-mémoire p. 175

Opérateurs mathématiques

Très logiquement, en Python, on utilise des opérateurs pour faire des calculs...

```
3 + 2           # Addition       => 5
10 / 3          # Division       => 3.3333... (avec decimales)
10 // 3         # Entier         => 3         (sans decimales)
2 * 3           # Multiplication => 6
5 ** 2          # Puissance      => 25 (5 au carre)
5 % 2           # Reste          => 1 (reste de la division de 5 / 2)
"oh" + 2*"la"   #                => ohlala
(1,2) + 2*(4,)  #                => (1, 2, 4, 4)

# Incrementation / Decrementation
val  = 10
val +=  1       # val=val+1      => 11
val -=  1       # val=val-1      => 10
val *=  2.5     # val=val*2.5    => 25.0
val /= 10       # val=val/10     => 2.5
val %=  2       # val=val%2      => 0.5

fruit = "pomme"
fruit += "s"    #                => pommes
```

CODE 3.7 – Opérateurs mathématiques

Note : en Python, les opérateurs ++ et −− n'existent pas !

Opérateurs de comparaison

Pour comparer des choses, on utilise :

— <, <=, >, >= pour les comparaisons numériques

— == pour vérifier l'égalité

— != pour tester la différence

— `is` pour vérifier si deux objets sont identiques

— `in` pour vérifier si un élément est dans un autre

Opérateurs logiques

— `and` : vrai si les deux côtés sont vrais

— `or` : vrai si au moins un côté est vrai

— `not` : inverse le résultat (vrai devient faux, et vice-versa)

```
print(True)                          # => True
1 <= 2                               # => True

"oui" == "OUI".lower()               # => True
[1,3] == [1,2]                       # => False
[1,3] != [1,2]                       # => True

a=[1,2] ; b=[1,2]; a is b            # => False /!\
a=[1,2] ; b=a;     a is b            # => True

2 in [1,2,3]                         # => True
"cd" in "abcdef"                     # => True

1 < 2  and  2 > 3                    # => False
1 < 2  or   2 > 3                    # => False
not False                            # True car inverse de False...
```

CODE 3.8 – Opérateurs logiques et de comparaison

3.1.4 Nomenclature

Vous trouverez ci-dessous les recommandations données par le guide de style PEP 8 et les conventions pour documents PEP 257 [1], toutes deux adaptées du guide de style écrit par Guido van Rossum [2]. Se référerer aux différents chapitres pour les définitions de certains termes.

Raisons d'être de cette nomenclature

L'objectif est ici, simplement, de normaliser le nom des choses, indiquant notamment ce qu'elles contiennent. Ces recommandations sont juste des conventions, pas des lois ou de la syntaxe gramaticale.

> *Une convention fournit de la clarté, de la cohérence et une base pour de bonnes habitudes de programmation. Ce qu'elle ne fait pas, c'est insister pour que vous la suiviez contre votre gré. C'est la philosophie de Python !*
>
> — Tim Peters, juin 2001

À notre que certains logiciels utilisent ces conventions, et les respecter vous permettra d'obtenir simplement de meilleurs résultats ;)

TABLE 3.1 – Styles de nomenclature les plus courants

Style	**Utilisation**
> b	Lettre minuscule simple, variables simples
> B	Lettre majuscule simple, variables simples
> minuscule	Fonctions, variables courantes et modules
> minuscule_avec_tirets_bas	Fonctions et variables de trois mots ou plus
> MAJUSCULE	Constantes courantes
> MAJUSCULE_AVEC_TIRETS_BAS	Constantes complexes
> premierMotMinuscule	Fonctions et variables de deux, trois mots max.
> MotPremieresLettresMajuscules ...	Les classes
> Mot_PremiereLettreMajuscule	Peu utilisé... pour les noms de fichier uniquement
> Mot_Premiere_Lettre_Majuscule ...	Pas recommandé en dehors des noms de fichier

1. Disponibles sur le site web de Python peps.python.org/pep-0008
2. Voir le Glossaire ou le petit historique sous Généralités pour savoir qui c'est :)

Remarques :

1. Lorsque on utilise des acronymes, on mettra toutes les lettres de l'acronyme en majuscule. Ainsi, HTTPServerError est préférable à HttpServerError
2. Existe aussi, dans certains langage, la convention d'utiliser un court préfixe unique pour regrouper des noms liés ; cette technique n'est pas beaucoup utilisé en Python...
3. Les formes spéciales utilisant des tirets bas au début ou à la fin d'un nom sont reconnues
   - **o** _tiret_bas_initial indique une utilisation interne faible (par exemple, l'instruction from M import * n'importera pas les objets dont les noms commencent par un tiret bas)
   - **o** tiret_bas_final_ est surtout utilisé pour éviter les conflits avec les mots-clés Python (par exemple class_="ClassName")
   - **o** __double_tiret_bas_initial est utilisé pour nommé un attribut dans une classe (à l'intérieur de la classe ClassName, __boo devient _ClassName__boo).
   - **o** __double_tiret_bas_initial_et_final__ sont des objets ou attributs "magiques" qui ont des raisons d'être particulière (notamment, __init__, __import__, __file__ ou __main__)... ne les utiliser que de manière convenue

Noms à éviter

N'utilisez jamais les caractères "l" (lettre minuscule el), "O" (lettre majuscule oh) ou "I" (lettre majuscule hi) comme noms de variables à caractère unique.

Recommendations pour nommer classes, modules, fonctions et variable

Les noms de **`classe`** doivent utiliser la convention MotsPremiereLettreMajuscule.

Les **`modules`** doivent avoir des noms courts, entièrement en minuscules. Des tirets bas peuvent être utilisés dans le nom du module seulement si cela améliore la lisibilité.

S'agissant des noms des **`fonctions`**, il est de coutume, de le faire débuter par un caractère minuscule. S'il est composé de plusieurs mots, on concatène ceux-ci et les faisant débuter chacun par une majuscule (exemple fctBienNommee). On séparera cependant les mots par des traits de soulignement pour éventuellement améliorer la lisibilité.

La dénomination des **`variables`** répond aux mêmes règles que les fonctions. À noter cependant que les noms sont sensibles aux majuscules / minuscules (PRINT ou Print retournant ainsi une erreur de type "not defined").

3.2 Containers

Les containers (ou types containers) en Python sont des types de données qui peuvent contenir plusieurs éléments. Contrairement aux types simples comme les booléens, entiers, ou nombres flottants, les containers peuvent stocker plusieurs valeurs.
On distingue fondamentalement trois catégories de containers, implémentés sous forme de 6 types de base (built-in) Python...

— les séquences comprenant les types : chaîne, liste et tuple

— les ensembles comprenant les types : set et frozenset

— les maps (aussi appelés hashs) avec le type dictionnaire

3.2.1 Chaînes de caractères

Une chaîne de caractères est une suite de lettres, chiffres ou autres symboles. En Python, les chaînes sont immuables, ce qui signifie que vous ne pouvez pas changer une chaîne directement.

```
meta1 = "argent"            # delimitation par apostrophes -
meta2 = "l'or"              # ou par guillemets

# Concatenation
meta = "L'" + meta1 + " et " + meta2  # => "L'argent et l'or"
type(meta)                  # => builtins.str

# Adressage
len(meta)                   # => 16
meta[:8]                    # idem que meta[0:8] => "L'argent"
meta[12:]                   # idem que meta[12:len(meta)] => "l'or"

# Modification
meta = meta[:9] + 'sale'    # => "L'argent sale"

# Chaine multi-ligne
chaine_multi = '''Ceci est
                  une chaine
                  sur plusieurs lignes'''
```

CODE 3.9 – Chaînes de caractères

3.2.2 Les fonctions range et enumerate

range Exercices p. 140, aide-mémoire p. 175

La fonction `range()` crée une séquence de nombres. Elle est souvent utilisée dans les boucles `for` pour répéter une action plusieurs fois (voir sur le sujet, le chapitre suivant).

```
for _ in range(5):
    print(_)                # Affiche les nombres de 0 a 4
```

CODE 3.10 – Fonction `range()` avec la boucle `for`

Vous pouvez spécifier un début, une fin et un pas pour `range()` :

```
print(*range(1, 10, 2))    # => 1, 3, 5, 7, 9... soit 1 > 10, par pas de 2
```

CODE 3.11 – Contrôler la numérotation de la fonction `range()`

enumerate

La fonctoin `enumerate()` est utilisée pour obtenir à la fois l'indice et la valeur des éléments d'une séquence (liste ou chaîne de caractères).

```
for indice, valeur in enumerate(["a", "b", "c"]):
    print(indice, valeur)  # => 0 a, 1 b, 2 c
```

CODE 3.12 – Fonction `enumerate()` avec la boucle `for`

Vous pouvez aussi spécifier une valeur de départ pour l'indice.

```
print(*enumerate(["a", "b", "c"], 1))  # => (1, "a"), (2, "b"), (3, "c")
```

CODE 3.13 – Contrôler la numérotation de la fonction `enumerate()`

3.2.3 Listes

Exercices p. 145, aide-mémoire p. 177

Une liste en Python est un type de donnée qui peut contenir plusieurs éléments, même de types différents (comme des nombres, du texte, ou même d'autres listes).

Créer et utiliser des listes

Pour créer une liste, on utilise des crochets `[]` avec des éléments séparés par des virgules.

```
ma_liste = [1, "deux", 3.0]
```

CODE 3.14 – Créer une liste

Accéder et modifier les éléments

On peut accéder aux éléments d'une liste en utilisant leurs indices (qui commencent à 0). On peut aussi modifier ces éléments.

```
element = ma_liste[0]  # => 1
ma_liste[1] = "trois"  # Change la denomination 'deux' en 'trois'
```

CODE 3.15 – Accéder et modifier les éléments d'une liste

Ajouter, parcourir et supprimer des éléments

Pour ajouter des éléments, on peut utiliser `append()` ou `insert()`. Pour les supprimer, on utilise `del` ou `pop()`.

```
ma_liste.append("quatre")  # Ajoute un element
del ma_liste[0]            # Supprime le premier element
```

CODE 3.16 – Ajouter et supprimer des éléments d'une liste

Pour parcourir tous les éléments d'une liste, on utilise souvent une boucle `for`.

```
for i in ma_liste:
    print(i)
```

CODE 3.17 – Parcourir une liste

Listes imbriquées et opérations

Les listes peuvent contenir d'autres listes. On peut aussi réaliser des opérations comme la concaténation ou la multiplication.

```
liste_imbriquee = [1, [2, 3], 4]    # Liste imbriquee
nouvelle_liste = ma_liste + [5, 6]  # Concatenation
liste_multiple = [0] * 5            # => [0, 0, 0, 0, 0]
```

CODE 3.18 – Listes imbriquées

Si l'on souhaite dupliquer les données d'une liste listeA=[...], il est très important de comprendre qu'avec Python une simple assignation listeB=listeA n'effectue pas de copie des données : les deux variables listeA et listeB référenceront en effet la même liste ! Pour réellement recopier les données d'une liste, il faut donc prendre des précautions particulières : utiliser une copie complète récursive (agissant dans toute la profondeur de la liste).

```
listeA = [1, 2, 3]
listeB = listeA  # listeB pointe vers listeA, verifiable avec print(...)

import copy
listeA = [1, 2, 3]
listeB = copy.deepcopy(listeB)
listeB[1] = "a"  # on agit dans ce cas, sur le nouvel objet
print(listeB)    # => [1, 'a', 3]
print(listeA)    # => [1,  2 , 3] :: donnees d'origine intouchees !
```

CODE 3.19 – Duppliquer une liste

3.2.4 Dictionnaires

Un dictionnaire est une collection d'éléments où chaque élément (ou occurence) est une paire clé : valeur. Les clés doivent être uniques et immuables (nombres ou chaînes), mais les valeurs peuvent être de n'importe quel type.

```
dic = {"cle1": "valeur1", "cle2": 2} # Creation du dictionnaire

dic["cle3"] = 3.0       # Ajoute un element au dictionnaire
dic["cle2"] = "deux"    # Moddifie la valeur d'une occurence

element = dic["cle1"]   # => valeur1

del dic["cle3"]         # Supprimer un element
```

CODE 3.20 – Dictionnaires

3.3 Contrôles

3.3.1 Indentation des blocs de code

En Python, l'indentation (espaces en début de ligne) est particulièrement importante. Elle détermine comment les blocs de code sont organisés.

— Utilisez quatre espaces pour chaque niveau d'indentation

— Ne mélangez pas les espaces et les tabulations

Notez encore qu'un bloc de code (ceci est notamment valable pour les fonctions, les structures `if`, `for`, `while`...) doit contenir au minimum une instruction. S'il n'en a pas, on peut utiliser l'instruction `pass` qui n'effectue aucune action.

3.3.2 Exécution conditionnelle avec if Exercices p. 137, aide-mémoire p. 175

On utilisera les conditions `if`, `elif` et `else` pour exécuter du code en fonction de conditions.

```
# Exemple de structure if - elif - else
a, b = 4, 5
if a > b:
    print("a est plus grand que b")
elif a == b:
    print("a est egal a b")
else:
    print("a est plus petit que b")
```

Expressions conditionnelles

Pour des conditions `if` simples, on utilisera plutôt des expressions conditionnelles.

```
resultat = "Pair" if val % 2 == 0 else "Impaire"
```

Fonctions logiques

Elles sont au nombre de deux :

— `any()` renvoie `True` si au moins un élément d'une séquence est vrai.

— `all()` renvoie `True` si tous les éléments d'une séquence sont vrais.

```
any(range(0,3))     # => True
                    # car elements 1 et 2 assimiles a vrai
all(range(0,3))     # => False
                    # car element de valeur 0 assimile a faux
```

Note : en Python, une valeur non nulle ou une séquence non vide est considérée comme vraie !

3.3.3 Boucles

Les boucles `for` et `while` peuvent inclure une clause `else` qui s'exécute après la fin de la boucle, sauf si la boucle est terminée par un `break`. Mais voyons cela de plus près..

For Exercices p. 140, aide-mémoire p. 175

La boucle `for` est utilisée pour répéter des instructions pour chaque élément d'une liste, d'un tuple, ou de tout objet itérable. Utilisez `range()` avec `for` pour itérer sur une séquence de nombres.

```
# Utiliser range avec for
for i in range(1, 4):
    print(i)  # Affiche 1, 2, 3

# Sur listes ou tuples
lst = [10, 20, 30]
for n in lst:
    print(n, end=" ")    # => 10 20 30

for index in range(len(lst)):
    print(index, lst[index])     # => affiche:  0 10, 1 20, 3 30

for index,val in enumerate(lst):
    print(index, val)
    # => meme affichage que ci-dessus

# Sur chaines
voyelles = "aeiouy"
for car in "chaine de caracteres":
    if car not in voyelles:
        print(car, end="")
        # => affiche les consonnes: chn d crctrs

# Sur dictionnaires
carres = {}
for n in range(1,4):
    carres[n] = n**2     # => {1: 1, 2: 4, 3: 9}

for k in carres:  # itere par defaut sur la cle !
    print(k, end=" ")    # => 1 2 3
```

While Exercices p. 140, aide-mémoire p. 176

La boucle `while` continue d'exécuter des instructions tant qu'une condition est vraie.

```
# Exemple de boucle while
i = 1
while i < 4:
```

```
    print(i)
    i += 1  # Important d'incrementer i
```

3.3.4 Instructions continue et break

Chacune joue un rôle différent :

— `continue` passe à l'itération suivante de la boucle

— `break` sort de la boucle courante

```
# Exemple d'utilisation de continue et break
for i in range(1, 10):
    if i % 2 == 0:
        continue  # Ignore les nombres pairs
    if i > 5:
        break     # Sort de la boucle si i > 5
    print(i)      # Affiche les nombres impairs jusqu'a 5
```

Liens avec les listes, sets et dictionnaires

Lorsqu'il s'agit de construire un container de type liste, set ou dictionnaire à l'aide d'une boucle For... assortie éventuellement d'une condition, Python offre une construction compacte. Sa syntaxe est :

liste = [expression for expr in iterable if cond]

set = expression for expr in iterable if cond

dict = expr1 :expr2 for expr in iterable if cond

```
# Comprehension de liste
carres = [x*x for x in range(1, 5)]  # [1, 4, 9, 16]

# Comprehension de set
impairs = {x for x in range(1, 10) if x % 2 != 0}

# Comprehension de dictionnaire
carres_dict = {x: x*x for x in range(1, 5)}
```

3.4 Fonctions, modules et scripts

3.4.1 Fonctions

Exercices p. 144

De façon générale, on implémente des fonctions lorsqu'un ensemble d'instructions est susceptible d'être utilisé plusieurs fois dans un programme. Cette décomposition en petites unités conduit à du code plus compact, plus lisible et plus efficace.

L'exemple ci-dessous illustre les principes de base de définition d'une fonction :

— La première ligne `def nomFonction(arguments):` (les deux points sont nécessaires) définit le nom avec lequel on invoquera la fonction, suivi entre parenthèses de ses éventuels arguments, ou paramètres d'entrée, séparés par des virgules

— Les instructions de la fonction (corps de la fonction) constituent ensuite un bloc qui doit donc être indenté

— au début du corps on peut définir, sous forme de chaîne ou de commentaire défini comme étant sur plusieurs lignes, le texte d'aide en-ligne (appelé docstrings) qui sera alors affiché avec la fonction help

— Avec l'expression `return` on sort de la fonction en renvoyant de manière optionelle des données de retour sous forme d'un objet de n'importe quel type (si l'on ne passe pas d'argument à return, la fonction retournera alors None)

```
# Definition d'une fonction
def nomFonction(param1, param2):
    """Description de la fonction."""
    return param1 + param2

# Utilisation de la fonction
resultat = nomFonction(3, 4)  # Donne 7
```

3.4.2 Modules

Utilisation

Un module Python (parfois appelé bibliothèque ou librairie) est un fichier rassemblant des classes et fonctions relatives à un certain domaine. On implémente un module lorsque ces objets sont susceptibles d'être utilisés par plusieurs programmes.

Pour avoir accès aux fonctions d'un module existant, il faut charger le module avec la commande import, ce qui peut se faire de différentes manières, avec :

A `from module import *`, on obtient l'accès direct à l'ensemble des fonctions du module indiqué sans devoir les préfixer par le nom du module

B `from module import fct1, fct2, ...`, on ne souhaite l'accès qu'aux fonctions fct1, fct2 spécifiées

C `import module1, module2, ...`, toutes les fonctions des modules spécifiés seront accessibles, mais seulement en les préfixant du nom du module

D `import module as nomLocal`, toutes les fonctions du module sont accessible en les préfixant du nomLocal que l'on a défini

Les méthodes (C) et (D) sont préconisées. La technique (A) ne devrait en principe pas être utilisée, car elle présente le risque d'écrasement d'objets si les différents modules chargés et/ou votre programme implémentent des objets de noms identiques. Elle rend en outre le code moins lisible (on ne voit pas d'où proviennent les fonctions utilisées).

```
# Admettons qu'on doive utiliser la constante 'pi' et la
# fonction 'sin', tous deux definis dans le module 'math'
# A ------------------------------------------------------------
from math import *
%who        # => on voit toutes fcts importees !
sin(pi/2)   # => 1.0

# B ------------------------------------------------------------
from math import pi, sin
%who        # %whos => seules 'pi', 'sin' accessibles
dir()       # => objets dans namespace, notamment ces fcts
sin(pi/2)   # => 1.0
cos(pi)     # => erreur "name 'cos' is not defined"

# C ------------------------------------------------------------
import math
%who               # => module 'math' importe
math.<tab>         # => liste les fonctions du module
help(math)         # => affiche l'aide sur ces fonctions
math.sin(math.pi/2) # => 1.0
cos(pi)            # => erreur (non prefixes par module)

# D ------------------------------------------------------------
import math as mt
%who                # => module math importe s/nom mt
mt.<tab>            # => liste les fonctions du module
mt.sin(mt.pi/2)     # => 1.0
math.sin(math.pi/2) # => erreur "name math not defined"
mt.cos(mt.pi)       # => -1.0
```

Écriture de modules

Créer un module revient simplement a créer un fichier nommé module.py dans lequel seront définies les différents objets (fonctions, classes...) du module. Le nom du module est en général défini en caractères minuscules seulement (a-z, avec éventuellement des _). Vous pouvez ensuite l'importer dans d'autres scripts.

```
# Contenu du fichier mon_module.py
def maFonction():
    return "Hello, World!"

# Utiliser le module dans un autre script
import mon_module
print(mon_module.maFonction())   # Affiche "Hello, World!"
```

Portée des variables dans les modules

Les variables et fonctions définies dans un module sont accessibles après l'importation du module.

```
# Fichier mon_module.py

def carre(nb):
    return nb*nb
def cube(nb):
    return nb*nb*nb

if ___name___ == "___main___":
    # module execute en tant que script
    print("Exemple de la fonction carre()")
    print("  carre(4) => ", carre(4))

# Utilisation du module
import mon_module
mon_module.cube(2)      # => 8

# Execution du module en tant que script
$ python mon_module.py    # => affiche ce qui suit :

Exemple de la fonction carre()
    carre(4) =>  16
Exemple de la fonction cube()
    cube(3) =>  27
```

3.4.3 Scripts

Un script (ou programme) est un fichier de code que l'on peut exécuter dans un interpréteur. En Python, son nom se termine en principe par l'extension .py.

On a déjà dit quelques mots sur (A) les deux lignes d'en-têtes d'un script (définition de l'interpréteur, encodage des caractères). Cette ligne, bien qu'utile dans la pratique, est, en revanche et pour des questions de lisibilité du code, optionels pour ce cours (!). Pour divers raisons, liées notamment à l'identification du code par l'enseignants, on mentionnera en revanche systématiquement l'auteur, le nom de l'exercice et la date de réalisation du script (B)[3]. Suivent (C) les instructions d'importation des modules utilisés (si nécessaire). Le script se poursuit par la définition (D) d'éventuelles fonctions. On trouve finalement (E) le corps principal du programme, entité souvent identifiée, dans la pratique, par l'interpréteur __main__[4].

On se rappellera que l'on peut exécuter monScript.py de différentes manières : depuis un shell (fenêtre de terminal / commande) en frappant, python script.py ou depuis un programme comme Thonny ou TigerJython.

```
# !/usr/bin/env python                                          # (A)
# Prenom et NOM de l'auteur :: Exercice monScript :: Date       # (B)

import sys                                                      # (C)

def pause(prompt):                                              # (D)
    print()
    input(prompt)

print("- nom de fichier du script :", sys.argv[0])              # (E)
print("- lancement avec", len(sys.argv)-1, \
        "arguments :", sys.argv[1:])

pause("Frapper <enter> pour terminer le programme")

# Exemple d'execution du script depuis un shell
$ python monScript.py un deux  # => affiche :
- nom de fichier du script : monScript.py
- lancement avec 2 arguments : ["un", "deux"]
```

3. /!\ Le nom du fichier sera alors soit
   — inclut dans un répertoire **PrenomNOM_TP00** et le fichier nommé **ex0_nomExercice_Date.py**
   — nommé **PrenomNOM_TP00ex0_nomExercice_Date.py**

Il est utile, pour éviter d'éventuels pertes de données, de prendre l'habitude de rendre les documents sous forme zippées. Ok compris, revenir aux exercices p. 120 ?

4. Nous ne demanderons pas dans les scripts, également pour des questions didactiques et pédagogiques, sauf exceptions, d'utiliser l'interpréteur __main__ comme défini à la page précédente.

3.5 Manipulation de fichiers

Exercices p. 146, aide-mémoire p. 177

3.5.1 Ouverture et fermeture de fichiers

Pour manipuler un fichier, il faut d'abord l'ouvrir, ce qui est exécuter avec la commande `open`, à savoir :
`fd = open(fichier, mode)`.

Le descripteur de fichiers ou `fd` retournra un objet – fichier, définit par le nom du fichier, avec son chemin d'accès (path) s'il n'est pas dans le répertoire du script - à moins de changer préalablement de répertoire courant avec `os.chdir(path)`.

Le mode d'accès ou `mode` est une chaîne définissant la manière d'accèder au fichier (voir après l'exemple) ; en l'absence de ce paramètre, fichier sera ouvert en mode `rt`.

```
fd = open("fich.txt", "w")
    # ouverture nouveau fichier en ecriture

print(fd)        # => affiche :
                      > name='fich.txt' mode='w' encoding='UTF-8'>
fd.closed        # => False
fd.name          # => 'fich.txt'
fd.mode          # => 'w'
fd.encoding      # => 'UTF-8'
fd.readable()    # => False (serait True si 'r')

# On doit le cas echeant gerer soi-meme les erreurs d'ouverture,
# p.ex. ici avec l'implementation de notre propre fonction 'error'

import sys

def error(msg): # fct d'affich. message & exit
    sys.stderr.write("Error: %s \n" % msg)
    sys.exit("Abandon...")

try:
    fd = open("inexistant.txt", "r")
except IOError as err:
    error(err.strerror)  # erreur => abandon
# ... suite du programme si OK

# Si fichier inexistant => cela affichera:
  Error: No such file or directory
  An exception has occurred, use %tb to
  see the full traceback.
  SystemExit: Aborting...
```

Les modes d'accès sont les suivants :

— `"r"` permet l'accès en lecture seule et une erreur s'affiche si le fichier n'existe pas

— `"w"` donne accès en écriture et, si le fichier n'existe pas, il est créé (/!\s'il existe, son contenu est écrasé)

— `"r+"` autorise l'accès en lecture et écriture, permettant ainsi de lire et intercaler des données dans le fichier

— `"a"` fournit un accès en mode ajout (append)
si le fichier n'existe pas, il est créé et s'il existe, son contenu est laissé intact, les écritures s'effectuent à la fin du fichier

La fermeture du fichier s'effectue logiquement avec `fd.close()`. En cas d'oubli, tous les fichiers ouverts sont automatiquement fermés à la sortie du programme. Pour des raisons pédagogiques et présentation, dans la théorie ci-dessous, nous continuerons d'utiliser les commandes `open` et `close`, même si, dans la pratique (y.c. dans l'aide mémoire), on préférera souvent utiliser soit `with open ... as fd:` qui fonctionne de la même manière qu'une boucle, et donc se prête bien au fichiers volumineux, fermant l'ouverture du fichier à la fin de celle-ci (voir solution 3b, point C du chapitre suivant), soit en passant, pour des fichiers peu volumineux, directement par des variables (voir point A du chapitre ci-après).

```
# ... suite de l'exemple ci-dessus :
fd.close()    # fermeture du fichier
fd.closed     # => True
```

3.5.2 Lecture de fichiers

Plusieurs méthodes peuvent être utilisées, selon que l'on souhaite lire le fichier :

— d'une traite

— par paquets d'un certain nombre de caractères

— ligne après ligne

Dans les exemples qui suivent, on utilisera le fichier essai.txt contenant les 3 lignes suivantes :

```
1ere ligne
2e ligne
fin
```

A Pour la lecture d'une traite (fichiers pas trop volumineux, moins de 5 Mb, soit moins de 50k lignes de texte ou 100k occurrences d'un champs d'une base de données), on dispose de deux méthodes :

— `read()` place le contenu sur une seule chaîne, y compris les retours à la ligne (\n)... on pourrait, dans ce cas, réaliser un découpage avec `chaîne.split("\n")`, mais on a dans ce cas meilleur temps d'utiliser la méthode suivante

— `readlines()` place le contenu sur une liste qui contiendra autant d'éléments que de lignes du fichier, chaque élément étant terminé par un retour à la ligne (\n)

```
# Ouverture
fd = open("essai.txt", "r")

# Lecture integrale sur une chaine
chaine = fd.read()
            # chaine = "1ere ligne \n 2e ligne \n fin \n"

fd.seek(0)  # repositionnement au debut fichier
            # on aurait pu aussi effectuer un close puis un open

# Seconde lecture integrale, sur une liste cette fois
liste = fd.readlines()
            # liste = ["1ere ligne \n", "2e ligne \n", "fin \n"]

# Fermeture
fd.close()

# Les trois instruction open, read/readline, close pourraient
# etre reunies en une seule:

chaine = open("essai.txt", "r").read()
liste  = open("essai.txt", "r").readlines()

# et c'est alors Python qui refermera les fichiers!
```

B Si le fichier est volumineux (plus de 5 Mb, soit 50k lignes de texte ou 100k occurrences d'un champs d'une base de données), la lecture d'une seule traite n'est pas appropriée (saturation mémoire), on peut alors utiliser la méthode `read(n)` de lecture par paquets de n caractères

```
fd = open("essai.txt", "r") # ouverture
chaine = ""
while True:
    paquet = fd.read(10)    # paquet de 10 caracteres
    if paquet:              # paquet non vide
        chaine += paquet
    else:                   # paquet vide => fin du fichier
        break
fd.close()                  # fermeture
                            # chaine = '1ere ligne \n 2e ligne \n
                            #           fin \n'
```

C Une lecture ligne par ligne peut s'avérer judicieuse si l'on veut analyser chaque ligne au fur et à mesure de la lecture, ou que le fichier est très volumineux (et qu'on ne veut / peut pas le charger en mémoire), plusieurs méthodes :

— `readline()` qui lit le fichier jusqu'à ce que la ligne soit vide, signe que l'on est alors arrivé à la fin du fichier (en cas de ligne vide dans le fichier, la variable ligne n'est

pas vide mais contient \n)

— `readlines()` vue au point A (et non pas `readline()` comme ci-dessus)

— Une simple itération du fichier, fournissant une ligne à chaque itération, qu'il est possible de compacter en s'appuyant sur l'instruction `with` dont le rôle est de définir un contexte au sortir duquel certaines actions sont automatiquement exécutées et qu'il n'est donc pas besoin de ferner explicitement

```
# Solution 1 ----------------------------------------------------
fd = open("essai.txt", "r")
while True:                 # boucle sans fin, sauf si break
    ligne = fd.readline()
    if not ligne: break     # sortie de boucle,
    print(ligne.rstrip())   # identique a if len(ligne)==0: break
fd.close()

# Solution 2, plus legere ---------------------------------------
fd = open("essai.txt", "r")
for ligne in fd.readlines():
    print(ligne.rstrip())
fd.close()

# Solution 3a, plus elegante -----------------------------------
fd = open("essai.txt", "r")
for ligne in fd:
    print(ligne.rstrip())
fd.close()

# Solution 3b, plus compacte -----------------------------------
with open("essai.txt", "r") as fd:
    for ligne in fd:
        print(ligne.rstrip())
```

3.5.3 Écriture ou ajout

En premier lieu il faut rappeler où débutera l'écriture après l'ouverture du fichier. Pour cela, il faut distinguer les différents modes d'accès d'écriture :

— `"w"`, débute au début du fichier

— `"a"`, débute à la fin du fichier

— `"r+"` qui s'effectue depuis la position courante suite à d'éventuelles opérations de lecture ou de positionnement

Pour garantir de bonnes performances, l'écriture est par défaut mise en mémoire, mémoire tampon qui est bien entendu entièrement vidée une fois que les données sont stockées dans le fichier et le fichier fermé avec l'instruction `fd.close()`. Les différentes méthodes d'écriture sont les suivantes :

A `write(chaîne)` écrit dans le fichier la chaîne donnée (n'envoie pas de retour à la ligne après la chaîne, donc on est appelé à écrire soi-même des \n là où l'on veut des retours à la ligne.

```
liste = ["1ere ligne", "2e ligne", "fin"]
      # elements non termines par \n
fd = open("ecrit.txt", "w")
for ligne in liste:
    fd.write(ligne + "\n")    # on ajoute les \n
fd.close()
```

B `writelines(sequence ou ensemble)` qui écrit dans le fichier une séquence (tuple ou liste) ou un ensemble (set ou frozenset) contenant exclusivement des chaînes (tout comme `write`, cette méthode n'envoie pas non plus de retours à la ligne)

```
liste = ["1ere ligne \n", "2e ligne \n", "fin \n"]
      # les elements doivent se terminer par \n
      # si l'on veut de retours a la ligne
fd = open("ecrit.txt", "w")
fd.writelines(liste)
fd.close()
```

C `print(...)` à qui l'on passe l'argument `file = fd` pour écrire dans un fichier (se comporte de façon standard, donc envoie par défaut un retour à la ligne après l'écriture, à moins d'ajouter le paramètre `end = ""` et terminer l'execution du script avec une chaîne vide).

```
liste = ["1ere ligne", "2e ligne", "fin"]
      # elements non termines par \n
fd = open("ecrit.txt", "w")
for ligne in liste:
    print(ligne, file=fd)
fd.close()
```

3.5.4 Autres fonctions utiles pour la manipulation de fichiers

Les méthodes de fichiers suivantes sont encore intéressantes à connaître :

- `tell()` : indique la position courante dans le fichier, exprimée en nombre de caractères depuis le début du fichier
- `seek(nb, depuis)` : déplace le curseur (position courante) dans le fichier, si depuis :
  - est ignoré ou vaut 0, on se déplace au nième caractère du fichier
  - vaut 1 : on se déplace de nb caractères par rapport à la position courante, en avant ou en arrière selon que ce nombre est positif ou négatif

— vaut 2 : on se déplace de nb caractères par rapport à la fin du fichier (nombre négatif)

```
fd = open("essai.txt", "r")
fd.tell()      # => 0 (position a l'ouverture)

fd.seek(6)     # aller au 6e car. depuis deb. fichier
fd.read(5)     # => lecture 'ligne'
fd.tell()      # => 11 (6 + 5)

fd.seek(0,2)   # aller a la fin du fichier
fd.tell()      # => 25 (nb de car. total fichier)

fd.seek(-4,2) # aller au 4e car avant la fin fichier
fd.read(3)     # => lecture 'fin'

fd.seek(0)     # "rewind" au debut du fichier
fd.tell()      # => 0 (position a l'ouverture et la fermeture)

fd.close()
```

3.6 Tableaux et matrices

Exercices p. 146, aide-mémoire p. 177

En Python, un tableau est une structure de données permettant de stocker plusieurs éléments. Un tableau à double entrée (ou matrice) est une liste de listes, où chaque liste interne correspond à une ligne.

Exemple

```
# Matrice 2x3
matrice = [[1, 2, 3], [4, 5, 6]]
```

Pour améliorer la lisibilité, on peut formater le code avec des retours à la ligne :

```
# Matrice 2x3 avec retours a la ligne
matrice = [
    [1, 2, 3],
    [4, 5, 6]
]
```

Autre exemple

Dans ce plan, le nombre '1' signifie qu'un siège est réservé et le '0' qu'il est libre :

```
# Plan des sieges d'un theatre
plan_sieges_theatre = [
    [0, 0, 1, 1, 1, 1, 1, 1, 0, 0],
    [0, 1, 1, 1, 1, 1, 1, 1, 1, 0],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
]
```

Ce format clair facilite la lecture et l'analyse du code.

3.6.1 Manipulation des éléments matriciels

Exemples

Afficher la première ligne de la matrice :

```
print(matrice[0])           # Resultat: [1, 2, 3]
```

Accéder à un élément spécifique :

```
element = matrice[1][2]
print(element)              # Resultat: 6
```

Modifier un élément :

```
matrice[0][1] = 10
print(matrice)
```

Afficher tous les éléments d'une matrice :

```
matrice = [
    [1, 10, 3],
    [4, 5, 6]
]
for ligne in matrice:        # Parcourt chaque ligne
    for element in ligne:    # Parcourt chaque element de la ligne
        print(element)
```

Ajouter une nouvelle ligne :

```
nouvelle_ligne = [10, 11, 12]
matrice.append(nouvelle_ligne)
print(matrice)
```

Résultat :

```
matrice = [
    [1, 10, 3],
    [4, 5, 6],
    [10, 11, 12]
]
```

3.6.2 Cas particulier des chaînes de caractères

Une chaîne de caractères fonctionne comme une liste où chaque lettre est un élément.

Exemple

```
ma_phrase = "Bonjour!"
print(ma_phrase[2])        # Affiche 'n'
```

On peut aussi manipuler une liste de phrases :

```
salutations = ["Bonjour!", "Tcho!", "Salut!"]
print(salutations[0][0])  # Premiere lettre de "Bonjour!" => "B"
print(salutations[1][0])  # Premiere lettre de "Tcho!"    => "T"
print(salutations[2][0])  # Premiere lettre de "Salut!"   => "S"
```

Chapitre 4

PRÉSENTATION DE MODULES PYTHON CLÉS

Pour chaque exemple, il est possible de
sélectionner le code source et l'insérer dans un éditeur pour être executé.

4.1 Matplotlib

Matplotlib est une bibliothèque standard pour la création de graphiques.

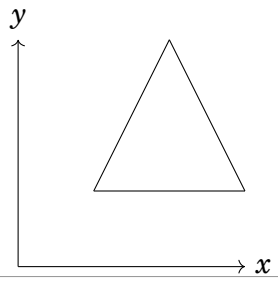
```
import matplotlib.pyplot as plt

plt.figure()      # Creation d'une figure

# Definition des limites pour les axes
plt.xlim(-3, 7)   # Limites pour l'axe des x
plt.ylim(-4, 6)   # Limites pour l'axe des y

plt.show()        # Affichage du graphique
```

Une fois une fenêtre ouverte, on peut alors dessiner librement à l'intérieur à l'aide de divers fonctions. La fonction `lines()` permet par exemple de dessiner des lignes et donc, par exemple, de créer un triangle.




```
import matplotlib.pyplot as plt
import matplotlib.lines as mlines

# Creer la premiere ligne
line1 = mlines.Line2D([1, 2], [1, 2])  # (1,1) a (2,2)
plt.gca().add_line(line1)

# Creer la deuxieme ligne
line2 = mlines.Line2D([2, 3], [2, 1])  # (2,2) a (3,1)
plt.gca().add_line(line2)

# Creer la troisieme ligne
line3 = mlines.Line2D([3, 1], [1, 1])  # (3,1) a (1,1)
plt.gca().add_line(line3)

plt.xlim(0, 4)
plt.ylim(0, 3)

plt.show()
```

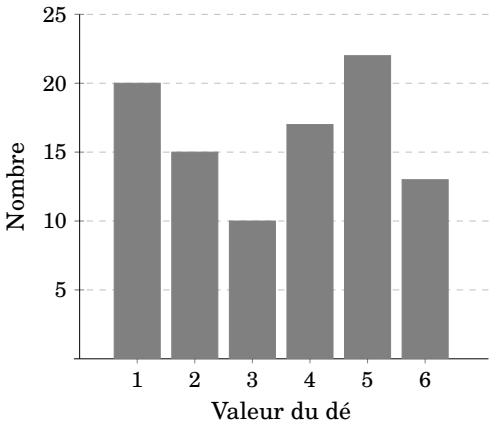
Ce genre de programme peut rapidement se compliquer, c'est pourquoi nous n'allons pas nous y attarder. Il est important de noter qu'il est souvent utilisé pour illustrer des événements. Nous l'utiliserons donc pour illustrer notre prochain module.

4.2 Random

Exercices p. 137 et 150

La chance joue un rôle important dans notre vie de tous les jours. On parle de "chance" ou de "hasard" pour désigner les événements qui ne sont pas prédictibles. Ce dernier joue un rôle important dans les jeux : si le dé n'est pas truqué, lorsque l'on jette un dé, le résultat que l'on obtient (entre 1 et 6) est complètement aléatoire. L'ordinateur se prête à merveille à la réalisation d'expériences stochastiques [1] car il permet d'exécuter, sans effort, un nombre important d'expériences.

La fonction `random()` retourne des nombres - flottants, qui ne sont donc pas des entiers - uniformément distribués entre 0 (inclus) et 1 (exclus). Pour l'utiliser, il est nécessaire d'importer le module random. La fonction randint(start, end) renvoie, elle, un nombre aléatoire entier compris entre start et end (les deux bornes sont incluses). Ci-contre, un exemple du résultat, pour le lancement de dés, de 100 essais.




```python
import matplotlib.pyplot as plt
from random import randint

ESSAI = 100
plt.title("# d'essai: " + str(ESSAI))
plt.xlabel("Valeur du de")       # Titre et labels de l'axe
plt.ylabel("Nombre")

# Valeurs aleatoires
essai = [randint(1, 6) for k in range(ESSAI)]

# Histogramme combinant les tirages et essais correspondant
values, counts = zip(*[(i, essai.count(i)) for i in range(1, 7)])

plt.bar(values, counts)          # Dessin

plt.xlim(0, 7)                   # Limite
plt.ylim(0, max(counts) * 1.1)

plt.show()                       # Afficher le graphique
```

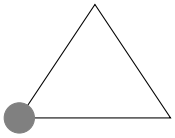
1. Qui se produit par l'effet du hasard.

4.3 Turtle

4.3.1 Déplacer une tortue

Ton premier programme Exercices p. 150, aide-mémoire p. 176

Cela peut devenir compliqué, n'est-ce pas ? Afin d'avoir une approche plus simple, voici le code (que tu peux toujours copier / coller) pour l'exécution d'un premier programme `turtle`, relativement facile à comprendre.



```
import turtle

jo = turtle         # ou jo = turtle.Turtle() pour etre plus rigoureux
pts, dist, cercle = 10, 100, 360

jo.dot(pts)         # instructions de deplacement, ici dessine un point
jo.forward(dist)    # ou jo.fd(100) pour avancer de 100 pixels
jo.left(cercle/3)   # ou jo.lt(120) pour tourner de 120deg vers la gauche
jo.forward(dist/2)  # ou jo.fd(50) pour avancer de 50 pixels
jo.right(cercle/4)  # ou jo.rt(90) pour tourner de 90deg vers la droite
```

Remarques : pour augmenter la vitesse de dessin, on peut utiliser la fonction `speed(0)` et que les fonctions `quit()` ou `done()` imposent de quitter le programme et libère ainsi la fenêtre de dessin.

Lignes et couleurs

La tortue dessine sa trace en utilisant un crayon de couleur qu'il est possible de personnaliser avec certaines instructions . Aussi longtemps que le crayon est posé, la tortue dessine. Il est possible d'interrompre ce comportement avec l'instruction `turtle.penup()` qui va lever le crayon et empêcher la tortue de dessiner. On reposera le crayon avec `pendown()` de sorte que la tortue se remette à dessiner la trace. L'option `turtle.dot()` permet par ailleurs de dessiner un rond d'une taille donnée.

Le programme suivant ordonne à la tortue de dessiner une bougie avec une ligne rouge très large. On règle la largeur de la trace avec l'instruction `turtle.width()`.

```
import turtle  # BOUGIE

t = turtle.Turtle()

t.left(90)      # Option: t.lt(90)
t.width(60)     # Largeur du trait
t.color("red") # Reglage de la couleur

t.forward(100) # Option: t.fd(90)
t.penup()      # Lever le stylo du dessin
```

```
# FLAMME
t.fd(50)
t.pendown()     # Baisser le stylo pour dessiner
t.color("yellow")
t.dot(40)

# MECHE
t.width(5)
t.color("black")
t.backward(15) # Reculer

# CACHER LA TORTUE
t.hideturtle()
```

4.3.2 Rappel de quelques concepts généraux

For... Exercices p. 150, aide-mémoire p. 175

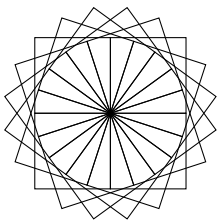
Pour dessiner un carré, la tortue avance tout droit, puis tourne de 90 degrés (360/4). Si l'on veut dessiner plusieurs carrés un tel programme deviendrait rapidement long... très long ! À l'aide de l'instruction `for...` on demande à la tortue de répéter une série d'instructions indentées. Attention de ne pas oublier les deux points : après la structure.

```
import turtle

# CARRE
for k in range(4):
    turtle.fd(100)    # Avancer de 100 pixels
    turtle.lt(360/4) # 90 degres a gauche
```

Séquences imbriquées

Maintenant, on va essayer de dessiner vingt carrés en les tournant légèrement les uns par rapport aux autres. Pour ce faire, il faut imbriquer deux instructions `for...` l'une dans l'autre. Dans le bloc d'instructions interne, la tortue dessine un carré et tourne ensuite de 360/20 degrés (soit 18°) vers la droite. L'autre instruction `for...` permet de répéter ce comportement n fois (ici 20). Faire bien attention d'indenter correctement les différents blocs d'instructions, sans quoi la tortue risque de faire n'importe quoi...



```
import turtle

# 20 REPETITIONS
for k in range(20):
```

```
    # CARRE
    for k in range(4):
        turtle.fd(80)      # Avancer de 80 pixels
        turtle.lt(360/4)   # 90 degres a gauche
    turtle.rt(360/20)      # 18 degres de rotation a droite apres avoir
                           # dessine le carre
```

Fonctions et paramètres Exercices p. 151, ainde-mémoire p. 176

Quand on souhaite dessiner des images avec plusieurs formes comme des carrés ou des triangles, on se heurte rapidement à un petit problème : la tortue ne sait dessiner que des lignes droites. Si on veut qu'elle dessine des formes plus complexes, on doit lui donner de nombreuses instructions, ce qui peut rendre le programme long et répétitif...

On va alors créer des "raccourcis", en apprenant à la tortue comment dessiner un carré ou un triangle. Pour cela, on définit des fonctions, ici nomée `square()`. On prendra bien soin d'indenter les instructions. L'option `turtle.pencolor()` permet de changer la couleur du crayon.

```
import turtle

def square():
    for k in range(4):
        turtle.fd(100)
        turtle.lt(360/4)

turtle.speed(1)         # Vitesse de la tortue

turtle.pencolor("red") # Carre rouge
square()

turtle.rt(120)          # Carre bleu
turtle.pencolor("blue")
square()
```

Quand on utilise la commande `forward()`, la valeur qu'on met entre les parenthèses est ce qu'on appelle un paramètre.

Pour rendre une commande plus flexible, on peut lui préciser un paramètre en écrivant quelque chose comme `def square(longueur):`. Quand on utilise `square(80)`, la tortue dessinera un carré de 80 pixels de côté.

```
import turtle

def square(longueur):
    for k in range(4):
        turtle.fd(longueur)
        turtle.lt(360/4)
```

```
turtle.color("red")
square(80)

turtle.lt(180)

turtle.color("green")
square(50)
```

On peut même ajouter plus de paramètre à la commande. Par exemple, on pourrait écrire `def square(longueur, couleur)` pour ajouter un paramètre de couleur... ainsi, `square(100, "red")` dessinerait un carré rouge de 100 pixels de côté.

```
import turtle

def square(longueur, couleur):
    turtle.color(couleur)
    for k in range(4):
        turtle.fd(longueur)
        turtle.lt(360/4)

square(100, "red")
```

Variables

Pour rendre notre programme plus interactif et capable de dessiner des carrés de différentes tailles, on peut demander à l'utilisateur de choisir la taille du carré. Pour cela, passera par la définition de `variables`.

Il est important de comprendre la différence entre `variable` et `paramètre`. Les `paramètres` sont utilisés pour donner des informations à une `fonction` et ne fonctionnent que dans cette `fonction`. Les `variables`, en revanche, peuvent être utilisées partout dans le programme.

```
import turtle

def square(longueur):
    for k in range(4):
        turtle.fd(longueur) # Utilisation du parametre
        turtle.rt(360/4)

cote = int(input("Longueur du cote: "))
square(cote)                # Variable saisie par l'utilisateur
```

On utilise ici `input()` pour saisir un nombre et `int` pour le traduire en nombre, nombre qui est stocké dans une variable nommée `cote`. Quand on appelle la fonction `square()`, on utilise la variable `cote` pour passer sa valeur au paramètre `longueur` de la fonction.

Boucles et conditions Exercices p. 152, aide-mémoire p. 175

Quand vous prenez des décisions dans la vie quotidienne, elles dépendent souvent de certaines conditions. Par exemple, comment vous allez à l'école peut dépendre du temps qu'il fait. En programmation, on utilise aussi des conditions pour décider quelles instructions le programme doit exécuter. Si une condition est vraie, le programme suit un chemin, sinon, il en prend un autre.

Si vous entrez la taille d'un carré dans un programme, ce dernier ne devrait dessiner le carré que si la taille est raisonnable. Si la taille est trop grande, il affichera un message d'erreur. En programmation, on utilise la structure `if` pour vérifier ce genre de conditions.

```
import turtle

def square(longueur):
    for k in range(4):
        turtle.fd(longueur)
        turtle.rt(90)

cote = int(input("Longueur du cote: "))
if cote < 300:
    square(25)   # Longueur saisie par l'utilisateur
else:
    print("Longueur trop importante")
```

Imaginez que vous voulez dessiner des carrés de différentes couleurs en utilisant la commande `turtle.fillcolor()`. En utilisant une structure `if`, le programme peut choisir la couleur en fonction d'un nombre que vous entrez. Si vous tapez 1, il dessine en rouge ; si c'est 2, en vert ; et ainsi de suite.

```
def square():
    for k in range(4):
        turtle.forward(100)
        turtle.right(90)

n = int(input("Saisir une valeur: 1:rouge 2:vert "))

if n == 1:
    turtle.fillcolor("red")
elif n == 2:
    turtle.fillcolor("green")
else:
    turtle.fillcolor("black")

turtle.begin_fill()
square()
turtle.end_fill()
```

Nous avons déjà utilisé la commande `for` à de nombreuses reprises pour répéter certaines

actions. Cependant, en Python, on utilise plus souvent la boucle `while` pour répéter des instructions tant qu'une condition donnée est vraie. Vous pouvez par exemple dessiner une spirale rectangulaire avec une boucle `while` en augmentant progressivement la taille de la spirale à chaque tour de la boucle de la manière suivante.

```
import turtle

a = 5
while a < 200:
    turtle.fd(a)
    turtle.rt(360/4)
    a = a + 2
```

En dessinant des figures, vous pouvez vouloir changer de couleur ou de forme en fonction de certaines conditions. Par exemple, vous pouvez alterner les couleurs ou choisir la taille d'une tranche en fonction des sa position.

```
import turtle

def triangle():
    for k in range(3):              # Dessiner un triangle
        turtle.fd(100)              # Avancer de 100
        turtle.rt(360/3)            # Tourner de 120

turtle.speed(1)                     # Vitesse de la tortue
i = 0
while i < 6:
    if i in [0, 2, 4]:              # Changer la couleur
        turtle.pencolor("red")      # Rouge
        turtle.fillcolor("red")
    else:
        turtle.pencolor("green")    # Vert
        turtle.fillcolor("green")

    turtle.begin_fill()             # Remplir la forme
    triangle()
    turtle.end_fill()
    turtle.rt(60)                   # Tourner de 60
    i = i + 1
```

Parfois, vous devez pouvoir arrêter une boucle même si sa condition est toujours vraie. Vous pouvez utiliser pour cela le mot-clé `break`.

```
import turtle

def square(longueur):
    for k in range(4):          # Carre
        turtle.fd(longueur)     # Avancer de la longueur du cote
        turtle.lt(90)           # Tourner de 90
```

```
turtle.speed(0)                 # Vitesse de la tortue
turtle.hideturtle()             # Cacher la tortue

i = 0
while True:
    if i > 120:
        break
    square(i)                   # Dessine un carre de cote i
    turtle.rt(360/60)           # Tourner de 6
    i += 2

print("i =", i)                 # Afficher valeur finale i
```

Afin d'évier les problèmes, il est préférable que, quand un programme demande de saisir une grandeur, il vérifie la saisie. Un bon programme répète la question jusqu'à ce qu'il obtienne une réponse acceptable...

```
import turtle

n = 0
while n < 1 or n > 3:
    n = int(input("Saisir une valeur entre 1 et 3: "))

if n == 1:
    turtle.pencolor("red")
elif n == 2:
    turtle.pencolor("green")
else:
    turtle.pencolor("yellow")

turtle.dot(200) # Affiche un point
```

4.3.3 Références récursives

Exercices p. 153

En programmation, on utilise souvent un concept appelé la récursivité, propriété de ce qui peut se répéter indéfiniment. On utilise ce principe qui fonctionne comme une boucle, lorsque une fonction s'appelle elle-même, exactement comme les poupées russes, où chaque poupée en contient une autre, plus petite, qui en contient une autre plus petite, et ainsi de suite.
Mais les choses ont une fin ! Dans l'exemple ci-dessus, pour dessiner un escalier de trois marches, on passera par une fonction `escalier` qui appelle la fonction `marche` de manière récursive...

```
def escalier(n):
    marche()
    escalier(n-1)
```

Et on terminera la boucle en indiquant qu'il n'est plus utile de construire des marches (que le nombre de marche est nul) à l'aide de la condition suivante :

```
if n == 0:
   return # similaire a quit()
```

Mis tout ensemble, cela donne :

```
import turtle

d, a = 50, 90       # d pour distance, a pour angle

def marche():
    turtle.fd(d)
    turtle.rt(a)
    turtle.fd(d)
    turtle.lt(a)

def escalier(n):
    if n == 0:      # Condition de sortie de la boucle
        return
    marche()
    escalier(n - 1) # Fonction recursive

escalier(3)

turtle.done()
```

4.3.4 Introduction à la programmation objet avec Turtle Exercices p. 153

Dans la nature, une tortue est un individu ayant sa propre identité. On pourrait d'ailleurs donner à chaque tortue un nom, par exemple, Quick et Flupke. On remarquera aussi que les tortues ont des caractéristiques communes : elles ont des carapaces et appartiennent à la classe animale des reptiles. Cette notion de classes et d'individus est une notion fondamentale en informatique, que l'on appelle programmation orientée objets (POO).
Lorsque on écrit `quick = turtle.Turtle()`, on crée une tortue nommée Quick. Si vous écrivez `flupke = turtle.Turtle()`, vous créez une autre tortue appelée Flupke. Vous aurez, ce qu'on appelle, en programmation, deux acteurs ou occurences de l'objet `Turtle`.
Les objets (acteurs ou occurences) similaires sont groupés par classes. Les objets d'une certaine classe sont fabriqués (on dit aussi instanciés) en utilisant le nom de la classe (suivi de parenthèses). Les fonctions rattachées à une certaine classe sont appelées des méthodes.
Illustrons cela par un exemple :

```
import turtle

quick = turtle.Turtle()  # Instance quick de la classe d'objet Turtle()
quick.color("red")       # Couleur de la tortue
quick.pencolor("green")  # Couleur du trait
quick.setpos(0, -200)    # Position initiale
```

```
d, b, n = 50, 4, 7        # d: distance, b: barreau, n: niveau

# DESSIN D'UNE ECHELLE
for _ in range(n):
    for _ in range(b):
        quick.forward(d)
        quick.right(360/b)
    quick.forward(d)

turtle.done()
```

Pour que les deux tortues soient ensemble, on va créer un espace commun, une fenêtre (en anglais `Screen`) appelée `enclos`. Dans le programme suivant, Quick dessine une échelle bleue verticale, et Flupke dessine une échelle noire horizontale.

```
import turtle

# VARIABLES
enclos = turtle.Screen()     # Instance de la classe Screen()
                             # pour avoir la fenetre de dessin
quick, flupke = turtle.Turtle(), turtle.Turtle()
qc, fc = "blue", "black"
d, b, n, p = 50, 4, 7, -200 # p: position

quick.color(qc), flupke.color(fc)
quick.pencolor(qc), flupke.pencolor(fc)
quick.setpos(p, 0), flupke.setpos(0, p)

# DESSIN DE DEUX ECHELLES
flupke.left(360/b)           # Oriente Flupke vers le haut
for _ in range(n):
    for _ in range(b):
        quick.fd(d)
        quick.rt(360/b)
        flupke.fd(d)
        flupke.lt(360/b)
    quick.fd(d)
    flupke.fd(d)

enclos.mainloop()            # Garde la fenetre ouverte
```

Pour simplifier le code, on pourrait utiliser une seule et même fonction, en lui passant le nom de l'objet en paramètre. Exemple : si on crée une fonction barre(o), on peut dire quelle tortue (par exemple, Quick ou Flupke) doit suivre les instructions pour dessiner un barreau d'échelle. En appelant barre(quick) puis barre(flupke), on fait bouger d'abord Quick, puis Flupke, de sorte qu'il n'y a pas de collisions (les mouvements sont supperposés).

```
import turtle
```

```
# FONCTION
def barre(o):
    for _ in range(b):
        o.fd(d)
        o.rt(360/b)
    o.fd(d)

# VARIABLES
enclos = turtle.Screen()
quick, flupke = turtle.Turtle(), turtle.Turtle()
qc = "green"
d, b, n, p = 50, 4, 7, -200

quick.pencolor(qc)
quick.setpos(p, 0), flupke.setpos(0, p)

# DESSIN DE DEUX ECHELLES
flupke.left(360/b)
for _ in range(n):
    barre(quick)
    barre(flupke)

enclos.mainloop()
```

4.4 Pygame

Développer un jeu vidéo sans utiliser la programmation orientée objet est un vrai défi, car les éléments du jeu et tous les objets qui interagissent dans leur environnement sont en fait des objets qui travaillent ensemble.
La première étape consiste à créer la fentre de jeu. Par exemple, nous pouvons utiliser une bibliothèque appelée "pygame"...

```
import pygame, time

pygame.init()

l, h = 600, 400
fenetre = pygame.display.set_mode((l, h))

time.sleep(10) # Pause de 10 sec.

pygame.quit()
```

4.4.1 Jeux et programmation orientée objet

En programmation, quand on crée un nouvel objet, on peut décider si on créé des fonctions ex nihilo ou si on se base sur une classe d'obets qui existe déjà. Si elle est basée sur une classe existante, elle hérite alors de tout ce que cette classe peut faire – que ce soit ses propriétés (caractéristiques) ou méthodes (actions).
Prenons l'exemple ci-dessous. Les objets dans ce jeu (des balles ou ballons) sont ce qu'on appelle des acteurs, définit par la classe `objetMouvement`.

```
import pygame
pygame.init()

l, h, enCours = 600, 400, True

fenetre = pygame.display.set_mode((l, h))

# ------------- class objet en deplacment ------------
class objetMouvement:
    def __init__(self, x, y, rayon, vitesse):
        self.x = x
        self.y = y
        self.rayon = rayon
        self.vitesse = vitesse

    def dessiner(self, surface):
        pygame.draw.circle(surface, "green", (self.x, self.y), self.rayon)

    def bouger(self):
```

```
        self.y += self.vitesse

# Creation d'instances d'objets en mouvement
balle = objetMouvement(l/2, 0, 10, 0.1)
ballon = objetMouvement(l/3, 0, 20, 0.2)

while enCours:
    for action in pygame.event.get():
        if action.type == pygame.QUIT:
            enCours = False

    fenetre.fill("gray")

    balle.dessiner(fenetre)
    ballon.dessiner(fenetre)
    balle.bouger()
    ballon.bouger()

    pygame.display.flip()

pygame.quit()
```

À noter, qu'afin de garantir une fermeture propre, il faut une boucle qui teste à chaque itération si la valeur booléenne enCours est toujours valable, pour, effectivement, ne pas avoir un jeu qui continue indéfiniment à fonctionner alors que la fenêtre est fermée.

4.4.2 Classes et objets

Dans la classe décrite ci-dessus, on a défini trois méthodes, qui sont des fonctions particulières. La première, sert à créer un objet, souvent appelé `__init__(self)`. On crée ensuite deux fonctions (dessiner et bouger) qui définissent comment l'objet se comportera.
Pour utiliser notre objet dans un jeu, on crée l'objet que l'on appelle ici soit "balle" soit "ballon". La création de ces objets peut également être "délégée", comme par exemple, ci-dessous, où un nouveau ballon est créé de manière aléatoire dans le jeu.

```
import pygame, random

pygame.init()

l, h, enCours = 600, 400, True
fenetre = pygame.display.set_mode((l, h))

# ------------- class objet en deplacement ------------
class objetMouvement:
    def __init__(self, x, y, rayon, vitesse):
        self.x = x
        self.y = y
        self.rayon = rayon
```

```
        self.vitesse = vitesse

    def dessiner(self, surface):
        pygame.draw.circle(surface, "green", (self.x, self.y), self.rayon)

    def bouger(self):
        self.y += self.vitesse

ballons = []                                    # Liste
AJOUT_BALLON = pygame.USEREVENT + 1
pygame.time.set_timer(AJOUT_BALLON, 1000)   # 1000ms == 1sec.

while enCours:
    for action in pygame.event.get():
        if action.type == pygame.QUIT:
            enCours = False
        elif action.type == AJOUT_BALLON:
            nouveauBallon = objetMouvement(random.randint(0, 1), 0, 10, 0.1)
            ballons.append(nouveauBallon)

    fenetre.fill("gray")

    for ballon in ballons:
        ballon.dessiner(fenetre)
        ballon.bouger()

    pygame.display.flip()

pygame.quit()
```

4.4.3 Interactions

Le but est d'empêcher que les ballons touchent le sol en cliquant dessus avec la souris. Si un ballon atteint le sol, le joueur perd alors un point.
Pour utiliser la souris dans le jeu, nous devons ajouter une fonction supplémentaire qui vérifie où l'on clique. Si on clique sur un ballon, il sera alors identifié et enlevé du jeu. Si on clique sur un endroit vide, il ne se passe rien de particulier.

```
import pygame, random

pygame.init()

l, h, enCours = 600, 400, True
fenetre = pygame.display.set_mode((l, h))

# ------------- class objet en deplacement ------------
class objetMouvement:
    def __init__(self, x, y, rayon, vitesse):
```

```
        self.x = x
        self.y = y
        self.rayon = rayon
        self.vitesse = vitesse

    def dessiner(self, surface):
        pygame.draw.circle(surface, "green", (self.x, self.y), self.rayon)

    def bouger(self):
        self.y += self.vitesse

    def est_clique(self, pos):
                                            # Verifie si le ballon est clique
        distance = ((self.x - pos[0]) ** 2 + (self.y - pos[1]) ** 2) ** 0.5
        return distance < self.rayon

ballons = []                                # Liste pour stocker les ballons
AJOUT_BALLON = pygame.USEREVENT + 1
pygame.time.set_timer(AJOUT_BALLON, 1000)   # Ajout d'un ballon toutes les sec.

while enCours:
    for action in pygame.event.get():
        if action.type == pygame.QUIT:
            enCours = False
        elif action.type == pygame.MOUSEBUTTONDOWN:
            pos = pygame.mouse.get_pos()
            for ballon in ballons[:]:
                if ballon.est_clique(pos): # Verifie si un ballon est clique
                    ballons.remove(ballon)
        elif action.type == AJOUT_BALLON:
            nouveauBallon = objetMouvement(random.randint(0, 1), 0, 10, 0.1)
            ballons.append(nouveauBallon)

    fenetre.fill("gray")

    for ballon in ballons:
        ballon.dessiner(fenetre)
        ballon.bouger()

    pygame.display.flip()

pygame.quit()
```

Chapitre 5

DÉVELOPPEMENT D'APPLICATIONS : EXEMPLES WEB

Nous aborderons ici les concepts fondamentaux tels que les métadonnées, les langages HTML et CSS, la planification et la création d'un site web simple, la gestion des fichiers, ainsi que les différentes méthodes pour publier votre site en ligne. Nous terminerons par une introduction aux systèmes de gestion de contenu (CMS).

5.1 Métadonnées

Aide-mémoire p. 178, exercices p. 155

Une métadonnée est une information qui décrit une autre donnée, facilitant ainsi sa compréhension, son organisation et sa gestion. Les métadonnées sont omniprésentes dans le monde numérique, que ce soit sur vos appareils personnels ou en ligne. Elles sont importantes pour diverses raisons :

- Elles permettent de classer et de retrouver facilement des fichiers.
- Elles fournissent des informations sur l'intégrité des données.
- Sur le web, les métadonnées aident les moteurs de recherche à comprendre le contenu des pages, améliorant ainsi le référencement.

5.1.1 Exemples de Métadonnées

Fichier informatique

- Date et heure de création
- Taille
- Extension (exemple .docx ou .html)
- Propriétaire du fichier

Photo

- Lieu où la photo a été prise (coordonnées GPS)
- Modèle de l'appareil photo
- Date et heure de la prise de vue

5.2 Utilisation du markdown pour produire un site web

5.2.1 De quoi s'agit-il ?

Exercices p. 156

Markdown est un langage de balisage léger inventé en 2004 par John Gruber (blogueur et podcasteur américain) avec pour objectif de permettre d'écrire des documents formatés (titres, listes, tableaux, images, code, etc.) avec une syntaxe **simple et lisible**, proche du texte brut. Même sans conversion (exportation), le texte reste compréhensible. Il est notamment reconnu dans GitHub, Moodle, les plateformes de blogs. Le site informaticLi permet également d'utiliser ce language. Un fois écrit, il est très facile, avec le même fichier, de pubier du contenu (site web, document pdf ou wiki). Quelques minutes suffisent pour produire un premier document structuré. Un fichier Markdown possède habituellement l'extension `.md`.

Exemple de code Markdown (iLi)

```
# Mon premier document
\
Bonjour, ceci est un texte en **gras** et en *italique*.
Voici une liste :
- élément A
- élément B
```

Principales commandes

Mise en forme du texte

Les commandes suivantes permettent de mettre en valeur certaines parties du texte : `**gras**` en **gras**, `*italique*` en *italique*, `_souligné_` en souligné et `` `code` `` (accents graves) en `code`. Pour ajouter un retour à la ligne, il suffit de taper `\`.

Titres et sous-titres

Utiliser le caractère # suivi d'un espace permet de créer des titres. Exemples : `# Titre 1` pour les titres et `## Titre 2` pour des sous-titres.

Listes

Les listes sont simplement créés grâce aux tirets −, `*` ou +, précédés éventuellement d'une tabulation (cas des listes imbriquées). Pour les listes numérotées, il suffit de commencer par un nombre, puis de continuer avec des 0.

```
1. Abricot                  1. Abricot
0. Pomme                    2. Pomme
    - golden      donnera       o golden
0. Raisin                   3. Raisin
```

Tableaux
On peut facilement organiser des données sous forme de tableau. Par exemple :

```
| Colonne A | Colonne B |
|-----------|-----------|
| a         | b         |
```

Bloc de code On affiche le code grâce aux trois accents graves (` ``). Tant que les trois accents graves ne sont pas remis, Markdown considère que le texte est un exemple de code.

Liens et images
Insérer un lien : `[Lien](http://eduge.ch)` représente un lien vers EDUGe
Insérer une image : `![Carré gris](https://placehold.co/100)`

Commentaires
Et enfin, pour ajouter un commentaire invisible (non affiché à l'impression), on utilisera soit le code html `<!-- Commentaire invisible -->`, ou, sur une ligne entière `[Commentaire non visible]::`.

5.3 Introduction au HTML et CSS

5.3.1 HTML

Aide-mémoire p. 178, exercices p. 159

HTML (HyperText Markup Language) est le langage standard utilisé pour créer et structurer le contenu des pages web. Il utilise des balises pour définir différents types de contenu tels que les paragraphes, les titres, les images, les liens, etc.

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, initial-scale=1.0">
    <title>SITE</title>
</head>
<body>
    <h1>Bienvenue</h1>
    <p>Ceci est le premier paragraphe.</p>
</body>
</html>
```

CODE 5.1 – HTML Structure de base

Composants clés

— `<!DOCTYPE html>` Indique au navigateur que le document est écrit en HTML.

— `<html>` Élément racine qui englobe tout le contenu de la page.

— `<head>` Contient des informations méta (métadonnées) sur la page.

— `<body>` Contient le contenu visible de la page.

Structurer le contenu iLi

```html
<header>
    <nav><!-- Liens de navigation --></nav>
</header>

<main>
    <h1 id="titre">Bienvenue</h1>
    <p>Ceci est un paragraphe.</p>
    <img src="https://placehold.co/100" alt="Carre gris">
</main>

<footer>
    <small>&copy; - Prenom</small>
</footer>
```

CODE 5.2 – Structurer les contenus entre les balises `<body></body>`

Explication Aide-mémoire p. 178, exercices p. 160

`<header></header>` zone d'en-tête du site, généralement placée en haut de la page. Contient habituellement le logo, un titre et un menu de navigation.

`<nav></nav>` section réservée aux liens de navigation (*menu*) permettant de se déplacer entre les différentes pages du site.

`<main></main>` contenu principal de la page (un seul `<main>` par page). Cela permet de différencier cette partie des en-têtes et pieds de page.

`<h1 id="titre"></h1>` le titre principal de la page. L'attribut `id="titre"` permet d'identifier le titre afin d'éventuellement créer un lien ou appliquer un style.

`<p></p>` paragraphe de texte.

`<img src="...">` insertion d'une image. `src` indique l'adresse de l'image et `alt` affiche un texte alternatif si l'image ne peut pas être chargée. Exemple : `alt="Carré gris"`.

`<footer></footer>` pied de page du site. Contient généralement les informations de copyright, les crédits ou des liens utiles.

`<small></small>` permet d'afficher un texte en plus petit.

5.3.2 CSS

Aide-mémoire p. 179, exercices p. 162

CSS (Cascading Style Sheets) est un langage de feuille de style utilisé pour décrire la présentation d'un document HTML. Il permet de contrôler l'apparence visuelle des pages web, notamment les couleurs, les polices, les espacements, les dispositions et bien plus encore.

```css
body { font-family: Arial, sans-serif; line-height: 1.5; }
h1   { color: gray; text-align: center; }
p    { color: #666666; line-height: 1.5; }
```

CODE 5.3 – Exemples de codes CSS

Intégration du CSS dans le HTML

— **Feuille de style externe** Recommandé pour séparer le contenu de la présentation

```html
<link rel="stylesheet" href="styles/style.css">
```

— **Style interne** Inséré dans la section `<head>`

```html
<style>
    /* Styles CSS */
</style>
```

— **Style en ligne** Directement dans les balises

```html
<p style="color: blue;">Texte en bleu</p>
```

Avantages de l'utilisation du CSS

— Facilite la maintenance et la cohérence du site

— Un seul fichier CSS peut être utilisé pour styliser plusieurs pages

— Permet des designs complexes et responsives

Exemple complet

5.4 Planification et conception de votre premier site

Avant de commencer à coder, il est essentiel de bien planifier et concevoir votre site web. Cette étape vous aidera à définir clairement vos objectifs et à structurer votre contenu de manière efficace.

5.4.1 Définir l'objectif

Posez-vous les questions suivantes :

— **Quel est le but principal de mon site ?** Portfolio, informations, vente, autre ?

— **Qui est mon public cible ?** B2B, B2C, autre ?

— **Quelles informations fournir ?** Textes, images, vidéos, liens, autre ?

5.4.2 Esquisser la structure

Créez une maquette du site en utilisant papier / crayon ou des outils numériques type PAO. Pensez à inclure...

— **Organisation** Page d'accueil, pages secondaires, contact, etc.

— **Disposition** En-tête, navigation, contenu principal, pied de page

— **Charte graphique** Couleurs, polices, images cohérentes

Mais aussi à :

— Choisir des images de qualité, optimisées pour le web (taille et format)

— Sélectionner des polices (Google Fonts) et des contrastes appropriés

— Respecter les droits d'auteur en utilisant des ressources libres ou en créant les votres

5.5 Structure de fichiers

Organiser correctement les fichiers de son site est essentiel pour assurer un développement simple et cohérent.

5.5.1 Arborescence

```
/projet_web
  /site
   index.html
    /images
    /css
    /js
```

5.5.2 Chemins vers les fichiers et les images

Pour que vos fichiers HTML puissent accéder aux ressources (images, feuilles de style, scripts), vous devrez utiliser les bonnes références, à savoir...

— **Chemin relatif** Pour référencer des fichiers situés dans le même projet, exemple :

```
<img src="images/nom_image.jpg">
```

CODE 5.4 – Chemins relatifs

— **Chemin absolu** Pour référencer des fichiers situés en dehors du site, exemple :

```
<img src="http://site-externe.ch/images/nom_image.jpg">
```

CODE 5.5 – Chemins absolus

5.6 Publier le site

Une fois le site web réalisé et testé en local, il est temps de le mettre en ligne pour qu'il soit accessible à tous.

5.6.1 Trouver un hébergeur et un nom de domaine

Il existe de nombreux fournisseurs pour l'hébergement : Infomaniak (à Genève), Hostpoint (à Zuerich), 000webhost (aux États-Unis), etc. Pour mémoire, le nom de domaine est une adresse unique (monsite.ch) louée pour faciliter la recherche d'un site.

5.6.2 Transfert FTP

Pour transférer vos fichiers vers le serveur, vous aurez besoin d'un client FTP comme FileZilla ou Cyberduck. Certains explorateurs de fichiers permettent également une connexion directe par FTP. Attention de toujours tester les modifications apportées pour s'assurer de leur bon fonctionnement.

5.7 Systèmes de gestion de contenu

Le CMS (Content Management System ou gestion de contenu en français) le plus utilisé dans le monde est sans nul doute WordPress : il permet de créer et gérer des sites web sans avoir à coder. Disponible en deux versions :

— **.org** Open-source, qu'on peut télécharger et installer sur un serveur compatible (idéal pour les utilisateurs avancés)

— **.com** Version hébergée, offrant une solution clé en main pour créer un site

La version hébergée propose des options gratuites et payantes pour personnaliser son site... cette contrainte rend malheureusement difficile son utilisation dans un environnement comme le notre !

5.7.1 Utilisation d'Odoo

Exercice p. 164

Pourquoi ? Bien que moins répendue, cette solution est plus complète que WordPress. Son utilisation s'avère également bien plus simple : elle permet, de manière centralisée, de gérer site web **et** gestion de l'organisation. Nous l'utiliserons comme base pour les ateliers CMS.
Note : Odoo est ce qu'on appelle un **ERP** (*Enterprise Resource Planning*). Il est open source.

Chapitre 6

BASES DE DONNÉES

Exercices p. 167, aide-mémoire p. 178

Une base de données permet de structurer et de stocker les données de manière organisée, évitant les redondances et les erreurs que l'on trouve souvent dans des fichiers comme les tableurs Excel. Un système de gestion de base de données (SGBD) permet de manipuler ces données efficacement et en toute sécurité. Il existe plusieurs types de bases de données, chacune optimisée pour des cas spécifiques.

Bases de données relationnelles
Elles structurent les données en tables (relations) et utilisent SQL pour les manipuler. C'est le modèle le plus largement utilisé pour la gestion de données structurées. Les plus connues sont MySQL, PostgreSQL, ou SQLite.

Bases de données déductives
Basées sur les bases relationnelles, elles permettent d'inférer de nouvelles informations à partir des données existantes à l'aide de règles logiques. Elles sont utilisées dans les systèmes experts et certaines applications d'intelligence artificielle.

Bases de données orientées objets
Elles stockent les données sous forme d'objets avec leurs méthodes, en suivant les principes de la programmation orientée objets. Elles sont adaptées pour des applications comme les systèmes de gestion de contenu ou les simulations.

Bases de données NoSQL
Conçues pour gérer des données non structurées ou semi-structurées, ces bases offrent une grande flexibilité et sont adaptées aux besoins des applications modernes. Pour la gestion documentaire, on mentionnera MongoDB.

Bases de données multimodales
Elles intègrent plusieurs modèles de données (relationnel, document, graphe) dans une seule base, offrant une flexibilité maximale pour les applications complexes. Elles peuvent être utilisées dans des systèmes qui interagissent avec des technologies comme les blockchains, notamment pour la gestion de données associées aux transactions de cryptomonnaies.

6.1 Modélisation

Pour numériser une réalité, il est nécessaire de la modéliser (par exemple, sous la forme d'un schéma du système étudié). Ce processus se déroule généralement étape par étape, en représentant d'abord les objets qui composent le système (les entités), puis les relations existantes entre ces objets. La modélisation d'une base de données se fait en procédant par niveaux (nous nous concentrerons ici uniquement sur les systèmes de gestion de bases de données relationnelles SGBDR) : du niveau conceptuel au niveau physique, en passant par le niveau logique.

6.1.1 Modélisation conceptuelle des données (MCD)

La création d'un MCD est la première étape dans la conception d'une base de données. Elle permet de définir le contenu de la base en identifiant les entités principales, leurs attributs, et les relations qui les lient. Une bonne modélisation aide à éviter les redondances, assure la cohérence des données, et facilite leur mise à jour.

Comment construire un schéma conceptuel La construction d'un schéma conceptuel peut se réaliser de la manière suivante :

1. Déterminer la liste des entités.
2. Pour chaque entité :
   — établir la liste de ses attributs ;
   — parmi ceux-ci, déterminer un identifiant.
3. Déterminer les relations entre les entités.
4. Pour chaque relation :
   — dresser la liste des attributs propres à la relation ;
   — définir les cardinalités.
5. Vérifier le schéma obtenu, notamment. qu'il répond aux exigences (formes normales, voir paragraphe suivant
6. Valider avec les utilisateurs

Entité et attributs

Une entité représente un objet ou un concept du monde réel, tandis qu'un attribut est une caractéristique de cette entité. Par exemple, dans une base de données pour un contrat d'hébergement web, les entités pourraient être CLIENT, SITE, CONTRAT, PROJET avec des attributs comme identifiant / référence, nom, nom de domaine, etc. On notera que l'entité CONTRAT n'existe que si l'entité CLIENT correspondante existe (l'inverse n'étant pas forcément vrai).

La présence d'un attribut peut être nécessaire et obligatoire pour décrire un objet ou alors, optionnelle. L'important est que chaque attribut soit non divisible, par exemple avec uniquement une seule adresse (ce qu'on appelle la première forme normale) et qu'il corresponde uniquement à l'entité qu'il décrit (deuxième forme normale).

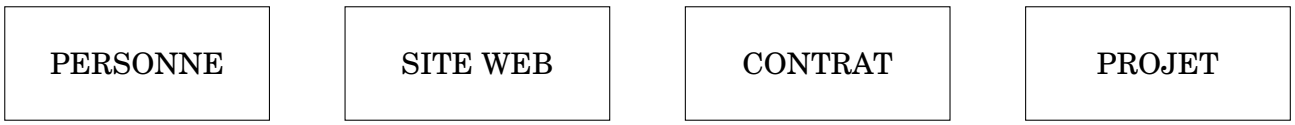


FIGURE 6.1 – Exemple de création des entités

L'identifiant est composé d'un ou plusieurs attributs qui permettent d'identifier de manière unique chaque enregistrement d'une entité. Par exemple, la référence du contrat peut servir d'identifiant unique pour l'entité CONTRAT. Une clé primaire identifie de manière unique chaque enregistrement dans une table, tandis qu'une clé étrangère permet de lier des tables entre elles. Par exemple, dans une relation entre un CLIENT et ses CONTRATS, la clé primaire du CLIENT pourrait être utilisée comme clé étrangère dans la table CONTRAT pour lier les enregistrements.

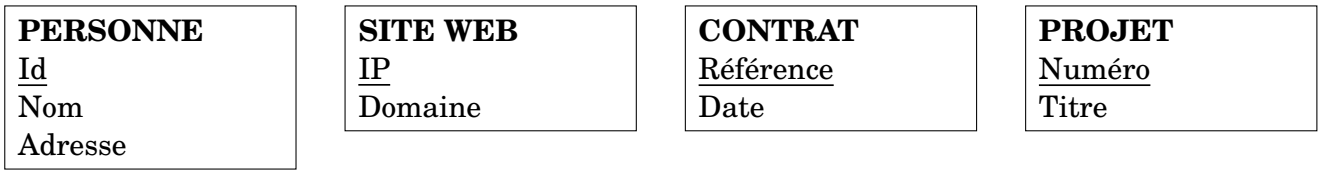


FIGURE 6.2 – Liste des attributs et identifiants

6.1.2 Modèle logique (MLD)

Le modèle logique est une étape clé dans le processus de conception d'une base de données relationnelle. Il s'agit de qualifier les attributs (int, char, date, etc.), d'ajouter les clés étrangères, d'éliminer les redondances et les anomalies en suivant les principes des formes normales ce qui, le cas échéant, signifie parfois de créer des tables supplémentaires.

Relations et cardinalité

Les relations décrivent les liens logiques entre les entités. Par exemple, un CLIENT peut avoir plusieurs CONTRATS, ce qui définit une relation "un à plusieurs". La cardinalité indique le nombre de participations d'une entité dans une relation, comme "1-1", "0-N", ou "1-N", décrivant les connexions minimales et maximales possibles.

— **Cadinalité un à un ou zéro à un** Si un site web ne peut être inclus que dans un seul contrat, et qu'un contrat n'a qu'un seul site d'ébergement possible.

— **Cadinalité un à plusieurs** Une personne peut avoir plusieurs contrats, mais chaque contrat n'est lié qu'à une seule personne.

— **Plusieurs à plusieurs** Par exemple, dans les cas où une personne peut travailler sur plusieurs projets, et qu'un projet donné peut intégrer plusieurs personnes à la fois.

Formes normales

Pour fonctionner de manière optimale, c'est-à-dire minimiser la redondance et assurer l'intégrité des données, un bon modèle relationnel doit respecter (au minimum) trois formes normales :

— **1ère forme normale (1FN)** Chaque champ doit contenir des valeurs uniques

— **2ème forme normale (2FN)** Chaque attribut dépend entièrement et exclusivement d'une entité donnée

— **3ème forme normale (3FN)** Aucun attribut ne doit dépendre d'un autre attribut non clé primaire

Prenons l'exemple d'une bibliothèque, avec une table qui recense les emprunts de livres par des adhérents :

Id	Nom	Livres	Auteurs
1	Alice	1984 et Hamlet	G. Orwell, W. Shakespeare

Problème des données non atomiques (1FN)

La colonne *Livres* contient plusieurs valeurs, rendant sa manipulation difficile. On divisera la colone en deux pour éviter ce problème de manipulation et identification des données :

Id	Nom	Livres	Auteurs
1	Alice	1984	G. Orwell
2	Alice	Hamlet	W. Shakespeare

Les données dans les colonnes sont maintenant uniques !

Problème des données non strictement dépendantes (2FN)

Les livres *1984* et *Hamlet* n'ont, en soi, au départ, pas directement de lien avec l'emprunteur et inversement. Nous avons ici des noms d'dhérents et des titres qui se retrouveront à plusieurs endroits dans la base de données, compliquant les recherches en cas de faute de frappe. Pour contourner ce problème, on va créer deux tables, l'une pour les adhérents et l'autre pour les ouvrages, que l'on relie avec une troisième (pour enregistrer les emprunts) :

Id	Id.A	Id.L
1	1	1
2	1	2

Id	Nom
1	Alice

Id	Livres	Auteurs
1	1984	G. Orwell
2	Hamlet	W. Shakespeare

A priori, plus aucun redondance dans les tables correspondantes aux divers entités !...

Problème des dépendances transitives (3FN)
Ce cas survient lorsqu'une colonne dépend d'une autre colonne qui n'est pas la clé primaire. Sachant que les auteurs peuvent rédiger plusieurs ouvrages, nous aurons, à terme, une répétition du nom d'auteur.

Id	Livres	Auteur
1	1984	G. Orwell
2	Hamlet	W. Shakespeare
3	Othello	W. Shakespeare

Pour éviter cela, on divise à nouveau la table en deux :

Id	Auteurs
1	G. Orwell
2	W. Shakespeare

Id	Livres	Id.A
1	1984	1
2	Hamlet	2
3	Othello	2

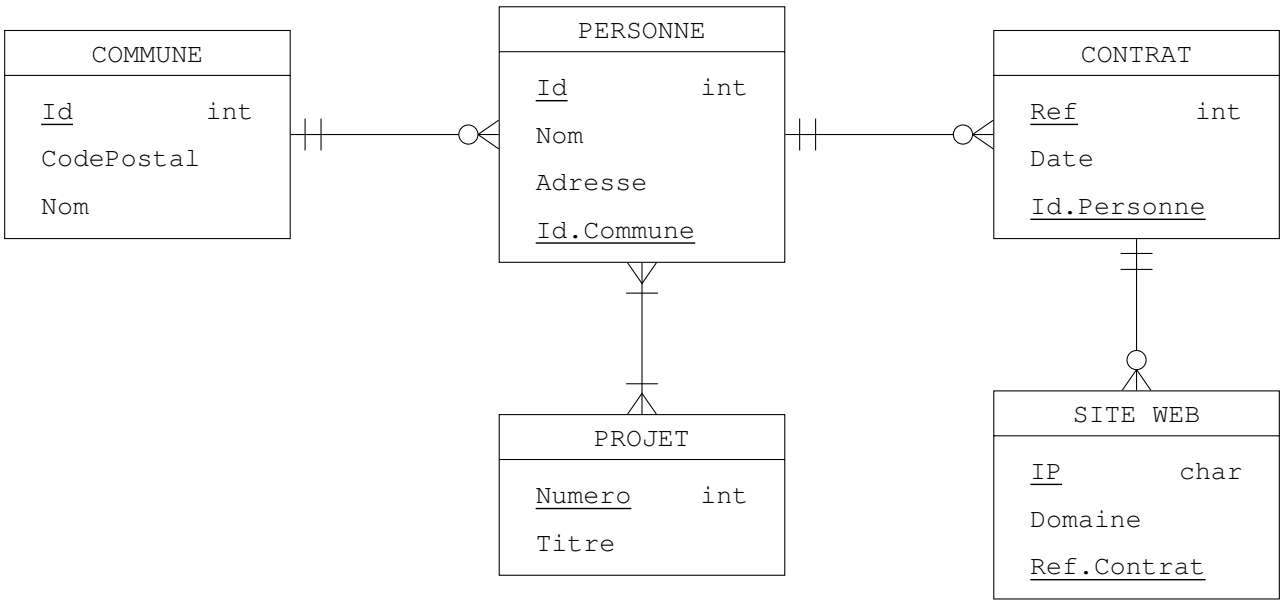
Ainsi, on garantit une base de données bien structurée et optimisée (rapide).

Illustrations par des schémas

Reprenons notre exemple du chapitre précédent : celui de la table `PERSONNE` avec l'"adresse de notre école, `Route de Base 24 1228 Plan-les-Ouates`. Ce type de données présente plusieurs risques d'erreurs : entre autres, répétition des noms de communes et incohérences dans l'orthographe de ces derniers. Typiquement, cela se produit si les attributs ne respectent pas les formes normales, où...

1. Chaque valeur des attributs est **atomique, non répétitive et constante** dans le temps
2. Une **dépendance stricte** des attributs vis-à-vis de l'identifiant unique (sans redondances) est exigée
3. Un attribut n'appartenant pas à la clé primaire **ne dépend pas d'un autre attribut** qui n'est pas non plus une clé primaire (ce qui serait par exemple le cas si l'on a une table `COMMANDE` qui inclut l'identifiant de la commande — clé primaire, l'identifiant de la personne et son nom...).

Pour éviter ces problèmes, nous procéderons donc à la normalisation des données en créant, dans l'exemple ci-dessus, une table séparée, `COMMUNE`, qui centralisera les informations spécifiques aux communes (nom, code postal, etc.). Cela permet de réduire la redondance, d'améliorer la cohérence des données et de respecter les exigences des formes normales. Illustré par un schéma, nous obtenons un modèle de données plus robuste.



Remarquons dans le schéma ci-dessus le problème entre PERSONNES et PROJET où il est impossible de relier une personne et un projet. C'est, ici aussi, une question de normalisation des données.

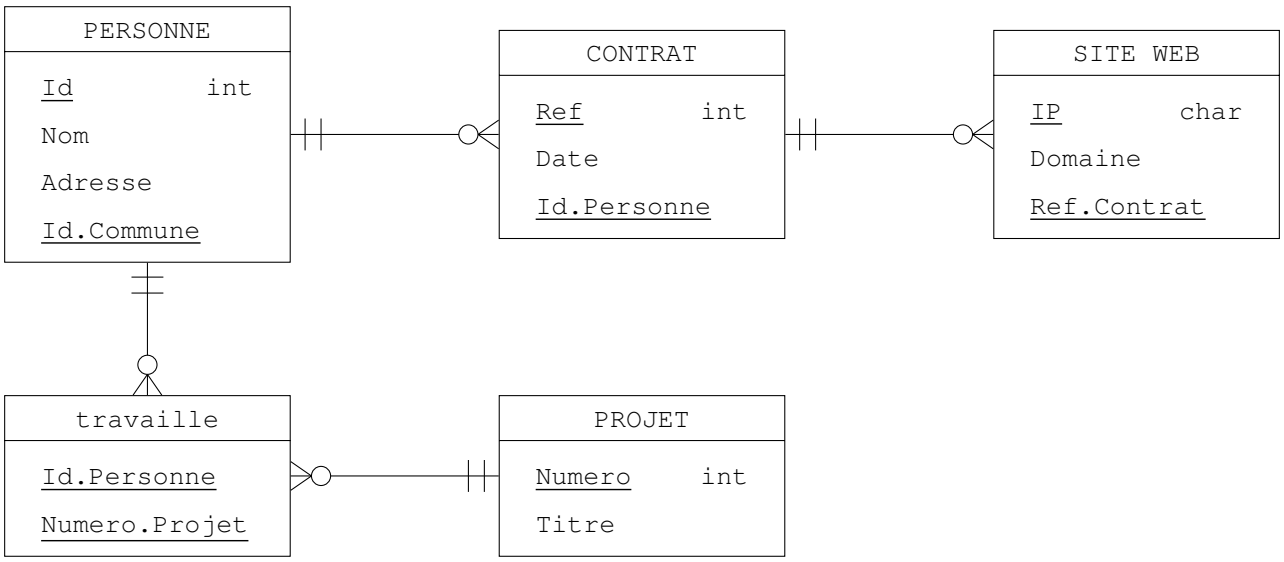


FIGURE 6.3 – Exemples de modèle logique respectant les formes normales

6.1.3 Modèle physique

Le modèle physique de données décrit comment les données seront stockées, organisées, et accédées physiquement dans la base de données. Ce qui inclut des détails spécifiques du type d'attributs, les méthodes d'indexation et les relations d'intégrité. Par exemple, on établira que pour enregistrer un contrat pour une personne donnée, son identifiant (et donc la personne) doit exister. Cette thématique sera le sujet de la section 6.2.1, commençant à la page suivante.

6.2 Language SQL et systèmes de gestion des bases de données (SGBD)

6.2.1 De la création à l'insertion de nouvelles données

Création d'une base Exercices p. 167, aide-mémoire p. 178

Pour créer la base de données, ouvrez une fenêtre de terminal et exécutez la commande suivante :

```
$ sqlite3 clients.db
```

Cela créera une nouvelle base de données nommée `clients`. Si le fichier existe déjà, SQLite ouvrira une connexion à la base existante ; s'il n'existe pas, SQLite la créera. À ce stade, votre invite changera, et un nouveau préfixe, `sqlite>`, apparaîtra.

Si le fichier `clients.db` n'existe pas encore et si vous quittez l'invite SQLite sans exécuter de requêtes, la base ne sera pas créée. Pour vous assurer que le fichier est bien créé, vous pouvez exécuter une requête vide en tapant `;` suivi de la touche `Enter`. Pour renommer le nom fournit par défaut, on utilisera la commande `ATTACH DATABASE 'clients.db' AS clients;`. En cas d'erreur, il est toujours possible de quiter la sessions avec les touches Ctrl + D.

La base de données `clients` étant créée (à vérifier avec la comande `.database`), nous allons maintenant créer une nouvelle table et y insérer des données.

Création d'une table

Les bases de données sont organisées en tables, qui stockent les informations (comme des feuilles de calcul Excel que l'on peut relier entre elles). Créons une première table inspirée de la table `COMMUNE` du chapitre précédent. Notez que, contrairement à Python, les commandes SQL se terminent par un point-virgule `;`.

— Un identifiant (id)
— Le code postal
— Le nom de la commune

Utilisez la commande suivante pour créer la table :

```
CREATE TABLE COMMUNE (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    cp INTEGER NOT NULL,
    nom TEXT NOT NULL
);
```

La création de la table peut être validé à l'aide de la commande `.table`. À noter que l'utilisation de `NOT NULL` rend ce champ obligatoire. Insérons maintenant trois lignes de valeurs dans la table `clients` créée précédemment :

```
INSERT INTO COMMUNE VALUES (1, 1100, "Lausanne");
INSERT INTO COMMUNE VALUES (2, 1200, "Geneve");
INSERT INTO COMMUNE VALUES (3, 1950, "Sion");
```

Comme nous avons spécifié `NOT NULL` pour chacune des colonnes de votre table, nous devons entrer une valeur pour chacune d'elles. Par exemple, si vous essayez d'ajouter un autre client sans indiquer sa taille :

```
INSERT INTO PERSONNES VALUES (4, "Fribourg");
```

Vous recevrez une erreur indiquant qu'il manque la valeur d'une colonne. La commande correcte étant

```
INSERT INTO COMMUNE VALUES (4, 1700, "Fribourg");
```

6.2.2 De la lecture à la suppression de données

Début du § p. 96

Lecture des tables

Durant cette étape, nous allons nous concentrer sur les méthodes les plus simples pour lire les données d'une table. Pour afficher la table avec toutes les valeurs insérées, utilisez la commande `SELECT` :

```
SELECT * FROM COMMUNE;
```

Vous verrez alors les entrées précédemment insérées :

```
1|1100|Lausanne
2|1200|Genève
3|1950|Sion
4|1700|Fribourg
```

Pour afficher une entrée spécifique en fonction de son identifiant, ajoutez la clause `WHERE` à la requête :

```
SELECT * FROM COMMUNE WHERE id = 2;
```

Cela retournera uniquement la commune identifiée par le numéro 2. Examinons cette commande :

1. Tout d'abord, nous sélectionnons avec `SELECT` toutes les valeurs (`*`) de notre table
2. Ensuite, nous appliquons une condition avec `WHERE` pour ne sélectionner que les entrées où `id` est égal à 2

Utilisation des opérateurs de comparaison

Un opérateur de comparaison dans une clause `WHERE` définit comment une colonne doit être comparée à une valeur. Voici quelques opérateurs de comparaison SQL courants :

=	Égalité	!=	Inégalité
IS NOT NULL	Valeur n'est pas `NULL`	IS NULL	Valeur est `NULL`
< / >	Inférieur / supérieur à	LIKE	Valeur correspond à un modèle donné
BETWEEN	Valeur est dans une plage donnée	IN	Valeur est dans un ensemble de valeurs

Ajout de colonnes

SQLite vous permet de modifier vos tables à l'aide de la commande `ALTER TABLE`. Utilisez `ALTER TABLE` pour ajouter une nouvelle colonne à la table `COMMUNE` pour identifier la langue parlée dans une ville donnée.

```
ALTER TABLE COMMUNE ADD COLUMN langue TEXT;
```

Vous avez maintenant une quatrième colonne `langue` dans votre table.

Mise à jour des valeurs

Utilisez la commande `UPDATE` pour ajouter des valeurs pour chacune des communes...

```
UPDATE COMMUNE SET langue = "fr" WHERE id = 1;
...
UPDATE COMMUNE SET langue = "fr/al" WHERE id = 4;
```

Lancer la commande suivante pour connaitre le résultat.

```
SELECT * FROM COMMUNE;
```

À noter qu'avec le table MySQL, il est possible d'ajouter directement une valeur par défault.

```
ALTER TABLE COMMUNE ADD COLUMN langue TEXT DEFAULT 'fr';
```

Vous avez ainsi modifié la structure de votre table et mis à jour les valeurs qu'elle contient.

Suppression d'informations

La commande suivante supprimera toutes les communes dont le code postal est > 1500

```
DELETE FROM COMMUNE WHERE cp > 1500;
```

En tapant `SELECT * FROM clients;` vous pourrez vérifier que `Sion` et `Fribourg` ont été supprimées. Seules restent les villes de `Lausanne` et `Genève`.

6.3 Gestion avancée des bases de données

6.3.1 Association d'informations

Début du § p. 96

Bien souvent il existe plusieurs tables : notre table actuelle `COMMUNE` et une table `PERSONNE`, avec une valeur `id` qui correspond à un identifiant dans votre table `COMMUNE`. Si vous voulez interroger des données des deux tables, vous devez soit utiliser l'une des quatre commandes de jointure suivante :

- `INNER JOIN`
- `OUTER JOIN`
- `LEFT JOIN`
- `CROSS JOIN`

Soit déclarer une clé étrangère. C'est cette deuxième option, plus fréquente que la première que nous allons étudier.

```
CREATE TABLE PERSONNE (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nom TEXT NOT NULL,
    adresse TEXT,
    id_commune INTEGER,
    FOREIGN KEY(id_commune) REFERENCES COMMUNE(id)
);
INSERT INTO PERSONNE VALUES (1,  "Prenom NOM", "Adresse", 2);
```

Joignez maintenant vos tables :

```
SELECT * FROM COMMUNE
  INNER JOIN PERSONNE on COMMUNE.id = PERSONNE.id_commune;
```

Votre résultat ressemblera à ceci :

```
2|1200|Geneve|fr|1|Prénom NOM|Adresse|2
```

Notez que le résultat inclut également la valeur `id` de `PERSONNE`. Vous pouvez spécifier l'affichage souhaitée avec une commande plus explicite :

```
SELECT PERSONNE.nom, COMMUNE.cp, COMMUNE.nom, COMMUNE.langue FROM COMMUNE
  INNER JOIN PERSONNE on COMMUNE.id = PERSONNE.id_commune;
```

Cette fois-ci, le résultat exclut le second `id` :

```
Prénom NOM|1200|Geneve|fr
```

Vous avez joint avec succès des informations provenant de plusieurs tables, bravo ! Généralisons maintenant quelques concepts.

6.3.2 Utilisation des caractères spéciaux

SQLite, comme la plupart des bases de données, permet l'utilisation de caractères spéciaux ou génériques. Ceux-ci sont utiles si vous essayez de trouver une entrée spécifique dans une table, mais que vous n'êtes pas sûr de ce qu'est cette entrée exactement.

Les signes de pourcentage (%) représentent zéro ou plusieurs caractères inconnus.

```
SELECT * FROM LATABLE WHERE colonne LIKE val\%;
```

Les traits de soulignement (_) sont utilisés pour représenter un seul caractère inconnu :

```
SELECT * FROM LATABLE WHERE colonne LIKE v_eur;
```

6.3.3 Les regroupements

Début du § p. 96

Compter le nombre d'entrées

La fonction COUNT est utilisée pour trouver le nombre d'entrées dans une colonne donnée. La syntaxe suivante renverra le nombre total de valeurs contenues dans la colonne :

```
SELECT COUNT(colonne) FROM LATABLE;
```

Vous pouvez affiner les résultats d'une fonction COUNT en ajoutant une clause WHERE, comme ceci :

```
SELECT COUNT(colonne) FROM LATABLE WHERE colonne = valeur;
```

Valeur moyenne

La fonction AVG est utilisée pour trouver la moyenne parmi les valeurs contenues dans une colonne spécifique. Notez que la fonction AVG ne fonctionnera qu'avec des colonnes contenant des valeurs numériques ; lorsqu'elle est utilisée sur une colonne contenant des valeurs de chaîne, elle peut renvoyersoit une erreur, soit 0.

```
SELECT AVG(colonne) FROM LATABLE;
```

Somme des valeurs

La fonction SUM est utilisée pour trouver le total de toutes les valeurs numériques contenues dans une colonne :

```
SELECT SUM(colonne) FROM LATABLE;
```

Comme avec la fonction AVG, si vous exécutez la fonction SUM sur une colonne contenant des valeurs de chaîne, elle peut renvoyer une erreur ou simplement 0, selon le système de gestion de base de données (SGBD) utilisé.

La plus grande valeur

Pour trouver la plus grande valeur numérique d'une colonne ou la dernière valeur par ordre alphabétique, utilisez la fonction `MAX` :

```
SELECT MAX(colonne) FROM LATABLE;
```

La plus petite valeur

Pour trouver la plus petite valeur numérique dans une colonne ou la première valeur par ordre alphabétique, utilisez la fonction `MIN` :

```
SELECT MIN(colonne) FROM LATABLE;
```

6.3.4 Tri des résultats

ORDER BY

Une clause `ORDER BY` est utilisée pour trier les résultats des requêtes. La syntaxe de requête suivante renvoie les valeurs des colonnes `column_1` et `column_2` et trie les résultats par les valeurs contenues dans `column_1` par ordre croissant ou, pour les valeurs de chaîne, par ordre alphabétique :

```
SELECT colonne_1, colonne_2 FROM LATABLE ORDER BY colonne_1;
```

Pour effectuer la même action, mais trier les résultats par ordre décroissant ou par ordre alphabétique inverse, ajoutez la requête avec `DESC` :

```
SELECT colonne_1, colonne_2 FROM LATABLE ORDER BY colonne_1 DESC;
```

GROUP BY

La clause `GROUP BY` est similaire à la clause `ORDER BY`, mais elle est utilisée pour trier les résultats d'une requête qui inclut une fonction d'agrégation telle que `COUNT`, `MAX`, `MIN` ou `SUM`. En soi, les fonctions d'agrégation décrites plus haut ne renverront qu'une seule valeur. Cependant, vous pouvez voir les résultats d'une fonction d'agrégation appliquée à chaque valeur correspondante d'une colonne en incluant une clause `GROUP BY`.

La syntaxe suivante comptera le nombre de valeurs correspondantes dans `column_2` et les regroupera par ordre croissant ou alphabétique :

```
SELECT COUNT(colonne_1), colonne_2 FROM LATABLE GROUP BY colonne_2;
```

Pour effectuer la même action, pour regrouper les résultats par ordre décroissant ou par ordre alphabétique inverse, ajoutez la requête avec `DESC` :

```
SELECT COUNT(colonne_1), colonne_1 FROM LATABLE GROUP BY colonne_1 DESC;
```

6.4 Python et les bases de données

Nous allons nous intéresser au module `sqlite3` de Python. Comme précédement, nous allons créer une connexion à une base de données SQLite, ajouter une table à cette base de données, insérer des données dans cette table, puis lire et modifier ces données, mais cette fois-ci directement avec un language de programmation, comme c'est le cas par exemple lors de la programmation de page web en php. Pour tirer le meilleur parti de ce tutoriel, il est donc recommandé d'avoir une certaine familiarité avec la programmation.
Partons du modèle utilisé précédement.

6.4.1 De la connexion à l'insértion de nouvelles données

Connexion

Lorsque nous nous connectons à une base de données SQLite, nous accédons aux données qui résident finalement dans un fichier sur notre ordinateur. Nous pouvons nous connecter à une base de données SQLite en utilisant le module Python `sqlite3` :

```
import sqlite3
connection = sqlite3.connect("clients.db")
```

La commande `import sqlite3` donne accès au module `sqlite3` dans notre programme Python. La fonction `sqlite3.connect()` renvoie un objet `Connection` que nous utiliserons pour interagir avec la base de données SQLite contenue dans le fichier `clients.db`. Le fichier `clients.db` est créé automatiquement par `sqlite3.connect()` si `clients.db` n'existe pas déjà sur notre ordinateur.

Nous pouvons vérifier que notre objet de connexion a été créé avec succès en exécutant :

```
print(connection.total_changes)
```

`connection.total_changes` est le nombre total de lignes de la base de données qui ont été modifiées par la connexion. Comme nous n'avons encore exécuté aucune commande SQL, 0 modifications est correct.

Remarque : nous pouvons supprimer à tout moment le fichier `clients.db` de notre ordinateur. il est également possible de se connecter à une base de données SQLite qui réside strictement en mémoire (et non dans un fichier) en passant la chaîne spéciale `":memory:"` à `sqlite3.con`
Par exemple, `sqlite3.connect(":memory:")`. Une base de données SQLite en mémoire disparaîtra dès que votre programme Python se termine. Cela peut être pratique si vous souhaitez un bac à sable temporaire pour essayer quelque chose, sans avoir besoin de persister les données après la sortie de votre programme.

Ajout de données

Maintenant que nous sommes connectés à la base de données SQLite `clients.db`, nous pouvons commencer à insérer et lire des données. Nous allons créer une table nommée `CONTRAT` qui contient les données suivantes :

```
ref     date         id_personne
1       2024-05-13   1
2       2024-09-01   1
```

Deux lignes d'exemples de contrats sont listées : une ligne pour un contrat signé en mai et un autre signé début septembre.

Nous pouvons créer cette table `CONTRAT` en utilisant la connexion que nous avons établie précédement :

```
cursor = connection.cursor()
cursor.execute("CREATE TABLE CONTRAT (
    ref INTEGER PRIMARY KEY,
    date DATE NOT NULL,
    id_personne INTEGER NOT NULL,
    FOREIGN KEY (id_personne) REFERENCES PERSONNE(id)
);")
```

`connection.cursor()` renvoie un objet `Cursor`. Les objets `Cursor` nous permettent d'envoyer des instructions SQL à une base de données SQLite en utilisant `cursor.execute()`. La chaîne `"CREATE TABLE CONTRAT ..."` est une instruction SQL qui crée une table nommée `CONTRAT` avec les trois colonnes décrites précédemment.

Maintenant que nous avons créé une table, nous pouvons insérer des lignes de données dans celle-ci :

```
cursor.execute("INSERT INTO CONTRAT (ref, date, id_personne) VALUES (1,
   '2024-05-13', 1);")
cursor.execute("INSERT INTO CONTRAT (ref, date, id_personne) VALUES (2,
   '2024-09-01', 1);")
```

Nous appelons `cursor.execute()` deux fois : une fois pour insérer une ligne pour le requin Sammy dans le bac réservoir 1, et une fois pour insérer une ligne pour la seiche Jamie dans le bac 7. `"INSERT INTO CONTRAT VALUES ..."` est une instruction SQL qui nous permet d'ajouter des lignes à une table.

6.4.2 De la lecture à la modification de données

Lecture des données

À l'étape précédente, nous avons ajouté deux lignes à une table SQLite nommée `CONTRAT`. Nous pouvons récupérer ces lignes en utilisant une instruction SQL `SELECT` :

```
rows = cursor.execute("SELECT * FROM CONTRAT").fetchall()
print(rows)
```

Si nous exécutons ce code, nous verrons une sortie similaire à celle-ci :

```
[(1, '2024-05-13', 1), (2, '2024-09-01', 1)]
```

La fonction `cursor.execute()` exécute une instruction `SELECT` pour récupérer les valeurs de la table `CONTRAT`. `fetchall()` récupère tous les résultats de l'instruction `SELECT`. Lorsque nous utilisons `print(rows)`, nous voyons une liste de deux tuples. Chaque tuple contient trois entrées ; une entrée pour chaque colonne de la table `CONTRAT`.

Si nous voulons récupérer des lignes dans la table `CONTRAT` qui correspondent à un ensemble de critères spécifiques, nous pouvons également utiliser une clause `WHERE` :

```
date_contrat = "2024-09-01"
rows = cursor.execute(
    "SELECT * FROM CONTRAT WHERE date = ?",
    (date_contrat),
).fetchall()
print(rows)
```

Si nous exécutons cela, nous aurons :

```
[(2, '2024-09-01', 1)]
```

Comme dans l'exemple précédent, `cursor.execute(<instruction SQL>)` suivi de `fetchall()` nous permet de récupérer tous les résultats d'une instruction `SELECT`. La clause `WHERE` dans l'instruction `SELECT` filtre les lignes où la `date` correspond à `date_contrat`. Remarquez que nous utilisons `?` pour substituer notre variable dans l'instruction `SELECT`.

Attention : N'utilisez jamais les opérations de chaîne Python pour créer dynamiquement une chaîne d'instructions SQL. Utiliser les opérations de chaîne Python pour assembler une chaîne d'instructions SQL vous rend vulnérable aux attaques. En effet, les attaques (par SQL) peuvent être utilisées pour voler, altérer ou modifier les données stockées dans votre base de données. Utilisez toujours le caractère de remplacement `?` dans vos instructions SQL pour substituer dynamiquement les valeurs de votre programme. Passez un tuple de valeurs comme second argument à `Cursor.execute()` pour lier vos valeurs à l'instruction SQL.

6.4.3 Clôture de session

Nous avons ici utilisé deux objets principaux pour interagir avec la base de données SQLite `clients.db` : un objet `Connection` nommé `connection`, et un objet `Cursor` nommé `cursor`. De la même manière que les fichiers Python doivent être fermés lorsque nous avons terminé de les utiliser, les objets `Connection` et `Cursor` doivent également être fermés lorsqu'ils ne sont plus nécessaires.

Nous pouvons pour cela utiliser une instruction `with` pour nous aider à fermer les objets `Connection` et `Cursor` de manière automatique :

```
from contextlib import closing

with closing(sqlite3.connect("clients.db")) as connection:
    with closing(connection.cursor()) as cursor:
        rows = cursor.execute("SELECT 1").fetchall()
        print(rows)
```

`closing` est une fonction utilitaire fournie par le module `contextlib`. Lorsque l'instruction `with` se termine, `closing` garantit que la méthode `close()` est appelée sur l'objet qui lui est passé. La fonction `closing` est utilisée deux fois dans cet exemple. Une fois pour s'assurer que l'objet `Connection` renvoyé par `sqlite3.connect()` est automatiquement fermé, et une deuxième fois pour s'assurer que l'objet `Cursor` renvoyé par `connection.cursor()` est automatiquement fermé.

Si nous exécutons ce code, nous verrons une sortie du genre :

```
[(1,)]
```

Comme `"SELECT 1"` est une instruction SQL qui renvoie toujours une seule ligne avec une seule colonne ayant une valeur de 1, il est logique de voir un seul tuple avec 1 comme seule valeur renvoyée par notre code.

6.5 Administration de SQLite à l'aide d'un système de gestion de base de données (SGBD)

À COMPLETER

Chapitre 7

POUR ALLER PLUS LOIN

7.1 Utilisation de son environnement de travail avec MS-DOS

7.1.1 Interpréteur de commandes

Sous Windows, vous disposez d'outils pour manipuler graphiquement les éléments de votre ordinateur. Mais comme sous Linux (voir partie sur les systèmes d'exploitation), vous disposez aussi d'un interpréteur de commandes vous permettant d'exécuter des commandes sur votre système.

Les commandes sont tapées dans un terminal en mode texte. Pour démarrer l'interpréteur de commandes sous Windows, il vous suffit d'aller dans **Démarrer → Tous les programmes → Accessoires → Invite de commandes**.

Plus simplement, vous pouvez aussi faire **Démarrer → Exécuter...** → `cmd`.

Une commande Windows se décompose en trois parties :

— La commande

— Des arguments, zéro, un nombre fixe ou variable d'arguments

— Des options, qui comme son nom l'indique, sont optionnelles (aucune, une ou plusieurs options)

Toutes les commandes sous Windows prennent la forme : `commande args [/OPTION]`

Aide sur les commandes

Un premier niveau d'aide vous permet d'avoir accès à la liste des commandes de base disponibles depuis votre interprète de commandes. Il vous suffit d'utiliser la commande `help`.

```
$ help
```

Il est aussi possible d'obtenir de l'aide sur une commande particulière afin de connaître les options et les arguments pour l'utiliser. Deux méthodes sont disponibles pour obtenir cette

aide sur une commande : soit à l'aide de la commande `help`, soit en utilisant l'option `/ ?` pour une commande donnée.

```
$ help commande
$ commande /?
```

7.1.2 Répertoires

Notion de répertoires

Dans le monde Windows, la racine d'un répertoire est appellé par un `\`. Il existe plusieurs racines, une par partition ou disque dur sous les systèmes d'exploitation. Chaque partition ou disque dur sous Windows est identifié par une lettre (attribuée suivant l'ordre de détection). Le premier disque dur (ou partition) sera noté `C:`, le deuxième `D:`, etc. Un système Windows peut donc inclure de nouveaux disque durs, partition ou périphérique (clé USB, etc.) jusqu'à la lettre `Z:` (`A:` et `B:` étant généralement réservés pour les lecteurs externes).

Un répertoire qui en contient un autre est dit *répertoire parent*. Lorsque d'un répertoire on veut aller au répertoire parent, celui-ci est désigné par `..`, comme sur la plupart des systèmes.

Visualisation des répertoires

La commande `dir` permet de lister le contenu d'un répertoire. Voici un tableau des options les plus courantes :

Option	
`/A`	Affiche tous les fichiers, y compris les fichiers cachés
`/N`	Affiche le contenu du répertoire au format long
`/P`	Affiche le contenu du répertoire en défilement par page
`/S`	Affichage récursif du contenu du répertoire courant

Les options de ce tableau peuvent être utilisées séparément ou conjointement. Par exemple :

```
$ dir /S /P
```

La commande `tree` permet l'affichage récursif du répertoire courant sous la forme d'un arbre.

```
$ tree répertoire
```

Déplacement dans les répertoires

La commande `cd` permet de se déplacer dans l'arborescence des fichiers. Elle s'utilise en lui donnant comme argument un répertoire.

```
$ cd répertoire
```

Note : comme sous Linux, il est possible d'utiliser des chemins relatifs et absolus. Ainsi, pour remonter dans le répertoire parent, on peut utiliser la commande :

```
$ cd ..
```

La commande `cd`, utilisée sans argument, vous informe sur le contenu du répertoire courant. La commande `cd` utilisée avec pour argument `\` vous ramène à la racine de l'unité (partition) en cours.

```
$ cd \
```

Pour changer d'unité, il vous suffit de désigner par sa lettre la nouvelle unité sur laquelle vous souhaitez aller. Imaginez que vous possédez deux unités appelées `C:` et `D:`. Vous pouvez taper la lettre en majuscule ou en minuscule.

```
$ d:
```

Création et gestion des répertoires

La commande `mkdir` ou `md` en version abrégée permet de créer un répertoire. Elle prend en argument le nom du répertoire que vous souhaitez créer. Par exemple :

```
$ md test\test
```

Pour copier un répertoire et son contenu, il faut utiliser la commande `xcopy` :

```
$ xcopy source destination /options
```

Option	
`/E`	Copie les répertoires et sous-répertoires y compris les répertoires vides
`/I`	Si la destination n'existe pas et que plus d'un fichier est copié, considérer la destination comme un répertoire
`/Q`	N'affiche pas le nom des fichiers lors de la copie
`/H`	Copie également les fichiers cachés et les fichiers systèmes
`/T`	Crée la structure des répertoires mais ne copie pas les fichiers
`/Y`	Supprime la demande de confirmation de remplacement des fichiers existants

Pour supprimer un répertoire, utilisez la commande `rmdir` ou `rd`. Par exemple :

```
$ rd test\test
```

7.1.3 Fichiers

Gestion des fichiers

Pour renommer un fichier, utilisez la commande `rename` ou `ren`. Par exemple :

```
$ rename ancien nouveau_nom
```

Pour supprimer un fichier, utilisez la commande `del` ou `erase`. Par exemple :

```
$ del fichier1.txt fichier2.txt
```

Option	
`/P`	Demande confirmation avant la suppression
`/F`	Force la suppression
`/S`	Supprime les fichiers dans tous les sous-répertoires

Utilisation de fichiers batch

Les fichiers batch (.bat) sont des exécutables qui permettent de réaliser une série de commande DOS en une seule opération. Exemple de fichier batch :
À COMPLÉTER.

7.2 Représentation algorithmique

Résolution de la multiplication russe à l'aide d'un tableur

	A	B	C	D	E	F	La colonne A représente les restes et la F, la sommes des multiplicandes. Les formules sont suivantes...
1	Si # ≤ 256	12	x		25		
2	0	6			50	0	A2 : = B1 - B2 x 2
3	0	3			100	0	B2 : = ARRONDI.INF(B1/2,0)
4	1	1			200	200	E2 : = E1 x 2
5	1	0			400	600	F2 : = SI (A2 = 1 ; E2 ; 0)
6	0	0			800	600	F3 : = SI (A2 = 1 ; E3 + F2 ; F2)
7	0	0			1600	600	G9 : = F9 / 2
8	0	0			3200	600	
9	0	0			6400	600	/ 2 = 300

7.3 Concepts spécifiques pour la rédaction de scripts

7.3.1 Containers

Tuples

Un tuple est comme une liste, mais vous ne pouvez pas changer ses éléments une fois créés (on dit qu'il est immutable). Les tuples utilisent des parenthèses `()`.

— Pour un tuple avec un seul élément, ajoutez une virgule après l'élément

— Accédez aux éléments du tuple avec des crochets `[]`

```
tpl = (1, "deux", 3.0) # Creer un tuple

singleton = (2)        # Tuple avec un seul element

element = tpl[0]       # Acceder a l'element no 0, qui donne 1
```

Les tuples sont utiles car ils prennent moins de mémoire et sont plus rapides à traiter que les listes.

Sets

Un set est une collection non ordonnée d'éléments uniques. C'est utile pour des opérations comme l'union, l'intersection, etc.

```
mon_set = {1, 2, 3}  # Creer un set

mon_set.add(4)       # Ajouter un element

mon_set.remove(2)    # Supprimer un element
```

Slicing

Le slicing permet d'accéder à des tranches d'une séquence (comme une liste ou un tuple).

```
ma_liste = [1, 2, 3, 4, 5] # Creer une liste

tranche = ma_liste[1:3]    # Acceder a une tranche, ici les nombres 2 et 3
```

Remarques

— Copier un dictionnaire ou un set avec `copy.deepcopy()` pour une copie complète

— Les frozensets sont comme les sets, mais immuables

Fonctions avec paramètres optionnels

Vous pouvez aussi définir des paramètres, dits optionnels, avec des valeurs par défaut...

```
def nomFonction(p1, p2=2): # Fonction avec parametres optionnels
    return p1 * p2

nomFonction(3)             # Donne 6 (3 * 2)
nomFonction(3, 3)          # Donne 9 (3 * 3)
```

Portée des variables

La portée est le périmètre dans lequel un nom (de variable, fonction...) est connu (visible) et utilisable. Les variables :

— Dans une fonction, sont locales à cette fonction

— À l'extérieur des fonctions, sont globales

```
# Exemple 1 ----------------------------------------------------
var_globale = 5

def nomFonction():
    var_locale = 10
    print(var_globale) # Peut acceder a var_globale

nomFonction()          # Affiche 5
print(var_locale)      # Erreur : var_locale n'est pas definie

# Exemple 2 ----------------------------------------------------
def incremCompt():     # Fonction sans param. d'entree
    global compteur    # Definition var. globale
    if "compteur" not in globals():
        compteur = 0   # Initialisation du compteur (si existant)
    compteur += 1
    print(compteur, "fois")

incremCompt() # => affiche: 1 fois
incremCompt() # => affiche: 2 fois ... etc.
```

Fonctions Lambda

Pour des fonctions simples, Python permet l'utilisation de fonctions lambda, qui sont des fonctions anonymes d'une seule ligne.

```
somme = lambda x, y: x + y  # Fonction Lambda

resultat = somme(5, 3)      # Utilisation de la fonction, donne 8
```

7.4 Modules et packages complémentaires

7.4.1 Tkinter : introduction au fenêtrage

Exercices p. 169

Tkinter est une bibliothèque (module) d'interface graphique intégrée à Python. Elle permet nottament de créer des applications avec des boutons, des champs de texte et d'autres éléments graphiques. Lors de la rédaction d'application Tkinter on procédera souvent en quatre étapes :

A) Importation du module

B) Création de la fenêtre principale

C) Ajout de widgets [1]

D) Lancement de la boucle principale

Voici un exemple de code :

```
import tkinter        # (A)
tk = tkinter

fenetre = tk.Tk()     # (B)

label = tk.Label(fenetre, text="Yo bro ;)")
label.pack()          # (C)

fenetre.mainloop()    # (D)
```

Pour intéragir avec les utilisateurs, on aura plus généralement recours aux widgets

— `Entry` qui permet de saisir du texte

— `Button` afin d'exécuter une action donnée

Ces éléments sont repris dans le code suivant qui permet de convertir des euros en francs suisses.

```
import tkinter

# FONCTION
def convertir_devise():
    montant = float(montantSaisi.get())
    taux = 1.08  # Taux de change donne
    resultat = montant * taux
    labelResultat.config(text=f"EUR {montant} = CHF {resultat}")

# VARIABLE
tk = tkinter

# WIDGETS
root = tk.Tk()
root.title("Convertisseur EUR/CHF")

labelMontant = tk.Label(root, text="Montant en EUR:")
```

1. Éléments graphiques que l'on peut ajouter dans une fenêtre

```
labelResultat = tk.Label(root, text="")

labelMontant.grid(row=0, column=0, padx=10, pady=10)
labelResultat.grid(row=2, column=0, columnspan=2, pady=10)

montantSaisi = tk.Entry(root)
montantSaisi.grid(row=0, column=1, padx=10, pady=10)

bouton = tk.Button(root, text="Convertir", command=convertir_devise)
bouton.grid(row=1, column=0, columnspan=2, pady=10)

root.mainloop()
```

7.5 Cryptographie

Exercices p. 170, aide-mémoire p. 177

La cryptographie est l'art de protéger des informations. Les définitions suivantes s'appliquent.

- **Le cryptage** est un processus de transformation d'un message, initialement lisible, en message codé, de manière :
  - **symétrique** si la clé pour chiffrer et déchiffrer est la même (ex. : César, Vigenère)
  - **asymétrique** si on utilise deux clés différentes, une publique pour chiffrer et une autre, privée, pour déchiffrer (ex. : RSA)
- **Un code** (ou message crypté) est un texte rendu illisible par cryptage (ou chiffrement)
- **Une clé** (de chiffrement) est une suite de caractères, plus ou moins longue, gardée secret, qui permet de chiffrer ou déchiffrer un message
- **Un caractère** `chr()` est représenté par un nombre, appelé Unicode, qui l'identifie de manière unique ; à l'inverse, la fonction `ord()` renvoie le nombre représentant l'unicode d'un caractère spécifique. Par exemple :

```
print(f"Lettre {chr(97)} <=> Unicode {ord('a')}")  # Lettre a <=> Unicode 97
print(f"Lettre {chr(65)} <=> Unicode {ord('A')}")  # Lettre A <=> Unicode 65
```

Notons ici trois points importants : tout d'abord nous considérerons, assez naturellement, un message comme une succession de mots, composés de lettres, ensuite, que, chaque opération de chiffrement et déchiffrement s'effectue sur un seul caractère à la fois et enfin, que les lettres sont différentes si elles sont écrites en majuscules ou minuscules. Les chiffrements symétriques (César, Vigenère) présentés ci-dessous sont aujourd'hui obsolètes, il n'est donc absolument pas recommandé de l'utiliser pour des communications sécurisées, mais servent de basent à la compréhension des concepts asymétriques !

7.5.1 Chiffrement par décalage

Exercices p. 170, aide-mémoire p. 177

Le chiffrement par décalage, ou cryptage symétrique de César, fait référence à l'imperator Jules César, consiste en un décalage des lettres de l'alphabet (soit 26 clés différentes possibles).

Fonction de chiffrement

Par exemple, le message `chat`", devient, avec un décalage de trois lettres, le code `fkdw`.

```
def cesar(msg, decal):
    txt = ""                            # Initialise le texte du message
    for char in msg:                    # Parcours chaque caractere
        code = ord(char) - 97 + decal   # ord('a') = 97, donc a devient 0 + decal
        txt += chr(code % 26 + 97)      # Convertir l'Unicode en lettre
    return txt                          # Rappel: % = modulo, soit le reste; expl
                                        # ord('y') - 97 + 3 = 27, 27 % 26 = 1 (b)
m, d = "chat", 3                        # [ m, d = "fkdw", -3 ]
print(cesar(m, d))                      #  ^ POUR DECRYPTER  ^
```

7.5.2 Chiffrement de Vigenère

Exercices p. 171, aide-mémoire p. 177

Le chiffrement de Vigenère, qui est également un cryptage symétrique, date de la fin du XIVe siècle. Au lieu d'un décalage par un nombre fixe, avec cette technique, on utilise une clé sous forme de mot. Chaque lettre de la clé définit un décalage différent pour chaque caractère du message à la position correspondante.

Explications

1. On choisit
   - un message à chiffrer, par exemple `hello`
   - une clé pour le chiffrement, par exemple `key` que l'on répète autant de fois que nécessaire pour qu'elle couvre toute la longueur du message, ici `keyke`
2. Chaque lettre du message en clair est décalée selon la lettre de la clé à la position correspondante. Pour mémoire, la 1e lettre de l'alphabet est en position 0 [ord(char) - 97].
   - `h +10 car k est la 11e lettre => r`
   - `e  +4 car e est la  5e lettre => i`
   - `l +24 car y est la 25e lettre => j`
   - `l +10 car k est la 11e lettre => v`
   - `o  +4 car e est la  5e lettre => s`

Fonction de chiffrement

```python
def vigenere(msg, cle):                    # v Ajuster la longueur de la cle
    crypto, cle = "", cle * (len(msg) // len(cle) + 1)
    for i in range(len(msg)):              # Option: ajouter if 'a' <= char <= 'z':
        crypto += chr((ord(msg[i]) - 97 + decal) % 26 + 97)
    return crypto                          # ^ Rappel: ord('a') = 97 / A -> 65
                                           #    + decal = ord(cle[i]) - 97
m, c = "hello", "key"                      #    - decal POUR CRYPTER
print(vigenere(m, c))                      # rijvs
```

On notera qu'une version encore légèrement améliorée, appelée le chiffrement de Vernam, a été utilisé durant la guerre froide (source : Yves Legrandgérard, 2019).

7.5.3 Chiffrement affine (asymétrique)

Exercices p. 172, aide-mémoire p. 177

Le chiffrement affine va, lui, substituer chaque lettre à l'aide d'une fonction mathématique, qui sera différente selon que l'on crypte (clé publique) ou décrypte (clé privée) le message !

$$f(x) = ax + b$$

où x est la position de la lettre dans l'alphabet (indépendamment de l'unicode, a=0, b=1, ..., z=25) et a et b sont des nombres entiers choisis comme clé de chiffrement. Le chiffrement par décalage étant un cas particulier où a = 1 et b = le décalage.

Exemple

Soit la clé de chiffrement (publique) : $f(x) = 17x+22$ pour chiffrer le message `test`, on convertit les lettres en nombres (t=19, e=4, s=18, on se souviendra que la position de a dans l'alphabet est de 0) auxquels on applique la transformation :

```
t > 17x19 + 22 = 345, int(345/26) = 13, comme 345 - 13x26 = 345 - 338 =  7 > h
e > 17x 4 + 22 =  90, int( 90/26) =  3, comme  90 -  3x26 =  90 -  78 = 12 > m
s > 17x18 + 22 =  32, int(328/26) = 12, comme 328 - 12x26 = 328 - 312 = 16 > q
t > 17x19 + 22 = 345, int(345/26) = 13, comme 345 - 13x26 = 345 - 338 =  7 > h
```

Le message chiffré est donc `hmqh`.
Nous allons maintenant inversé le procédé avec une clé de déchiffrement (privée et différente) : $g(x) = 23x + 14$. On convertit d'abord les lettres en nombres (h=7, m=12, q=16) sur lesquels on applique, ensuite, la transformation :

```
h > 23x 7 + 14 = 175, int(175/26) =  6, comme 175 -  6x26 = 175 - 156 = 19 > t
m > 23x12 + 14 = 290, int(290/26) = 11, comme 290 - 11x26 = 290 - 286 =  4 > e
q > 23x16 + 14 = 382, int(382/26) = 14, comme 382 - 14x26 = 382 - 364 = 18 > s
h > 23x 7 + 14 = 175, int(175/26) =  6, comme 175 -  6x26 = 175 - 156 = 19 > t
```

Le code déchiffré est donc bien `test`.

Fonction de chiffrement

```
def affine_encrypt(msg, a, b):
    return ''.join(
       chr((x * a + b) % 26 + 97) for y in msg if (x := ord(y) - 97) >= 0)

def affine_decrypt(msg, a_inv, b):
    return ''.join(
       chr(((x - b) * a_inv) % 26 + 97) for y in msg if (x := ord(y) - 97) >= 0)

a, b, a_inv = 17, 22, 23 # Cles de chiffrement
                         # sans utiliser b_inv, qui fausserait le decryptage
r = input("Crypter (c) ou Decrypter (d) le message? ").lower()
m = input("Message (en minuscules)? ").strip()

print((affine_encrypt(m, a, b) if r == "c" else "") + \
      (affine_decrypt(m, a_inv, b) if r == "d" else ""))
```

Pour bien comprendre la cryptgraphie, il faut saisir que, si on vous demande combien font 5 x 7, vous répondez aisément, 35. Qu'à l'inverse, si on vous demande de factoriser 35 vous pouvez aussi assez aisément répondre 5 × 7. Ces deux questions ne sont pas cependant pas du même ordre de difficulté ! Si on vous demandais de factoriser 1591, vous allez devoir effectuer plusieurs tentatives pour résoudre le problème, alors que si l'on vous avais directement dit d'effectuer le calcul de 37 x 43, cela n'aurait a priori pas posé de problème.
En d'autres termes, étant donnée une fonction f, il est assez facile, connaissant x, de calculer f(x). Mais l'inverse, n'est pas si facile (si la fonction n'est pas bijective)... en effet, connaitre

le résultat de f(x) ne permet pas de connaitre aisément x (parfois même avec des supercalculateurs) ! A noter aussi que le chiffrement symétrique est beaucoup plus rapide / aisé que le chiffrement asymétrique, mais a l'inconvénient de nécessiter de partager une clé sensée être privée et secrète...

7.5.4 Chiffrement par hachage

Exercices p. 172, aide-mémoire p. 177

Le hachage est un outil essentiel en cryptographie. Ces fonctions permettent, par des procédés complexes que nous n'aborderont pas ici, de transformer des données de taille quelconque en une grandeur de longueur fixe (appelée hash ou empreinte), facile à stocker (par exemple dans une base de donnée). Ces fonctions sont extrémement difficile à inverser : il est en effet quasiment impossible de retrouver les données originales à partir du hash.

Exemple : stockage des mots de passe

Au lieu de stocker un mot de passe directement, on stockera son hash. Par exemple `secret123` donnera, à l'aide la la fonction SHA-256...

```
import hashlib

pwd = "secret123"
ash = hashlib.sha256(pwd.encode()).hexdigest()

print("Mdp :", pwd)
print("Hash:", ash)
              # 3e23e8160039594a33894f6564e1b1348bbd7a7848b9cce3b5d5f30795f4787d
```

Illustration : on a trouvé des mots de passe, sur youtube.com/watch?v=_1ONcmFUOxE

7.5.5 À retenir

— Définitions de la cryptographie et du vocabulaire associé

— Utiliser de manière débranchée les chiffrements de César et Vigenère

— À l'aide de Python, utiliser le chiffrement de César

— Attaquer un mot de passe par "force brute" (TP05 p. 172)

Annexe A

EXERCICES

Pour le rendu d'exercices, se référer aux instructions données en notes de bas de page, p. 57.

A.1 Les systèmes informatiques

A.1.1 Softwares (logiciels)

Soft_TP01. Les systèmes d'exploitation Théorie p. 2

ex1 QCM

i) Quel est le rôle principal d'un système d'exploitation ?

- ◯ Créer des présentations animées
- ◯ Permettre la communication entre matériel et logiciels
- ◯ Remplacer le processeur machine
- ◯ À jouer aux jeux vidéo
- ◯ Stocker les fichiers dans le cloud

ii) Lequel des éléments suivants n'est **pas** un système d'exploitation ?

- ◯ Linux
- ◯ macOS
- ◯ Android
- ◯ Windows
- ◯ Word

iii) Quel système d'exploitation est basé sur le noyau Unix ?

- ◯ Windows
- ◯ macOS
- ◯ MS-DOS
- ◯ ChromeOS

iv) Parmi ces affirmations, lesquels sont corrects concernant Linux ?

- ☐ C'est un logiciel propriétaire
- ☐ Il fonctionne sur smartphone
- ☐ C'est un système opensource et souvent gratuit
- ☐ Comme pour Windows, il n'existe qu'un seul fabricant produit
- ☐ Il est principalement utilisé pour les jeux vidéo

ex2 Rôle d'un système d'exploitation

Expliquer, en quelques mots, pourquoi un ordinateur ne peut pas fonctionner sans système d'exploitation. Donner deux exemples de tâches que le système exécute automatiquement pour l'utilisateur.

.....

.....

.....

.....

.....

.....

ex3 **Comparaison** Compléter le tableau suivant en indiquant pour chaque système d'exploitation (SE / OS) un avantage ou une caractéristique notable.

	Caractéristique / Avantage
Windows	
macOS	
Linux	
Android	
iOS	

ex4 **Étude de cas** Un ami souhaite acheter un nouvel ordinateur portable. Il hésite entre un modèle sous Windows 11, un MacBook (macOS) et un ordinateur sous Linux Ubuntu.

i) Indiquer deux critères qu'il doit considérer pour choisir son système d'exploitation

ii) Donner un exemple de situation où a) Linux et b) Windows est préférable

.....

.....

.....

.....

.....

.....

.....

.....

Soft_TP02. Formats de fichiers Corrigé p. 203, théorie p. 6

ex1 QCM

i) Quel format est le plus adapté pour un document qui doit être imprimé et conservé sans modification ?

◯ .docx ◯ .jpg ◯ .pdf ◯ .txt ◯ .xls

ii) Quels formats sont utilisés pour stocker des images compressées ?

☐ .bmp ☐ .gif ☐ .jpg ☐ .png ☐ .txt

iii) Quel format est *non compressé* et génère donc des fichiers lourds ?

◯ .bmp ◯ .docx ◯ .jpg ◯ .mp3 ◯ .png

iv) Parmi les formats ci-dessous, lesquels sont utilisés pour des présentations assistées par ordinateur ?

☐ .docx ☐ .jpg ☐ .odp ☐ .pdf ☐ .pptx

ex2 Type de fichier Indiquer, pour chaque extension, le type de fichier et donner deux logiciels qui permet de l'ouvrir.

Extension	**Type de fichier**	**Logiciels utilisables**
.docx		
.png		
.mp4		
.xlsx		
.pdf		

ex3 Étude de cas Un étudiant doit envoyer un rapport de 20 pages avec images par courriel. Il hésite entre les formats suivants : .docx, .pdf, .jpeg.

i) Expliquer quels formats sont les plus adaptés et pourquoi

ii) Indiquer les inconvénients d'envoyer le rapport en .jpeg

.....

.....

.....

.....

.....

.....

Soft_TP03. Travailler avec différents formats

ex1 Fichier texte

Comparer la taille, la qualité et la compatibilité des différents formats de fichiers de texte... `.docx`, `.pdf`, `.txt` et `.odt`.

i) Ouvrir MS Word

ii) Copier/coller du web, un texte d'environ une page, incluant deux ou trois images et différentes mises en forme (gras, listes à puces, etc.)

iii) Enregistrer ce document en quatre formats : `.docx`, `.pdf`, `.txt`, `.odt`

iv) Dans l'explorateur de fichier, comparer la taille des fichiers

v) Ouvrir chaque fichier avec différents programmes (Adobe Acrobat Reader, MS Word, LibreOffice Writer, Bloc-notes, mais aussi Google Docs) pour vérifier la compatibilité et le fonctionnement de ces outils

Production attendue : Quatre fichiers et une brève note décrivant les différences observées en termes de qualité, de taille et de mise en forme entre chaque format.

ex2 Travailler avec des images

Comprendre les avantages et inconvénients de différents formats d'image (`.jpg`, `.png`, `.gif`).

i) Ouvrir un logiciel de traitement d'image (MS Word puis GIMP)

ii) Importer ou créer une image contenant des éléments avec transparence

iii) Sauvegarder / exporter l'image en trois formats : `.jpg`, `.png`, `.gif`

iv) Comparer la taille et la qualité de ces fichiers

Production attendue : Une image par format et une brève note décrivant les différences constatées en termes de qualité et de taille.

Soft_TP04. Rendu de fichiers

ex1 Organiser son dossier personnel et rendre les fichiers selon les consignes suivante...

i) Ouvrir l'explorateur Windows

ii) Accéder à votre dossier personnel

iii) À la racine du dossier personnel :

— Si le dossier `Informatique` n'existe pas, le créer

— Si le dossier `Informatique` existe déjà, déplacer tous les dossiers et fichiers existants (sauf les dossiers Windows, comme `Documents` ou `Audio`) dans un dossier nommé `Z_Anciens`

iv) Dans le dossier `Informatique`, créer les cinq sous-dossiers suivants :

- A_Donnees
- B_Documents
- C_Programmation
- D_Reseaux
- E_MesProjets

v) Ouvrir le dossier `\\Echanges\Distribution\PrenomNOM\Soft_TP04ex1` (remplacez la partie `PrenomNOM` par le nom de votre enseignant·e / formateurice)

vi) Dans une autre fenêtre, ouvrir son dossier personnel (sous PC)

vii) Copier le fichier `Ex_Rendu.docx` et le coller dans son dossier `A_Donnees`

viii) Renommer ce fichier en `Prénom_Ex_Rendu.docx` (en remplaçant `Prénom`)

ix) Ouvrir le fichier avec Microsoft Word, modifier ce qui est demandé, enregistrer, puis fermer le fichier (`Ctrl + W`)

x) Rendre le fichier compressé (`zip`) en utilisant l'outil **Rendu** et sur la plateforme **Moodle**

xi) Pour l'exercice spécifique de rendu sur Moodle, effectuer les trois dernières opérations ci-dessus avec le fichier `Ex_Moodle.docx`

ex2 Manipulations de dossiers

i) Télécharger l'archive `MaisonQuartier.zip`, la dézipper, renommer le fichier `Prénom_Mais` et le déplacer dans son répertoir personel

ii) Supprimer les fichiers suivants :

- `Photo bonhomme hiver.bmp`
- `Cours improvisation.odt`

iii) À l'intérieur du dossier `MaisonQuartier`, créer l'arborescence de dossiers indiquée dans le schéma suivant :

```
MaisonQuartier
│
├─Administration
│  ├─Comptabilité
│  └─Employés
├─Cours_Ateliers
│  ├─Multi-sports
│  └─Poterie
└─Manifestations
   ├─Fête de quartier
   └─Halloween
```

iv) Trier tous les fichiers dans les bons dossiers d'après leur nom et les indications ci-après :
   — Les fichiers `Compta fête de quartier.xlsx` et `Compta Halloween.ods` doivent figurer (par un lien) dans le dossier `Comptabilité` **ET** dans les dossiers des `manifestations` correspondantes
   — Créer un nouveau fichier Word nommé `tour de potier.docx` et le placer dans `Cours_Ateliers/Poterie`

v) Compresser le dossier (le zipper) en lui donnant un nom adéquate, puis, selon les insctructions de votre votre enseignant·e / formateurice, le rendre !

Soft_TP05. Recherche de dossiers et fichiers

ex1 Recherche simple

Retrouver rapidement un fichier précis dans son dossier personnel.

i) Si pas encore effectué, copier le répertoire `Soft_TP04ex1` sur le serveur de distribution

ii) Ouvrir l'Explorateur Windows, accéder à son dossier personnel et rechercher un fichier nommé `Photo_Vacances.jpg`

iii) Noter le chemin complet où se trouve le fichier

.....

ex2 Recherche par extension

Dans le dossier `Informatique`, utiliser la zone de recherche de Windows pour afficher uniquement les fichiers texte (`.txt`). Noter le nombre de fichiers trouvés et leur emplacement.

.....

.....

.....

.....

ex3 Recherche avancée

Toujours dans le dossier `Informatique`, rechercher tous les fichiers dont le nom commence par `Compta` (par exemple `Compta_2025.xlsx`, `Compta_budget.docx`) en utilisant le caractère générique `*`. Noter les noms exacts des fichiers trouvés et leurs extensions.

.....

.....

.....

A.1.2 Hardware (la machine)

Hard_TP01. Théorie hardware Théorie p. 11

ex1 Compréhension du système binaire

Traduire les nombres 5, 12 et 23 en binaire (base 2), puis indiquer pourquoi l'ordinateur ne peut pas utiliser le système décimal pour ses calculs.

.....

.....

.....

ex2 Catégorisation des périphériques

Classer les périphériques suivants dans la bonne catégorie (entrée, sortie, mixte) :

— Casque des salles informatiques .....

— Clavier .....

— Disque dur externe .....

— Écran tactile .....

— Imprimante .....

— Microphone .....

ex3 Mémoire vive et mémoire morte

Expliquer en quelques mots la différence entre mémoire vive (RAM) et mémoire morte (ROM). Indiquer un exemple d'utilisation pour chacune.

.....

.....

.....

ex4 Évolution du matériel

Définir puis expliquer, en quelques mots, comment ont évolué les éléments suivants au cours des 20 dernières années. Donner un exemple concret pour chacun (par expl. fréquence du CPU, taille, caractéristiques techniques).

— Puissance des processeurs .....

.....

— Capacité de stockage .....

.....

— Mémoire vive .....

.....

.....

— Périphériques d'entrée/sortie .....

.....

.....

ex5 QCM

i) Quel est le rôle principal du processeur (CPU)?

- ◯ Afficher les images à l'écran
- ◯ Alimenter l'ordinateur
- ◯ Exécuter les instructions programmes
- ◯ Sauvegarder les données dans le cloud (automatiquement)
- ◯ Stocker des fichiers

ii) Quel type de mémoire est volatile (s'efface à l'arrêt de l'ordinateur)?

◯ DiDu ◯ USB ◯ RAM ◯ ROM ◯ SSD

iii) Quels sont des supports de stockage persistants?

- ☐ Clé USB
- ☐ Mémoire cache
- ☐ RAM
- ☐ HDD
- ☐ SSD

iv) Quel composant est souvent intégré à la carte mère?

- ◯ Carte réseau
- ◯ Disque externe
- ◯ Imprimante
- ◯ Clavier
- ◯ Écran

v) Parmi ces unités, laquelle mesure la fréquence d'un processeur?

◯ GHz ◯ Go ◯ Mb/s ◯ Mo ◯ ms

Hard_TP02. Diagnostic et choix matériel Corrigé p. 204, théorie p. 11

Identifier les composants principaux d'un ordinateur, comprendre leur rôle et analyser leur adéquation avec les besoins (travaillez sur un poste à l'école ou à domicile).

ex1 Ouvrir le gestionnaire des tâches (ou la config. système sous Linux/macOS) et noter :

— Le nom du processeur (CPU) .....

— La quantité de mémoire vive (RAM) installée .....

— L'espace libre sur le disque dur local et sur les disques réseaux .....

.....

ex2 Identifier, depuis l'explorateur de fichiers ou via les préférences système, le type de carte graphique (GPU) utilisée .....

.....

ex3 Relever au moins deux périphériques connectés (externe ou intégré) et préciser leur rôle

.....

.....

.....

ex4 Rédiger une courte synthèse (5–6 lignes) présentant les forces et limites de la configuration observée .....

.....

.....

.....

.....

.....

.....

ex5 Imaginer, sur la base de ce qui précède...

i) La configuration d'un ordinateur idéal pour les études, en justifiant ses choix (processeur, RAM, stockage, carte graphique, périphériques) .....

.....

.....

.....

.....

.....

.....

ii) Fournir un lien (avec les coordonnées du vendeur) pour facillement effectuer l'achat

.....

.....

A.1.3 Networks (réseaux)

Théorie p. 17

Net_TP01. Réseau local

ex1 Ouvrir l'invite de commande (`cmd`) et afficher la configuration réseau de votre machine (commande `ipconfig` sous Windows ou `ifconfig` sous Linux/macOS). Noter son adresse réseau par défault et expliquer les différences entre une adresse `IP` qui commence par 10.134..., 127.0... et une autre qui commence par 255.255...?

ex2 Identifier et compléter vos connaissances sur les composants des systèmes informatiques (CPU, etc.), le rôle des switch, routeurs, câbles et points d'accès WiFi

ex3 Sur la base des élément suivants, rédiger un schéma (page 132) montrant la connexion de votre poste au réseau local et à Internet.

Nom de l'hôte :
Système d'exploitation :
Organisation ou propriétaire du réseau :
Date de la dernière installation système :
Nom et type du processeur utilisé :
Options régionales appliquées :
Mémoire totale du disque dur (en Go) :
Domaine (DNS) :

Net_TP02. Internet et noms de domaine

Vérifier le fonctionnement des protocoles de communication réseaux.

ex1 Ouvrir l'invite de commande (`cmd`) et exécuter un `ping` vers l'adresse `IP` trouvée précédement. Noter le résultat.

ex2 Toujours dans l'invite de commande, exécuter un `ping`, mais cette fois vers un site externe (par expl. `www.eduge.ch`). Noter et expliquer ce qui se passe.

ex3 Rappeler, en une phrase, le rôle d'un DNS dans l'accès aux sites web. Puis utiliser la commande `nslookup` avec votre adresse `IP`. Qu'y a-t-il de nouveau?

ex4 Demander à un·e camarade son adresse `IP`. Exécuter un `ping` vers cette adresse et observer le résultat. Quelle différence / similitude y'a-t-il avec l'exercice précédent?

Net_TP03. Connectivité non-sécurisée et sécurisée

ex1 Connexion SSH (si OpenSSH installé sur votre poste...)

i) Que signifie SSH ? .....

ii) Depuis l'invite de commande, taper `ssh prenom@adresseIP` (remplacez `prenom` par l'utilisateur d'un·e camarade et `adresseIP` par son `IP`) .....
.....
.....

iii) Observer le message du système (demande de mot de passe, avertissement de sécurité, etc) .....
.....
.....

iv) À quoi sert SSH et en quoi est-elle une communicaton sûre (à discuter éventuellement en groupe) ? .....
.....
.....

ex2 Connexion à un serveur FTP

i) Que signifie FTP ? .....

ii) Depuis l'invite de commande, taper `ftp ftp.dlptest.com`

iii) Se connecter avec l'utilisateur `dlpuser` et
le mot de passe donné sur le site `dlptest.com`

iv) Utiliser les commandes `ls` (lister les fichiers)

v) Que peut-on constater et quelle différence pourait-il exister entre un serveur FTP et SFTP (basé sur SSH) ? .....
.....
.....
.....

ex3 QCM

i) Quel appareil permet de connecter plusieurs ordinateurs et de filtrer les informations reçues / envoyées ?

◯ Hub ◯ Routeur ◯ Switch
◯ Pare-feu ◯ Répéteur

ii) Lequel de ces protocoles est utilisé pour transférer des pages web ?

◯ DHCP ◯ DNS ◯ FTP ◯ HTTP ◯ SMTP

iii) Quelle est la principale différence entre un réseau câblé et un réseau sans fil ?

- ◯ Le câblé est plus rapide (à capacité égale)
- ◯ Le câblé utilise des câbles, le sans fil utilise des ondes radio
- ◯ Le sans fil est gratuit
- ◯ Le sans fil ne permet pas (encore) d'accéder à Internet
- ◯ Le sans fil ne peut pas connecter plus de 3 appareils

iv) Parmi ces affirmations, lesquelles concernent un réseau LAN ?

- ☐ Peut être câblé ou WiFi
- ☐ Réseau local (maison, école, bureau)
- ☐ Souvent en configuration étoile
- ☐ Utilisé pour relier des continents
- ☐ Synonyme de WAN

v) Quel service est assuré par le protocole DNS ?

- ◯ Chiffrer la connexion
- ◯ Fournir une adresse MAC
- ◯ Gérer la bande passante
- ◯ Traduire un nom de domaine en adresse IP
- ◯ Transférer des mails

A.2 Notions d'algorithmique

Pour le rendu d'exercices, se référer aux instructions données en notes de bas de page, p. 57.

A.2.1 Introduction à l'algorithmique, étape par étape

Algo_TP01. Demander, enregistrer et afficher en suivant un algorithmes Théorie p. 26

Commence par définir, la question ou le problème posé ; fournit ensuite, les trois ou quatre éléments qui définissent les spécifications tel que définies dans le chapitre de théorie spécifié ci-dessus (à droite).

ex1 Écouter la réponse

i) Demander à l'utilisateur d'écrire une réponse à "Alors, tu dis quoi ?␣"

ii) Stocker la valeur de la reponse dans la variable reponse

iii) Afficher la valeur de la variable reponse

ex2 Un bonjour personnalisé

i) Demander à l'utilisateur son prénom en affichant "Quel est ton prénom ?␣"

ii) Stocker la valeur de la reponse dans une variable

iii) Demander le nom de famille (stocké dans une autre variable)

iv) Afficher une phrase de type "Bonjour Prénom NOM"

ex3 On vérifie !

i) Vérifier la réponse
- Demander un nombre entre 1 et 6
- Enregistrer la réponse
- Si la réponse > 6 ou < 1, reposer la question
- Si la réponse est dans l'intervalle, remercier l'utilisateur

ii) Avec une interface améliorée
- idem que précédement, mais ...
- Si la réponse < 1, demander un chiffre plus grand et retourner au premier point
- Si la réponse > 6, demander un chiffre plus petit et retourner au premier point
- Sinon, remercier l'utilisateur et lui rappeler le nombre fournit

ex4 On calcule !

i) Une fois, hein...
- Poser la question "Donne-moi s'il te plait un nombre"
- Enregistrer la réponse
- Ajouter un au nombre donné
- Afficher la somme

ii) Demander une fois une chose, mais pas plusieurs fois la même chose...

— Poser la question "Donne-moi s'il te plait un nombre"

— Enregistrer la réponse

— Prendre la partie entière n du nombre donné

— Pour i de 0 à 10, répéter : ajouter i à n et afficher la somme

Algo_TP02. Recherches de solutions simples Théorie p. 30

Réprésente les processus suivants sous forme d'ordinogramme, puis, pour chaque exercice, écrit l'algorithme linéaire correspondant.

ex1 Demander successivement 10 nombres à un utilisateur, et lui dire ensuite quel était le plus grand nombre parmi les réponses (exemple : réponse 1 = 12, réponse 2 = 14, etc. le plus grand nombre étant le 14).

ex2 Afficher de surcroît en quelle position ce nombre a été saisie ce nombre (ci dessus en deuxième position)

ex3 Cela change-t-il quelque chose si on ne connaît pas d'avance combien l'utilisateur souhaite saisir de nombres ? Détailler votre réponse.

A.2.2 Algorithme et démarche pour la résolution d'un problème

Algo_TP03. Comment afficher un sapin de Noël ? Théorie p. 27, 30 et suivantes

Représente les processus sous forme d'ordinogramme puis d'algorithme linéaire afin d'indiquer comment dessiner un arbre de Noël – d'une grandeur définie dans un premier temps, puis en demandant ensuite la taille du sapin à un utilisateur (5 pour le sapin ci-dessous). Vérifier bien (éventuellement avec un ou une camarade) que vos processus et algorithmes fonctionnent.

```
    *
   ***
  *****
 *******
*********
    |
```

A.3 Concepts généraux

A.3.1 Acquérir des bases solides

Pour le deux TP suivants, rédiger d'un part l'algorithme linéaire (si vu en cours) et, d'autre part, le code à exécuter. Pour la reddition, se référer aux instructions p. 57.

Bases_TP01. Afficher des valeurs et des variables Théorie p. 41, aide-mémoire p. 175

ex1 Écrire un programme qui affiche "Bonjour !" à l'écran.

ex2 2 variables ? Écrire un programme qui :

i) donne la valeur 5 à la variable a

ii) donne la valeur 20 + 45 à b

iii) affiche la valeur de la variable a

iv) affiche la valeur de la variable b

ex3 Multiplier ? Écrire un programme qui :

i) donne la valeur 10 à la variable nombre

ii) donne la valeur 3 à la variable facteur

iii) affiche le résultat de l'opération nombre fois facteur

Bases_TP02. Sans nécessairment passer par un algorithme, demander, enregistrer et afficher à l'aide d'un script Python Théorie p. 41, aide-mémoire p. 175

ex1 Pas à pas...

i) Demande à l'utilisateur d'écrire quelque chose en affichant "Alors, tu dis quoi ?␣"

ii) Enregistre ce qu'a écrit l'utilisateur dans la reponse (dans une variable)

iii) Affiche la valeur de la variable reponse

ex2 Un bonjour personnalisé

i) Ecrire un programme qui demande à l'utilisateur son prénom en affichant "Quel est ton prénom ?␣", puis affiche "Bonjour " suivi de la valeur de la reponse

ii) Modifie ton programme pour qu'il demande le nom de famille (stocké dans une autre variable), et qu'il affiche une phrase de type "Bonjour Prénom NOM".

ex3 On vérifie et on calcule !

i) Ecrire un programme qui :

— Demande à l'utilisateur un premier nombre (en affichant "Premier nombre :␣")

— Demande à l'utilisateur un deuxième nombre (en affichant "Deuxième nombre :␣")

— Vérifie que les réponses données sont des nombres

— Affiche "En additionnant ces deux nombres j'obtiens :␣" suivi de la réponse du calcul (sur la même ligne)

ii) Modifie ton programme pour qu'il affiche aussi le résultat de la soustraction, puis de la multiplication et enfin de la division des deux nombres.

A.3.2 Bases et exécutions conditionnels...

Bases_TP03. Jeux de hasard Théorie p. 68

Soit le code ci-dessous. Sur cette base, compléter le code pour réaliser les exercices suivant.

```
import random                    # Module hasard
hasard = random.randint(0,1)     # Tire un nombre au hasard (0 ou 1)
```

ex1 Pile ou face ?

i) Dire si le tirage donne pile (0) ou face (hasard = 1)

ii) Demander à l'utilisateur sur quoi il veut parier : "PILE (écrire 0) ou FACE (écrire 1)?" et enregistrer sa reponse

iii) Annoncer au joueur s'il gagné ou non (si réponse = hasard –> "Gagné")

ex2 Jeu de dé

i) Demander à l'utilisateur de choisir un nombre de 1 à 6 et enregistrer sa réponse

ii) Tirer un nombre, au hasard, entre 1 et 6 et enregistrer le tirage

iii) Afficher le résultat du tirage (sur quel nombre le dé est tombé) et dire si l'utilisateur a gagné ou perdu

Bases_TP04. À condition que ça fonctionne... Théorie p 50, aide-mémoire p. 175

Entrées : à partir de code ci-dessous, suivre les consignes et corriger les erreurs une à une

Sorties : tableau de toutes les erreurs
Pré-condition : interpréteur type Thonny
Post-condition : code fonctionel

```
mon age = input(Quel age as-tu ?)
if age <= 18:
    print ("Tu es majeur-e")
else:
print("Tu es mineur-e")
prin("Fin du questionnaire")
```

iLi - TP04

i) Soit le code ci-dessus...

ii) Préparer un tableau avec le numéro de l'erreur corigée, le type d'erreur et la solution mise en place

iii) Tant qu'il y a une erreur :

— Lire le message d'erreur

— Noter le type d'erreur (NameError, SyntaxError, IndentationError, TypeError)

— Corriger l'erreur

— Remplir le tableau

— Relancer le code

Bases_TP05. Exercices complémentaires

ex1 Assignation et affichage `print`

i) À l'allumage d'un ordinateur, l'écran reste noir... d'où peut provenir le problème ?

ii) En informatique, int est l'abréviation pour.. international, entier, interne, intelligent ?

iii) Pourquoi ce programme ne donne-t-il pas le nombre 24 ?

```
nb = 24
print("nb")
```

iv) Pourquoi ce programme ne donne-t-il pas le nombre 24 ?

```
nb = "17"
print(nombre+4)
```

v) Quelles sont les variables utilisées dans le programme suivant ?

```
age = 15
delai = 18 -- age
print("Vous serez majeur dans", delai, "ans")
```

vi) Combien de variables sont utilisées dans le programme suivant et que vaut la variable `a` une foix cette séquence d'opérations terminée ?

```
a = 8, b = 3
a = a - 4
b = 2 * b
a = a + b
```

vii) Que fais l'algorithme suivant ?

```
monAge = 25
print("J'ai", monAge, " !")
```

viii) Écrire un programme qui demande à l'utilisateur, son prénom, son nom et son âge et qui revoit "Bonjour je m'appelle votrePrenom votreNom, j'ai votreAge ans".

ex2 Affichage `print` avancé

i) Modifier les insctructions 2 et 3 d'un des programme précédent pour qu'il affiche ton âge en 2030

ii) Corriger les programmes suivants :

— Programme a

```
monAge = tonAge + 1
tonAge = 15
print(monAge)
```

— Programme b... le nombre de frere (f) et soeur (s) n'est pas correct !

```
f = 2
s = 1
print("Nombre freres et soeur:", f + f)
```

— Programme c

```
 age=input("Quel est ton age? ")
print("L'an prochain, tu auras ",age+1, " ans")
```

— Programme d

```
x = input("Donner un entier ")
y = input("Donner un autre entier ")
print("la somme de", x, "et", y, "vaut", x + y)
```

— Programme e

```
 print("je suis sense calculer la moyenne de deux notes...")
moyenne = 5 + 4 / 2
print(moyenne)
```

iii) Qu'affiche le programme suivant ? Le fonctionnement du programme te surprend-t-il ? Justifie ta réponse...

```
contenance = 60
print(contenance)
contenance = contenance - 15
print("Il reste:", contenance, "litres")
```

ex3 Lecture calvier `input` et calculs avancés

i) Calculer la moyenne de deux notes rentrées par un utilisateur

ii) Un boulanger désire un programme qui demande à l'utilisateur le nombre de croissants et qui affiche le prix à payer (sachant qu'un croissant coute 1 franc 80).

iii) Convertir une température donnée par un utilisateur en degré Celsius et afficher l'équivalent en degré Fahrenheit sachant que la tF = 1.8 x tC + 32

iv) Proposer un programme qui permet d'échanger les valeurs stockées dans les variables a et de b, demandées à un utilisateur. Exemple : un utilisateur donne les variable a = 5 et b = 3 ; le programme affiche : a = 3 et b = 5 !

A.3.3 Containers, contrôles et fonctions

Coco_TP01. Être for ! Théorie p. 51, aide-mémoire p. 175

ex1 Affiche trois fois le mot "Youpi" en utilisant seulement deux lignes de code !

ex2 Encore plus for... Affiche :

i) Tous les nombres de 1 à 100 avec le même nombre de lignes de code

ii) Avec une ligne de code supplémentaire, demande cette fois-ci à un utilisateur de fixer la limite

ex3 *Dessine un motif en demandant à l'utilisateur* :

i) Un motif à répéter (comme par exemple `_.-'-`)

ii) Le nombre de répétion horisontales et verticales

Afficher ensuite un rectangle fait du motif dans les dimensions demandées [1]

```
# Lancement du programme, exemple
Ton motif? _.-'-
Combien de fois dois-je repeter ce motif? 5
_.-'-_.-'-_.-'-_.-'-_.-'-
_.-'-_.-'-_.-'-_.-'-_.-'-
_.-'-_.-'-_.-'-_.-'-_.-'-
_.-'-_.-'-_.-'-_.-'-_.-'-
_.-'-_.-'-_.-'-_.-'-_.-'-
```

Coco_TP02. On tourne ! Théorie p. 51, aide-mémoire p. 176

Souvent, lorsqu'on demande à un utilisateur de choisir entre plusieurs propositions, on doit s'assurer que la réponse est parmi les options que l'on attends. Pour cela, la plupart du temps, on utilise la fonction `while`.

ex1 Écrit un programme qui demande à l'utilisateur : `Qu'est-ce qu'on dit?`. La question sera répétée tant que l'utilisateur n'aura pas répondu soit `merci`, soit `Merci`. Le programme ajoutera ensuite `Aahh... quand même!`.

ex2 Demande à l'utilisateur :

i) S'il veut tourner à `droite (d)` ou `gauche (g)` et le demander tant que le choix n'est pas gauche

ii) Afficher `Tu as tourné à...` suivi du choix effectué

iii) Répéter la question tant que le choix n'est ni `d`, ni `g`

ex3 Décompte les sauts de deux-cents moutons à l'aide d'une variable `compteur` et d'une boucle `while`

```
Saut du mouton 1, mouton 2, etc...
```

1. Multiplier la chaîne de caractère par le nombre donné, exemple 5 * motif

A.3.4 Création d'une histoire interactive

De quoi s'agit-il ?

Créer une histoire interactive, sous Python et juste avec du texte, où l'utilisateur agit en choisissant une action à effectuer parmi des choix donnés par la machine (vous choisissez le lieu de l'histoire, le scénario, la manière de gagner ou de perdre).
/!\ bien suivre les contraintes et consignes.

Contraintes (Pré-conditions)

— L'histoire est interactive, mais présentée de manière linéaire
(les étapes se suivent, peu importe les choix du joueur)

— Il doit y avoir du texte qui explique clairement où est le joueur et ce que le joueur a le droit de faire ou non

— Les choix doivent influencer le résultat final
(il peut y avoir du hasard, mais pas seulement)

— Utilisez au moins une variable que certains choix pourront modifier
(par exemple des points vie, un nombre de pièces, présence ou absence d'une clé, etc.)

```
nbEpee = 0

print("Tu es a Paris")
print("Tu trouves une epee par terre.")

choix = input("Tu la Ramasses (R) ou Pas (P) ? ")

if choix == "R":
    print("Tu as ramasse l'epee.")
    nbEpee = nbEpee + 1
else:
    print("Tu ne la ramasses pas.")

print("Tu as maintenant", nbEpee, "epee")
```

Régle générale A : être capable d'expliquer le code produit fait partie intégrante de l'évaluation ; en particulier, si un·e élève plagie du code (c'est à dire sans mentionner ses sources), son évaluation ne pourra pas être réalisée (!). L'élève qui, en outre, ne peut pas expliquer son fonctionnement (du code), n'obtiendra au mieux et en conséquence, pas plus que la moyenne. À noter par ailleurs, que si tous le groupe sait expliquer le fonctionnement du code, un bonus de 10% pourra être accordé.

Partie 1 : Définitions (première semaine, à rendre au tout début de la deuxième)

Créer un document de définition du jeu qui contient (sous traitement de texte, tableur ou logiciele de présentation par ordinateur, selon consignes orales données par l'enseignant) :

— Le nom et la classe des deux personnes qui travaillent sur le projet

— Le titre de l'histoire ou du jeu

— Dans quelle ambiance l'histoire se passe (présent, futuriste, médiéval, farwest, mythologique, abstrait, etc)

— Un scénario qui contiendra les trois éléments suivants :
Le fils rouge de l'histoire
> Scène 1 : le joueur est à tel endroit. Il peut soit faire ça, soit faire ça.
> Scène 2 : le joueur est arrivé à tel endroit. Il peut soit faire ça, soit faire ça, soit faire ça. > Scène finale : le joueur est arrivé à tel endroit. Il peut soit faire ça, soit faire ça.
Conditions de victoire (Post-conditions)
Le joueur gagne, par exemple, soit s'il a une clé et qu'il choisit d'ouvrir le coffre, soit s'il a plus de 50 points de vie et choisit de porter le coffre jusqu'au chef. Dans les autres cas il perd, le joueur doit savoir s'il a gagné ou s'il a perdu
Variables du jeu (exemple)
> Vie (varVie = 100 # au début)
> Clé (varCle = 0 # au début)

Critères d'évaluation	Échelle de 0 à 3
Conditions de victoire clairement définies et compréhensibles	○ ○ ○ ○
Document clair et organisé (bien structuré, avec titres, sous-titres et listes)	○ ○ ○ ○
— " — complet (contient tous les éléments demandés)	○ ○ ○ ○
— " — original (histoire attrayante, ambiance et scénarios démontrent de la créativité)	○ ○ ○ ○
Investissement personnel et travail de groupe (x2)	○ ○ ○ ○ ○ ○ ○ ○
Variables du jeu pertinentes et bien expliquées	○ ○ ○ ○
Variété des choix proposés	○ ○ ○ ○

TABLE A.1 – Grille d'évaluation de la première semaine

Partie 2 : État d'avancement (deuxième semaine)

À la fin de la deuxième semaine, joindre le fichier de votre jeu/histoire, dans sa version du moment. Un message d'accompagnement sous forme de vidéo (deux minutes maximum) décrit ce qui a été effectué et ce qu'il reste à faire, notamment des problèmes rencontrés ou questions en suspens.

Critères d'évaluation	Échelle de 0 à 3
Communication vidéo agréable à suivre (discours structuré et fluide) .	○ ○ ○ ○
Consignes et fonctionnalités attendues mentionnées	○ ○ ○ ○
Documentation annexe visuellement agréable, bien organisée et structurée (grammaire et orthographe inclues)	○ ○ ○ ○
Identification claire de ce qui reste à faire	○ ○ ○ ○
Investissement personnel et travail de groupe (x2)	○ ○ ○ ○ ○ ○ ○ ○
Lisibilité du code (mise en plage claire et commentaires pertinents) ..	○ ○ ○ ○
Problèmes rencontrés et réalisations faciles à comprendre	○ ○ ○ ○
Temps de la vidéo inférieur ou égale à deux minutes	○ ○ ○ ○

TABLE A.2 – Grille d'évaluation de la partie intermédiaire (deuxième semaine)

Partie 3 : Déposer le script (à rendre à la fin de la troisième et dernière semaine)

Joindre le fichier de votre jeu/histoire, dans sa version finale.

Critères d'évaluation	Échelle de 0 à 3
Absence d'erreurs lors de l'exécution (x2)	○ ○ ○ ○ ○ ○ ○ ○
Fluidité et cohérence de l'interaction	○ ○ ○ ○
Investissement personnel et travail de groupe (x2)	○ ○ ○ ○ ○ ○ ○ ○
Lisibilité et organisation du code (commentaires et explications) (x2)	○ ○ ○ ○ ○ ○ ○ ○
Respect des contraintes (résentation linéaire, position du joueur et influence des choix)	○ ○ ○ ○
Utilisation appropriée de variables influençant le déroulement ou le résultat de l'histoire	○ ○ ○ ○
Variété et intérêt des choix proposés	○ ○ ○ ○

TABLE A.3 – Grille d'évaluation de la production du code (troisième semaine)

Régle générale B : un code (partie 2 et 3) qui ne respecte pas scrupuleusement les spécifications établie (lors de la partie 1), est, par définition hors sujet ; le travail réalisé (et donc la note) ne pourra, dès lors, pas être considérée comme suffisant.

Coco_TP03. Création et utilisation de fonctions Théorie p. 54

Créé la fonction ...

ex1 `bonjour()` qui affiche "Comment vas-tu yo !".

ex2 `multiplication(a, b)` avec deux nombres en paramètre d'entrée et qui renvoie leur produit ; appeler cette fonction avec deux nombres pour vérifier qu'elle fonctionne correctement.

ex3 `nomEntier(prenom, nom)` avec deux variables "codées en dur" [2] en entrée et qui renvoie leur concaténation [3] sous la forme Prenom NOM.

ex4 `prixTotal()` avec en argument le prix HT d'un article et la quantité afin qu'elle retourne le prix TTC (taux TVA à chercher sur internet).

ex5 `excuse()` qui ne prend aucun argument, mais retourne une excuse tirée au hasard [4] dans une liste. Utiliser cette fonction pour donner deux excuses dans une phrase de type "Merci de bien vouloir excuser mon absence. En effet, je n'ai pas pu venir car d'une part X et Y, en parallèle, d'autre part." [5].

```
explExcuses = ["j'ai ete attaque-e par un chaton trop mignon",
"j'ai oublie de me reveiller", "j'ai cru que c'etait demain",
"j'ai ete happe-e par un ouragan",
"j'ai ete teleporte-e dans le passe",
"j'ai ete aspire-e dans un trou noir",
"j'ai ete enleve-e par deux extraterrestres",
"j'ai ete transporte-e dans un univers parallele",
"j'ai ete aspire-e dans un reve eveille",
"j'ai ete echange-e a la naissance",
"j'ai ete enleve-e par des pirates de l'espace"]
```

2. Ce qui signifie que vous les définissez vous.
3. Action de mettre ensemble
4. Aidé du module d'`import` avec la fonction `random.choice()` pour tirer au sort un élément de la liste
5. Remplacer X et Y par les excuses tirées au hasard.

Coco_TP04. Listes et dictionnaires Théorie p. 48, aide-mémoire p. 177

ex1 Pour s'entraîner à la gestion de listes (de courses), effectue les étapes suivantes :

- Créer une liste vide nommée listeCourses
- Ajouter cinq éléments à la liste (un par un) à l'aide de la fonction `append()`
- Afficher le nombre d'éléments dans la liste à l'aide de la fonction `len()`
- Utiliser une boucle `for` et la fonction `print()` pour afficher chaque élément de la liste précédé d'un tiret
- Supprimer l'un des éléments de la liste à l'aide de la fonction `remove()`
- Vérifier le nombre d'éléments de la liste
- Modifier le programme pour afficher chaque élément de la liste avec son index (0, 1, 2, 3...)

ex2 Gère les séries comme sur Netflix ;)

- Créer une liste de séries
- Demander à l'utilisateur de saisir un nom de série à chercher et utiliser une variable (par exemple varRechercher) pour stocker la valeur fournie
- Vérifier si la série demandée se trouve dans la liste des séries à l'aide de la fonction `in` et du contrôle `if`
- Afficher un message indiquant si la série est dans la liste ou non (afficher le nom de la série saisie et le message d'information)

```
series = ["Vikings", "Sherlock", "Westworld", "Fargo", "Gotham", "Homeland"]
```

```
# Lancement du programme, exemple 1
Choisir une serie: 1
Sherlock va bientot demarrer, bonne serie!

# Lancement du programme, exemple 2
Choisir une serie: Casa
Cette serie n'est malheureusement pas disponible.
```

Coco_TP05. Manipulation de fichiers Théorie p. 58, aide-mémoire p. 177

ex1 Tester différents modes d'ouverture

Créer un fichier nommé test.txt contenant la phrase "Bonjour le monde". Écrire ensuite un script Python pour :

i) Ouvrir test.txt en mode lecture ("r") et afficher le contenu à l'écran

ii) Reouvrir en mode écriture ("w") et écrire le mot "Salut"

iii) Ouvrir le fichier pour vérifier que l'ancien contenu est bien écrasé

iv) Ouvrir le même fichier en mode ajout ("a") et ajouter "le monde", sans effacer la phrase existante

ex2 Lecture, écriture et fermeture

Créer un fichier donnees.txt manuellement (par exemple via un Notepad) et y ajouter quelques lignes de texte.

i) Ouvrir donnees.txt en mode r+ (lecture/écriture), lire la première ligne et l'afficher

ii) Ajouter ensuite la ligne de texte "FIN DU DOCUMENT" à la fin du fichier.

iii) Fermer le fichier avec fd.close().

Coco_TP06. Tableaux et matrices Théorie p. 64

ex1 Révision

Créer un fichier `mes_animaux.txt`

```
ÉléphantZèbré
Pangouin
*Serpent à plumes
Chat-poisson
Kangourou volant
*Girafe-licorne
*Tortue-lumineuse
Poule multicolore
Écureuil-sauterelle
*Chauve-souris étoilée
```

i) Ouvrir ce fichier (depuis un ide Python)

ii) Stocker son contenu dans une liste

iii) Supprimer les lignes commençant par "*"

iv) Tirer un animal au sort avec `random.choice()`

v) Afficher l'animal sélectionné

ex2 Créer une image en noir et blanc avec une matrice et l'afficher à l'aide de boucles. Pour cela, utiliser :

```
print("X", end="")  # Croix (0)
print(" ", end="")  # Vide  (1)
```

```
image = [
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 1, 1, 0, 0, 1, 1, 0],
    [0, 1, 1, 0, 0, 1, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 0, 1, 0],
    [0, 0, 1, 1, 1, 1, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0]
]
```

ex3 Carte de jeu

i) Créer un tableau nommé carte pour représenter une carte de jeu de 4x4 avec des emplacements vides représentés par le caractère "·"

. . . .

. . . .

. . . .

. . . .

ii) Placement du trésor

— Utiliser la fonction randint du module random pour choisir aléatoirement une ligne et une colonne sur la carte

— La fonction `randint()` prend deux nombres en paramètres et renvoyer un entier au hasard (l'utiliser pour attribuer la ligne et la colonne)

— Placer le trésor représenté par le caractère "X" à la position choisie

iii) Affichage de la carte

— Créer une boucle pour parcourir chaque ligne de la matrice, puis, chaque élément de chaque ligne

— Afficher le contenu de chaque case, en ajoutant , end="" pour éviter les retours à la ligne

ex4 Organisation d'une classe de sport

Un prof. de sport organise un tournoi de badminton avec 4 équipes. Chaque équipe joue 3 matchs, et les scores sont enregistrés dans un tableau (liste de listes).

i) Créer une matrice pour enregistrer des scores de badminton :

```
scores = [
    [15, 20, 25],  # Equipe 1
    [10, 30, 22],  # Equipe 2
    [18, 25, 28],  # Equipe 3
    [12, 19, 27]   # Equipe 4
]
```

ii) Calculer la somme des scores et déterminer l'équipe gagnante

ex5 Places de cinéma

Un cinéma dispose de 5 rangées de sièges, chacune contenant 8 places. Les sièges réservés sont représentés par 1, et les sièges libres par 0. La situation actuelle est la suivante :

```
sieges = [
    [0, 1, 1, 0, 1, 0, 0, 1],
    [1, 1, 0, 1, 0, 1, 1, 0],
    [0, 1, 1, 1, 1, 0, 0, 0],
    [1, 0, 1, 1, 0, 1, 0, 1],
    [0, 0, 0, 1, 1, 1, 0, 0]
]
```

i) Calculer le nombre total de places réservées (somme des 1) et afficher les numéros de rangées contenant au moins 5 sièges disponibles

ii) Modifier le tableau pour ajouter une 6ème rangée où tous les sièges sont libres (tous 0), et afficher le nouveau tableau

ex6 Analyse de l'évolution de températures

Une station météo a enregistré les températures quotidiennes de 7 villes pendant 3 jours. Les données sont organisées dans un tableau de la forme suivante :

```
temperatures = [
    [12, 15, 14],  # Ville 1
    [10, 13, 17],  # Ville 2
    [14, 14, 16],  # Ville 3
    [9,  10,  8],  # Ville 4
    [15, 17, 20],  # Ville 5
    [11, 12, 14],  # Ville 6
    [13, 16, 18]   # Ville 7
]
```

i) Afficher la température moyenne pour chaque ville

ii) Déterminer quelle ville a eu la température maximale pendant ces 3 jours, et préciser la journée où elle a été enregistrée.

A.4 Modules Python clés

A.4.1 Matplotlib

À venir...

A.4.2 Random

Théorie p. 68

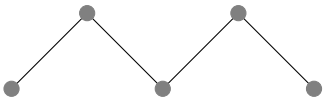
Voir Bases_TP03 p. 137 sur les jeux de hasard avec les boucles au conditionnel.

A.4.3 Turtle

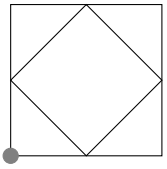
Turtle_TP01. Déplacer une tortue Théorie p. 69, aide-mémoire p. 176

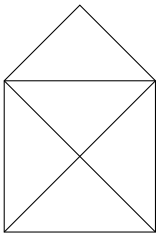
Dans ces exercices, tu peux utiliser l'instruction `back()` *permettant de faire faire un demi-tour à la tortue.*

ex1 Génère les zig-zag ci-contre.



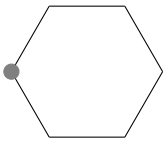
ex2 Dessine deux carrés emboîtés.



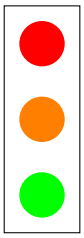


ex3 Voici un jeu pour enfants nommé "Maison de Nicolas" : le but est de la dessiner sans lever le crayon avec exactement 8 segments droits sans repasser deux fois sur un même segment.

ex4 Dessine un hexagone régulier et assure-toi que chaque côté à une couleur différente.

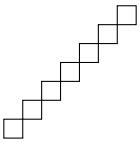


ex5 Dessine un feu de circulation. Tu peux aussi dessiner, un rectangle noir avec un crayon de largeur 80 et les cercles avec dot(40)...

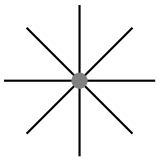


Turtle_TP02. Répétitions Théorie p. 70, aide-mémoire p. 175

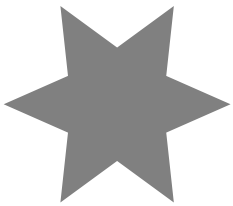
ex1 Dessine un escalier comportant sept marches.



ex2 Dessine une étoile en utilisant l'instruction back().

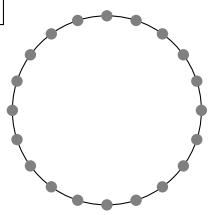


ex3 Dessine une belle étoile en utilisant les angles de 140° et 80°.



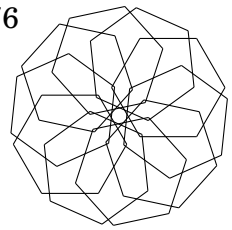
ex4 Dessine la figure ci-contre en utilisant deux instructions `for...` imbriquées. L'instruction `for...` de l'intérieur doit dessiner un carré.

ex5 Dessine un collier de perles.

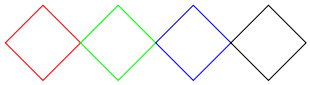


ex6 Dessine un oiseau.

Turtle_TP03. Fonctions et paramètres Théorie p. 71, aide-mémoire p. 176

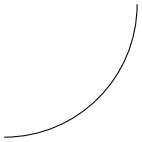


ex1 Définir une commande `hexagon()` permettant de dessiner un hexagone. Utiliser ensuite cette nouvelle fonction pour dessiner la figure ci-contre.

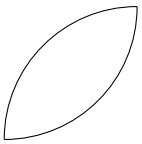


ex2 Définir une fonction permettant de dessiner un carré posé sur l'un de ses sommets. Utiliser ensuite cette fonction pour dessiner la figure ci-contre.

ex3 *Dans cet exercice, on va découvrir comment résoudre un problème complexe par étapes en utilisant des fonctions. Il s'agit d'un concept central en programmation nommé : programmation procédurale.*



— Définir une fonction `arc()` qui demande à la tortue de dessiner un arc de cercle d'angle au centre 90°. Pour accélérer le dessin, on peut régler la vitesse de la tortue.

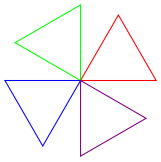


— Ajouter au programme la fonction `petale()` chargée de dessiner deux arcs de cercles consécutifs de sorte que la tortue se retrouve dans sa position et son orientation de départ à la fin du dessin.

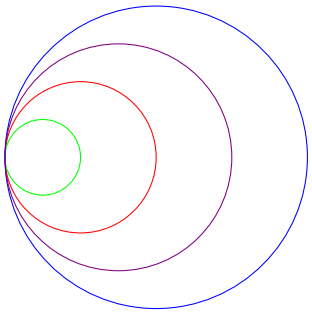
— Modifier le programme précédent pour de sorte que la fonction `petale()` dessine une forme remplie de jaune sans bord visible.

— Étendre le programme avec la fonction `fleur()` permettant de dessiner une fleur à cinq pétales. Pour accélérer encore davantage le dessin, on peut utiliser la fonction `hideturtle()` pour rendre la tortue invisible.

On peut également se poser la question s'il n'existe pas une manière plus rapide de réaliser ce dernier dessin, une fonction qui par exemple permettrait de gagner du temps ?...



ex4 Définir une fonction `triangle(couleur)` qui demande à la tortue de dessiner des triangles colorés. Elle doit dessiner quatre triangles de couleur rouge, verte, bleue et violette.

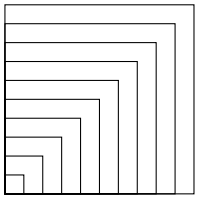


ex5 Définir une fonction `colorCircle(rayon, couleur)` telle que la tortue dessine un cercle coloré. La fonction `circle(rayon, angle)` peut être utilisée dans ce but. Employer cette fonction par la suite pour dessiner la figure ci-contre.

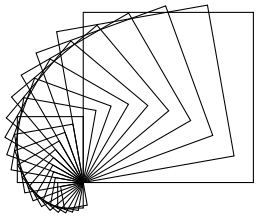
Turtle_TP04. Variable

ex1 Développer un programme qui demande à l'utilisateur un nombre entier n et qui dessine ensuite un polygone régulier à n côtés avec la tortue. Par exemple, si l'utilisateur saisit le nombre 8, il faut dessiner un polygone régulier à 8 côtés (un octogone). Le programme doit calculer l'angle de rotation approprié après chaque segment droit.

ex2 Demander à la tortue de dessiner 10 carrés. Définir tout d'abord une fonction square prenant le paramètre longueur. Le côté du premier carré mesure 8 et celui des carrés suivants est toujours supérieur de 10 à celui du carré précédent.



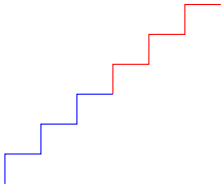
ex3 Lire le côté du plus grand carré depuis une boîte de dialogue. Dessiner ensuite 20 carrés dont le côté du carré suivant diminue toujours d'un facteur 0.9 par rapport au côté du carré précédent et qui effectue une rotation de 10 degrés.



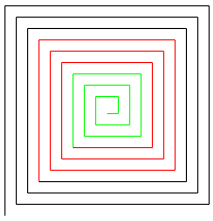
Turtle_TP05. Conditions Théorie p. 73, aide-mémoire p. 175

ex1 Demander à l'utilisateur de saisir la longueur du côté du carré à dessiner. Si elle est inférieure à 50, dessiner un carré rouge de la taille correspondante. Sinon, dessiner un carré vert de la taille indiquée.

ex2 Dessiner un escalier comprenant 6 marches en utilisant une instruction for _ in range. Dessiner les 3 premières marches en bleu et les marches restantes en rouge.

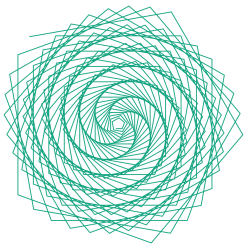


ex3 Dessiner une spirale en utilisant d'abord du vert, ensuite du rouge et, finalement, du noir.

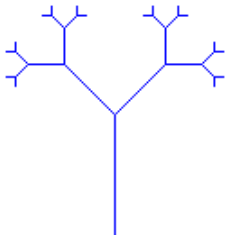


Turtle_TP06. Boucles while Théorie p. 73, aide-mémoire p. 176

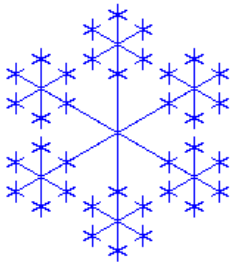
ex1 La tortue avance sur un segment droit de longueur 5 puis tourne de 70° vers la droite. Ensuite, la longueur du segment est incrémentée de 0.5. Répéter ces étapes tant que la longueur du segment est inférieure à 150.



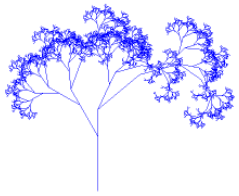
Turtle_TP07. Références récursives Théorie p. 75



ex1 La figure ci-contre représente un arbre binaire (graphe bien connu des mathématiciens) de profondeur 4. Le tronc unique est rajouté en plus et n'en fait pas partie à strictement parler. Définir une fonction récursive arbreBinaire(t) qui dessine un arbre binaire complet (sans le tronc qu'il faut rajouter a posteriori) de taille t.



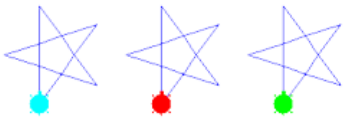
ex2 Dessiner l'étoile ci-contre en définissant la fonction récursive flocon(t) où t représente la taille du flocon. À chaque itération, t diminue à 1/3 de la valeur précédente. Dessiner le flocon avec l'appel flocon(180) et définir l'ancrage de la récursion de telle sorte qu'elle s'arrête lorsque t < 20 (en cachant la tortue avec hideTurtle, le dessin se fera plus rapidement).



ex3 Pour aller plus loin... Dessinez un arbre qui ressemble presque à un arbre réaliste. Dans ce but, définir une fonction arbreFractal(t) dont la branche mesure t construit de la manière suivante :

- Définir l'ancrage de la récursion de telle sorte qu'elle s'arrête lorsque la longueur t de la branche est inférieur à 5
- Avant de commencer le dessin de l'arbre, sauver les coordonnées x et y ainsi que son orientation de manière à pouvoir les restaurer facilement à la fin du dessin
- Avancer de s/3, tourner de 30 deg. vers la gauche et dessiner un arbre de taille 2s/3
- Tourner de 30 degrés vers la droite, avancer de s / 6, tourner de 25 degrés vers la droite et dessiner l'arbre de taille s/2
- Tourner encore une fois de 25 degrés vers la droite, avancer de s/3 et dessiner à nouveau un arbre de taille n/2

Turtle_TP08. Introduction à la programmation objet Théorie p. 76



ex1 Trois tortues doivent dessiner des étoiles à cinq branches simultanément. Chacune doit tracer un segment à tour de rôle. Elles doivent être de couleur cyan (la couleur par défaut), rouge et verte respectivement. La couleur sera spécifiée dans un paramètre additionnel du constructeur.

ex2 Une maman tortue verte tourne en rond à vitesse constante. Au début, la position du bébé tortue rouge est éloignée de la maman. Au fil du temps, le bébé tortue avance par petits pas en direction de sa maman. Le bébé tortue, nommé enfant peut déterminer la direction pointant vers sa maman avec `direction = enfant.towards(maman)` et `enfant.setheading(direction)`.

A.4.4 Pygame

Pygame_TP01. Bases de la programmation orientée objet

ex1 Créer sa propre image de fond à l'aide d'un éditeur d'images puis l'ajouter au dossier Images, lui-même présent dans le même dossier que vos scripts Python. À l'aide de recherches sur Internet, inclure cette image comme fond d'écran d'un jeu.

ex2 Ajouter une barre de statut de 30 pixels de haut qui donne le nombre de ballons qui ont malgré tout atterri.

Pygame_TP02. Interactions et classes d'objet

ex1 Les balles ou ballons, une fois atterris, ne devraient pas disparaître mais être remplacés par une forme différente et immobile ; un petit carré bleue fera par exemple très bien l'affaire.

ex2 Les objets qui ont atterris communiquent aux entrants leur position d'atterrissage de sorte que les futurs ballons ne tombent pas dans un espace déjà occupé. Une fois que toutes la base (organisée par colonnes) est complète, le jeu se termine avec le message "Game Over".

A.5 Sites web

Web_TP01. Les métadonées Aide-mémoire p. 178, théorie p. 83

ex1 Repérer les métadonnées

i) Choisir un fichier de l'ordinateur (document texte ou image). Afficher ses propriétés (avec Informations ou Propriétés). Noter trois métadonnées

1. .....
2. .....
3. .....

ii) Définir une métadonnée .....
.....

iii) En quoi une métadonnée se distingue du contenu du fichier .....
.....

iv) Pourquoi il est important de vérifier ces informations ? .....
.....

ex2 Créer des métadonnées

i) Soit un fichier `meta.html` contenant le code suivant :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <meta name="author" content="Prenom NOM">
    <meta name="description" content="Page metadonnees">
    <meta name="keywords" content="web, metadonnees, exercice">
    <title>META</title>
  </head>
  <body>
    <h1>Bienvenue</h1>
    <p>Ceci est une page exemple.</p>
  </body>
</html>
```

ii) Identifier les balises qui contiennent des métadonnées .....
.....
.....

iii) Expliquer l'utilité de ces balises (référencement, lisibilité, etc.).
.....
.....

ex3 Manipuler les métadonnées

i) Télécharger une photo libre de droit et afficher ses informations EXIF. Noter trois métadonnées trouvées

1. .....
2. .....
3. .....

ii) Pourquoi des métadonnées pourraient-elles poser un problème de confidentialité ?

.....

iii) Donner un exemple de métadonnées utiles pour :

– un photographe .....
– un site de voyage .....
– un réseau social .....

iv) Rechercher comment supprimer ou modifier les métadonnées d'une image. Indiquer l'outil utilisé et comment vous avez fait .....

.....

.....

.....

Web_TP02. Document Markdown Théorie p. 84

ex1 Bases du Markdown

i) Préparation

— Remplacer `#### Prénom / ...` par
`#### Prénom / Web - DocMD - Bases _ TP02ex1`

— Enregistrer le document dans son répertoire personnel (remarquer le nom donné)

ii) Modification

— Changer `# Titre du document` en `# Bonjour le monde`

— Remplacer / Ajoutr le paragraphe :
`Ceci est mon premier document **Markdown**, il est *simple*.`

— Enregistrer (`Ctrl + s`)

iii) Publication

— Ajouter une citation (>), simple, de votre choix

— Enregistrer à nouveau

— Récupérer le lien de partage du fichier (`.md`)

— Le coller dans un nouvel onglet pour voir le résultat

— Déposer le fichier (.md) et le lien dans Moodle

Résultat attendu (exemple)

```
#### Prénom / Web - DocMD - Bases _ TP02ex1
---
\
# Bonjour le monde
Ceci est mon premier document **Markdown**, il est *simple*.

> Le Markdown est facile à apprendre !
```

ex2 Carte d'identité numérique

i) Préparation

- Copier le nom et revenir au modèle standard en cliquant sur Réinitialisé (bouton jaune).
- Remplacer `#### Prénom / Objet-TP01` par `#### Prénom / Web - DocMD - CarteId _ TP02ex2`
- Exporter le document en `.md`, puis le ré-ouvrir
- Changer `# Titre du document` en `# Ma bio`
- Enregistrer dans votre répertoire personnel

ii) Modification

- Mettre un mot en gras, un autre en italique et un petit terme technique en code
- Modifier la citation avec une phrase inspirante (auteur précédé de –)
- Créer un sous-titre `## Centres d'intérêt`
- Ajouter une `### Liste` de 3 à 5 puces d'intérêts
- Mettre en forme un `### Tableau` de `Compétence / Niveau` (3 lignes)
- Sous `### Code`, écrire en Python : `Bonjour, {VotrePrénom} !` (à tester au préalable dans l'atelier Python...)
- Sous `### Tâches`, cocher "Terminé" et ajouter une case "Relecture"
- Sous `### Divers`, insérer un lien vers un site au choix (texte explicite), conserver le commentaire HTML
- Placer une image carrée de 120×120 : `https://placehold.co/120`

iii) Publication

- Conserver la ligne `[Commentaire non visible]: :` et ajouter un mot-clé
- Enregistrer à nouveau le document et récupérer le lien de partage
- Le coller dans un nouvel onglet pour voir le résultat
- Recommencer les deux points précédent, mais avec le partage d'aperçu
- Déposer le fichier et les liens dans Moodle

Résultat attendu (exemple)

````
#### Prénom / Web - DocMD - CarteId _ TP02ex2
---
\
# Ma bio
Je suis **motivé**, *curieux*, et j'aime apprendre
de nouveaux mots en `markdown`.

> "On ne naît pas génie, on le devient." - Anonyme

## Centres d'intérêt
### Liste
- Programmation créative
- Graphisme et mise en page
- Sport & montagne

### Tableau
| Compétence     | Niveau        |
|----------------|---------------|
| Markdown       | Intermédiaire |
| Python         | Avancé        |
| Organisation   | Débutant      |

### Code
```python
# Affichage
prenom = "Antoine"
print(f"Bonjour, {prenom} !")

### Tâches
- [x] Rédaction
- [ ] Relecture

### Divers

[Lien](http://eduge.ch) <!-- EDUGe -->

![Carré gris](https://placehold.co/120)

[Commentaire non visible]:: Ma bio personelle
````
````

Web_TP03. Bases HTML Aide-mémoire p. 178, théorie p. 85

ex1 Créer une page

- Reprendre le code proposé par défaut
- Créer un nouveau fichier avec pour titre
  `<!- Prénom / Web - Bases HTML - Creer _ TP03ex1 ->`
- Enregistrer puis ouvrir dans un navigateur web
- Vérifier que le titre de l'onglet et le contenu s'affichent correctement

ex2 Modifier une page

i) Ajouter le commentaire sur la première ligne et enregistrer le fichier dans son répertoire personnel (`<!- ... - Modifier _ TP03ex2 ->`)

ii) Ouvrir la page `https://www.mozilla.org/fr/` et copier / coller le script html (boutton droit / Afficher le code)

iii) Rechercher, ajouter ou modifier
   - le titre principal <h1>, remplacer par "Copie Mozilla"
   - remplacer le contenu de la balise `<title>` par TEST

iv) Ouvrir le fichier enregistré dans un navigateur web

v) Constater les changements dans un navigateur web

ex3 Enrichir une page

- Repartir avec un fichier standard nommé
  `<!- Prénom / Web - Bases HTML - Enrichir _ TP03ex3 ->`
- Ajouter un titre secondaire `<h2>` Premier sous-titre `</h2>`
- Ajouter un lien vers `https://eduge.ch`
- Ajouter une image `https://placehold.co/200x150`
- Améliorer la lisibilité en indentant correctement le code
- Rendre les trois pages créées (ex1, 2 et 3) dans Moodle

Notes : .....
.....
.....
.....

Web_TP04. Balises, listes et tableaux Aide-mémoire p. 178, théorie p. 86

ex1 Identifier les balises

— Observer le code suivant :

```
<!DOCTYPE html>
<html>
  <head>
    <title>PAGE</title>
  </head>
  <body>
    <h1>Titre</h1>
    <p>Ceci est un paragraphe.</p>
    <a href="https://eduge.ch">Lien vers EDUGe</a>
  </body>
</html>
```

— Repérer et expliquer le rôle des balises

`<html></html>` .....

`<head></head>` .....

`<body></body>` .....

`<h1></h1>` .....

`<p></p>` .....

`<a></a>` .....

ex2 Créer une page avec des balises simples

i) Créer un nouveau fichier `<!- Prénom / Web - Balises _ TP04ex2 ->`

ii) Écrire le squelette minimal (`<!DOCTYPE>`, `<html>`, `<head>`, `<title>`, `<body>`).

iii) Dans le `<body>`, insérer :

— un titre principal `<h1>` : « Ma première page avec des balises »

— un sous-titre `<h2>` : « Introduction »

— deux paragraphes `<p>` avec du texte libre

— une image `<img src="https://placehold.co/150x100">`

— un lien `<a>` vers le site de votre école

iv) Enregistrer puis ouvrir dans un navigateur pour vérifier.

ex3 Structurer et hiérarchiser correctement

— Partir du fichier précédent et le renommer
`<!- Prénom / Web - Listes_Tableau _ TP04ex3 ->`

— Ajouter une section avec :

— un titre `<h2>` : « Mes centres d'intérêt »

— une liste non ordonnée `<ul>` contenant 3 éléments `<li>`

— une liste ordonnée `<ol>` contenant 3 éléments `<li>`

— Ajouter un tableau simple :

```
<table border="1">
  <tr><th>Balise</th><th>Utilite</th></tr>
  <tr><td>h1</td><td>Titre principal</td></tr>
  <tr><td>p</td><td>Paragraphe</td></tr>
</table>
```

— Vérifier l'affichage et corriger l'indentation du code.

Web_TP05. Contenus

ex1 Structurer le texte et les images

i) Texte et listes

— Reprendre la structure minimale HTML
(`<!DOCTYPE>`, `<html>`, `<head>`, `<title>`, `<body>`).

— Modifier l'en-tête
`<!- Prénom / Web - Contenus - Textes _TP05ex1i ->`

— Dans le `<body>` insérer :

— un titre principal `<h1>` Page de contenu

— un sous-titre `<h2>` : Introduction

— deux paragraphes `<p>` avec texte libre

— une liste non ordonnée `<ul>` avec 2 éléments `<li>`

— une liste ordonnée `<ol>` avec 3 éléments `<li>`

— Enregistrer et afficher la page dans un navigateur.

ii) Images et média

— Renommer l'en-tête du fichier précédent
`<!- Prénom / Web - Contenus - Media _ TP05ex1ii ->`

— Ajouter :

— une image distante
`<img src="https://placehold.co/200x150" alt="Exemple">`

— une vidéo avec `<video src="....mp4" controls>` (lien libre de droit)

— Aligner les contenus médias sur une seule ligne

— Intégrer un pied-de-page à votre document

Enregistrer le tout et le tester dans un navigateur

ex2 Liens internes et externes

i) Reprendre la structure minimale HTML et créer un fichier avec l'en-tête suivant :
`<!- Prénom / Web - Contenus - Liens _ TP05ex2 ->`

ii) Dans le `<body>` insérer :

— un titre principal `<h1>` : Page de navigation

— un paragraphe contenant un lien externe `<a>` vers `https://www.eduge.ch`

— un second paragraphe contenant un lien vers un fichier créé dans un TP précédent

— trois ancres internes (`<a href="#...">`) permettant de naviguer vers :

— une section `<h2>` Présentation

— une section `<h2>` Images

— une section `<h2>` Contact

— En bas de la page, insérer un lien `<a href="#top">` pour revenir en haut

iii) Tester dans un navigateur / Cliquer sur chaque lien pour vérifier la navigation

Web_TP06. CSS et Formulaires Aide-mémoire p. 179, théorie p. 87

ex1 Appliquer une feuille de style interne

— Reprendre la structure minimale HTML

— Modifier l'en-tête
`<!- Prénom / Web - CSSint _ TP06ex1 ->`

— Dans la section `<head>`, insérer un bloc :

```
<style>
  body { background-color: #f0f0f0; }
  h1   { color: blue; text-align: center; }
  p    { color: #333333; font-family: Arial, sans-serif; }
</style>
```

— Enregistrer, ouvrir dans un navigateur et observer les changements de couleur / formatage

ex2 Feuille de style externe et mise en forme
Modifier l'en-tête de l'exercice précédent en
`<!- Prénom / Web - CSSext _ TP06ex2 ->`

i) Lier la page avec une feuille de style externe et modifier les mises en forme

```
<link rel="stylesheet" href="style.css">
```

— Retirer les blocs h1 et p des styles internes

— Modifier le fond de la page en blanc

— Dans `style.css`, ajouter les bloc suivant :

```
h1  > rouge foncé, taille de police 28
h2  > vert, taille de police 22
p   > hauteur de ligne de 1.5 et marge de 10px
img > bordure noire fine
```

ii) Dans le corps du texte, ajouter un espace de navigation et un pied de page
Compléter ce dernier avec votre nom

iii) Dans la partie principale (à créer), ajouter

— un titre `<h1>`

— un sous-titre `<h2>`

— deux paragraphes `<p>`

— une image locale ou distante

iv) Enregistrer et tester votre page en vérifiant que le style s'applique à tous les éléments

ex3 Créer un formulaire de contact simple
Modifier l'en-tête de l'exercice précédent en
`<!- Prénom / Web - FormulaireSimple _ TP06ex3 ->`

i) Modifier le titre principal `<h1>` en "Formulaire de contact"

ii) Insérer une balise `<form>` contenant :

— un champ texte `<input type="text">` pour le nom

— un champ texte pour l'adresse électronique

— une zone de texte longue `<textarea>` pour le message

— un bouton d'envoi `<input type="submit" value="Envoyer">`

iii) Ajouter des `<label>` associés à chaque champ

iv) Enregistrer et tester dans un navigateur : saisir puis cliquer sur « Envoyer » Que ce passe-t-il ? Expliquer .....
.....

ex4 Enrichir le formulaire avec de nouveaux champs
Modifier l'en-tête de l'exercice précédent en
`<!- Prénom / Web - FormulaireComplet _ TP06ex4 ->`

i)

ii) Ajouter dans le `<form>` :

— un menu déroulant `<select>` avec trois options (expl. Info, Question, Autre)

— des cases à cocher `<input type="checkbox">` (ex. Newsletter, Partage...)

— un bouton radio `<input type="radio">` pour choisir un genre (H/F/Autre)

iii) Tester le tout dans un navigateur

Web_TP07. CMS Théorie p. 89

ex1 Créer un compte

- Se rendre sur la base démo (cliquer sur le mot démo [6])
- Copier le site en tant qu'étudiant
- Saisir une adresse éléctronique valide
- Choisir un nom de base de données à vous (exemple : `classe-prénom-web`)
  Note : ce nom identifie votre espace de travail (intitulé base de données)

ex2 Paramètres

- Sous Paramètres, donner les détails liés au site (nom d'école, adresse, etc.)
- Ajouter un ou une camarade à votre base de données
- Régler la **langue** (français) et la **devise** (CHF)
- Ajouter les trois modules : `Site Web`, `Facturation`, `Comptabilité`
  Note : chaque module est une application ajoutée à votre base de données

ex3 Personnaliser le site

i) Choisir un thème
   - Ouvrir le module `Site Web`
   - Choisir un nouveau thème

ii) Pages principales
   - **Accueil** Brève description de l'exercice
   - **À propos** Page vous présentant (utiliser éventuellement l'exercice Markdown)
   - **Contact** Formulaire (message)

   Note : pour modifier les pages, utiliser l'éditeur visuel (glisser-déposer).

ex4 Ajouter un produit

- Activer le module `Ventes`
- Ouvrir Produits puis Créer
- Remplir / Ajouter...
  (a) Nom : ex. `T-shirt ECCG`
  (b) Prix : `20 CHF`
  (c) Image : importer une photo de son choix
- Cliquer sur Publier pour rendre le produit visible en ligne

Note : dans une démarche Agile, vérifier avec vos camarades que le site est bien visible !

6. https ://www.odoo.com/odoo-enterprise/template/34355 ?token=8d77607e-4f65-4208-999e-a8d6d3456659

ex5 Générer une facture

- Se rendre dans le module `Comptabilité` ou `Facturation`
- Créer un client fictif (ex. Jean Dupont)
- Ajouter une Nouvelle facture
- Lier avec le produit (créé dans `Ventes`)
- Vérifier la TVA (8.1%)
- Enregistrer la facture
  Note : la liaison entre Ventes et Comptabilité est automatique

ex6 Aller plus loin

i) Activer `Inventaire` pour gérer les stocks

ii) Activer `Achats` pour enregistrer les fournisseurs

iii) Installer `E-commerce` pour transformer le site en boutique

Web_TP08. Révisions Théorie p. 83

Consignes générales

Préparer le site d'un club de boxe (exemple)
(trois pages : `index.html`, `presentation.html`, `contact.html`, accompagné d'un fichier de style `style.css`), où :

- Chaque page contient un menu (en-tête, fond gris clair, texte en noir)
- Chaque page contient un pied de page (fond noir, texte en blanc)
- L'ensemble du texte est en Arial
- Les titres sont clairement définis

Note : les quatre fichiers doivent être placés dans un répertoire nommé :
`PrenomNOM_Web-Boxe_TP08`

ex1 Page d'accueil

Photo : `accueil.jpg`

i) Horaires (sous forme de tableau)

- Tous les jours, sauf le dimanche
- Entraînements enfants et jeunes, en groupe : 16h–18h
- Entraînements individualisés : 18h–20h
- Entraînements au combat : 20h–22h

ii) Annonces

Clôture des inscriptions, 1er décembre 2025.

Le club clôture les inscriptions pour la section boxe éducative, baby boxe et boxe loisir (féminine également). Selon la fréquentation, il est possible que le club puisse, sans garantie, ré-ouvrir une session d'inscription en janvier.

ex2 Présentation

La boxe : présentation

La boxe est un sport de combat où deux adversaires s'affrontent dans un ring, en utilisant uniquement leurs poings, protégés par des gants rembourrés. Reconnue pour sa discipline exigeante et son intensité physique, la boxe repose sur la maîtrise des techniques de frappe, la défense et l'endurance. Au-delà de la simple confrontation physique, la boxe met en avant des valeurs de respect, de contrôle de soi et de résilience. Chaque combat est une démonstration de stratégie et de détermination. Pratiquée depuis des siècles, la boxe est aujourd'hui l'un des sports les plus populaires et compte des compétitions prestigieuses à travers le monde, de l'amateur au professionnel.

Les groupes

Une photo (boxeurs participants aux compétitions) avec la légende
Photo : `groupe.jpg`

ex3 Contact

Boxing Club

Route de la Boxe 24, 1227 Carouge
+41 22 725 2424
`contact@boxingclub.ch`

Entraîneurs

— Nicolas de Vitivier

— Florian Fortich

— Antoine Matleau

Présidente du club

Véronique Embrasslair

ex4 Pied-de-page

— Adresse à gauche

— Lien vers la page de contact à droite

— En dessous : ajouter un lien vers la Fédération suisse de boxe (`https://www.swissboxing.ch/fr`)

A.6 Bases de données

Aide-mémoire p. 178, théorie p. 90

SGBD_TP01. Histoire d'entraînements

Une association / club de sport souhaite gérer l'inscription de ses membres et l'organisation des cours proposés par différents entraîneurs. Les contraintes sont les suivantes :

i) Un cours est donné par un seul entraineur

ii) Chaque cours a une heure de début

iii) Un entraineur peut donner plusieurs cours
(relation 1-N, ce qui implique la création d'une clé externe dans la table des cours)

iv) Un membre peut s'inscrire à plusieurs cours
(relation de type N-N, ce qui implique la création d'une table Inscription)

ex1 Identifier les quatres entités et les relier aux attributs ci-dessous. En déduire le MCD.
Id, Nom, Courriel, Date_Naissance | Id, Nom | Id, Nom_Cours, Heure

ex2 Construire le modèle logique de données correspondant (y.c. les cardinalités).

SGBD_TP02. Silence on lit !

La bibliothèque de l'école souhaite mettre en place un système informatique pour gérer les livres, les emprunts et les personnes qui empruntent des livres pour SoL. Travailler en binôme pour concevoir un modèle conceptuel et logique pour ce faire. Les contraintes sont les suivantes :

i) Un livre est identifié par son titre, son auteur, son ISBN, et sa catégorie (fiction, non-fiction, etc.)

ii) Un abonné est identifié par son numéro d'abonné, son nom, et son adresse

iii) Un emprunt est fait par un abonné pour un ou plusieurs livres
(inclut la date d'emprunt prévue)

iv) Un abonné peut emprunter plusieurs livres, mais un livre ne peut pas être emprunté par plusieurs abonnés en même temps

ex1 Identifier les entités (livres, personnes , emprunts, ...) et définir leurs attributs

ex2 À l'aide des contraintes, déterminer les relations entre les entités

ex3 Transformer le modèle conceptuel en tables E/R (entités / relations ou MLD) avec...

— indication des cardinalités
(1,N pour décrire le fait qu'une personne peut emprunter plusieurs livres, mais qu'un livre ne peut pas être emprunté par plusieurs personnes)

— le type de données (entier / int ou alphanumérique / char)

SGBD_TP03. Bon appetit !

Un restaurant souhaite gérer ses commandes de repas et menus. Le système doit permettre de suivre les commandes des clients, les plats du menu, et les employés qui préparent ces plats. Les contraintes sont les suivantes :

i) Un client est identifié par un numéro de commande, son nom, et son numéro de table

ii) Un plat est identifié par son nom, son prix, et sa catégorie (entrée, plat principal, dessert)

iii) Une commande est passée par un client et contient un ou plusieurs plats. La commande est identifiée par son numéro et contient la date et l'heure de la commande

iv) Un plat est préparé par un employé de cuisine, et chaque employé est identifié par son numéro d'employé et son nom

v) Un client peut commander plusieurs plats, mais chaque plat appartient à une seule commande

ex1 Créer le MCD, en représentant les entités principales

ex2 Définir les cardinalités entre les entités
(par exemple, un employé peut préparer plusieurs plats, mais un plat est préparé par un seul employé)

ex3 Transformer le modèle conceptuel en MLD

SQL_TP01. Tuto / Créations de bases et des tables Théorie p. 96

SQL_TP02. Tuto / De la lecture à la supression Théorie p. 97

SQL_TP03. Tuto / Les jointures Théorie p. 99

SQL_TP04. Tuto / Grouper - Trier Théorie p. 100

A.7 Devenir expert en programmation

A.7.1 Modules spécifiques

Tkinter_TP01 Théorie p. 114

En vous basant sur le code donné dans la théorie, et en cherchant les formules sur le web, créer des applications qui permettent de ...

ex1 Convertir les miles en kilomètres

ex2 Calculer l'indice de masse corporelle (IMC) d'une personne

A.7.2 Cryptographie

Crypto_TP01. Notions Théorie p. 116, aide-mémoire p. 177

ex1 ASCII (ou en luge) Dans un tableau, convertir chaque :

i) Lettre (minuscule a, b, ..., z et majuscule A, B, ..., Z) en son code ASCII avec `ord()`

```
import string                           # abcdefghijklmnopqrstuvwxyz
alphabet = string.ascii_lowercase       # variante: _uppercase ou _letters
```

ii) Nombre (1, 2, ..., 9) en son code ASCII

iii) Code ASCII (48, 49, ..., 57 ansi que 65, 66, ..., 90 et 97, 98, ..., 122) avec `chr()`

iv) Découvrir quels sont les sigles correspondants aux code 58 à 64 et 91 à 96

ex2 Remplacer *Écrire un programme qui demande à l'utilisateur :*

i) D'entrer une phrase

ii) Quelle lettre il veut remplacer

iii) Par quelle lettre il veut la remplacer

Affichez la phrase modifiée.

ex3 Compter Écrire une fonction qui compte combien de lettres (minuscules) se trouvent dans le texte suivant, *"La beauté est dans les yeux de celui qui regarde..."*

ex4 Majuscules / minuscules Écrire une fonction qui prend une phrase en entrée et retourne le nombre de majuscules et de minuscules.

ex5 Clé Écrire un programme qui demande à l'utilisateur un mot secret de cinq caractères. Transformez chaque lettre en son code ASCII et affichez la clé secrète générée sous la forme `115-99-114-101-116`.

Crypto_TP02. Décaler Théorie p. 116, aide-mémoire p. 177

ex1 Chiffrer le mot `chat` avec un décalage de 23.

ex2 Déchiffrer ce texte, codé avec un décalage de 5, `Qj gtsmjzw jxy ifsx qj htij`.

ex3 Faille Un espion a découvert que la lettre S est chiffrée en J dans `Jétlizké urej tcé`. Trouver le décalage et décrypter le message.

ex4 Probabilité On a chiffré un message avec un décalage inconnu compris entre 1 et 25. Tester tous les décalages possibles pour retrouver le message original du code `tm bncd rdbqds dwhrsd`. Tester automatiquement tous les décalages possible et afficher, sur la base d'un dictionnaire de mots courants, le résultat le plus probable.

```
dictionnaire = {
    "il", "est", "un", "le", "la", "les", "ce", "cette", "ces", "secret",
    "message", "ici", "pour", "tu", "vous", "nous", "elle", "ils", "dans",
    "sur", "sous", "avec", "sans", "et", "ou", "mais", "donc"
}
```

Crypto_TP03. Vigenère Théorie p. 117, aide-mémoire p. 177

ex1 Comprendre Soit le message `python` et la clé `code`, appliquer le chiffrement de Vigenère en suivant les étapes suivantes :

i) Répéter la clé jusqu'à la longueur du message

ii) Convertir chaque lettre du message et de la clé en valeurs numériques (a=0, bB=1, ... z=25).

iii) Appliquer le décalage de chaque lettre du message en fonction de la clé

iv) Convertir le message chiffré en lettres

v) Afficher le message chiffré

ex2 Chiffrer Écrire un programme qui :

i) Prend (demande) un message en entrée

ii) Demande une clé à l'utilisateur

iii) Applique l'algorithme de Vigenère pour chiffrer le message

iv) Affiche le texte chiffré

ex3 Déchiffrer Étant donné le message chiffré `lvyfs` et la clé `key`, appliquer le déchiffrement en :

i) Répétant la clé jusqu'à la longueur du message

ii) Convertissant chaque lettre du message et de la clé en valeurs numériques

iii) Appliquant le décalage inverse pour retrouver le message original

iv) Convertissant le message déchiffré en lettres

ex4 Programme de déchiffrement Modifier le programme ex2 pour ajouter une option de déchiffrement :

i) L'utilisateur entre (c) pour chiffrer et (d) pour déchiffrer

ii) L'algorithme doit fonctionner pour récupérer le message original à partir du texte chiffré

ex5 Ça dégénère Un espion a intercepté un message chiffré `dsn-xmz` sans connaître la clé. Cependant, il sait que la clé est un mot anglais de trois lettres... *Proposition de méthode de résolution :*

i) Créer un programme qui teste toutes les combinaisons possibles de clés de trois lettre

```
alphabet = string.ascii_lowercase  # abcdefghijklmnopqrstuvwxyz
print(cles_possibles[:10])         # Afficher dix cles pour verification
```

ii) Vérifier si le texte obtenu contient un des mots français ou anglais de six lettres

```
francais = {"bal-bal", "bon-bon"}
anglais  = {"big-big", "top-top"}
```

Crypto_TP04. Être ou ne pas être fin Théorie p. 117, aide-mémoire p. 177

ex1 Fin limier Décrypter :

i) `nzwpa` avec la clé $f(x) = g(x) = x + 13$. Quelle est la particularité de cette clé ?

ii) `prfne` avec le chiffrement affine $g(x) = 23x + 14$

ex2 Casser un texte codé Un espion a découvert...

i) que `thppbxh phruhk` a été chiffré avec un chiffrement affine inconnu. La lettre K est devenue D, et N est devenue O. Déduire les clés de chiffrement et de déchiffrement et décrypter le message.

ii) qu'un message a été chiffré avec une clé affine (a=11, b=7). Déchiffrez le message `ihwgg sbyw!` et écrire un programme qui teste automatiquement toutes les paires (a, b) possibles pour la clé privée.

Crypto_TP05. La force brute Aide-mémoire p. 177

ex1 PIN et sapin Tester toutes les combinaisons possibles de `0000` à `9999`.

i) Combien de temps met-on pour passer en revue toute les combinaisons ?

ii) Et pour découvrir le code 1228 ?

ex2 Mot de passe avec dictionnaire

i) Un ingrédient secret, en six lettres, a été chiffré avec un chiffrement de César. Complétez le programme pour tester tous les décalages et vérifier si un mot du dictionnaire est retrouvé pour l'ingrédient codé (exemple avec `zywwo`).

```
ingredients = ["oignon", "pomme", "tomate", "piment", "amande"]
```

ii) Un site utilise des mots de passe à quatre lettres uniquement en minuscule (ex : abcd)

i) Écrire un script qui essaie toutes les combinaisons possibles pour retrouver un mot de passe (exemple avec `vent`)

ii) Optimiser l'algorithme pour ne tester que les mots présents dans un dictionnaire

```
mdp = [
    "chat", "pain", "vent", "noir", "bleu", "jour", "nuit", "ciel",
    "rose", "bain", "gris", "faux", "loin", "plus"
    ]
```

ex3 Bonus Un espion a intercepté un message codé mais ne sait pas s'il a été chiffré avec César, Vigenère ou par transformation affine. Écrire un script qui analyse les fréquences de lettres de l'alphabet et essaye d'identifier le type de chiffrement utilisé.

Crypto_TP06. Hâcher Théorie p. 119, aide-mémoire p. 177

i) Stocker le hash d'un mot de passe avec hashlib.sha256()

ii) Demander à l'utilisateur de saisir son mot de passe et vérifier s'il est correct

Dbapi_TP

À venir...

Request_HTMLParser_TP

À venir...

Annexe B

FICHES DE RÉVISIONS

Programme 1e année ECG

Voir également le plan de formation, p. viii.

1. Introduction à l'algorithmique et résolution de problèmes, p. 26, p. 28 et p. 133
2. Introduction à la programmation (bases et containers), p. 37 - 49
3. Contrôles (conditions et boucles), p. 50 - 52
4. Fonctions (paramètres et données), p. 54 - 55
5. Dessin / Orientation, p. 69 - 75
6. Projet et culture numérique (introduction à l'informatique)

Programme 2e année ECG

1. Application algorithmique, rappels (voir ci-dessus) et bonnes pratiques, p. 35 et p. 134
2. Révison 1e année (voir ci-dessus) et gestion de listes, p. 37
3. Introduction à la programmation orientée objet (avec Turtle, p. 69+ et Pygame p. 79+)
4. Jeux et fenêtrage p. 79
5. Projet et culture numérique (diverses notions d'actualité)

Programme 3e année ECG

1. Systèmes et sécurité informatique
2. Algorithmie et résolution de problèmes, p. 25 - 35 et p. 135
3. Web - HTML et CSS p. 83 et p. 155
4. Révison programmation 1e et 2e année (voir ci-dessus) et gestion de fichiers, p. 58, p. 110, p. 146
5. Matrices, tableaux et encodages p. 64 et p. 146
6. Les bases de données p. 96

PYTHON

Commentaires et opérations

Théorie p. 38 et p. 42

```
''' Prenom NOM :: TPi_SujetTravail / Ex... '''
# addition +, soustraction -, multiplication *, division /, puissance **, etc.
```

Assigner / Afficher

Théorie p. 41, exercices p. 136

```
var, prenom = 0, "Croco"          # Assignation de variables
x = x + 1 / round(x)              # Operation (idem x += 1) et arrondi
print("Hello", prenom)            # Affiche 'Hello Croco'
rep = input(question)             # Demande standard / Question (input)
```

Demander / Ajuster les formats

Théorie p. 41, exercices p. 136

```
msg = f"La reponse est {rep}."    # Variable dans une phrase [print(msg)]
txt = str(input("Texte? "))       # Format texte (string, par defaut)
entier = int(input("Nombre? "))   # Nombre entier (integer)
reel = float(a)                   # Nombre reel (float)
```

Répétitions for (boucles basiques)

Théorie p. 47, 51 et 70, exercices p. 140 et 150

```
for _ in range(1,n,p):            # Boucle de 1 jusque n, pas p
    print("no =", _)              # Affiche la valeur du compteur
''''''
for _ in range(n):                # Boucle standard (de 0 jusqu'a n) / carre si n
                                  #         = 4
    t.fd(100)                     # Voir module de dessin Turtle ci-apres
    t.rt(360/n)                   # /! Indentation
```

Conditions if et comparaisons

Théorie p. 50 et 73, exercices p. 137 et 152

```
if note < 4:                      # Si ... alors, <, >, <=, >=, == ou !=
    print ("np")                  # Affiche np pour 'non promu'
elif note > 5:                    # Condition if invalide
    print ("c")                   # Affiche c pour 'certificat'
else:                             # Dans tous les autres cas ...
    print ("p")                   # Affiche p dans tous les autres cas
```

Modules basiques

```
import math / math.sqrt(a)        # Racine carree de a
import time / time.sleep(1)       # Systeme en pause 1 seconde
import turtle / t = turtle        # Dessiner avec turtle (programmation objets)
from nomModule import *           # Synthaxe alternative (sans objets)
```

Dessin (un cercle fait 360°...)

Théorie p. 69, exercices p. 150

```
t.fd(var)                          # Avancer (forward) de var (distance en px)
t.lt(angle) / t.rt(360/n)          # Rotation gauche (left) / droite (right)
t.bk(var)                          # Reculer (back)
t.dot(p)                           # Point d'un rayon p
t.circle(r,a)                      # Cercle de rayon r et angle a
t.penup() / t.pendown()            # Souleve / descend le crayon
t.pencolor("red")                  # Couleur rouge du trait
t.hideturtle() / t.speed(1)        # Cacher la tortue  / accelere
```

While / Tant que (boucles avancées)

Théorie p. 51 et 73, exercices p. 140 et 152

```
i = 0                              # Definit la valeur du compteur
while True:                        # Definit une boucle, possible tant que
    print("i =", i)                # Ecrit la valeur du compteur
    if i == 10: break              # Sort du programme avec break
    else: i = i + 1                # Ajouter un a la variable i (idem: i += 1)
''''''
rep = ""                           # Annoncer la varaiable
while rep != reponse:              # Tant que ..., <, >, <=, >=, == ou !=
    rep = input(question)          # Poser une question
```

Fonctions

Théorie p. 54 et 71, exercices p. 144 et 151

```
def fonction(p):                   # Fonction avec un parametre p
    return p                       # /! indentation (tabulation ou quatre espaces)
fonction(var)                      # Appel de la fonction avec variable
''''''
def forme(n):                      # Fonction forme (parametre n)
    for _ in range(n):             # Repetition n fois
        t.fd(100)                  # Avancer l'objet t de 100px
        t.rt(360/n)                # Rotation de l'objet t
rep = int(input("Nb? "))           # Variable rep = nombre de cote de la forme
forme(rep)                         # Appel de la fonction forme
''''''
if __name__ == "__main__":         # Mode avance de programmation
    fonction(var)
```

Modules avancés

```
import matplotlib                  # Graphiques
import os / os.system("clear")     # Nettoye l'ecran du contenu
import pygame                      # Jeu
import random / random.randint(0,1)  # Nombre aleatoire entre 0 et 1
import tkinter                     # Fenetres
nomModule.nomFonction()            # Utilisation d'une fonction dans un module
```

Listes

Théorie p. 48, exercices p. 145

```
liste = []                          # Creer une liste vide
element = liste[i]                  # Element specifique de la liste
element = matrice[i][j]             # Element specifique du tableau
liste.append(var)                   # Ajouter une variable var dans liste
liste.remove("txt")                 # Supprimer l'element 'txt' de la liste
liste[i] = var                      # Attribuer la valeur var dans une liste
len(liste)                          # Taille de la liste
''''''
for e in liste:                     # Parcourir une liste
    print(e, end="")                # Afficher les elements (end => meme ligne)
''''''
for k in range(len(liste)):         # Parcourir toute la liste (len => longueur)
    print(f"{k}. {liste[k]}")       # Afficher chacun des elements et index
''''''
if e in liste:                      # Verifier si un element dans liste
    print(e, "est present !")
```

Fichiers

Théorie p. 58, exercices p. 146

```
fichier = "Docs/Nom_Fichier"        # Fichier dans repertoire Docs
with open(fichier, "w") as fd:      # Mode ecriture (w pour write)
    fd.write("L1\nL2")              # Ecrire dans fichier (\n => saut de ligne)
''''''
liste = ["L1", "L2"]                # Avec une liste et encodage des accents
with open(fichier, "w", encoding="utf-8") as fd:
    fd.write("\n".join(liste))      # Ecriture des elements de la liste dans fichier
''''''
liste = open(fichier, "r").readlines() # Mode lecture (r pour read)
                                    # Sous forme de liste, ici ['L1\n', 'L2']
''''''
with open(fichier, "r") as fd:      # Mode lecture (yc. gros fichiers > 100k lignes)
    c = fd.read()                   # Lire contenu, sans les saut de ligne \n
''''''
with open(fichier, "r") as fd:      # Avec un index / indice
    for i, ligne in enumerate(fd, start=1):
        print(f"{i}. {ligne.strip()}")
```

Cryptotographie

Théorie p. 116, exercices p. 170

```
code = ord(caractere)               # Convertir caractere en code numerique
caractere = chr(code)               # Convertir code numerique en caractere
''''''                              # ord('a') = 97 / ord('A') = 65
dec, txt = 3, ""                    # Decallage (negatif si decodage du message)
for char in "message":              # Parcours chaque caractere du message
    code = ord(char) - 97 + dec # Donc a => 0 + dec, b => 1 + decal, etc.
    txt += chr(code % 26 + 97)      # Convertir l'Unicode en lettre
```

SQL

Théorie p. 90, exercices p. 167

```
SELECT...
*               FROM table       # Tous les champs de la table
champ           FROM table       # Uniquement le champ
     ... WHERE conditions        # Conditions specifiques
     ... WHERE champs LIKE 'B%'  # Commence par B
champ1, champ2 FROM table        # Deux champs
min(champ)      FROM table       # min, max, sum ou avg
date AS date    FROM table       # Creation d'un alias
DISTINCT champ FROM table        # Uniquement les champs distincts
id, id FROM table1, table2 WHERE tbl2.id = tbl2.id # Jointure
champ  FROM table ORDER BY champ# ASC ou DESC
id, id FROM table1, table2 WHERE tbl2.id = tbl2.id
GROUP BY champ HAVING avg(champ) > 100
```

HTML

Théorie p. 83, exercices p. 155

```
# Structure de base
<!DOCTYPE html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>TITRE</title>
        <link rel="stylesheet" href="style/style.css">
        <style></style>
    </head>
    <body>
        <!-- Contenu de la page -->
    </body>
</html>
```

Balises web

Théorie p. 86, exercices p. 160

```
# Balises de structure
<html>                           # Racine du script
<head>                           # Metadonnees (title, meta, link, style)
<title>                          # Titre de la page (dans onglet)
<body>                           # Contenu de la page
<header>                         # En-tete visible (titre, logo, navigation)
<nav>                            # Partie de navigation
<section>                        # Section thematique
<article>                        # Bloc type article
<aside>                          # Contenu en relation (ex: barre laterale)
<footer>                         # Pied de page
```

```
# Balises de texte / liste
<h1></h1>                       # Niveau des titres h1 a h6
<br>                            # Saut a la ligne
<p></p>                         # Paragraphe
<ul></ul>                       # Declare une liste de puces
<ol></ol>                       # - " - numerotee (expl. <ol start="1"></ol>)
<li></li>                       # Element d'une liste (dans balise ul ou ol)
<a href=""></a>                 # Lien hypertexte, ajouter ev. target="blank"
<code></code>                   # Extrait de codes
<blockquote></blockquote>       # Citation
                           # Espace insecable
```

```
# Balises semantiques
<main></main>                   # Contenu principal de la page
<div></div>                     # Conteneur specifique
<span></span>                   # Conteneur d'une partie de ligne
<table></table>                 # Tableau lignes <tr></tr> et colonnes <td></td>
<img src="" alt="">             # Image avec attributs source et alternative
```

```
# Exemple avec formulaire et divers champs (text, email, password, submit, ...)
<form action="" method="post">
    <label for="nom">Nom</label>
    <input type="text" id="nom" name="nom" required><br>
    <textarea id="msg" name="msg" rows="4" cols="50"></textarea>
    <input type="submit" value="Envoyer">
</form>
```

Styles / CSS

Théorie p. 87, exercices p. 162

```
# Balises sous <style></style>
p {text-align: center;}         # <p>Texte</p>
span {color: red;}              # <span>rouge</span>
```

```
# Styles directement dans le code et les balises <p></p> ou <span></span>
<p style="text-align: center;">Texte <span style="color: red;">rouge</span></p>
```

```
# Selection par classe            <div class="container"></diV>
.container {
    background-color: silver;   # Gris clair
}
```

```
# Selection par identifiant       <div id="menu"></diV>
#menu {                         # Hashtag necessaire
    background-color: gray;     # Gris fonce
    color: white;               # Blanc
}
```

```
# Attributs de mise en forme  /  Expl. style = "width:800px"
text-align                      # Alignement (left, center, right ou justify)
justify-content                 # - " - / Container (flex-start ou flex-end)
border                          # Bordure (1px solid gray)
border-radius                   # Bords arrondis (10px)
color                           # Couleur du texte [red ou rgb(255,0,0)]
background-color                # Couleur d'arriere-fond (white)
text-decoration                 # Decoration (underline ou none)
height ou width                 # Dimension de l'element (50vh/vw, 800px ou 100%)
font-weight                     # Epaisseur du texte (normal ou bold)
line-height                     # Interligne
margin-left ou margin-right     # Marges (auto, 50% ou 5px)
font-family                     # Police utilisee ("Arial", sans-serif)
position                        # Position (absolute, fixed ou relative)
float                           # - " - images (left ou right)
font-style                      # Style du texte (normal ou italic)
font-size                       # Taille de la police (12px, 2em ou 150%)
display                         # Types d'affichage (block > par ligne,
                                # - inline-block > sur une seule ligne,
                                # - flex > alignement flexible
                                # - grid - sur plusieurs colonnes ou rangs
                                # - none - invisible / cache)
```

```
<html>                          # Exemple de grilles, type tableau et carte
    <head>
        <style>
            .grille {
                display: grid;
                grid-template-columns: 1fr 2fr 1fr; # Trois colonnes
            }
            .card {border: 1px solid gray;}
            @media (max-width: 768px) {             # Responsive
                .grille {
                    grid-template-columns: 1fr;     # Une seule colonne
                }
            }
        </style>
    </head>
    <body>
        <div class="grille">
            <div>Gauche</div><div>Centre</div><div>Droite</div>
        </div>
        <div class="card">
            <img src="https://via.placeholder.com/150" alt="image" style="...">
            <p>Texte</p>
        </div>
    </body>
</html>
```

RÉFÉRENCES

Glossaire

algorithme Méthode permettant de résoudre un problème de manière systématique. 25

arrondir . 40

break Sort de la boucle courante. 39, 52, 74

chaîne de caractères Suite de lettres, chiffres ou autres symboles. 46

commentaires Parties de votre code qui ne sont pas exécutées. 36, 38

conditions Critères de sélection pour exécuter du code en fonction de conditions. 50, 73

Copyright Droit exclusif que détient un auteur ou son représentant d'exploiter une œuvre (symbole ©). vii

dictionnaire Collection d'éléments où chaque élément est une paire clé : valeur. 49

enumerate Pour obtenir à la fois l'indice et la valeur des éléments d'une séquence (liste ou chaîne de caractères). 47

fichiers . 58

fonctions Regroupement d'un ensemble d'instructions susceptible d'être utilisé plusieurs fois dans un programme. 54, 71, 144

for Répéter des instructions pour chaque élément d'une liste, d'un tuple, ou de tout objet itérable. 39, 70, 140

Guido van Rossum Créateur du langage Python, en 1991. 37

IDE / GUI Interface utilisateur graphique d'un environnement de développement intégré. 38

if Exécuter du code en fonction de conditions. 28, 39, 50, 73

input Retourne sous forme de chaîne ce que l'utilisateur a saisi. 42, 72

int Traduit une chaîne de caractère en nombre. 72

interpréteur Programme qui lit et exécute le code. 38

liste Type de donnée qui peut contenir plusieurs éléments. 48, 112

Matplotlib Bibliothèque standard pour la création de graphiques. 66

module Bibliothèque ou librairie de classes et fonctions relatives à un domaine d'activité donné. 54

multiplication russe Méthode de multiplication. 29

opérateurs Correspondent aux opérations arithmétiques (+ - x ÷) et aux comparaisons / logiques (= > < et ou). 42

paramètre Valeur qu'on met entre les parenthèses dans l'appel d'une fonction. 58, 71

print Afficher des informations à l'utilisateur. 41

processus Façon de procéder pour résoudre un problème de manière systématique. 30

random Retourne des nombres flottants uniformément distribués entre 0 (inclus) et 1 (exclus). 68

range Créer une séquence de nombres. 47

script fichier ou programme qui finit par ".py" que l'on peut exécuter à l'aide d'un interpréteur de code. 38, 57

set Collection non ordonnée d'éléments uniques [*pour aller plus loin*]. 112

slicing Permet d'accéder à des tranches d'une séquence (liste ou tuple) [*pour aller plus loin*]. 112

spécifications Plan plus ou moins détaillé qui explique ce que doit faire un programme. 26

séquenciel Consiste à effectuer des opérations les unes à la suite des autres (synonyme: linéaire). 33

tuple Liste non modifiable (immutable) [*pour aller plus loin*]. 112

turtle Robot sous forme de tortue. 69

turtle.dot Dessiner un rond d'une taille donnée. 69

turtle.fillcolor Remplit des formes en couleurs. 73

turtle.pencolor Changer la couleur du crayon. 71

turtle.penup Va lever le crayon et empêcher la tortue de dessiner. 69

turtle.width Règle la largeur de la trace. 69

variables Parties de votre code qui ne sont pas exécutées. 28, 39, 41, 72, 113

while Exécuter des instructions tant qu'une condition est vraie. 28, 39, 51, 74

Bibliographie

[Arnold *et al.*, 2022] ARNOLD, J., KOHN, T. et PLÜSS, A. (2022). *Concepts en programmation en Python avec la plateforme TigerJython*. https://programmierkonzepte.ch/franz.

[Bonjour, 2021] BONJOUR, J.-D. (2021). *Introduction à la programmation en Python*. https://www.jdbonjour.ch/cours/.

[Jouët, 1991] JOUËT, J. (1991). *Technologies de communication*. Rapport technique, Organisation des Nations Unies pour l'éducation, la science et la culture (UNESCO). ISBN 92-3-202678-1.

[Python Software Foundation, 2025] PYTHON SOFTWARE FOUNDATION (2025). *Documentation Python 3.13*. https://docs.python.org/fr/3.

[Solnon, 2007] SOLNON, C. (2007). *Support de cours d'algo.* http://perso.citi.insa-lyon.fr/csolnon/.

[Swinnen, 2012] SWINNEN, G. (2012). *Apprendre à programmer avec Python*. Eyrolles, Paris, 3e éd édition. ISBN 22-1-213434-6.

Environnements de développement

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE ZÜRICH, *Plateforme TigerPython*.
https://webtigerpython.ethz.ch/

PROJET JUPYTER, *Google Drive Colaboratory*.
https://colab.research.google.com

INFORMATICLI, *IDE gratuit et sans publicité*.
https://informatic.li

Autres outils utiles

Caractères unicode

— Tables unicode (caractères, symboles, emojis...)

Icônes et emojis

— EmojiDB

— Font Awesome

— Google Fonts

— W3Schools (recherches Font Awesome, Google, Bootstrap)

Couleurs

— HTML Color Picker

Expressions

— Regex Generator (Générateur d'expressions)

Table des figures

Liste des tableaux

Liste des codes

Table des matières

À propos

L'ECCG Aimée-Stitelmann, située à Plan-les-Ouates, à Genève, propose des formations de l'École de Culture Générale (ECG) et de la filière professionnelle commerciale (EC). La cohabitation de formations généralistes dans les domaines socio-éducatifs, de la communication, de l'information, et de la formation professionnelle commerciale y est une richesse tant pour les enseignants que les apprenants.

Antoine Melo est enseignant d'informatique, gestion, économie et droit à l'ECCG Aimée-Stitelmann. Il a auparavant travaillé pendant dix ans en tant que Directeur Administratif et Financier (CFAO), puis directeur des opérations (COO), supervisant notamment les activités opérationnelles et de développement de platformes de reporting et trading en Europe, Asie, Amérique Latine et du nord. Titulaire d'un diplôme en formation professionnelle de la Haute École fédérale en Formation Professionnelle (HEFP), d'une maîtrise ès sciences et ingénierie de l'École Polytechnique Fédérale (EPF) et d'un Master en Business Administration (Executive MBA) des Hautes Études Commerciales (HEC) de l'université de Lausanne, en Suisse, il a également travaillé durant plusieurs années sur diverses missions d'ingénierie en Europe, Amérique Latine et Afrique. Il est marié à Cynthia et l'heureux père de trois grands enfants.

LIENS - QR CODES

Concepts généraux

EXERCICES	ex1	ex2	ex3
Bases_TP01 (p. 136)			
Bases_TP02 (p. 136)			
Bases_TP03 (p. 137)			
Bases_TP04 (p. 137)			
Bases_TP05 (p. 138)			

CORRIGÉS	co1	co2	co3
Bases_TP01			
Bases_TP02			
Bases_TP03			
Bases_TP04			

Containers, contrôles et fonctions (partie 1)

Création d’une histoire interactive

Containers, contrôles et fonctions (partie 2)

Turtle

Pygame

...

Sites web

	ex1	ex2	ex3
Web_TP02 (p. 156)			
Web_TP03 (p. 159)			

...

...

...

CORRIGÉ

SYSTÈMES INFORMATIQUES

Softwares (logiciels)

Exercices p. 120

Sys_TP01. Les systèmes d'exploitation

ex1 QCM

i) Rôle principal d'un système d'exploitation ? Permettre la communication entre matériel et logiciels ; **ii)** Éléments n'est pas un système d'exploitation ? Word ; **iii)** Système d'exploitation basé sur le noyau Unix ? macOS **iv)** Corrects concernant Linux ? Il fonctionne sur smartphone, système opensource et souvent gratuit.

ex2 Rôle d'un système d'exploitation

Interface entre matériel et utilisateur. Exemples : mémoire, gestion fichiers, gestion périphériques (clavier, souris, imprimante), sécurité (comptes, mises à jour).

ex3 Comparaison

Windows : compatibilité logicielle très large (notamment métier et bureautique), jeux et entreprise. **MacOS** : interface intuitive, intégration écosystème Apple, applications pour créatifs. **Linux** : gratuit, opensource, stable (idéal pour les serveurs), personnalisable et léger (aussi pour ordinateurs anciens). **Android** : système mobile très répandu, flexible. **iOS** : grande sécurité, fluidité, stabilité.

ex4 Étude de cas

Critères : budget, compatibilité logiciels, sécurité, simplicité d'utilisation, écosystème (smartphone/tablette/Cloud). **Linux et Windows** : voir ci-dessus.

Sys_TP02. Formats de fichiers

ex1 QCM

i) Format le plus adapté pour un document à conservé sans modification ? .pdf ; **ii)** Images compressées ? .gif, .jpg, .png ; **iii)** Non compressé et donc fichiers lourds ? .bmp ; **iv)** Présentations assistées par ordinateur ? .odp, .pdf, .pptx.

ex2 Type de fichier

`.docx`, document texte (Word), MS Word, LibreOffice Writer ou Google Docs ; `.png`, image compressée (transparente), MS Photos, GIMP ou navigateur web ; `.mp4`, vidéo, VLC Media Player ou Windows Media Player ; `.xlsx`, feuille de calcul, MS Excel, LibreOffice Calc ou Google Sheets ; `.pdf`, document semi-figé / protégé, Adobe Reader ou Foxit Reader.

ex3 Étude de cas

Formats adaptés : `.pdf` car fige la mise en page, léger et universel) ; dans le cas d'un partage pour modifications `.docx` (version souvent juste envoyée, par modifiable par touxtes en même temps) ou Google docs. Inconvénients du `.jpeg` : fichier lourd (une image par page, images séparées), perte de qualité (compression), pas d'édition, mauvaise lisibilité pour un rapport.

Hardware (la machine)

Exercices p. 126

Sys_TP01. Théorie hardware

ex1 Traduction binaire : 5 = 101, 12 = 1100, 23 = 10111
Pourquoi pas décimal : ordinateur basé sur 2 états électriques (0/1).

ex2 Périphériques
Casque des salles informatique > mixte (sortie audio, entrée micro.), clavier > entrée, disque dur externe > mixte (lecture/écriture), écran tactile > mixte (affichage + interaction), imprimante > sortie, microphone > entrée.

ex3 RAM = mémoire vive, volatile, s'efface à l'arrêt (expl. programmes en cours d'exécution), ROM = mémoire morte, non volatile (expl. BIOS, firmware).

ex4 **Évolution processeurs** Fréquence + nombre de coeurs en hausse constante (expl. 1 GHz simple coeur à 4 GHz multi-coeurs) ; **Stockage** Capacité en hausse constante, passage du HDD au SSD (expl. 40Go HDD à 1To SSD beaucoup plus rapide) ; **RAM** Quantité et vitesses en hausse constante (expl 256Mo à 16 Go) ; **Périphériques** Plus rapides et plus de connectiques (expl. USB-C, sans fil, périphériques Bluetooth, écran plat tactile). Ces augmentations constantes suivent une loi empirique appelée **loi de Moore**, selon laquelle la densité des transistors (et donc la puissance) double tous les 18 à 24 mois environ.

ex5 QCM
i) Rôle du processeur ? executer les instructions ; **ii)** Mémoire volatile ? RAM ; **iii)** Supports persistants ? USB, HDD, SSD ; **iv)** Composant intégré à la carte mère ? Carte réseau ; **v)** Fréquence d'un processeur ? GHz.

Sys_TP02. Diagnostique matériel

ex1 **Configuration CPU** expl. Intel Core i5 (processeur central, exécute les instructions), **RAM** expl. 8 Go (mémoire vive, rapide, volatile, utilisée par les programmes), **Disque** expl. SSD 512 Go (stockage permanent, si SSD, alors plus rapide).

ex2 GPU expl. Nvidia GTX ou AMD Radeon (processeur graphique, gère l'affichage).

ex3 Périphériques : entrée (clavier, souris, micro.), sortie (écran, imprimante, haut-parleurs), mixte (casque-micro)

ex4 **Forces** SSD rapide, RAM suffisante, bonne compatibilité (Windows) ;
Limites processeur moyen de gamme, carte graphique basique.

ex5 Ordinateur idéal
CPU quad-core récent (Intel i5 ou Ryzen 5), RAM de 16Go, Stockage SSD de 512Go + Cloud, périphériques (webcam, micro, ports USB-C, capacités Wi-Fi)

Net_TP01. Réseau local

ex1 `10.134...` Adresse privée, utilisée uniquement dans un réseau local. `127.0...` Adresse de bouclage (loopback), renvoie toujours vers sa propre machine. `255.255...` Adresse de diffusion (broadcast), permet d'envoyer un message à toutes les machines du réseau.

ex2 Composants d'un réseau
CPU, RAM, disque Composants internes de l'ordinateur. **Switch** Relie plusieurs ordinateurs dans un même réseau local. **Routeur** Connecte le réseau local à Internet. **Câbles Ethernet** Assurent la transmission filaire des données. **Point d'accès WiFi** Connexion sans fil.

ex3 Exemple de fiche
`PC-Etudiant01`, sous `Windows 11`, dans l'organisation nommée `DIP`. Date d'installation = `10.08.2025`, sur processeur `Intel Core i5`, avec les options régionales `fr-CH`. Disque dur `512 Go` au sein du domaine (DNS) : `eduge.ch`.

Net_TP02. Internet et noms de domaine

ex1 Adresse locale (quatre essais), réponses positives, preuve que la machine est joignable sur le réseau.

ex2 Ping vers `www.eduge.ch` : suite de réponses avec temps de latence. Connexion Internet opérationnelle et nom de domaine résolu si aucun rejet.

ex3 Le rôle d'un DNS est de traduire les noms de domaine (ex. `www.eduge.ch`) en adresses `IP`. Avec `nslookup`, on obtient l'adresse `IP` et le serveur DNS utilisé.

ex4 Ping / adresse interne : pas de passage par Internet, mais uniquement l'intranet.

Sys_TP03. Connectivité

ex1 Connexion SSH
SSH = *Secure Shell*. Message typique : demande de mot de passe et avertissement de sécurité. Intérêt : SSH permet une communication chiffrée et donc sécurisée, contrairement à Telnet.

ex2 Connexion FTP
FTP = *File Transfer Protocol*. Différence FTP / SFTP : FTP transmet les données en clair (non sécurisé), alors que SFTP chiffre les échanges car il s'appuie sur SSH.

ex3 QCM
i) Filtrer les informations ? Routeur ; **ii)** Transférer pages web ? HTTP ; **iii)** Différence entre réseau câblé et sans fil ? Le sans fil utilise des ondes radio ; **iv)** LAN ? Câblé ou WiFi, réseau local, souvent en étoile ; **v)** Service DNS ? Nom de domaine et adresse IP.

BASES DE DONNÉES

Modélisation

Exercices p. 167

SGBD_TP01. Histoire d'entraînements

Schéma conceptuel (MCD)

Une relation N-N (à gauche) => Table intermédiaire INSCRIPTION

Entités et attributs

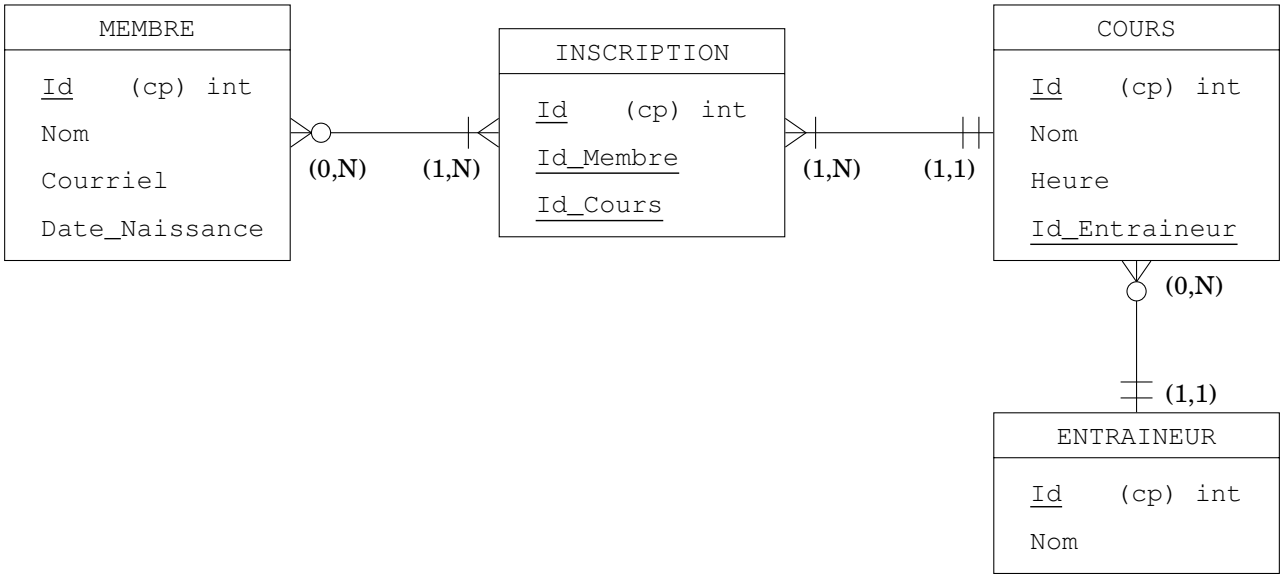
MEMBRE
Nom
Courriel
Date_Naissance

INSCRIPTION
... voir MLD

COURS
Nom
Heure

ENTRAINEUR
Nom

Schéma entités - relations avec clés primaires et étrangères (MLD)



Cardinalités dans le modèle conceptuel / logique

- Un membre peut s'inscrire à plusieurs cours (0,N)
- Une inscription est liée à un seul membre et un seul cours (1,1)
- Un cours peut avoir plusieurs inscriptions (1,N)
- Un cours peut exister sans entraîneur (0,N)
- Un entraîneur donne un et un seul cours à la fois (1,1)

ENTRAINEUR - COURS (0,N) => COURS contient donc Id_Entraineur (clé étrangère ce)

SGBD_TP02. Silence on lit !

Schéma conceptuel (MCD)

Une relation N-N (à droite) => Table intermédiaire EMPRUNT_LIVRE

Entités et attributs

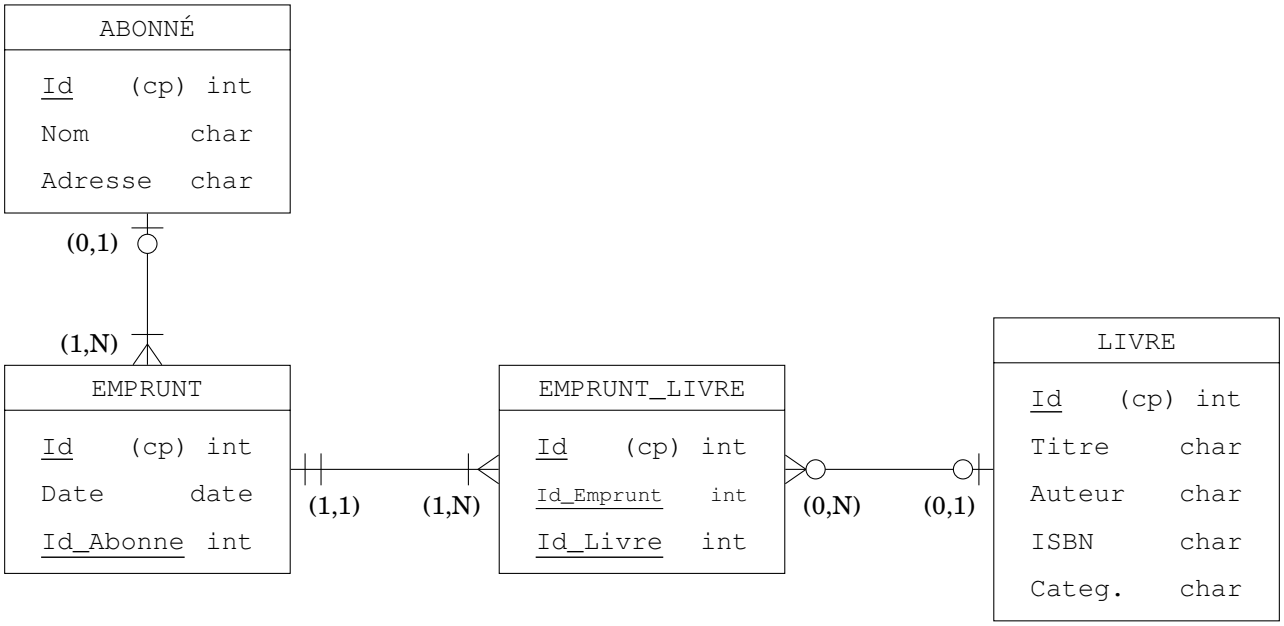
ABONNÉ
Nom
Adresse

EMPRUNT
Date

EMPRUNT_LIVRE
... voir MLD

LIVRE
Titre
Auteur
ISBN
Catégorie

Schéma entités - relations avec clés primaires et étrangères (MLD)



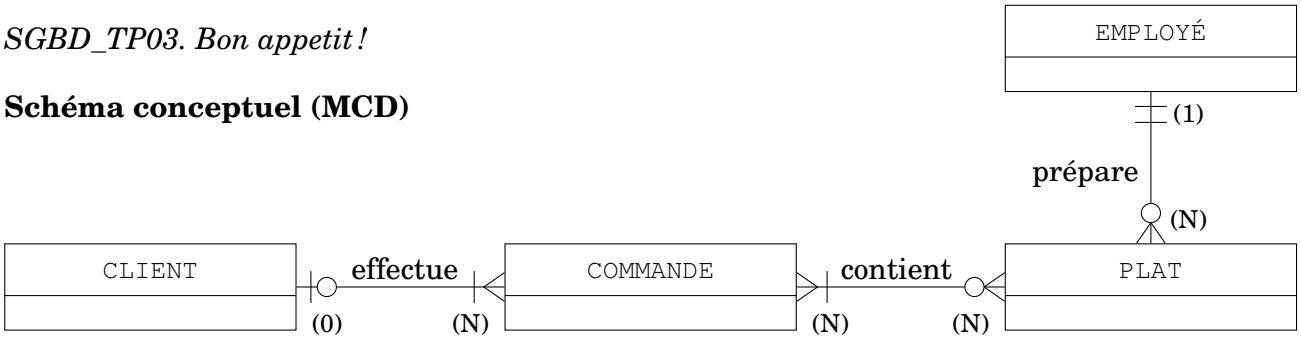
Cardinalités dans le modèle conceptuel / logique

- Un abonné peut ne pas effectuer d'emprunts (0,1)
- Un abonné peut effectuer plusieurs emprunts (1,N)
- Un emprunt est fait par un seul abonné (1,1)
- Un emprunt concerne au moins un livre (1,N)
- Un livre peut ne jamais être emprunté, mais s'il l'est, il peut l'être plusieurs fois (0,N)

ABONNE - EMPRUNT (1,N) => EMPRUNT contient donc Id_Abonne (clé étrangère ce)

SGBD_TP03. Bon appetit !

Schéma conceptuel (MCD)



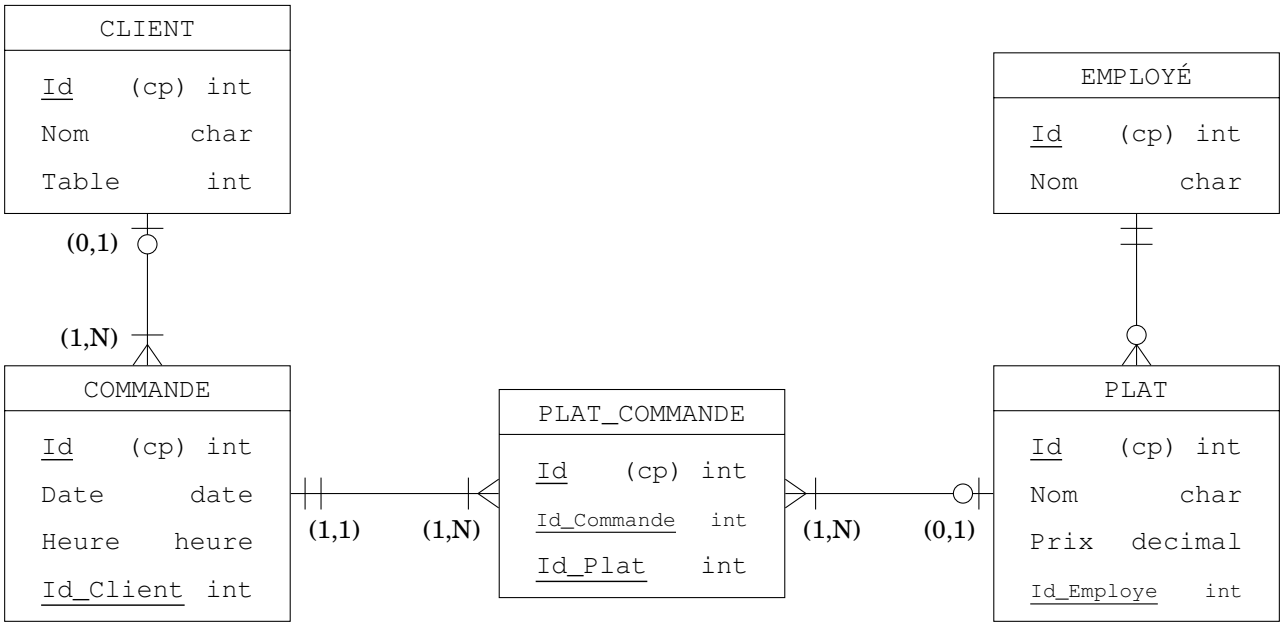
Une relation N-N (à droite) => Table intermédiaire PLAT_COMMANDE

Entités et attributs

CLIENT	**COMMANDE**	**PLAT_COMMANDE**	**PLAT**
Nom	Date	... voir MLD	Nom
Table	Heure		Prix

EMPLOYÉ
Nom

Schéma entités - relations avec clés primaires et étrangères (MLD)



Cardinalités dans le modèle conceptuel / logique

—

—

—

—

—

—

(clé étrangère ce)