

SNT 2nd GT

E. Suong

5 février 2026

Cours de SNT

I	Les données et leur traitement	4
1	Les données : structure, stockage, utilisations	5
1	Activités d'introduction	6
2	Données et bases de données	7
3	Stockage des données	9
4	Confidentialité des données	10
5	Activités de fin de chapitre	12
2	Formats et tailles de fichiers	13
1	Les fichiers informatiques	13
2	Les formats données structurées	16
II	Algorithmique et programmation	25
1	Algorithmique	26
1	Les variables informatiques	27
2	Les structures conditionnelles	28
3	Les structures itératives	29
4	Les fonctions	31
	Exercices	32
2	Programmer en Python	42
1	Les interfaces	43
2	Les logiciels	44
3	Python : variables et types	46
1	Affectation en Python	46
2	Les différents types	47
3	Entrer du texte en Python	51
4	(Bonus) Liste des mots-clés en Python	52
	Exercices	53
4	Python : fonctions et modules	55
1	Les fonctions	55
2	Les modules Python	57
3	(Bonus) Apprendre à lire les erreurs	60
	Exercices	61
5	Python : boucles et conditionnement	63
1	Les structures itératives	63
2	Les structures conditionnelles	64
	Exercices	67
6	Python : les listes	71
7	Les langages du web	72

III Projets de fin d'année

73

Thème d'étude I

Les données et leur traitement

1 Les données : structure, stockage, utilisations

Table des matières

1	Activités d'introduction	6
2	Données et bases de données	7
3	Stockage des données	9
4	Confidentialité des données	10
A	Les données sensibles	10
B	Les données ouvertes	11
5	Activités de fin de chapitre	12

Programme

Contenus	Capacités attendues
Données	Définir une donnée personnelle. Identifier les principaux formats et représentations de données.
Données structurées	Identifier les différents descripteurs d'un objet. Distinguer la valeur d'une donnée de son descripteur. Utiliser un site de données ouvertes, pour sélectionner et récupérer des données.
Traitement de données structurées	Réaliser des opérations de recherche, filtre, tri ou calcul sur une ou plusieurs tables.
Métadonnées	Retrouver les métadonnées d'un fichier personnel.
Données dans le nuage (cloud)	Utiliser un support de stockage dans le nuage. Partager des fichiers, paramétrer des modes de synchronisation. Identifier les principales causes de la consommation énergétique des centres de données ainsi que leur ordre de grandeur.

Les activités sont tirées de <https://ressources.forge.apps.education.fr/snt/Donnees/presentation>. Vous pouvez consulter ces ressources pour trouver des corrections.

1 – Activités d'introduction

Activité 1

- 1► Déterminer 5 critères permettant à une personne extérieure d'identifier directement n'importe qui dans la classe.
- 2► Proposer plusieurs manières de présenter ces informations. Quels sont leurs avantages et inconvénients ?

Activité 2

Regardez la vidéo [MOOC SNT / Données, comment les manipuler ?](#) de l'Inria - Médiation Scientifique sur YouTube, et répondez aux questions suivantes :

- 1► Qu'est-ce que l'informatique ?
- 2► Qu'est-ce qu'une donnée ?
- 3► Pourquoi avoir plusieurs données pour définir un objet ?
- 4► Pourquoi structurer des données ?
- 5► Comment structurer des données ?
- 6► Que sont les métadonnées, et à quoi servent-elles ?
- 7► Que sont les bases de données, et à quoi servent-elles ?

Activité 3

Dans votre quotidien ou celui de vos proches, quels applications, sites ou comportements génèrent des données ? À quoi servent ces données ?

Activité 4

Voici une liste d'événements :

- Utilisation des cartes perforées comme premier support de stockage de données :
- Invention du disque dur
- Invention du modèle relationnel pour la structuration et l'indexation des bases de données
- Loi informatique et liberté
- Création du premier tableur
- Open Gouvernement Initiative du président B. Obama
- Charte du G8 pour l'ouverture des données publiques
- Adoption du RGPD

- 1► Choisissez 5 événements dans la liste ci-dessus, et placez-les sur une frise chronologique. Pour chaque événement, vous placerez un autre événement qui s'est produit la même année (la page Wikipédia de l'année en question pourra vous être utile pour cela).
- 2► Sur une demi feuille blanche :
 - Inscrivez au recto un QCM : en question, le nom d'un événement, et en réponses possibles, des dates (ou l'inverse : en question, une date, et en réponses possibles, des événements)
 - Au verso, donnez la réponse et créez une mini fiche qui détaille l'événement (courte description, que s'est-il passé, pourquoi c'est important)
 - Si vous avez le temps, illustrez votre carte.

2 – Données et bases de données

Définition 5

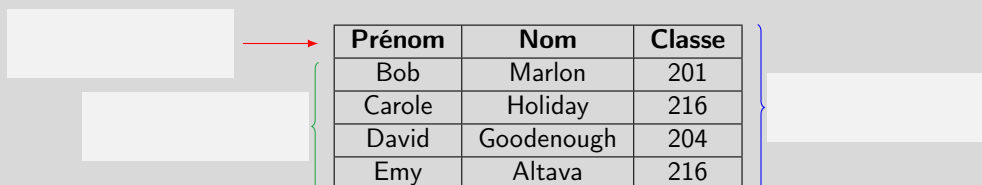
Donnée

Exemple 6

Définition 7

Données structurées

- Plusieurs peuvent être utiles pour décrire un même objet.
- Une regroupe des objets partageant les mêmes descripteurs.
- La structure de permet de présenter une collection : les objets en ligne, les descripteurs en colonne et les données à l'intersection. Les données sont alors dites .



Prénom	Nom	Classe
Bob	Marlon	201
Carole	Holiday	216
David	Goodenough	204
Emy	Altava	216

Exemple 8

Dans un carnet d'adresses,

- Descripteurs caractérisant un contact :
- L'ensemble des contacts forme une collection (de données) : ils ont tous un nom, un prénom, une adresse et un numéro de téléphone.

Définition 9

Base de données

Exemple 10

La base de données d'une bibliothèque conserve les données sur les livres, les abonnés et les emprunts effectués.

Remarque 11

- La recherche dans des données structurées a d'abord été effectuée selon une indexation préalable faite par l'homme. Des algorithmes ont ensuite permis d'automatiser l'indexation à partir de textes, d'images ou de sons.
- La recherche dans une base comportant plusieurs collections peut aussi croiser des collections différentes sur un descripteur commun ou comparable.

Une table de données peut faire l'objet de différentes opérations : rechercher une information précise dans la collection, trier la collection sur une ou plusieurs propriétés, filtrer la collection selon un ou plusieurs tests sur les valeurs des descripteurs, effectuer des calculs, mettre en forme les informations produites pour une visualisation par les utilisateurs.

Définition 12***Traitement des données***

Définition 13***Métadonnée***

Exemple 14

- Pour un document texte, les métadonnées peuvent être _____

- Pour une image, cela peut-être _____

- Pour une musique, _____

Information 15

3 – Stockage des données

Activité 16 (Travail de recherche)

Réaliser un document texte (avec Word, Libre Office, Open Office, ...) présentant différents supports de stockage de l'information à travers l'histoire. Vous devez :

- ▶ Réaliser une page de garde avec le sujet du document, vos nom, prénom et classe,
- ▶ Pour chaque support de stockage :
 - ▶ indiquer une date ou une période d'invention, éventuellement la ou les personnes impliquées dans son invention,
 - ▶ décrire brièvement son fonctionnement,
 - ▶ lister quelques cas d'usage les plus fréquents,
 - ▶ inclure une ou plusieurs illustrations,
- ▶ Tracer à la fin de votre document une frise chronologique plaçant les différents supports présentés.

Votre document ne doit pas faire plus de 5 pages *illustrations non comprises*. Vous êtes libre d'ajouter tout élément qui vous semble pertinent. **Vous devez inclure les URL des sources de vos informations.**

Pour guider vos recherches, voici une liste non exhaustive de mots-clés à explorer :

- | | | | |
|-------------------------|------------------|--------------------|---------------------|
| ▶ Écriture | ▶ Disquette | ▶ Disque dur | ▶ Donnée numérique |
| ▶ Gramophone | ▶ CD/DVD | ▶ SSD | ▶ Donnée analogique |
| ▶ Disque phonographique | ▶ Clé USB | ▶ Cassette | |
| ▶ Gravure de sillon | ▶ Carte perforée | ▶ Bande magnétique | ▶ Carte SD |

✨ L'utilisation des intelligences artificielles est tolérée dans la mesure où vous n'en copiez pas directement les résultats. Reformulez ses réponses avec vos propres mots. **Attention, l'IA n'est pas une source valide, et ne se substitue pas à des recherches supplémentaires.**

🧑 Vous avez le droit de recevoir de l'aide, mais votre document doit vous rester propre et unique. Votre document ne doit pas avoir strictement le même texte ni être présenté de la même manière qu'un autre.

Information 17

Lieux et accessibilité

Les données sont stockées sur différents supports :

- ▶ soit , c'est-à-dire à l'intérieur de la machine qui exploite cette donnée, comme un disque dur ou un SSD,
- ▶ soit , c'est-à-dire dans une support qui ne fait pas partie de la machine, comme une clé USB ou une carte SD.

Les données peuvent être accessibles :

- ▶ , c'est-à-dire depuis la machine qui les contient,
- ▶ , c'est-à-dire en les récupérant directement des données stockées dans une autre machine accessible *via* Internet (on parle alors de de données, ou de stockage).

Exercice 18

Donner, si possible, plusieurs exemples de données qui complètent les cases du tableau :

	Interne	Externe
Local		
Distant		

Remarque 19

Les supports pouvant subir des dommages entraînant des altérations ou des destructions des données, il est nécessaire de réaliser des sauvegardes sur plusieurs supports.

Remarque 20

Les grandes bases de données sont souvent implémentées sur des serveurs dédiés (machines puissantes avec une importante capacité de stockage sur disques). Les **centres de données** (**datacenter** en anglais) stockent ces serveurs mettant à disposition les données et des applications les exploitant.

Leur fonctionnement nécessite des ressources (en eau pour le refroidissement des machines, en électricité pour leur fonctionnement, en métaux rares pour leur fabrication) et génère de la pollution (manipulation de substances dangereuses lors de la fabrication, de la destruction ou du recyclage). De ce fait, les usages numériques doivent être pensés de façon à limiter la transformation des écosystèmes (notamment le dérèglement climatique) et à protéger la santé humaine.

Exercice 21 (Travail de recherche)

Chercher un ordre de grandeur de la consommation d'un centre de données, comparée à la consommation annuelle d'un certain nombre d'habitants.

4 – Confidentialité des données

A – Les données sensibles

Définition 22*Donnée personnelle*

L'évolution des capacités de stockage, de traitement et de diffusion des données fait qu'on assiste aujourd'hui à un phénomène de surabondance des données et au développement de nouveaux algorithmes capables de les exploiter.

L'exploitation de données massives (**Big Data** en anglais) est en plein essor dans des domaines aussi variés que les sciences, la santé ou encore l'économie. Les conséquences sociétales sont nombreuses tant en termes de démocratie, de surveillance de masse ou encore d'exploitation des données personnelles.

Mais on assiste aussi au développement d'un marché de la donnée où des entreprises collectent et revendent des données sans transparence pour les usagers. D'où l'importance d'un cadre juridique permettant de protéger les usagers, préoccupation à laquelle répond le **règlement général sur la protection des données (RGPD)**.

RGPD**Information 23**

Le **RGPD (Règlement Général sur la Protection des Données)** est une réglementation européenne en vigueur depuis le 25 mai 2018. Elle vise à protéger les données personnelles des individus et à harmoniser les pratiques au sein de l'Union européenne.

Ses principes fondateurs sont :

- ▶ Transparence : informer clairement la personne concernée.
- ▶ Finalité : collecter les données pour un objectif précis et légitime.
- ▶ Minimisation : ne recueillir que ce qui est strictement nécessaire.
- ▶ Sécurité : protéger les données contre tout accès ou usage non autorisé.
- ▶ Droits des personnes : Chaque individu dispose notamment du droit d'accès, de rectification, d'effacement (« droit à l'oubli »), de portabilité et d'opposition au traitement de ses données.

Lorsque nous naviguons sur Internet, de nombreux sites utilisent des traceurs (c'est souvent le cas avec certains cookies, prêtez attention à la fenêtre de consentement qui s'affiche à l'ouverture d'une page), qui enregistrent des informations sur notre activité : pages consultées, recherches effectuées, temps passé, produits regardés, localisation approximative... Ces données seuls semblent anodines, mais combinées avec celles provenant d'autres sites, elles permettent de créer un profil détaillé de chaque internaute.

Pour les entreprises, l'intérêt est avant tout commercial : ce profil détaillé permet une publicité ciblée et des recommandations personnalisées. Mais ces informations présentent également un risque : des escrocs peuvent exploiter ces informations pour créer des arnaques sur mesure, comme du **hameçonnage (phishing en anglais)** (faux messages ou sites imitant un service que nous utilisons), ou d'autres fraudes ciblées, par exemple une fausse offre commerciale adaptée à nos centres d'intérêt. Comprendre ce fonctionnement et régler les paramètres de confidentialité de nos appareils permet de réduire ces risques.

B – Les données ouvertes

Données ouvertes**Définition 24****Exemple 25**

Il peut s'agir d'articles de presse, de publications sur les réseaux sociaux, de vidéos en ligne, de documents officiels ou encore d'images satellites.

Activité 26 (OSINT)

Regarder la vidéo « [On vous dévoile les coulisses de nos enquêtes en OSINT](#) » du journal Le Monde sur YouTube et répondre aux questions suivantes.

ATTENTION : le sujet et les images de 4:25 à 5:44 peuvent être choquantes. Vous n'êtes pas obligés de visionner ce segment et il n'est pas utile

pour répondre aux questions.

- 1► Que signifie l'acronyme « OSINT » ?
- 2► Qu'est-ce que l'OSINT ?
- 3► Donner des exemples de sources d'informations exploitées en OSINT.
- 4► Quels éléments d'une image ou d'une vidéo peuvent donner des indications sur :
 - a► le lieu géographique ?
 - b► le jour ?
 - c► l'heure ?
- 5► Quel peut être l'intérêt l'OSINT ?
- 6► À quels dangers peut-on s'exposer lorsqu'on publie en ligne des données ?

5 – Activités de fin de chapitre

Activité 27

Définir les métiers du traitement des données listés ci-dessous :

- | | |
|--|---------------------------------|
| ► Analyste de données | ► Analyste financier |
| ► Data Scientist | ► Data Engineer |
| ► Ingénieur en traitement des données | ► Architecte de données |
| ► Analyste de business intelligence | ► Analyste en cybersécurité |
| ► Ingénieur en apprentissage automatique | ► Data Analyst en santé |
| ► Analyste de données marketing | ► Analyste en sciences sociales |

Activité 28

Voici une collection d'expressions :

Internet des objets (IoT)	Réplication des données	Entrée de données	Recherche de données
Cloud computing	Sélection des données	Algorithme de tri	Normalisation des données
Archivage des données	Scraping de données	Acquisition des données	Extraction de données
Perte de données	Sécurité des données	Indexation des données	Fuite de données
Agrégation des données	Erreur humaine	Stockage en mémoire flash	Capteurs
Transformation des données	Sondages	Gestion des identités et des accès (IAM)	Serveur de données
Enquêtes	Base de données distribuée	API	Structuration des données
Compression des données	Filtrage des données	Sécurité des données	Nettoyage des données
Collecte en temps réel	Violation de la vie privée	Conformité réglementaire	Piratage informatique

En effectuant des recherches, classer ces expressions entre les quatre catégories :

- | | | | |
|------------|-------|------------|-----------|
| ► COLLECTE | ► TRI | ► STOCKAGE | ► RISQUES |
|------------|-------|------------|-----------|

(Profitez du fait d'être en groupe pour vous répartir le travail !)

2 Formats et tailles de fichiers

Une partie des activités est inspirée de [d'un parcours Algorea](#). Vous pouvez le consulter en parallèle de ce cours.

1 – Les fichiers informatiques

A – Les formats de fichiers

Activité 1 (Classez les langues)

- 1► Vous avez 20 textes à disposition. Classez les textes selon la langue dans laquelle ils sont écrits. Est-ce simple ? Expliquer pourquoi.
- 2► Vous disposez de 20 autres textes, mais cette fois-ci une étiquette indique la langue dans laquelle le texte est écrit. Classez les textes selon leur langue. Pourquoi est-ce plus simple que précédemment ?
- 3► Dans quelle situation est-il plus simple de traduire les textes ? Expliquer pourquoi et comment.

Information 2

Exemple 3

- Extensions correspondant à des images : .jpg, .png, .webp, ...
- Extensions correspondant à des documents : .pdf, .odt, .docx, ...
- Extensions correspondant à des sons : .wav, .mp3, .ogg, ...
- Extensions correspondant à des vidéos : .mp4, .avi, .m4v, ...

Activité 4 (Erreur de traduction)

Essayez d'ouvrir une image avec le Bloc Notes.

- 1► Qu'observez-vous ?
- 2► Essayez de modifier un caractère, n'importe lequel. Sauvegardez et ouvrez l'image. Qu'observez-vous ?
- 3► Essayez d'expliquer la situation avec vos propres mots. Pourquoi après avoir modifié le fichier ainsi, l'image a-t-elle été corrompue ?

Information 5**Remarque 6**

Comme abordé dans l'exemple précédent, il existe plusieurs formats, et donc plusieurs extensions, pour un même type de fichier. Le choix d'un format dépend de l'usage que l'on fait du fichier.

Exemple 7

Pour une image :

- ▶ _____

- ▶ _____

Exemple 8

Pour un document textuel :

- ▶ _____

- ▶ _____

►

Remarque 9

B – Les unités de taille de fichiers

Information 10

Dans un appareil numérique, l'information est traitée et stockée en *binaire*, c'est-à-dire à l'aide d'une succession de 0 et de 1 (respectivement un état « éteint », autrement dit le courant ne passe pas, et un état « allumé », autrement dit le courant passe).

Bit, octet**Définition 11**

L'unité fondamentale d'information numérique, autrement dit l'alternative « 0 ou 1 », est appelée *bit*. Un *octet* (*byte* en anglais, prononcé « *baïte* ») est un regroupement de 8 bits.

Information 12

L'octet est traité comme les unités physiques (le gramme, le mètre, ...), et est souvent affublé des différents préfixes du Système international :

Unité	Symbole	Valeur
Octet	o	1o
Kilo-octet	Ko	10^3o
Méga-octet	Mo	10^6o
Giga-octet	Go	10^9o
Téra-octet	To	10^{12}o
Péta-octet	Po	10^{15}o

Remarque 13

On ne parle pas de milli-octet ou de déci-octet car, contrairement aux grammes, aux mètres, ..., 1 octet ne peut pas être indéfiniment découpé en mesures plus petites.

Exemple 14

- ▶ Un cours de mathématiques complet de seconde, de plus de 200 pages, pèse un peu moins de 2Mo.
- ▶ Une photo prise avec un smartphone récent pèse un peu moins de 5Mo.
- ▶ Une musique de qualité moyenne pèse environ 1Mo par minute.
- ▶ Un vidéo filmée en 4K pèse de l'ordre de 2Go par minute.
- ▶ On estime que plus de 29 000Go de données transitent par Internet chaque seconde dans le monde.
- ▶ On estime que le volume total de données sur le Web est de l'ordre de 6×10^{22} octets.

2 – Les formats données structurées

A – Les formats CSV et TSV

Activité 15 (Relevé de notes)

- 1▶ Dans son carnet de notes, un professeur a écrit :

NOTES DU 1ER TRIMESTRE

À la 1ère évaluation, Bob a eu 40, David a eu 55, Alice a eu 60 et Carole a eu 50.

À la 2e évaluation, Carole a eu 80, Bob a eu 50, Alice a eu 65 et David a eu 55.

- a▶ Décrivez les inconvénients de cette présentation.
 - b▶ Proposer une manière plus pertinente de présenter les notes des élèves aux différentes évaluations du trimestre.
Rédiger un court texte expliquant comment lire votre présentation.
 - c▶ Expliquer les avantages de votre présentation.
- 2▶
- a▶ Présenter de nouveau les notes des élèves, mais avec une contrainte : vous n'avez pas le droit de tracer des traits (autrement dit vous pouvez uniquement écrire du texte).
 - b▶ Donner un algorithme (une règle) permettant de transformer cette présentation en tableau classique.
- 3▶ Sur la page suivante de son carnet de notes, le professeur a écrit :

NOTES DU 2EME TRIMESTRE

À la 1ère évaluation, Alice WONDERLAND a eu 40, Bob MARLON a eu 55, Carole HOLIDAY a eu 60 et David GOODENOUGH a eu 50. À la 2e évaluation, Alice WONDERLAND a eu 80, Bob MARLON a eu 50, Carole HOLIDAY a eu 65 et David GOODENOUGH a eu 55.

- a► De même que précédemment, présenter les notes des élèves en ne traçant aucun trait.
- b► L'algorithme formulé précédemment fonctionne-t-il toujours ? Si non, expliquer pourquoi et proposer une solution.

4► Sur la page suivante de son carnet de notes, le professeur a écrit :

NOTES DU 3EME TRIMESTRE

À la 1ère évaluation, Alice WONDERLAND a eu 40,5, Bob MARLON a eu 55,3, Carole HOLIDAY a eu 60 et David GOODENOUGH a eu 50,9. À la 2e évaluation, Alice WONDERLAND a eu 80,1, Bob MARLON a eu 50,6, Carole HOLIDAY a eu 65,7 et David GOODENOUGH a eu 55.

- a► De même que précédemment, présenter les notes des élèves en ne traçant aucun trait.
- b► L'algorithme formulé précédemment fonctionne-t-il toujours ? Si non, expliquer pourquoi et proposer une solution.

Information 16

Les formats CSV (*Comma-separated Values*) et TSV (*Tab-separated Values*) servent à représenter des *tableaux* de données.

Sur la première ligne se trouvent les descripteurs, et sur chacune des suivantes les données associées à chaque objet. Les éléments d'une même ligne sont séparés par des *virgules* ou des *points-virgules* dans un fichier CSV, et par des *tabulations*⁽¹⁾ dans un fichier TSV.

Exemple 17

Tableau de données

Article	Réserve	Prix
Cookie	50	1
Éclair	30	2
Flan	3	15

Fichier CSV correspondant

```
Article,Réserve,Prix
Cookie,50,1
Éclair,30,2
Flan,3,15
```

Fichier TSV correspondant

Article	Réserve	Prix
Cookie	50	1
Éclair	30	2
Flan	3	15

Remarque 18

Dans un fichier CSV, si une donnée contient une virgule (par exemple un nombre décimal), alors on l'encadre entre guillemets. Si une donnée contient des guillemets, alors on l'encadre également entre guillemets. On peut toujours délimiter une donnée entre guillemets dans le doute même si ce n'est pas nécessaire.

(1). Une *tabulation* est un caractère représenté par un espace large. La touche Tabulation est en général au-dessus de la touche Verr. maj.

Exemple 19

Tableau de données

Titre	Prix (€)
Le Seigneur des agneaux	32,50
"Hairy" Potter	57,99
Le Mouton de M. Seguin	12,30

Fichier CSV correspondant

```
Titre,Prix (€)
Le Seigneur des agneaux,"0,50"
""Hairy" Potter","1,40"
Le Mouton de M. Seguin,"2,25"
```

Fichier TSV correspondant

```
Titre          "Prix (€)"
"Le Seigneur des agneaux"  0,50
""Hairy" Potter"         1,40
"Le Mouton de M. Seguin"  2,25
```

Exercice 20

- 1► Traduire le tableau en un fichier CSV :

Candidat	Épreuve n°1	Épreuve n°2	Mention
Alice	5,6	17	Sans mention
Bob	12	19	Bien
Carole	20	17	Très bien
David	20	0	Sans mention

- 2► Traduire le fichier CSV suivant en tableau :

```
Date,Activité,Dépense
"14/03/15","Restaurant","56,89"
"15/03/15","Bowling","23,18"
"16/03/15","Restaurant","45,67"
```

Remarque 21

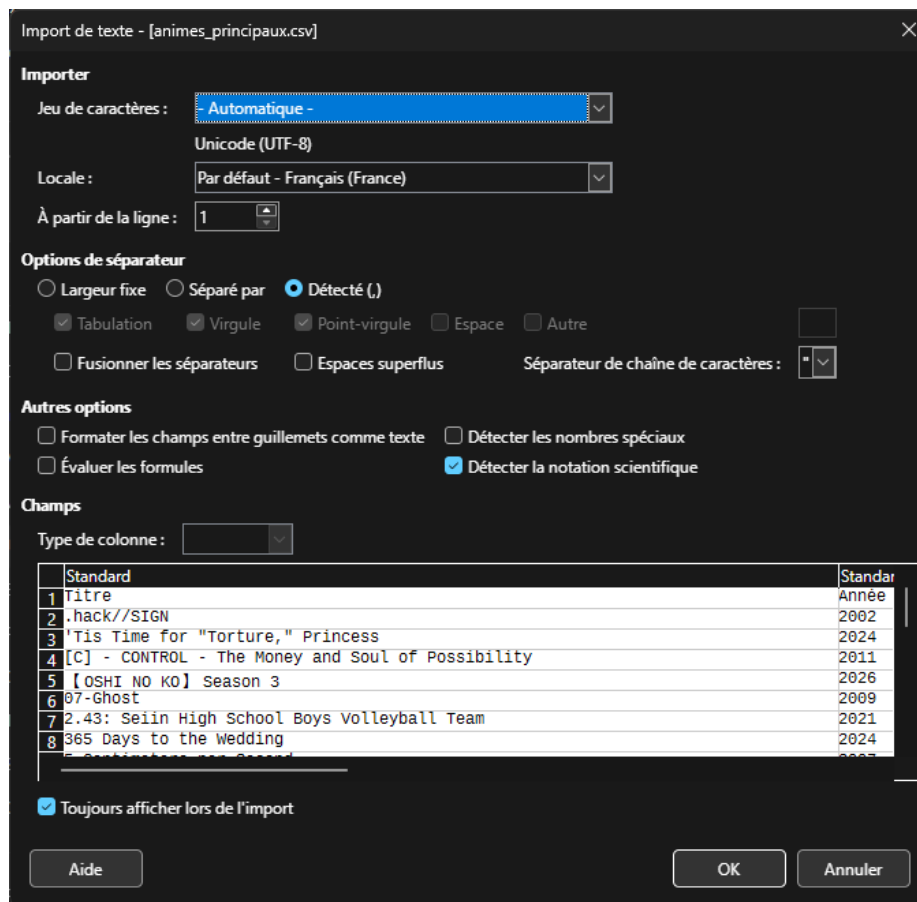
Le format TSV et le format CSV remplissent les mêmes fonctions, mais le format TSV est plus flexible (les guillemets sont moins souvent nécessaires), et peut être plus lisible humainement.

Remarque 22

Le Bloc Notes permet de lire et éditer un fichier CSV, mais cela est peu lisible pour un être humain. Un fichier CSV représentant un tableau de données, un logiciel tableur est plus adapté.

Activité 23 (Animés populaires)**Partie A : Ouverture des données**

- 1► Rendez-vous sur <https://esuong-maths.fr/snt-activites.html> et télécharger le fichier de l'activité.
- 2► Quelle est l'extension du fichier ?
- 3► Ouvrez le fichier avec le Bloc Notes ou Notepad.
 - a► Lisez la première ligne : quelles informations sur chaque animé semble contenir ce fichier ?
 - b► Essayez de trouver l'animé le plus ancien. Est-ce facile ? Expliquez pourquoi.
 - c► Quel symbole semble séparer les différentes informations sur une même ligne ?
- 4► Ouvrez le fichier avec LibreOffice. Cela vous affiche la fenêtre suivante :



Cliquez sur « OK ». Comment s'appelle le type de logiciel, ou de document affiché par LibreOffice ?

Partie B : Trier les données

- 1► Combien y a-t-il d'animés référencés ?
- 2► Surlignez les colonnes A à H (cliquez, maintenez et glissez le long des lettres des colonnes, tout en haut). Dans la barre de navigation, allez dans Données, puis cliquez sur AutoFiltre. À quoi servent les petites flèches qui sont apparues ?
- 3►
 - a► À l'aide de la petite flèche de l'AutoFiltre, triez les animés selon leur date de parution, du plus ancien au plus récent.
 - b► Surlignez les lignes 35 à 42 (cliquez, maintenez et glissez le long des numéros des lignes, tout à gauche). Dans la barre de navigation, allez dans Données, puis cliquez sur Trier. Une fenêtre apparaît.
 - c► Dans Clé de tri 1, sélectionnez la colonne E, et cliquez sur OK. Que s'est-il passé ? En déduire l'animé le mieux noté de 1993.
 - d► Répétez l'opération : dans Clé de tri 1, sélectionnez cette fois-ci la colonne F. Quel est l'animé le plus populaire de 1993 ?

Partie C : Exploitation des données

En vous servant des techniques apprises dans la partie B, **et selon ce que l'on peut lire directement dans la base de données**, répondre aux questions suivantes :

- 1► Quel est l'animé le plus populaire tiré d'une œuvre de :
 - a► Yoshihiro Togashi ?
 - b► Jun Mochizuki

- 2► Quel est l'animé le plus populaire du studio :
 - a► A-1 Pictures ?
 - b► CloverWorks ?
- 3► Quel est le troisième animé le plus populaire du studio MAPPA ?
- 4► Quel est l'animé :
 - a► le moins populaire ?
 - b► le plus populaire ?
- 5► Quel est l'animé :
 - a► le moins bien noté ?
 - b► le plus populaire parmi les mieux notés ?
- 6► Quel est l'animé le plus vieux ?
- 7► Quel est l'animé le plus ancien de :
 - a► Hajime Yatate ?
 - b► Mamoru Hosoda ?
 - c► Yasuhiro Nightow ?
- 8► Quel est l'animé le plus populaire des années :
 - a► 1980 à 1989 ?
 - b► 1990 à 1999 ?
 - c► 2000 à 2009 ?
 - d► 2020 à 2025 ?
- 9► Parmi les travaux du Studio Ghibli, l'œuvre la mieux notée est-elle la plus populaire ?
Si oui, préciser le titre de l'œuvre, si non, préciser les titres des deux œuvres concernées.
- 10► Parmi les travaux du Studio Pierrot, l'œuvre la mieux notée est-elle la plus populaire ?
Si oui, préciser le titre de l'œuvre, si non, préciser les titres des deux œuvres concernées.
- 11►
 - a► Parmi les films (les animés ayant 1 épisode), lequel est le plus populaire ?
 - b► Quel est l'animé avec le plus d'épisodes ?
- 12►
 - a► En triant la base de données de manière pertinente, déterminer le nombre d'animés faisant partie du genre "Action".
 - b► Pourquoi la méthode de tri utilisée à la question ne peut pas s'appliquer au genre "Comédie" ?
 - c► Proposer une méthode alternative pour déterminer le nombre d'animés de genre "Comédie".
- 13► Un peu d'esprit critique : la base de données permet-elle de déterminer la popularité totale de la saga *Demon Slayer : Kemetsu no Yaiba* ? Expliquer pourquoi/comment.

B – Les formats XML et JSON

Dans la continuité de l'activité précédentes, on considérera dans cette section le jeu de données suivant :

« L'animé *My Hero Academia* » est produit depuis **2016** par le studio *bones* d'après l'œuvre originale de **Kouhei Horikoshi**. Il enregistre **784051** vues pour une note de **76** sur 100. »

Le but de cette section est de découvrir de nouveaux formats pour organiser ces données.

Activité 24 (Limites du format CSV)

- 1► Essayez de présenter ces informations dans un tableau. Quelles difficultés rencontrez-vous ?
- 2► Le format CSV est-il pertinent pour présenter ces données ? Justifier.

Format XML

Information 25

Le format XML (*Extensible Markup Language*, extension .xml) est un autre format permettant de structurer des données. Il présente ces dernières sous la forme de paires « clé/valeur », autrement dit d'un nom décrivant la donnée, associé à ladite donnée.

Les clés sont des *balises* qui encadrent leur valeur correspondante : <clé>valeur</clé>. L'ensemble du jeu de données est lui-même encadré entre deux balises.

```
<oeuvre>
  <titre>My Hero Academia</titre>

  <genres>
    <genre>Action</genre>
    <genre>Aventure</genre>
    <genre>Comédie</genre>
  </genres>

  <production>
    <date>2016</date>
    <studio>bones</studio>
    <auteur>Kouhei Horikoshi</auteur>
  </production>

  <popularite>
    <notePublic>76</notePublic>
    <vues>784051</vues>
  </popularite>
</oeuvre>
```

Exemple 26

Un tableau peut être représenté au format XML. Il y a plusieurs manières d'organiser les données, l'exemple ci-dessous n'est qu'une façon parmi tant d'autres.

Tableau de données

Article	Réserve	Prix
Cookie	50	1
Éclair	30	2
Flan	3	15

Fichier XML correspondant

```
<patisserie>
  <article>
    <nom>Cookie</nom>
    <réserve>50</réserve>
    <prix>1</prix>
  </article>
  <article>
    <nom>Éclair</nom>
    <réserve>30</réserve>
    <prix>2</prix>
  </article>
  <article>
    <nom>Flan</nom>
    <réserve>3</réserve>
    <prix>15</prix>
  </article>
</patisserie>
```

Exercice 27

1► Traduire le tableau en un fichier XML :

Candidat	Épreuve n°1	Épreuve n°2	Mention
Alice	5,6	17	Sans mention
Bob	12	19	Bien
Carole	20	17	Très bien
David	20	0	Sans mention

2► Traduire le fichier XML suivant en tableau :

```
<notes>
  <matière>
    <nom>Maths</nom>
    <moyenne>85</moyenne>
    <coefficient>3</coefficient>
  </matière>
  <matière>
    <nom>Français</nom>
    <moyenne>70</moyenne>
    <coefficient>3</coefficient>
  </matière>
  <matière>
    <nom>Philo</nom>
    <moyenne>60</moyenne>
    <coefficient>8</coefficient>
  </matière>
</notes>
```

Format JSON**Information 28**

Le format JSON (*JavaScript Object Notation*, extension `.json`) remplit les mêmes fonctions que le JSON, mais sa présentation est différente, plus versatile et plus légère.

Le jeu de données est encadré entre accolades, les clés sont des textes (entre guillemets), suivi d'un deux-points et de la valeur associée. Une virgule sépare les différentes paires « clé/valeur ». Si la valeur est une liste de données, celle-ci est encadrée entre crochets. Si la valeur est elle-même un jeu de données, on ouvre une nouvelle paire d'accolades et on recommence.

```
{
  "titre": "My Hero Academia",
  "genres": ["Action", "Aventure", "Comédie"],
  "production": {
    "date": 2016,
    "studio": "bones",
    "auteur": "Kouhei Horikoshi"
  },
  "popularité": {
    "note public": 76,
    "vues": 784051
  }
}
```

Exemple 29

Un tableau peut être représenté au format JSON. Il y a plusieurs manières d'organiser les données, l'exemple ci-dessous n'est qu'une façon parmi tant d'autres.

Tableau de données

Article	Réserve	Prix
Cookie	50	1
Éclair	30	2
Flan	3	15

Fichier JSON correspondant

```
{
  "Cookie": {
    "réserve": 50,
    "prix": 1
  },
  "Éclair": {
    "réserve": 30,
    "prix": 2
  },
  "Flan": {
    "réserve": 3,
    "prix": 15
  }
}
```

Exercice 30

- 1► Traduire le tableau en un fichier JSON :

Candidat	Épreuve n°1	Épreuve n°2	Mention
Alice	5,6	17	Sans mention
Bob	12	19	Bien
Carole	20	17	Très bien
David	20	0	Sans mention

- 2► Traduire le fichier JSON suivant en tableau :

```
{
  "Alice": {
    "sport": "course",
    "temps": 37.9
  },
  "Bob": {
    "sport": "natation",
    "temps": 65.7
  },
  "Carole": {
    "sport": "vélo",
    "temps": 16.3
  }
}
```

Exercice 31

Voici un fichier JSON présentant des informations au sujet du jeu vidéo *Super Mario Galaxy*.

```
{
  "titre": "Super Mario Galaxy",
  "développement": {
    "développeur": "Nintendo EAD Tokyo",
    "éditeur": "Nintendo",
    "éditeur": "Yoshiaki Koizumi",
    "musique": {
      "compositeurs": ["Mahito Yokota", "Koji Kondo"],
      "chef d'orchestre": "Koji Haishima"
    },
    "producteurs": ["Shigeru Miyamoto", "Takao Shimizu"]
  },
  "date de parution": 2007,
  "genre": "Plates-formes",
  "consoles": ["Wii", "Nintendo Switch"]
}
```

En vous inspirant de l'exemple, créez un fichier JSON (entre 15 et 20 lignes avec une présentation similaire à l'exemple) sur le jeu *The Legend of Zelda : Breath of the Wild*.

Vous sélectionnerez les informations dans le bandeau résumé de la [page Wikipédia](#) du jeu. Vous n'êtes pas obligés de traduire toutes les informations, ni de vous restreindre aux informations dans le bandeau.

Thème d'étude II

Algorithmique et programmation

1 Algorithmique

Table des matières

1	Les variables informatiques	27
2	Les structures conditionnelles	28
A	Les blocs <i>SI... SINON SI...</i> , <i>SINON...</i>	28
B	Les opérateurs logiques	29
3	Les structures itératives	29
A	La boucle inconditionnelle (ou bornée)	29
B	La boucle conditionnelle (ou non bornée)	30
4	Les fonctions	31
	Exercices	32

Programme

Utiliser les variables et les instructions élémentaires

Contenus

- Variables informatiques de type entier, booléen, flottant, chaîne de caractères.
- Affectation (notée $\leftarrow$ en langage naturel).
- Séquence d'instructions.
- Instruction conditionnelle.
- Boucle bornée (for), boucle non bornée (while).

Capacités attendues

- Choisir ou déterminer le type d'une variable (entier, flottant ou chaîne de caractères).
- Concevoir et écrire une instruction d'affectation, une séquence d'instructions, une instruction conditionnelle.
- Écrire une formule permettant un calcul combinant des variables.
- Programmer, dans des cas simples, une boucle bornée, une boucle non bornée.
- Dans des cas plus complexes : lire, comprendre, modifier ou compléter un algorithme ou un programme.

Notion de fonction

Contenus

- Fonctions à un ou plusieurs arguments.
- Fonction renvoyant un nombre aléatoire. Série statistique obtenue par la répétition de l'appel d'une telle fonction.

Capacités attendues

- Écrire des fonctions simples ; lire, comprendre, modifier, compléter des fonctions plus complexes. Appeler une fonction.
- Lire et comprendre une fonction renvoyant une moyenne, un écart type. Aucune connaissance sur les listes n'est exigée.
- Écrire des fonctions renvoyant le résultat numérique d'une expérience aléatoire, d'une répétition d'expériences aléatoires indépendantes.

Définition 1**Algorithme**

Un *algorithme* est une suite finie d'instructions qui permet de faire quelque chose.

Exemple 2

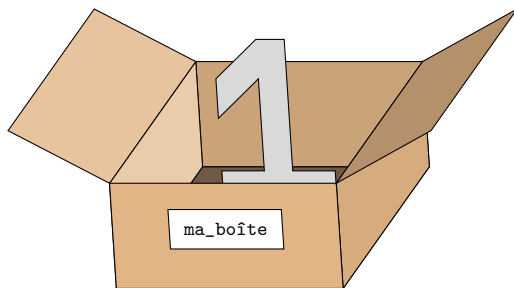
Une recette de cuisine est une sorte d'algorithme : c'est une suite d'instructions qui transforme des ingrédients en un plat. Les directions données par un GPS forment un algorithme, qui permettent d'aller d'un endroit à un autre.

1 – Les variables informatiques

Définition 3

- ▶ Une *variable* (informatique) est un élément dans la mémoire de l'ordinateur, qui possède un nom et contient une valeur.
- ▶ Le *type* d'une variable est la nature de la valeur qu'elle contient : un nombre entier, décimal, un mot, une image ...
- ▶ Lorsqu'on donne une certaine valeur à une variable, on dit qu'on lui *affecte* cette valeur.

On illustre souvent une variable par une *boîte* : une boîte qui porte un nom permettant de l'identifier, et contenant une certaine valeur que l'on peut sortir et remettre dans la boîte à volonté.



Cette boîte (cette variable), qui s'appelle `ma_boîte` contient la valeur 1.
On peut écrire `ma_boîte = 1`.

En *langage naturel*, l'affectation d'une valeur à une variable est représentée par une flèche. Par exemple, $X \leftarrow 5$ signifie « on affecte la valeur 5 à la variable X ». La variable X est donc de type *entier*.

Exemple 4

Voici ci-contre un exemple d'algorithme en langage naturel. Détaillons son fonctionnement par étapes :

- 1▶ On commence par affecter la valeur 2 à une variable qu'on appelle a.
- 2▶ Ensuite, on affecte la valeur 3 à une variable qu'on appelle b.
- 3▶ Ensuite, on affecte à une variable c la valeur de $a + b$: ici, on lui affecte la valeur 5, puisque $5 = 2 + 3$.
- 4▶ Enfin, on affiche la valeur contenue dans la variable c, autrement dit on affiche 5.

```

1  a ← 2
2  b ← 3
3  c ← a + b
4  AFFICHER c

```

Exercices conseillés : 23, 24

Remarque 5

Une *variable* s'appelle ainsi car son contenu est amené à varier, à changer le long du déroulement de l'algorithme. On peut donc réaffecter une nouvelle valeur à une variable déjà existante : auquel cas, l'ordinateur oublie

purement et simplement l'ancienne valeur, et ne garde que la nouvelle. C'est comme si on vidait le contenu de la boîte pour le remplacer.

On peut également redéfinir une variable à partir d'elle-même : par exemple, $a \leftarrow 2 * a$ indique que l'on remplace la valeur de a par son double.

Exercices conseillés : 25, 26, 27, 28

2 – Les structures conditionnelles

A – Les blocs *SI... SINON SI..., SINON...*

Définition 6

- ▶ Une *condition* est la valeur de vérité d'une phrase pouvant dépendre d'une variable. Une condition est donc soit Vraie, soit Fausse.
- ▶ Une *structure conditionnelle* est une suite de *blocs d'instructions* que l'on exécute si une certaine condition est Vraie, et que l'on n'exécute pas sinon.
- ▶ Les blocs d'instructions sont signalés en décalant les instructions d'un cran vers la droite : on dit qu'on les *indente* (l'espace est appelée un *indentation*).
- ▶ Une variable qui vaut soit Vrai, soit Faux est dite *booléenne* (ou de *type booléen*).

Exemple 7

Cet algorithme contient une structure conditionnelle (composée de deux blocs d'instructions) qui dépend de la variable *âge* : si celle-ci est supérieure ou égale à 18, alors l'algorithme affiche le texte "Vous êtes majeur", et dans le cas contraire (c'est-à-dire si *âge* est strictement inférieur à 18), il affiche le texte "Vous êtes mineur". En langage naturel, la structure conditionnelle est délimitée, en langage naturel, par SI et FIN SI.

```

1  SI âge ≥ 18:
2      AFFICHER "Vous êtes majeur"
3  SINON:
4      AFFICHER "Vous êtes mineur"
5  FIN SI

```

Remarque 8 (Enchaîner les test)

On peut aussi enchaîner les tests les uns après les autres avec le mot-clé SINON SI... :

```

1  SI x ≥ 5:
2      AFFICHER "x est plus grand que 5:"
3  SINON SI x ≥ 3:
4      AFFICHER "x est plus grand que 3"
5  SINON:
6      AFFICHER "Je ne sais pas..."
7  FIN SI

```

Cette structure conditionnelle est constituée de trois blocs. Tout d'abord, l'algorithme va vérifier si $x \geq 5$: si c'est effectivement le cas, alors l'algorithme affiche "x est plus grand que 5" et ignore les autres blocs d'instruction de la structure conditionnelle. Dans le cas contraire, il lance le test suivant, et ainsi de suite. Le bloc SINON est exécuté seulement quand tous les tests précédents ont échoué.

Notez bien cette subtilité : l'algorithme exécute un et un seul bloc d'instructions dans chaque structure conditionnelle, même si parfois les conditions se superposent. Par exemple avec $x = 6$, l'algorithme ci-dessus affiche "x est plus grand que 5" car c'est le premier test réussi – qu'importe que le deuxième soit aussi valide.

Remarque 9

Le bloc **SINON** : est optionnel : si on ne souhaite exécuter aucune instruction dans le cas où tous les tests échouent, il suffit de ne pas écrire le bloc **SINON** :

Exercices conseillés : 29, 30, 31, 32, 33

B – Les opérateurs logiques

Définition 10

Soient A et B deux variables booléennes.

- ▶ Le booléen A **ET** B est le booléen qui est Vrai si, et seulement si A et B sont simultanément Vrais.
- ▶ Le booléen A **OU** B est le booléen qui est Vrai si, et seulement si au moins l'une des deux variables A ou B est vraie Vrais.
- ▶ Le booléen **NON** A est le booléen qui est Vrai si, et seulement si A est Faux.

On résume cette définition dans des *tables de vérité* :

A	B	A ET B
V	V	V
V	F	F
F	V	F
F	F	F

A	B	A OU B
V	V	V
V	F	V
F	V	V
F	F	F

A	NON A
V	F
F	V

Remarque 11

Parfois, on entoure les conditions de parenthèses, soit pour améliorer la lisibilité de l'algorithme, soit parce que c'est nécessaire.

Exercice conseillé : 34

3 – Les structures itératives

Définition 12

- ▶ Une *structure itérative*, ou *boucle*, est une suite d'instruction qu'on répète un certain nombre de fois.
- ▶ Chaque exécution du bloc d'instruction est appelée *itération* de la boucle.

A – La boucle inconditionnelle (ou bornée)

Définition 13

Une *boucle inconditionnelle*, ou *boucle bornée*, est un bloc d'instructions que l'on répète un nombre de fois déterminé à l'avance.

Une boucle inconditionnelle est délimitée, en langage naturel, par **POUR** et **FIN POUR** (à ce titre, on l'appelle également « *boucle Pour* »).

Exemple 14

Dans l'algorithme suivant, le *compteur* n prend successivement les valeurs 1, puis 2, puis 3, ... jusqu'à 10. Pour chaque valeur de n , l'algorithme exécute les instructions du bloc d'instructions repéré par une indentation : ici, on affiche le double de n (on a donc affiché la table de 2!).

```

1  POUR n ALLANT DE 1 A 10:
2      AFFICHER 2 * n
3  FIN POUR

```

Exercices conseillés : 35, 36, 37, 38

B – La boucle conditionnelle (ou non bornée)**Définition 15**

Une *boucle conditionnelle*, ou *boucle non bornée*, est un bloc d'instructions que l'on répète un nombre de fois inconnu en avance.

Une boucle conditionnelle est délimitée, en langage naturel, par TANT QUE et FIN TANT QUE (à ce titre, on l'appelle aussi « *boucle Tant que* »).

Exemple 16

Dans cet algorithme, la variable x prend successivement les valeurs 0, puis 2, puis 4, et ainsi de suite, jusqu'à ce que la condition $2 * x < 10$ ne soit plus vérifiée (autrement dit, ici jusqu'à ce que x dépasse la valeur 5).

```

1  x ← 0
2  TANT QUE 2*x < 10:
3      x ← x + 2
4  FIN TANT QUE

```

Remarque 17

Il est possible de ne pas exécuter le bloc d'instruction d'une boucle Tant que si sa condition est fausse dès le départ. Par exemple, l'algorithme suivant n'affichera rien.

```

1  x ← 10
2  TANT QUE x ≤ 5:
3      AFFICHER "x est plus petit que 5"
4  FIN TANT QUE

```

Définition 18

Incrémenter une variable (de type entier), c'est augmenter sa valeur de 1.

Remarque 19

Il est possible de bricoler une boucle Pour à partir d'une boucle Tant que en créant un compteur que l'on incrémente à la main à chaque itération. En revanche, il n'est pas possible de bricoler une boucle Tant que à partir d'une boucle Pour!

En pratique, il est préférable d'utiliser une boucle Pour dès que c'est possible, car une boucle Pour est plus lisible et plus sûre d'utilisation. En effet, la boucle Tant que ci-contre tourne indéfiniment!

```

1  x ← 10
2  TANT QUE x > 0:
3      x ← x * 2
4  FIN TANT QUE

```

Exercices conseillés : 39, 40, 41, 42

4 – Les fonctions

Définition 20

- ▶ Une *fonction* est un bloc d'instructions qui porte un nom, et qui est exécuté uniquement sur demande.
- ▶ Une fonction peut admettre en entrée des *arguments* (ou *paramètres*), et peut, après avoir exécuté ses instructions, *renvoyer* un résultat.
- ▶ Dans l'algorithme principal, quand on utilise une fonction, on dit qu'on *appelle* cette fonction.

Une fonction est délimitée, en langage naturel, par **FONCTION** et **FIN FONCTION**. Si une fonction admet des arguments, on les précise entre parenthèse, séparés par des virgules, après le nom de la fonction (on laisse une paire de parenthèses vide s'il n'y a pas d'arguments). Le bloc d'instructions définissant une fonction est signalé par une indentation.

Pour appeler une fonction, on écrit son nom dans l'algorithme, suivi de ses paramètres entre parenthèses (ou une paire de parenthèses vide).

Exemple 21

Dans de nombreux jeux ou certaines activités d'initiation à la programmation, un personnage peut faire certaines actions, déclenchées à l'aide de fonctions. Par exemple `avancer()`, `droite()`, `poser()`. Certains tests sont aussi effectués grâce à des fonctions qui renvoient un booléen, par exemple `est_bleu()`, `mur_devant()`, `est_libre()`, ...

Exemple 22

Dans les exemples suivants, on propose une fonction à gauche, et un exemple

- 1▶ La fonction `BonjourMonde` suivante ne prend pas d'argument en entrée et ne renvoie aucune valeur. Elle se contente d'afficher "Bonjour, monde !" et "Hello, world !".

```
1  FONCTION BonjourMonde():
2      AFFICHER "Bonjour, monde !"
3      AFFICHER "Hello, world !"
4  FIN FONCTION
```

```
1  AFFICHER "Salutation du monde en
2      cours..."
3  BonjourMonde()
4  AFFICHER "Le monde est salué !"
```

- 2▶ La fonction `Doubler` suivante prend comme argument un nombre, et renvoie son double.

```
1  FONCTION Doubler(n):
2      RENVOYER 2 * n
3  FIN FONCTION
```

```
1  AFFICHER "Le double de 5 est:"
2  AFFICHER Doubler(5)
3  AFFICHER "Et le double de 8 est:"
4  AFFICHER Doubler(8)
```

- 3▶ La fonction `Additionner` suivante prend comme argument trois nombres, et renvoie leur somme après avoir affiché un petit texte.

```
1  FONCTION Additionner(a, b, c):
2      AFFICHER "On additionne
3          tout ça !"
4      RENVOYER a + b + c
5  FIN FONCTION
```

```
1  AFFICHER "Prenons les nombres 1, 2
2      et 3..."
3  AFFICHER Additionner(1, 2, 3)
```

Exercices conseillés : 43, 44, 45, 46, 47

Exercices

Variables et affectation

Exercice 23

Donnez les valeurs affichées dans les deux algorithmes ci-dessous (remarque : le symbole $*$ indique une multiplication) :

```
1  facteur_1 ← 8
2  facteur_2 ← 5
3  resultat ← facteur_1 * facteur_2
4  AFFICHER resultat
```

```
1  x ← 6
2  y ← 7
3  z ← x * y - 2 * x
4  AFFICHER z - 3 * y
```

Exercice 24 (Déclic 2022)

On considère l'algorithme ci-contre, où la variable x est un nombre donné en amont. Les affirmations suivantes sont-elles vraies ou fausses ? Justifier.

```
1  a ← x + 1
2  b ← a*a
3  c ← 3*b
```

- 1► Si la variable x contient la valeur 2 avant l'exécution de l'algorithme, alors la variable c contiendra 27.
- 2► Si la variable x contient la valeur -3 avant l'exécution de l'algorithme, alors la variable c contiendra -12.
- 3► La variable c contient toujours un réel positif.

Exercice 25 (Tableau d'avancement)

Compléter le tableau suivant représentant les valeurs des variables x et y à chaque étape de l'algorithme :

```
1  x ← 3
2  y ← 2
3  x ← 2*x - 1
4  y ← y*y + 3
5  x ← x - y
6  y ← x*y
```

Étape	x	y
1		
2		
3		
4		
5		
6		

Exercice 26

Donnez les valeurs affichées dans les deux algorithmes ci-dessous (remarque : le symbole $/$ indique une division) :

```
1  a ← 12
2  a ← a - 4
3  b ← 2
4  c ← a / b
5  b ← 10
6  AFFICHER c
```

```
1  a ← 5
2  b ← 0
3  b ← a
4  a ← a / b
5  AFFICHER a
```

Exercice 27 (Déclic 2022)

On considère le programme de calcul suivant : « choisir un nombre X , prendre son double, soustraire 1, puis élever au carré. »

Parmi les algorithmes ci-dessous, indiquer ceux pour lesquels la variable X contient le résultat du programme de calcul à la fin de l'exécution.

Algo A

```
1  A ← 2*X
2  B ← A - 1
3  X ← B*B
```

Algo B

```
1  X ← 2*X
2  X ← X - 1
3  X ← X*X
```

Algo C

```
1  X ← 2*X
2  X ← 2*X - 1
3  X ← (2*X - 1)*(2*X - 1)
```

Exercice 28

Écrire un algorithme permettant d'échanger les valeurs de deux variables a et b .

Structures conditionnelles et logique

Exercice 29 (Déclic 2022)

On considère l'algorithme ci-contre où N est une variable entière.

```

1  SI  $N \geq 10$ :
2     $X \leftarrow 2*N$ 
3  SINON:
4     $X \leftarrow N + 3$ 
5  FIN SI

```

- 1► Si N vaut 5, alors X vaut-elle 10, 8 ou 5 ?
- 2► Si N vaut 10, alors X vaut-elle 20, 13 ou 10 ?
- 3► Si N vaut 15, alors X vaut-elle 30, 18 ou 15 ?

Exercice 30

Qu'affichent les deux algorithmes suivants pour $x = 2$? Et pour $x = 8$?

Algo A :

```

1  SI  $x \leq 5$ :
2    AFFICHER "Plus petit que 5"
3  SINON SI  $x \leq 10$ :
4    AFFICHER "Plus petit que 10"
5  FIN SI

```

Algo B :

```

1  SI  $x \leq 5$ :
2    AFFICHER "Plus petit que 5"
3  FIN SI
4
5  SI  $x \leq 10$ :
6    AFFICHER "Plus petit que 10"
7  FIN SI

```

Exercice 31

Les variables a et b représentent des nombres. Que fait l'algorithme suivant ?

```

1  SI  $a > b$ :
2     $M \leftarrow a$ 
3  SINON:
4     $M \leftarrow b$ 
5  FIN SI
6  AFFICHER M

```

Exercice 32

L'algorithme suivant affiche un message en fonction du score dans un jeu, représenté par la variable score.

```

1  SI score = 100:
2    AFFICHER "Score parfait !"
3  SINON SI score  $\geq 75$ :
4    AFFICHER "Excellent !"
5  SINON SI score  $\geq 50$ :
6    AFFICHER "Bravo !"
7  SINON:
8    AFFICHER "Dommage !"
9  FIN SI

```

Qu'affiche ce programme pour les valeurs de score suivantes ?

► 100 ► 90 ► 70 ► 50 ► 24 ► 0

Exercice 33

Écrire un algorithme qui affiche la plus grande valeur entre trois variables a , b et c .

Exercice 34

Qu'affiche l'algorithme suivant...

Lorsque jour = "Mardi" et meteo = "Pluie" ?

Lorsque jour = "Dimanche" et meteo = "Soleil" ?

```

1  SI jour = "Lundi" ET meteo = "Pluie":
2    AFFICHER "Au boulot..."
3  SINON SI (NON jour = "Dimanche") OU meteo = "Pluie":
4    AFFICHER "N'oublions pas le parapluie"
5  SINON SI (NON jour  $\neq$  "Dimanche") OU (NON meteo  $\neq$  "Pluie"):
6    AFFICHER "Direction la plage !"
7  SINON:
8    AFFICHER "Que vais-je faire aujourd'hui..."
9  FIN SI

```

Boucle bornée

Exercice 35 (Déclic 2022)

Dans l'algorithme ci-dessous, la variable N désigne un entier plus grand que 2.

```

1  S ← 0
2  POUR i ALLANT DE 1 A N:
3      S ← S + 10
4  FIN POUR

```

1► Dans cette question on choisit $N = 3$ avant d'exécuter l'algorithme.

a► Recopier et compléter le tableau d'avancement suivant :

i	S	N
	0	
1		
2		
⋮	⋮	⋮

b► En déduire la valeur de S à la fin du programme.

2► Déterminer la valeur finale de S lorsqu'on donne à N la valeur...

a► 4 b► 5 c► 6 d► 10

Exercice 36

Déterminer la valeur de la variable S à l'issue de cet algorithme :

```

1  S ← 0
2  POUR n ALLANT DE 1 A 10:
3      S ← S + n
4  FIN POUR

```

Exercice 37 (Math'x)

Déterminer l'affichage des deux scripts suivants :
Algo A :

```

1  a ← - 5
2  POUR k ALLANT DE 1 A 5:
3      a ← 10*a - 8
4  FIN POUR
5  AFFICHER a

```

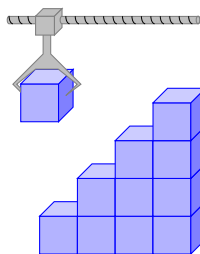
Algo B :

```

1  a ← - 5
2  POUR k ALLANT DE 1 A 5:
3      a ← 10*a - 8
4      AFFICHER a
5  FIN POUR

```

Exercice 38 (Défi)



Une pince mécanique dépose des blocs (on supposera que le prochain bloc apparaît magiquement entre ses crochets après avoir posé le bloc précédent). La pince dispose de trois commandes :

- Écrivez `droite()` dans le programme pour que la pince se décale d'un cran vers la droite.
- Écrivez `gauche()` dans le programme pour que la pince se décale d'un cran vers la gauche.
- Écrivez `poser()` pour que la pince pose un bloc.

Objectif : écrire un programme qui construit un escalier à N marches (sur l'illustration, $N = 4$).

Boucle non bornée

Exercice 39

- 1► Déterminez la valeur de la variable e à l'issue de cet algorithme.
- 2► Interpréter cet algorithme.

(Remarque : la notation $**$ indique l'opération Puissance. Par exemple, $3**2 = 3^2 = 9$.)

```

1  e ← 1
2  TANT QUE 2**e < 500:
3      e ← e + 1
4  FIN TANT QUE

```

Exercice 40 (Suite de Syracuse)

On construit une suite de nombres de la manière suivante : on choisit un nombre, et :

- si ce nombre est pair, alors on le divise par 2,
- sinon, on le triple et on ajoute 1,

```

1  ...
2  TANT QUE ...:
3      SI N%2 ...:
4          N ← ...
5      SINON:
6          N ← ...
7      FIN SI
8      ...
9  FIN TANT QUE

```

et on recommence avec le nouveau résultat, ainsi de suite.

La conjecture de Syracuse stipule que quelque soit le nombre que l'on choisit au départ, on finit par atteindre 1.

Compléter le programme ci-dessus pour qu'il calcule et affiche les nombres successifs de la suite. On rappelle que $N\%2$ vaut le reste de la division euclidienne de N par 2.

Exercice 41 (Objectif bac)

On modélise l'évolution du nombre d'abeilles (en milliers) dans un rucher. Au départ, il y a 1 000 abeilles. À cause de la pollution, 20% du rucher se meurt chaque mois. Pour pallier leur disparition, on introduit 20 milliers d'abeilles à la fin de chaque mois. On donne l'algorithme correspondant à la fin de l'énoncé.

- 1► Recopier et compléter le tableau d'avancement suivant :

abeilles	n	abeilles > 500 ?
	1	
1000	1	Vrai
⋮	⋮	⋮

- 2► Déterminez la valeur de la variable n à l'issue de cet algorithme.
- 3► Interpréter cette valeur.

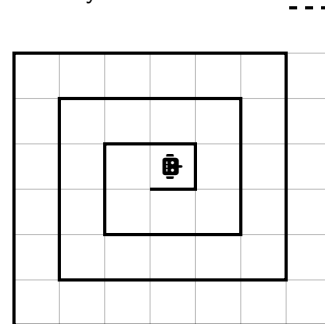
```

1  n ← 1
2  abeilles ← 1000
3  TANT QUE abeilles > 500:
4      abeilles ← 0.8*abeilles + 20
5      n ← n + 1
6  FIN TANT QUE

```

Exercice 42 (Défi)

Votre robot est au centre d'un « labyrinthe » en forme de spirale comme illustré ci-dessous. Vous devez mettre à jour son programme pour le faire sortir de là, mais problème... il ne vous reste de la place que pour 6 lignes de code, et vous ne savez pas quelle est la taille du labyrinthe !



Votre robot dispose des commandes suivantes :

- Effectuez le test `SI mur_devant()` : (ou analogue) pour savoir si votre robot fait face à un mur et réagir en conséquence.
- Écrivez `avancer()` pour que votre robot avance d'un pas.
- Écrivez `droite()` pour que le robot tourne sur lui-même vers la droite. De même avec `gauche()` pour tourner à gauche.
- Effectuez le test `SI est_libre()` : (ou analogue) pour savoir si le robot est sorti du labyrinthe.

Comment allez-vous vous sortir de cette situation ?

Fonctions

Exercice 43 (Déclic 2022)

On considère la fonction `mystere` ci-contre.

- 1► Quels sont les paramètres de cette fonction ?
- 2► Quels sont les renvois des appels suivants ?

a► `mystere(3,4)` b► `mystere(12,5)`

```

1  FUNCTION mystere(x, y):
2      SI x < 10:
3          RENVOYER 3*x - y
4      SINON:
5          RENVOYER x + y
6      FIN SI
7  FIN FONCTION

```

```

1  FUNCTION OperationMystere(a, b):
2      q ← 0
3      TANT QUE a > b:
4          q ← q + 1
5          a ← a - b
6      FIN TANT QUE
7      RENVOYER q, a - b * q
8  FIN FONCTION

```

Exercice 44

On définit une fonction `rectangle` qui renvoie l'aire et le périmètre d'un rectangle dont les dimensions sont données en paramètres.

Compléter cette fonction.

```

1  FUNCTION rectangle(a, b):
2      aire = ...
3      peri = ..
4      RENVOYER ..., ...
5  FIN FONCTION

```

Exercice 46

Programmer une fonction `f` qui prend en paramètre un nombre x et renvoie $3x + 2$ si $x < 1$, et $5 - 2x$ sinon.

Exercice 45

On définit la fonction `OperationMystere` ci-dessous, qui prend deux nombres en paramètres et renvoie deux nombres.

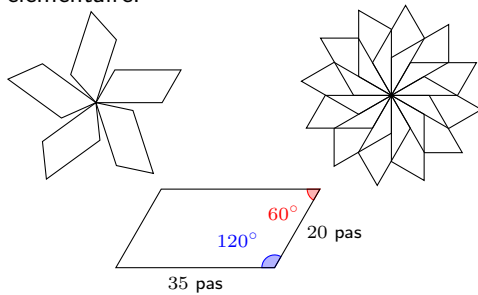
- 1► Que renvoient les appels suivants ?
 - a► `OperationMystere(10, 3)`
 - b► `OperationMystere(3, 10)`
 - c► `OperationMystere(16, 2)`
- 2► À quoi sert cette fonction ?

Exercice 47

Un logiciel de dessin par programmation dispose des commandes suivantes :

- `poser_crayon()` pour commencer à dessiner,
- `lever_crayon()` pour arrêter de dessiner,
- `avancer(x)` pour avancer de x unités vers l'avant,
- `droite(angle)` pour tourner le crayon d'un angle de $angle$ degrés dans le sens horaire .
- `gauche(angle)` similaire à la précédente, mais dans le sens inverse,

On veut dessiner un motif de fleur à partir d'un pétale élémentaire.



Un pétale

- 1▸ Compléter la fonction `Petale` ci-dessous qui dessine un pétale élémentaire du motif.

```

1  FONCTION Petale():
2      ...
3      POUR i ALLANT DE ... A ...:
4          avancer(...)
5          gauche(...)
6          avancer(...)
7          gauche(...)
8      FIN POUR
9      ...
10  FIN FONCTION

```

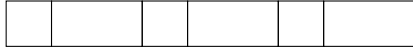
- 2▸ Écrire la fonction `Fleur` qui prend en paramètre le nombre de pétales à dessiner et dessine la fleur avec des pétales également espacés.

Problèmes de géométrie

On reprend les fonctions de l'exercice 47.

Exercice 48

- 1► Écrire une fonction `Carre` qui trace un carré de côté 50 unités.
- 2► Écrire une fonction `Rectangle` qui trace un rectangle long de 100 unités et large de 50 unités.
- 3► Écrire un programme réalisant la frise suivante :



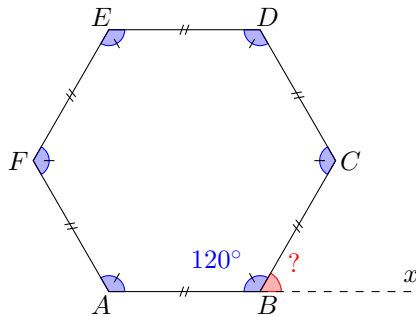
- 4► Écrire un programme réalisant la frise suivante :



Exercice 49

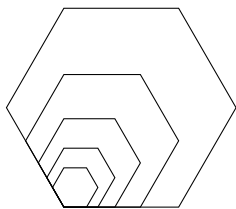
Un hexagone régulier est un polygone à 6 côtés de même longueur, et dont tous les angles mesurent 120° .

- 1► Quelle est la mesure de l'angle $\widehat{CBx}$ ci-dessous ?



Dans chacune des questions suivantes, la fonction `Hexagone` prend en paramètre la variable `cote`, et trace un hexagone tel que ci-dessus dont les côtés mesurent `cote` unités.

- 2► Dans cette question, on souhaite construire la figure suivante. Le plus petit hexagone a un côté de longueur 32, et chaque hexagone a un côté 1,5 fois plus grand que le précédent.

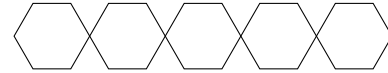


- a► Écrire la fonction `Hexagone` de sorte que l'hexagone soit tracé en partant du point `A`

et en tournant dans le sens inverse des aiguilles d'une montre (autrement dit on part de `A`, puis on va en `B`, puis en `C`, etc. jusqu'à revenir en `A`).

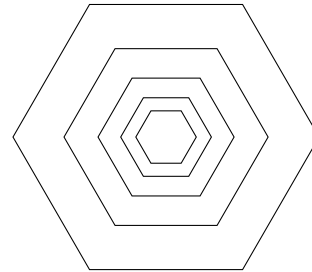
- b► Écrire une fonction `Grossissement` prenant un paramètre entier `n`, et dessinant la figure demandée avec `n` hexagones.

- 3► Dans cette question, on souhaite construire la figure suivante. La longueur des côtés de chaque hexagone est 30 unités.



- a► Modifier la fonction `Hexagone` de la question précédente pour débiter du point `F` au lieu du point `A`.
- b► Dans les hexagones de la figure à tracer, quelle distance sépare deux sommets opposés ?
- c► En déduire une fonction `Frise` prenant un paramètre entier `n`, et dessinant la frise demandée avec `n` hexagones.

- 4► Dans cette question, on souhaite construire la figure suivante. Le plus petit hexagone a un côté de longueur 32, et chaque hexagone a un côté 1,5 fois plus grand que le précédent.



- a► Modifier la fonction `Hexagone` de la question précédente de sorte que l'hexagone soit tracé selon la logique suivante :
 - On part d'un point qui sera le centre de l'hexagone,
 - On décale le crayon sur le sommet de droite,
 - On trace l'hexagone,
 - On ramène le crayon au centre.
- b► Écrire une fonction `Tunnel` prenant un paramètre entier `n`, et dessinant la figure demandée avec `n` hexagones.

Des boucles un peu plus corsées

Exercice 50

Que fait la fonction suivante ?

```

1  FONCTION mystere(n):
2    POUR i ALLANT DE 1 A n-1:
3      POUR j ALLANT DE 1 A n-1:
4        SI i+j = n:
5          AFFICHER i, " + ", j, " = ", n
6        FIN SI
7      FIN POUR
8    FIN POUR
9  FIN FONCTION

```

Exercice 51

En vous inspirant de l'exercice 50, écrire une fonction qui trouve toutes les manières d'écrire un entier n comme le produit de deux entiers différents de 1.

Exercice 52

En utilisant une boucle POUR i ALLANT DE 0 A 10, afficher un compte à rebours de 10 à 0.

Exercice 53

L'objectif de cet exercice est d'écrire une fonction qui détermine si un nombre est premier. La procédure est la suivante : pour savoir si un nombre n est premier, on vérifie qu'il n'est divisible par aucun entier entre 2 et $n - 1$.

- ▶ Écrire le squelette de la fonction `est_premier` qui prend en paramètre un entier n .
- ▶ Compléter la phrase suivante : « n est divisible par k si, et seulement si le ... de la division euclidienne de ... par ... vaut ... »
- ▶ En déduire un test étant vrai lorsque n n'est pas divisible par une variable k en langage naturel.

Le corps de notre fonction sera le suivant, à compléter avec les prochaines questions :

```

1  k <- ...
2  TANT QUE ...:
3    k <- ...
4  FIN TANT QUE
5  SI ...:
6    RENVOYER ...
7  SINON:
8    RENVOYER ...
9  FIN SI

```

- ▶ Le boucle doit s'arrêter lorsque l'on trouve un diviseur. Compléter les deux premières lignes.
- ▶ Quel est le rôle de la variable k ? Compléter la ligne 3.

Étudions les deux cas de figure, où n est soit premier, soit non.

- ▶ Dans cette question, supposons n premier.
 - a▶ Rappeler la définition d'un nombre premier.
 - b▶ Quel est le plus petit diviseur positif de n qui n'est pas 1 ?
 - c▶ En déduire la valeur de k à la sortie de la boucle.
- ▶ Dans cette question, supposons n non premier.
 - a▶ On note a le plus petit diviseur de n positif et différent de 1. Comparer a et n le plus précisément possible.
 - b▶ Que renvoie le test de la boucle pour les valeurs de k entre 2 et a .
 - c▶ En déduire la valeur de k à la sortie de la boucle.
- ▶ Déduire des deux questions précédentes un test à effectuer sur la valeur finale de k pour savoir si n est premier ou non.
- ▶ Compléter les dernières lignes du corps de la fonction pour qu'elle renvoie VRAI si n est premier, et FAUX sinon.
- ▶ Bonus : simplifier la fin de la fonction pour remplacer tout le bloc SI ... SINON ... en une seule ligne RENVOYER ...

Jeux élémentaires

Exercice 54 (Jeu des bâtonnets – ★★☆☆☆)

Dans le jeu des bâtonnets, deux joueurs s'affrontent. Un certain nombre de bâtonnets sont disposés sur la table, et tour à tour les deux joueurs prennent 1, 2 ou 3 bâtonnets. Le joueur qui est forcé de prendre le dernier bâtonnet perd.

- 1► On veut écrire une fonction `afficher_batons` qui prend en paramètre le nombre de bâtonnets et affiche ces bâtonnets sous la forme de traits verticaux entrecoupés d'espaces.

```
| | | | |
```

- a► Les bâtonnets et les espaces représentent un frise. Quel est son motif élémentaire ?
- b► On note n le paramètre de la fonction `afficher_batons`. Combien de motifs élémentaires va-t-on tracer ?
- c► En déduire le type de boucle à utiliser.
- d► Une seule instruction se trouvera à l'intérieur de la boucle, celle qui affiche un motif élémentaire. L'implémenter.

- 2► Définir une variable `nbBatons` qui vaut 15.

On va désormais implémenter la *boucle (principale) de jeu*, c'est-à-dire ce qui doit se passer tant qu'il reste des bâtonnets.

- 3► Sait-on à l'avance le nombre de tours que le jeu durera ? En déduire le type de boucle que l'on va utiliser, et l'implémenter.
- 4► À chaque tour de jeu (autrement dit à chaque tour de boucle), on doit réaliser les actions suivantes :
 - Demander au joueur combien de bâtons il souhaite retirer,
 - Mettre à jour le nombre de bâtons restant,
 - Afficher les bâtons restant.

Compléter le bout de code suivant :

```
1  ENTRER retirer
2  nbBatons <- ...
3  ...
```

- 5► Vous avez un support élémentaire pour le jeu. Laissez parler votre créativité, et améliorez certains éléments ! Tenir compte du tour de chaque joueur, déterminer qui gagne et qui perd, afficher un message de victoire...

Exercice 55 (Le juste prix – ★★★★★)

Le jeu du juste prix est le suivant : le joueur doit deviner un prix en faisant des propositions, et le présentateur dit au joueur si sa proposition est au-dessus ou en-dessous du prix recherché.

- 1► Définir une variable `juste_prix` qui prend un entier aléatoire entre 1 et 100. On utilisera la fonction `ENTALEA` qui prend deux paramètres a et b , et tire un entier aléatoire entre a et b inclus.
- 2► Définir une variable `proposition`, qui va contenir les propositions successives du joueur. On fera attention à bien choisir sa valeur initiale.

On va désormais implémenter la *boucle (principale) de jeu*, c'est-à-dire ce qui doit se passer tant que le joueur n'a pas gagné.

- 3► Sait-on à l'avance le nombre de tours que le joueur prendra pour gagner ? En déduire le type de boucle que l'on va utiliser, et l'implémenter.
- 4► À chaque tour de jeu (autrement dit à chaque tour de boucle), on doit réaliser les actions suivantes :
 - On récupère la proposition du joueur et on l'affiche,
 - Si sa proposition est plus grande que le juste prix, on dit au joueur "C'est moins",
 - Si sa proposition est plus petite que le juste prix, on dit au joueur "C'est plus".

Compléter le bout de code suivant :

```
1  ENTRER proposition
2  AFFICHER ...
3  SI ...:
4    AFFICHER ...
5  SINON SI ...:
6    AFFICHER ...
7  FIN SI
```

- 5► Ajouter un message de victoire quand le joueur a gagné.
- 6► Bonus 1 : définir une fonction qui prend en paramètre un entier n et qui déroule le jeu en faisant deviner un nombre entre 1 et n .
- 7► Bonus 2 : faire en sorte que le joueur n'ait droit qu'à 10 essais.

Exercice 56 (Jeu d'argent – ★★★★★)

Dans un jeu d'argent, on mise 1 euro pour jouer, et on lance deux dés à 6 faces : si on tombe sur un double 6, on gagne 30 euros, sinon on ne gagne rien et la mise est perdue.

On souhaite évaluer les risques de ce jeu : on va simuler un certain nombre de joueurs, et voir combien d'entre eux ressortent perdants au bout de 10 tours de jeu.

- 1► Écrire une fonction `lancer` qui ne prend aucun paramètre, simule de lancer de deux dés à 6 faces et renvoie "gagnant" si le lancer est gagnant (paire de 6) et "perdant" s'il est perdant perdant.

On utilisera la fonction `ENTALEA` qui prend deux paramètres `a` et `b`, et tire un entier aléatoire entre `a` et `b` inclus.

- 2► a► Créer le squelette d'une fonction `partie` qui prend en paramètre une variable `capitalDepart`.
Les questions suivantes servent à remplir le corps de la fonction.
- b► Initialiser une variable `argent` avec la valeur de `capitalDepart`. Cela correspond à l'argent dont on dispose avant de jouer.
- c► Le jeu est composé de 10 tours. Quel type de boucle va-t-on utiliser ? L'implémenter.
- d► À chaque tour de jeu, on :
- mise 1 euro,
 - lance les deux dés,
 - gagne de l'argent ou non selon le résultat.

Compléter et expliquer le code suivant, et le simplifier si possible :

```

1  argent <- ... # mise
2  SI lancer() == "gagnant":
3      argent <- ...
4  SINON:
5      ...
6  FIN SI
```

- e► Renvoyer l'argent du joueur à la fin de la partie.

- 3► Afin de répondre à la question initiale, on va procéder ainsi :

- on initialise un compteur `gagnants` à 0, qui comptera le nombre de gagnants,

- on crée une boucle qui fait 1000 parties et incrémente `gagnants` pour chaque partie gagnante,
- on calcule la proportion de parties gagnantes.

On suppose que chaque joueur dispose de 100 euros au départ. Compléter le programme suivant :

```

1  gagnants <- 0
2  POUR i ALLANT DE 1 A ...:
3      resultat <- partie(...)
4      SI ...:
5          gagnants <- gagnants + 1
6      FIN SI
7      AFFICHER gagnants/...
8  FIN POUR
```

- 4► Interpréter ce résultat.

2 Programmer en Python

Table des matières

1	Les interfaces	43
A	La console	43
B	Exécuter un script	43
C	Le Notebook Jupyter	43
2	Les logiciels	44
A	Basthon	44
B	EduPython	45

Programme

Utiliser les variables et les instructions élémentaires

Contenus

- ▶ Variables informatiques de type entier, booléen, flottant, chaîne de caractères.
- ▶ Affectation (notée $\leftarrow$ en langage naturel).
- ▶ Séquence d'instructions.
- ▶ Instruction conditionnelle.
- ▶ Boucle bornée (for), boucle non bornée (while).

Capacités attendues

- ▶ Choisir ou déterminer le type d'une variable (entier, flottant ou chaîne de caractères).
- ▶ Concevoir et écrire une instruction d'affectation, une séquence d'instructions, une instruction conditionnelle.
- ▶ Écrire une formule permettant un calcul combinant des variables.
- ▶ Programmer, dans des cas simples, une boucle bornée, une boucle non bornée.
- ▶ Dans des cas plus complexes : lire, comprendre, modifier ou compléter un algorithme ou un programme.

Notion de fonction

Contenus

- ▶ Fonctions à un ou plusieurs arguments.
- ▶ Fonction renvoyant un nombre aléatoire. Série statistique obtenue par la répétition de l'appel d'une telle fonction.

Capacités attendues

- ▶ Écrire des fonctions simples ; lire, comprendre, modifier, compléter des fonctions plus complexes. Appeler une fonction.
- ▶ Lire et comprendre une fonction renvoyant une moyenne, un écart type. Aucune connaissance sur les listes n'est exigée.
- ▶ Écrire des fonctions renvoyant le résultat numérique d'une expérience aléatoire, d'une répétition d'expériences aléatoires indépendantes.

Introduction

Python est un langage de programmation, sorti dans sa première version en 1991, devenu l'un des langages les plus populaire pour sa simplicité écrasante et ses possibilités illimitées. Un langage de programmation n'est rien d'autre qu'un moyen de traduire un algorithme qu'un humain peut comprendre en une suite d'instructions qu'un ordinateur peut comprendre. Python est un langage *interprété*, c'est-à-dire qu'un *interpréteur* va traduire votre code en langage machine à *chaque fois* que vous l'exécutez (par opposition aux langages *compilés* qui sont traduits une bonne fois pour toutes).

1 – Les interfaces

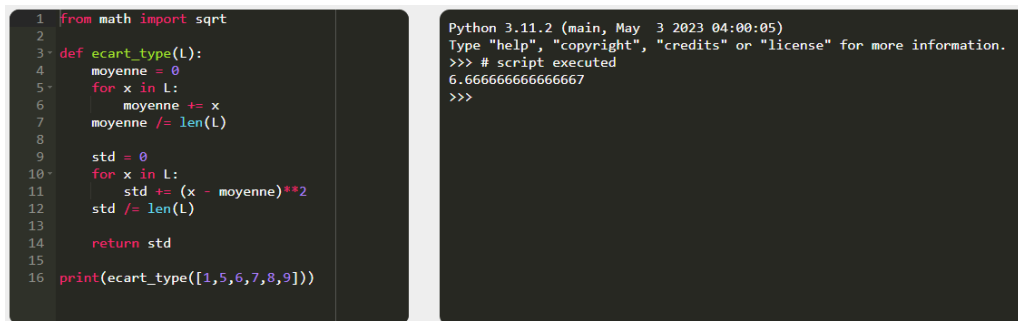
A – La console

La *console* de Python est un espace dans lequel vous pouvez entrer du code et l'interpréter au fil de l'eau. On écrit les lignes de code devant les `>>>`, et on presse la touche Entrée pour valider. Elle est idéale pour tester des codes élémentaires, avec une seule ligne ou une seule boucle par exemple. Les résultats s'affichent systématiquement. Elle fonctionne également comme une calculatrice.

```
>>> 2 + 5
7
>>> 2 - 5
-3
>>> a = 6
>>> a = a + 5
>>> a
11
```

B – Exécuter un script

Pour des projets plus complexes, on préférera écrire un *script* à part, et l'*exécuter* une fois tout le script rédigé. Un script Python est un fichier avec l'extension `.py`, bien que, dans les éditeurs que nous verrons plus bas, souvent le système de fichiers est caché.



Exemple avec un script complet à gauche et son exécution à droite

C – Le Notebook Jupyter

Un *Notebook Jupyter* (ou *Notebook* pour faire court) est à mi-chemin entre la console et le script. Il est constitué de *cellules*, qui sont des morceaux de scripts que l'on peut exécuter indépendamment, et dont les résultats s'affichent en-dessous. C'est idéal pour présenter des résultats intermédiaires et produire des documents mêlant du code et du texte, voire des images.

```

Entrée[1]: from turtle import *
           from random import randint

           print("Les modules ont été importés")

           Les modules ont été importés

Entrée[2]: alice = Turtle()

           print("La tortue s'appelle Alice")

           La tortue s'appelle Alice

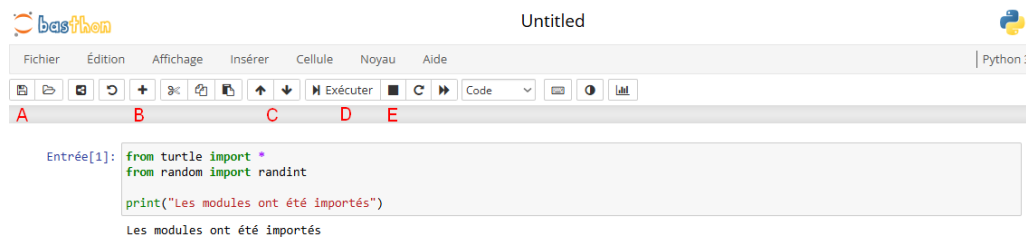
Entrée[3]: for _ in range(200):
           alice.forward(10)
           alice.left(randint(-90,91))
           done()

```



Exemple de Notebook avec trois cellules

Bien que son implémentation peut différer d'un logiciel à l'autre, les commandes d'un Notebook Jupyter sont toujours sensiblement les mêmes :



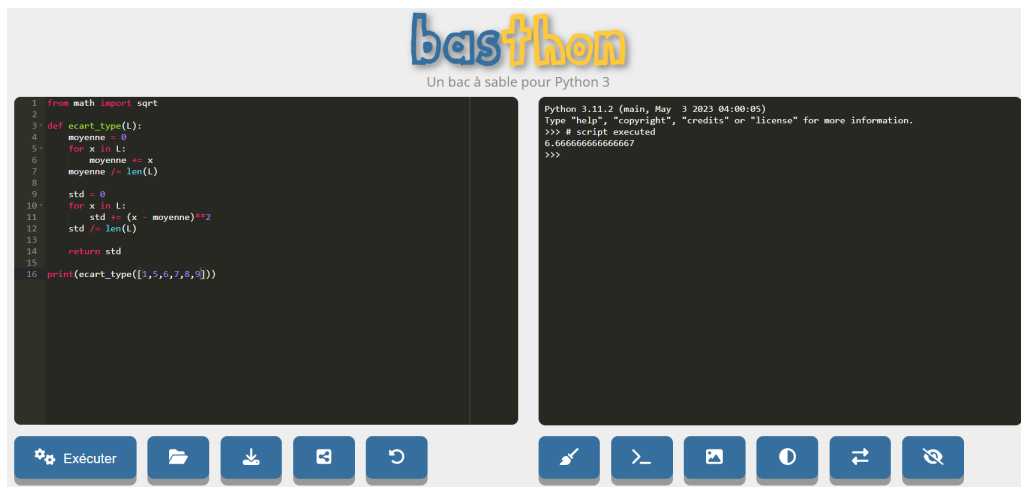
- A► Sauvegarder le projet (Raccourci clavier : Ctrl + S)
- B► Ajouter une cellule en-dessous (Raccourci clavier : a)
- C► Déplacer une cellule vers le haut ou le bas
- D► Exécuter la cellule sélectionnée (Raccourci clavier : Ctrl + Entrée pour simplement exécuter la cellule, Shift + Entrée pour exécuter la cellule et sélectionner la suivante)
- E► Interrompre le programme quand une cellule est coincée

2 – Les logiciels

Python n'est pas le nom d'un logiciel permettant de coder en Python. Il ne désigne que le langage en lui-même, autrement dit le programme qui transforme et exécute le code que l'on écrit en Python. Pour écrire ledit code, il faut un *éditeur de code* ou un *environnement de développement intégré* (*IDE* en anglais).

A – Basthon

Le site basthon.fr permet de coder en Python de toutes les manières possibles, listées dans la précédente section. En mode console on peut écrire le script à gauche et l'exécuter avec le bouton Exécuter en bas à gauche, ou directement écrire dans la console à droite, où s'affichent les résultats.



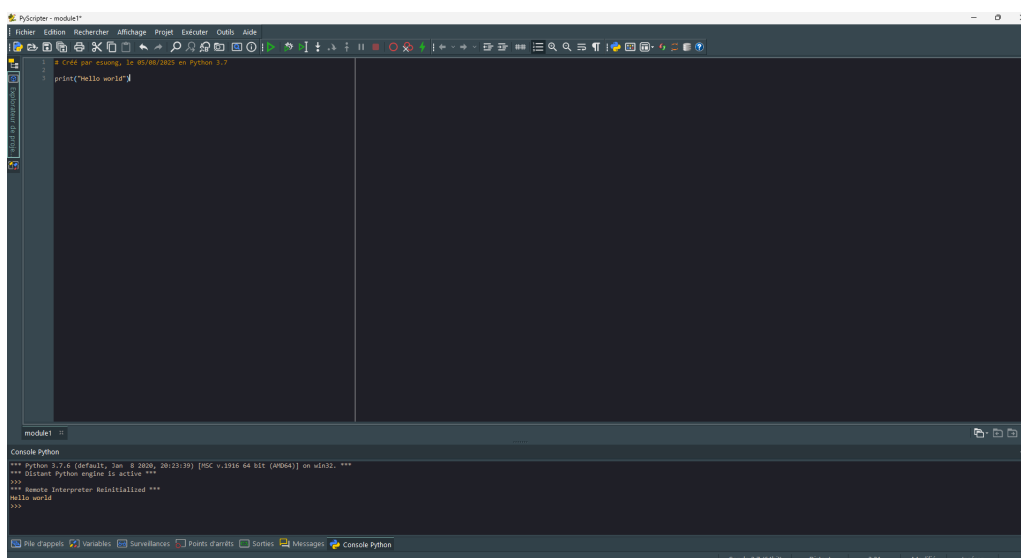
Bien que pratique pour de petits projets à la volée, cette méthode a quelques inconvénients : elle ne dispose pas de toutes les fonctionnalités disponibles dans les autres méthodes, notamment tout ce qui touche aux graphiques et aux dessins, et sa puissance de calcul est limitée. Un programme trop exigeant peut faire planter la page.

Basthon propose également un Notebook, qui donne accès aux fonctionnalités graphiques comme illustré sur les captures de la précédente section, mais qui souffre du même problème de performances. En pratique, il sera bien suffisant pour l'année de seconde.

Le site Capytale de l'académie de Paris, à l'adresse <https://capytale2.ac-paris.fr>, utilise Basthon.

B – EduPython

EduPython est un logiciel tout-en-un qui permet de tout faire : écrire directement dans la console, faire des scripts complets, utiliser les fonctionnalités graphiques mentionnées précédemment, et faire des Notebooks. Il est en général préinstallé sur les ordinateurs de lycée, et sert parfois dans les autres disciplines. Son interface est certes plus complète que Basthon, mais elle est d'autant plus austère.



Au lancement du logiciel, vous disposez d'un éditeur de code à gauche, de la console en bas, et d'un bouton vert triangulaire dans la barre de navigation en haut pour exécuter votre script.

3 Python : variables et types

1 – Affectation en Python

Pour affecter une valeur à une variable (déjà existante ou non) en Python, on utilise le symbole = :

Langage naturel

```
1 var ← 3
```

Scratch

mettre var à 3

Python

```
var = 3
```

Remarque 1

Vous ne pouvez pas nommer vos variables n'importe comment :

- ▶ Un nom de variable doit commencer par une lettre (ou un tiret bas, mais ça a une signification particulière...). Par exemple, *truc ou 2bidule ne sont pas des noms valides.
- ▶ Un nom de variable peut contenir des chiffres ou des tirets bas, mais pas d'autres caractères spéciaux. Par exemple, liste1, nombre_2, de3gil et carte_4_monde sont des noms valides.
- ▶ On ne peut pas utiliser un mot clé réservé par Python : par exemple, on ne peut pas nommer une variable `if`, `for`, `while`, ... La liste complète des mots-clé réservés se trouve à la fin du chapitre.
- ▶ Un nom de variable ne doit pas contenir de caractère accentué.
- ▶ Les majuscules et minuscules ne sont pas équivalentes : ainsi, Truc et truc désignent deux variables différentes. (On dit que les noms de variables sont *sensibles à la casse*).

Lorsque vous choisissez un nom pour une variable, préférez l'explicite à l'implicite : par exemple, si une variable contient un prix, appelez-la `prix` plutôt que `juste p...` L'essentiel, c'est que vous puissiez déterminer d'un coup d'œil ce que représente une variable dans un code. Exemple avec une fonction (voir le paragraphe sur le sujet) qui calcule un intérêt :

Les variables sont mal nommées

```
1 def itr(t, c, a):
2     for i in range(a):
3         c = c * t
4     return c
5
6 itr(1.1, 1000, 5)
```

Les variables sont bien nommées

```
1 def interets(taux, capital, annees):
2     for i in range(annees):
3         capital = capital * taux
4     return capital
5
6 interets(1.1, 1000, 5)
```

Vous pouvez retenir la règle suivante : lorsque votre variable n'a aucune interprétation particulière, vous pouvez la nommer avec juste une lettre. Sinon, nommez-la de manière à décrire ce qu'elle représente. Toutefois, évitez les noms à rallonge !

Exception : quand vous tapez du code sur une calculatrice lycée, vous pouvez vous permettre de nommer vos variables avec juste une lettre... c'est déjà assez pénible comme ça.

Remarque 2 (Pour les experts)

Il arrive très fréquemment de devoir redéfinir une variable à partir d'elle-même. Par exemple, lui ajouter 1 (autrement dit, l'*incrémenter*) :

Langage naturel

```
1 var ← var + 1
```

Scratch



Python

```
var = var + 1
```

En Python, il existe une notation raccourcie quand on redéfinit une variable à partir d'elle-même par une opération simple (addition, soustraction, multiplication, division). Par exemple, `var = var + 1` se raccourcit en `var += 1`. Autre exemple, `var = var * 5` se raccourcit en `var *= 5`.

2 – Les différents types

Voici une liste non exhaustive des types de données que vous serez les plus susceptibles de rencontrer dans un programme Python :

Type (en français)	Type Python	Exemple
Entier (relatif)	<code>int</code>	<code>n = 2</code>
Flottant	<code>float</code>	<code>x = 2.3</code>
Chaîne de caractères	<code>str</code>	<code>texte = "Bonjour monde !"</code>
Booléen	<code>bool</code>	<code>A = True</code>

Pour connaître le type d'un objet `x`, on utilise la fonction `type` : par exemple, `type(12)` affiche `<class 'int'>`.

Remarque 3 (Pour les experts)

Python est un langage que l'on dit *orienté objet*. Pour résumer très grossièrement, ça veut dire que tout en Python est un *objet*. Il y a les objets de type `int`, les objets de type `bool`, ... Une fonction renvoie toujours un objet – quand elle ne renvoie rien, elle renvoie en fait un objet de type `NoneType`. La fonction `type` renvoie un objet de type `type` : ce n'est pas une chaîne de caractère (autrement dit, ce n'est pas un objet de type `str`) ! Pour vérifier, par exemple, si un objet est de type `int`, on ne peut pas écrire le test (voir le paragraphe sur les tests) `type(x) == 'int'` : on doit plutôt écrire `isinstance(x, int)`, ce qui renvoie un booléen.

Exercice conseillé : 21

A – Les entiers

Dans Python, les nombres entiers sont les nombres sans virgule, positifs comme négatifs.

Remarque 4

Bien que mathématiquement, 12 et 12,0 sont égaux, dans Python, ils n'ont pas le même statut : le premier est de type `int`, le second est de type `float`.

Les nombres entiers en Python sont principalement utilisés dans deux contextes : pour faire des compteurs, et pour faire de l'arithmétique. Pour l'utilisation arithmétique, voici les opérations que l'on peut effectuer entre des entiers que l'on rencontre le plus souvent :

Opération	Notation Python	Exemple
Addition	+	>>> 2 + 3 5
Soustraction	-	>>> 2 - 3 -1
Multiplication	*	>>> 2 * 3 6
Puissance	**	>>> 2 ** 3 8
Quotient de la division euclidienne	//	>>> 13 // 4 3
Reste de la division euclidienne	%	>>> 13 % 4 1

Remarque 5

On peut calculer une division entre deux entier, mais le résultat sera toujours de type `float`, même si mathématiquement il est entier. Par exemple, l'instruction `12/3` renvoie `4.0` et non `4`.

B – Les flottants

Un nombre de type flottant est un nombre décimal qui contient une virgule apparente. Par exemple, `2.3` est de type `float` (attention, Python utilise la convention anglaise pour les nombres décimaux, on doit donc écrire un point et non une virgule!). Cependant, il faut bien noter une subtilité, où Python se détache du monde mathématique. En effet, `12.0` est de type `float`, bien que mathématiquement il est égal à un nombre entier. C'est pourquoi on définit les nombres de type flottant comme des nombres décimaux *avec une virgule apparente* (on rappelle qu'un nombre entier est un type particulier de nombre décimal!).

Le type `float` supporte les opérations d'addition (symbole `+`), de soustraction (symbole `-`), de multiplication (symbole `*`), de puissance (symbole `**`) et de division (symbole `/`).

Remarque 6

En Python, comme dans de nombreux langages de programmation, on ne travaille qu'avec des nombres décimaux qui ont un certain nombre de chiffres après la virgule (par exemple, on peut représenter tous les nombres qui ont au plus 55 décimales après la virgule). En effet, pour stocker toutes les décimales d'un nombre non décimal... il faudrait une infinité d'espace mémoire.

Python travaille donc *toujours* avec des arrondis : essayez donc de calculer `0.1 + 0.2` avec Python...

Voir l'activité ?? pour plus d'explications.

Remarque 7

Pour transformer un nombre entier en nombre flottant, on peut soit rajouter un simple point si on donne le nombre explicitement (exemple : `12.`, qui est un raccourci pour `12.0`), soit écrire `float(x)` pour transformer une variable `x`.

Remarque 8

La syntaxe `int(3.4)` renvoie `3` (qui est de type `int`). Attention cependant, la syntaxe `int(-3.4)` renvoie `-3` (qui est aussi de type `int`).

Ainsi, appliquer la fonction `int` à un nombre flottant ne renvoie pas sa partie entière : elle renvoie le nombre tronqué.

Remarque 9

Python comprend l'écriture scientifique d'une certaine manière l'écriture scientifique : par exemple, $1,5 \times 10^3$ s'écrit `1.5e3`. Autre exemple, $3,95 \times 10^{-8}$ s'écrit `3.95e-8`. Dernier exemple, $8,113 \times 10^{275}$ s'écrit `8.113e275`. Il suffit d'écrire la mantisse, puis un e, puis l'exposant sans parenthèses.

C – Les chaînes de caractères

Les chaînes de caractère (*string* en anglais, d'où le type `str`) sont délimitées par des guillemets simples ou doubles. Exemples : `"Bonjour monde !"` et `'Hello world !'` sont des chaînes de caractères.

Remarque 10

La fonction `input` renvoie toujours une chaîne de caractères ! Même si on tape un nombre, par exemple `1.2`, le résultat renvoyé sera la chaîne de caractères `'1.2'`.

Remarque 11

Appeler les fonctions `int` ou `float` sur une chaîne de caractère qui représente un nombre transformera cette dernière en le nombre qu'elle contient. Par exemple, `float('1.2')` renvoie `1.2` (de type `float`). En revanche, cela ne fonctionnera pas si vous essayez de convertir autre chose qu'un nombre : Python n'y arrivera pas et vous affichera une erreur.

Passons en revue quelques fonctions et opérations utiles avec les chaînes de caractères :

Fonction/Opération	Description	Exemple
Concaténation	Coller bout à bout deux chaînes de caractères	<pre>>>> "Bonjour" + "Monde" "BonjourMonde"</pre>
Répétition	Répéter bout à bout une chaîne de caractères un certain nombre de fois	<pre>>>> "Hello"*3 "HelloHelloHello"</pre>
Longueur	Obtenir le nombre de caractères	<pre>>>> len("monde !") 7</pre>

Remarque 12 (Pour les confirmés)

On peut également accéder à un caractère spécifique d'une chaîne de caractères :

```
>>> texte = "Bonjour"
>>> texte[2]
'n'
```

Ainsi on accède au caractère en position 2... ce qui correspond à la troisième lettre ! En effet, comme la quasi-totalité des langages de programmation, *Python compte les positions (les indices) à partir de 0*.

Remarque : on ne peut pas changer un caractère d'une chaîne déjà existante : autrement dit, la syntaxe `texte[2] = 'e'` ne fonctionne pas (on dit que les chaînes de caractères sont *immuables*).

Remarque 13 (Pour les experts)

Le *slicing* correspond à l'action de récupérer une sous-chaîne : par exemple, de "Bonjour" je souhaite récupérer "njou". Pour ça, on utilise la syntaxe suivante : `texte[2:6]`. Le premier nombre correspond à la position du premier caractère (elle est donc incluse dans le découpage), et le second nombre correspond à la position du dernier caractère +1 (autrement dit, c'est la position qui suit la dernière de la sous-chaîne), elle est donc exclue. Pour récupérer les 3 premiers caractères (donc jusqu'à la position 2!), c'est-à-dire "Bon", on peut écrire `texte[0:3]`, mais il existe la notation raccourcie `texte[:3]`.

Pour récupérer tous les caractères à partir du quatrième (donc en position 3!), on peut écrire `texte[4:7]`, mais il existe la notation raccourcie `texte[4:]`.

On peut facilement accéder au dernier caractère par la syntaxe `texte[-1]`. On peut se servir de cela pour récupérer, par exemple, toute la chaîne sauf le dernier caractère, on peut écrire `texte[:-1]`.

Dernière possibilité de *slicing* : la syntaxe `texte[1:6:2]` va extraire la sous-chaîne partant de la position 1 jusqu'à la position 5, en prenant un caractère sur deux.

Dans la même veine, une astuce utile les la syntaxe suivante : la syntaxe `texte[6:1:-1]` va donner la sous-chaîne des caractères entre les positions 2 et 6, mais à l'envers... en particulier, `texte[::-1]` permet de renverser la chaîne.

Remarque 14 (Pour les experts)

Souvent, on a besoin d'inclure une variable dans une chaîne de caractères. La manière naïve de le faire est la suivante :

```
print("La somme de " + str(x) + " et " + str(y) + " vaut " + str(x + y))
```

(Remarque, il faut impérativement convertir les variables `x` et `y` en chaînes de caractères *via* la fonction `str`!) C'est lourd, et c'est pénible à écrire. Une manière plus pratique est la suivante :

```
print("La somme de {0} et {1} vaut {2}".format(x, y, x+y))
```

C'est ce qu'on appelle *formater* une chaîne de caractères. Ici, 0, 1, et 2 entre accolades renvoie à l'ordre des arguments de la fonction `format` (n'oubliez pas qu'on compte à partir de 0!)

Depuis la version 3.6 de Python, il existe une manière encore plus rapide :

```
print(f"La somme de {x} et {y} vaut {x+y}")
```

Le petit `f` devant la chaîne de caractère (que l'on appelle en conséquence un *f-string*) indique à Python qu'il doit remplacer ce qui est entre accolades par les variables qui s'y trouvent.

Une autre syntaxe très pratique est la suivante :

```
print(f"La valeur de x est environ {x:.2f}")
```

Le `:.2f` signifie qu'on tronque le nombre avec deux chiffres après la virgule (le 2 indique le nombre de décimales, le `f` est pour *floating (point)*, il indique la virgule). Il y a plein d'autres petites astuces de formatage du genre, cherchez-les sur Internet.

D – Les booléens

Une variable booléenne (autrement dit de type `bool` ne peut prendre que deux valeurs : soit `True`, soit `False`.

Remarque 15

Quand on applique la fonction `bool` à un objet quelconque, on le transforme en booléen. Mais quelle est la valeur de vérité de 12 ou de "Bonjour" ?

Python a fait le choix suivant : quand une variable numérique vaut 0 ou 0., et quand une chaîne de caractères est vide (c'est-à-dire juste ""), la valeur de vérité est `False`. Dans tous les autres cas, elle vaut `True`.

Mais il est très (très) rarement utile de devoir convertir une variable quelconque en booléen...

Remarque 16

Un test mathématique, ou d'égalité, ou d'identité, renvoie un booléen. Par exemple, `5 < 9` renvoie un booléen (`True` en l'occurrence). Ainsi, écrire `var = 5 < 9` affecte à la variable `var` la valeur `True`. On peut mettre des parenthèses pour améliorer la lisibilité (exemple : `var = (5 > 9)` – ici, `var` prend la valeur `False` puisque 5 n'est pas plus grand que 9), mais ce n'est pas nécessaire.

3 – Entrer du texte en Python

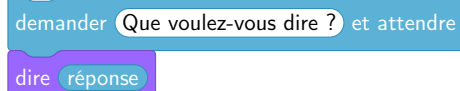
En Python, la fonction permettant d'entrer du texte est `input`.

Exemple 17

Langage naturel

```
1 Afficher "Que voulez-vous dire ?"
2 entree ← ...
3 Afficher entree
```

Scratch



Python

```
1 entree = input("Que voulez-vous dire ?")
2 print(entree)
```

La fonction `input` renvoie une chaîne de caractères, autrement dit du texte (même si on entre un nombre!).

Remarque 18

La fonction `input` renvoie toujours une chaîne de caractères.

Par exemple, si vous demandez à l'utilisateur d'entrer un nombre avec `input`, et que celui-ci entre 2, la fonction `input` renvoie "2", de type `str`, et non 2, de type `int`.

Cela est d'autant plus problématique quand on veut en exploiter le résultat : par exemple, le test `"2" > 1` ne renvoie ni vrai ni faux, il est invalide et soulève un erreur.

On doit donc convertir le résultat de `input` dans le type attendu. Par exemple, si on attend de l'utilisateur qu'il rentre un entier, on peut écrire :

```
x = input("Entrez un nombre")
x = int(x)    # Convertir en int
```

Ou, de manière plus compacte :

```
x = int(input("Entrez un nombre"))
```

4 – (Bonus) Liste des mots-clés en Python

Il existe 35 *mots-clés stricts* ; on ne peut pas nommer une variable ou une fonction avec ces noms, ils sont réservés au fonctionnement de Python.

- | | | | | |
|-------------------------|------------------------|-----------------------|-------------------------|----------------------|
| ▸ <code>and</code> | ▸ <code>del</code> | ▸ <code>from</code> | ▸ <code>None</code> | ▸ <code>True</code> |
| ▸ <code>as</code> | ▸ <code>elif</code> | ▸ <code>global</code> | ▸ <code>nonlocal</code> | ▸ <code>try</code> |
| ▸ <code>assert</code> | ▸ <code>else</code> | ▸ <code>if</code> | ▸ <code>not</code> | ▸ <code>while</code> |
| ▸ <code>break</code> | ▸ <code>except</code> | ▸ <code>import</code> | ▸ <code>or</code> | ▸ <code>with</code> |
| ▸ <code>class</code> | ▸ <code>False</code> | ▸ <code>in</code> | ▸ <code>pass</code> | ▸ <code>yield</code> |
| ▸ <code>continue</code> | ▸ <code>finally</code> | ▸ <code>is</code> | ▸ <code>raise</code> | |
| ▸ <code>def</code> | ▸ <code>for</code> | ▸ <code>lambda</code> | ▸ <code>return</code> | |

Il existe en plus 3 *mots-clés souples*, qui correspondent à des fonctionnalités de Python, mais on peut tout de même nommer des variables ou des fonctions avec ces noms :

- | | | |
|---------------------|----------------------|---------------------|
| ▸ <code>case</code> | ▸ <code>match</code> | ▸ <code>type</code> |
|---------------------|----------------------|---------------------|

Exercices

Exercice 19

Recopier et compléter le tableau d'avancement suivant :

Instruction	Valeur de a	Valeur de b
a = 1		
b = 5		
a = b-3		
b = 2*a		
a = b		
b = a		

Exercice 20

La Terre a un rayon d'environ 6 400km. Compléter les instructions Python suivantes pour que la variable `equateur` contienne la longueur de l'équateur de la Terre en km. La valeur de π est supposée stockée dans une variable appelée `pi`.

```
rayonTerre = ...
equateur = ...
```

Exercice 21

Compléter le tableau suivant :

Exemple de variable	Type (en français)	Type (en Python)
2	Entier	<code>int</code>
		<code>str</code>
		<code>bool</code>
3.14		
"c"		
	Chaîne de caractères	
	Flottant	

Exercice 22

Compléter le tableau suivant :

Opération	Résultat	Type du résultat
$3 + 19$		
$2.5 + 3.1$		
$2.5 + 5$		
"SNT" + "NSI"		
$17//5$		
$15/5$		

Exercice 23

Traduire l'expression $x^3 - 4x^2 + \frac{1}{3}x - 5$ en Python.

Exercice 24

Que vaut la variable x à la fin de ces instructions ?

```
x = 5
x = x*2
x = x - 1
x = x**2
```

4 Python : fonctions et modules

1 – Les fonctions

Une fonction en Python se construit avec le mot-clé `def`, et éventuellement renvoie une valeur avec le mot-clé `return` :

Exemple 1

Langage naturel

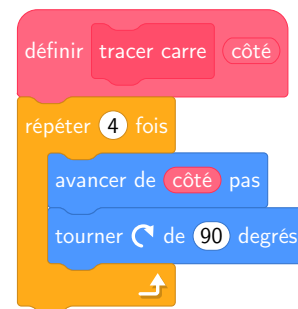
```
1  FONCTION carre(x):  
2    RENVOYER x*x  
3  FIN FONCTION  
4  
5  carre(2)
```

Python

```
1  def carre(x):  
2    return x*x  
3  
4  carre(2)
```

Remarque 2

On peut créer des fonctions (appelés *blocs*) avec Scratch, mais celles-ci ne peuvent pas renvoyer de valeur.



Exemple 3

Une fonction peut prendre plusieurs argument. Une fonction peut également renvoyer plusieurs valeurs. Elles apparaîtront alors entre parenthèses. On peut récupérer séparément ces valeurs dans plusieurs variables, comme dans l'exemple suivant :

```
>>> def vitesse(distance, temps):  
...     return distance/temps  
...  
>>> vitesse(60,2)  
30.0  
>>> vitesse(100,8)  
12.5
```

```
>>> def rectangle(a, b):  
...     perimetre = 2*a + 2*b  
...     aire = a*b  
...     return perimetre, aire  
...  
>>> rectangle(4, 5)  
(18, 20)  
>>> peri, aire = rectangle(4, 5)  
>>> peri  
18  
>>> aire  
20
```

```
def fonction():  
    for i in range(5):  
        print(i)  
        if i == 2:  
            return "Sorti !"
```

Une chose importante à noter : le mot-clé `return` arrête la fonction ! Par exemple, le code à gauche contient une fonction qui lance une boucle censée faire cinq itération, afficher le compteur à chaque itération, et renvoyer "Sorti !" à l'itération `i = 2`.

Cependant, si vous appelez cette fonction, vous constaterez que les itérations `i = 3` et `i = 4` sont ignorées.

```
>>> f()  
0  
1  
2  
'Sorti !'
```

Une fonction s'arrête au premier `return` qu'elle rencontre. Tout code qui suivrait un `return` serait ignoré.

Remarque 4

Observez le code suivant : selon-vous, que va-t-il produire ?

```

1  a = 4
2
3  def f():
4      b = 9
5      print(a)
6
7  print(b)
```

Remarquez que `a` est définie à l'extérieur de `f`, et `b` est définie à l'intérieur de `f`. En exécutant ce code, `f` va réussir à afficher la valeur de `a` même s'il elle est définie à l'extérieur, mais `print(b)` va renvoyer une erreur, selon laquelle la variable `b` n'est pas définie. En effet, c'est ce qu'on appelle la *localité des variables*. Les variables peuvent passer des environnement les plus grands aux plus petits, mais pas faire le chemin inverse.

Remarque 5 (Pour les experts)

Lorsqu'on précise des arguments à une fonction comme on l'a fait jusque là, on parle d'*arguments positionnels*, car leur position dans les parenthèses détermine leur interprétation. Par exemple, dans l'exemple 3, `dimensions(18,20)` et `dimensions(20, 18)` ne renvoient pas la même chose.

On peut donner une valeur par défaut à certains arguments : on parle alors d'*arguments nommés*. Lors de l'appel de la fonction, on peut alors ne pas préciser la valeur de ces arguments si on n'en a pas besoin.

```

>>> def interets(capital, annee_fin, annee_debut=2000, taux=1.1):
...     temps = annee_fin - annee_debut
...     for _ in range(temps):
...         capital *= taux
...     return capital
...
>>> interets(1000, 2010)
2593.7424601000025
>>> interets(1000, 2010, annee_debut=2009)
1100.0
>>> interets(1000, 2010, taux=2)
1024000
>>> interets(1000, 2010, annee_debut=2009, taux=2)
2000
```

Attention, les arguments nommés doivent être placés après les arguments positionnels.

Remarque 6 (Pour les experts)

Lorsque vous tapez, par exemple, `help(sqrt)`, s'affiche à l'écran une aide, une description de ce que fait la fonction `sqrt`, ce qu'elle prend en entrée et renvoie en sortie.

Ce que s'affiche, c'est ce qu'on appelle la *documentation* de la fonction. Dans les gros projets, il est souvent vital de rédiger la documentation de vos propres fonctions : en effet, si vous travaillez à plusieurs, ou sur une longue période de temps, vous serez amenés à relire des fonctions qui vous sembleront inconnues. Plutôt que de devoir réfléchir et prendre le temps de comprendre le code que vous avez sous les yeux, il sera bien plus efficace de lire la documentation directement.

Pour insérer une documentation, faites comme dans l'exemple suivant :

```

1  def dimensions(perimetre, aire):
2      """
3      À partir de son périmètre et de son aire, retrouver la largeur et la
4      longueur d'un rectangle.
5      Entrée:
6          - perimetre (float): le périmètre du rectangle
7          - aire (float): l'aire du rectangle, dans l'unité au carré du
8                        périmètre !
9      Sortie:
10         - Largeur du rectangle (float)
11         - Longueur du rectangle (float)
12      """
13
14     d = sqrt(perimetre**2 - 16*aire)
15     return (perimetre - d)/4, (perimetre + d)/4

```

(Remarque : ici, la documentation n'était pas vraiment nécessaire, car les noms de la fonction et des variables étaient déjà transparents.)

Remarque 7 (Pour les experts)

Le *type* (ou *type hints*) a été introduit dans la version 3.5 de Python. C'est un complément de la documentation, qui la rend encore plus transparente en permettant de préciser la *signature* de la fonction, c'est-à-dire les types qu'elle reçoit en entrée et ceux en sortie.

```

1  def ecart_longueur(chaine1: str, chaine2: str) -> int:
2      """ Calcule l'écart de longueur entre deux chaînes de caractères. """
3      ecart = len(chaine1) - len(chaine2)
4      if ecart < 0:
5          ecart = -ecart
6      return ecart

```

(Remarque : si la fonction renvoie plusieurs valeurs, alors il faut écrire -> *tuple*.)

2 – Les modules Python

Bien que Python vient initialement avec une très large gamme de fonctions prédéfinies, certaines fonctions plus spécifiques (par exemple, les fonction mathématiques, les fonctions aléatoires, ...) ne sont pas disponibles de base : il faut les *importer* à partir d'un *module*.

A – Les modules mathématiques

Le module `math` contient toute une variété de fonctions mathématiques : `cos`, `sin`, `tan`, `sqrt` (la racine carrée, *square root* en anglais), ... mais aussi des variables spéciales : par exemple, le module `math` contient la variable `pi` qui, comme son nom le suggère, contient la valeur de π .

Pour importer un module, la syntaxe est la suivante : `from math import *`.

Remarque 8 (Pour les experts)

Cette syntaxe pour l'importation d'un module est à éviter au possible. En effet, prenons un exemple : imaginez que vous avez défini une fonction `cos` dans votre programme, puis que vous importez le module `math`. Comme

`math` contient lui-même une fonction `cos`, cette-dernière va simplement supprimer la vôtre et la remplacer.

Une meilleure syntaxe est la suivante : `from math import fonction_1, fonction_2`

On précise les fonctions qu'on souhaite importer, et on importe uniquement celles-ci.

La meilleure syntaxe est la suivante : `import math`

Grâce à cette syntaxe, vous pourrez avoir, par exemple, plusieurs fonctions qui s'appellent `cos`. En effet, pour appeler une fonction du module `math`, la syntaxe est de la forme `math.fonction`. Par exemple, on écrirait `math.cos(math.pi / 2)`.

Pour connaître la liste des fonctions des modules `math` et `random`, exécutez `help(math)` et `help(random)`. Voici tout de même quelques fonction incontournables :

Module	Fonction(s)	Description
<code>random</code>	<code>randint</code>	Nombre aléatoire entre deux bornes (incluses) <code>randint(1,6)</code> renvoie un nombre aléatoire entre 1 et 6
<code>math</code>	<code>sqrt</code>	Racine carrée
<code>math</code>	<code>cos</code> , <code>sin</code> , <code>tan(x)</code>	Fonctions trigonométriques (angles en degrés)
<code>math</code>	<code>floor</code>	Partie entière
/	<code>abs</code>	Valeur absolue
/	<code>round</code>	Arrondi à 10^{-n} <code>round(x,n)</code> arrondit x avec n décimales
/	<code>min</code> , <code>max</code>	Minimum / maximum de deux valeurs

Le module `math` fournit également `pi`.

B – Des modules graphiques – exemple de `turtle`

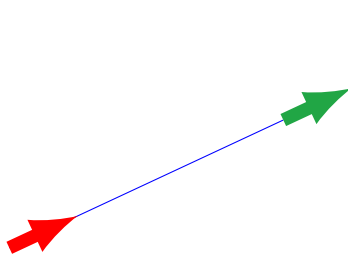
Il existe de nombreux modules permettant l'affichage d'éléments graphiques, parmi eux on peut noter `matplotlib` qui permet la génération de graphiques mathématiques (courbes de fonctions, histogrammes, nuages de points, ...). Dans ce paragraphe, nous allons plutôt nous concentrer sur `turtle`, un module qui permet de faire du dessin de manière assez similaire à Scratch.

Dans ce paragraphe, on liste une poignée de fonctions utiles, avec une petite explication. Il est conseillé de ne pas simplement lire ce morceau de cours, mais plutôt choisir un projet de dessin et consulter la liste des fonctions afin de compléter le projet. L'utilisation du module `turtle` n'est pas un objectif du programme, elle est une application ludique de Python.

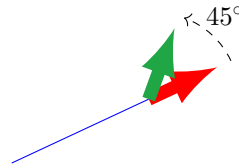
Toute utilisation du module `turtle` débutera par son import, et *dans Basthon*, tout dessin doit se terminer par `done()` pour lancer le tracé.

```
from turtle import *
...
done()
```

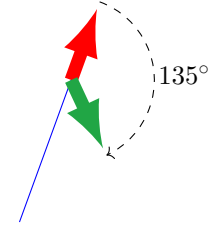
Déplacement et orientation



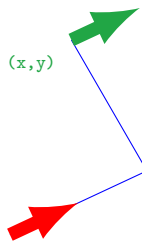
`forward(n)` / `backward(n)`
Avancer/reculer de n pixels



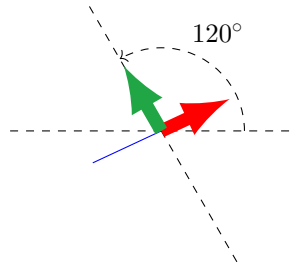
`left(45)`
Tourner de 45° vers la gauche *par rapport à là où la tortue fait face*



`right(135)`
Tourner de 135° vers la droite *par rapport à là où la tortue fait face*



`goto(x,y)`
Déplacer la tortue aux coordonnées (x,y) *sans la tourner*



`setheading(120)`
Tourner la tortue face à la direction 120° *par rapport à l'horizontale* (à droite = 0°)

`clear()`
Efface tous les tracés

Modification du style

`up()` / `down()`
Relève le stylo (la tortue ne trace plus) / Abaisse le stylo (la tortue trace)

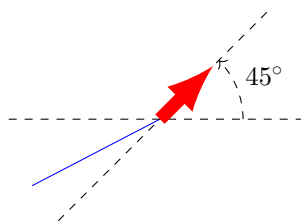
`color("red")`
Change la couleur du tracé

`width(n)`
Change l'épaisseur du trait à n pixels

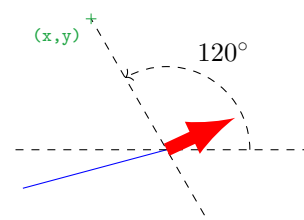
Fonctions de mesure

`position()`
Renvoie les coordonnées (x,y) actuelles de la tortue

`distance(x,y)`
Renvoie la distance en pixels entre la position actuelle de la tortue et le point de coordonnées (x,y)



`heading()`
Renvoie une mesure de l'angle auquel fait face la tortue par rapport à l'horizontale



`towards(x,y)`
Renvoie une mesure de l'angle de la droite horizontale passant par la tortue à la droite passant par la tortue et le point de coordonnées x,y

Remplir une région

Pour remplir une région, écrivez `begin_fill()` puis tracez le contour de cette région. Pour arrêter le mode remplissage, écrivez `end_fill()`. Pour changer la couleur de remplissage, écrivez `fillcolor("red")` pour, par exemple, remplir en rouge.

3 – (Bonus) Apprendre à lire les erreurs

Voici un exemple d'erreur :

```
>>> a = 3
>>> b = "Bonjour"
>>> a + b
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Une erreur se décompose en trois parties :

- ▶ « Traceback (most recent call last) : » indique qu'une erreur est survenue. Nous y reviendrons dans l'exemple suivant.
- ▶ Ensuite, Python indique le lieu de l'erreur, en particulier la ligne dans le code. C'est le plus utile quand on stocke le code dans un fichier, plutôt que l'écrire dans la console. (Remarque : « File : » indique le nom du fichier, et ici « <stdin> » désigne la console.)
- ▶ Le type d'erreur et une description de la raison. Ici, on a essayé d'ajouter un nombre et une chaîne de caractères.

Voici un exemple plus complexe :

```
>>> def f_division(a,b):
...     return a/b
...
>>> f_division(5,0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in f_division
ZeroDivisionError: division by zero
```

Ici, l'erreur a lieu à l'intérieur d'une fonction. Python va retracer le fil de son exécution : en effet, il nous indique une division par zéro dans la fonction. Mais le problème ne vient pas de la fonction, il vient des arguments qu'on lui a donnés ! Pour nous aider à les retrouver, Python nous indique aussi à quelle ligne on a appelé la fonction avec les mauvais arguments. La phrase « Traceback (most recent call last) : » signifie donc qu'on remonte nos pas, avec en dernière ligne, l'opération de la fonction qui a échoué, en avant-dernière ligne, l'endroit où on a appelé cette fonction, éventuellement, avant encore, l'endroit où on appelé la fonction qui a appelé la fonction, etc. Python refait l'historique des événements qui ont conduit à l'erreur.

Exercices

Exercice 9

Traduire les programmes suivants dans la langage Python :

```
1  FONCTION f(x):
2      RENDRE x**2 - 3*x + 1
3  FIN FONCTION
```

```
1  FONCTION moyenne(a,b,c):
2      S ← a + b + c
3      moy ← S/3
4      RENDRE moy
5  FIN FONCTION
```

a et b en fonction de x puis démontrer la conjecture.

Instruction	a	b
$a = x$		
$b = x$		
$a = a - 5$		
$a = a*a$		
$a = a - 13$		
$b = b*(b-10)$		
$b = b + 12$		

Exercice 10

On considère la fonction :

```
1  def joliot():
2      a = 1
3      b = 5
4      a = b - 3
5      b = 2*a
6      a = b
7      b = a
8      return a, b
```

Déterminer les valeurs renvoyées par cette fonction.

Exercice 12

On considère la fonction :

```
def produits(a, b, c):
    return (a*c, b*c)
```

- 1► Quel est le nom de cette fonction ?
- 2► Combien d'arguments possède cette fonction ?
- 3► Que renvoie l'instruction `produits(4,3,2)` ?
- 4► Écrire une instruction qui appelle la fonction `produits` avec des arguments bien choisis de façon à ce que les valeurs renvoyées soient (15,20).

Exercice 11

On considère la fonction :

```
def curie(x):
    a = x
    b = x
    a = a - 5
    a = a*a
    a = a - 13
    b = b*(b-10)
    b = b + 12
    return a, b
```

- 1► Déterminer les valeurs renvoyées par l'appel `curie(x)` pour :
 - a► $x = 3$
 - b► $x = 12.5$
 - c► $x = 5$
- 2► Que peut-on conjecturer ?
- 3► Compléter le tableau ci-dessous en exprimant

Exercice 13

- 1► Écrire une fonction `peri_carre` qui prend en argument la longueur du côté d'un carré et renvoie son périmètre.
- 2► Faire un appel de cette fonction afin d'afficher le périmètre d'un carré de côté 10.

Exercice 14

- 1► Écrire une fonction `calcul` qui traduit le programme de calcul suivant : on choisit un nombre, on lui ajoute 5, et on double le résultat.
- 2► Faire un appel de cette fonction afin d'afficher le résultat pour la valeur 10 au départ.

Exercice 15

Écrire une fonction `pythagore` qui prend comme arguments les longueurs des deux côtés perpendiculaires d'un triangle rectangle, et renvoie la longueur de son hypoténuse.

On utilisera la fonction `sqrt` du module `math`.

Exercice 16

Écrire une fonction `disque` qui prend comme argument le rayon d'un disque et renvoie son aire.

On utilisera la constante `pi` fournie par le module `math`.

Exercice 17

Écrire une fonction `des` qui ne prend aucune argument et renvoie la somme du lancer de deux dés à 6 faces.

On utilisera la fonction `randint` du module `random`.

Exercice 18

Réalisez les figures des exercices 47, 48 et 49 du chapitre d'algorithmique à l'aide du module `turtle`.

5 Python : boucles et conditionnement

1 – Les structures itératives

A – La boucle inconditionnelle (ou bornée) : la boucle *for*

En Python, la boucle inconditionnelle (ou bornée) se construit à partir des mots-clés *for* et *in* :

Exemple 1

Langage naturel

```
1 POUR n ALLANT DE 1 A 10:  
2   AFFICHER 2 * n  
3 FIN POUR
```

Python

```
1 for n in range(1, 11):  
2     print(2*n)
```

Scratch



Remarque 2

- Python n'a pas d'équivalent de FIN POUR : la boucle s'arrête là où l'indentation s'arrête. C'est pourquoi vous devez prêter le plus grand soin à vos indentations !
- Remarquez que pour dire que le compteur *n* doit varier entre 1 et 10, on utilise la syntaxe `range(1,11)` : en particulier, la première borne est incluse, et la seconde est exclue !
- Constatez qu'avec Scratch, le compteur n'est pas automatiquement créé par la boucle ! Il faut le définir et l'incrémenter à la main.

Remarque 3

Quand on se fiche du compteur et qu'on souhaite juste répéter un bloc d'instructions un certain nombre de fois (comme Scratch le fait par défaut), on peut écrire `for i in range(20)` : pour répéter les instructions 20 fois. (On peut même écrire `for _ in range(20)` : pour signifier que l'on n'a pas besoin d'utiliser le compteur.)

Remarque 4 (Pour les experts)

- Lorsque la première borne n'est pas précisée, elle est implicitement définie à 0. Autrement dit, `range(15)` est un raccourci pour `range(0,15)`.
- La fonction `range` prend un troisième paramètre qui correspond au pas (de deux en deux, trois en trois, voire -1 pour indiquer un parcours décroissant). Exemples : `range(1,10,2)` donne 1, 3, 5, 7, 9 ; `range(5,0,-1)` donne 5, 4, 3, 2, 1.

B – La boucle conditionnelle (ou non bornée) : la boucle *while*

En Python, la boucle conditionnelle (ou non bornée) se construit à partir du mot-clé *while* :

Exemple 5

Langage naturel	Python	Scratch
<pre> 1 x ← 0 2 TANT QUE 2*x ≤ 10: 3 x ← x + 2 4 FIN TANT QUE 5 AFFICHER x </pre>	<pre> 1 x = 0 2 while 2*x <= 10: 3 x = x + 2 4 print(x) </pre>	

Remarque 6

Notez la différence entre la boucle conditionnelle de Scratch et elle de Python. En Scratch, on boucle *jusqu'à* ce que la condition soit vraie, autrement dit *tant que* la condition est fausse. En Python, on boucle *tant que* la condition est vraie. Ainsi, la condition de la boucle non bornée est inversée entre Python et Scratch !

2 – Les structures conditionnelles

A – Les opérateurs logiques

Python vient avec son lot d'opérateurs de comparaison. Ce sont des opérateurs *mathématiques* : ainsi, Python considère que 2 et 2.0 ont la même valeur (mathématique), bien qu'ils soient de types différents.

Opération	Notation	Exemple 1	Exemple 2
Égal à	==	>>> 2 == 2.0 True	>>> 3 == 15 False
Différent de	!=	>>> 2 != 2.0 False	>>> 3 != 15 True
Inférieur (strict) à	<	>>> 2 < 2.0 False	>>> 3 < 15 True
Inférieur ou égal à	<=	>>> 2 <= 2.0 True	>>> 3 <= 15 False
Supérieur (strict) à	>	>>> 2 > 2.0 False	>>> 3 > 15 False
Supérieur ou égal à	>=	>>> 2 >= 2.0 True	>>> 3 >= 15 False

On peut également tester l'égalité ou la différence de deux chaînes de caractères :

Opération	Notation	Exemple 1	Exemple 2
Égal à	==	>>> "Python" == "Python" True	>>> "Fourchette" == "Couteau" False
Différent de	!=	>>> "Python" != "Python" False	>>> "Fourchette" != "Couteau" True

Remarque 7

Les opérateurs `<`, `<=`, `>`, `>=` fonctionnent aussi avec les chaînes de caractères : ils utilisent l'ordre *lexicographique* (ce qui signifie l'ordre alphabétique, étendu aux chiffres et aux symboles, et en distinguant les majuscules des minuscules).

B – Les mots-clés `if`, `elif` et `else`

La structure conditionnelle la plus simple se construit avec le mot clé `if` :

Exemple 8

Langage naturel

```

1  SI n < 10:
2    AFFICHER "C'est plus petit que 10"
3  FIN SI

```

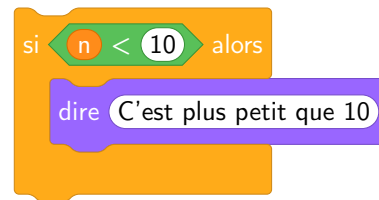
Python

```

1  if n < 10:
2    print("C'est plus petit que 10")

```

Scratch

**Remarque 9**

En Python, l'indentation seule définit un bloc de commandes ; celui-ci se termine là où l'indentation s'arrête. Il n'y a pas d'équivalent à `Fin Si` en Python. N'oubliez pas les `:` à la fin de la condition !

La structure conditionnelle avec alternative se construit avec le mot-clé `else` :

Exemple 10

Langage naturel

```

1  SI n < 10:
2    AFFICHER "C'est plus petit que 10"
3  SINON:
4    AFFICHER "C'est au moins égal à 10"
5  FIN SI

```

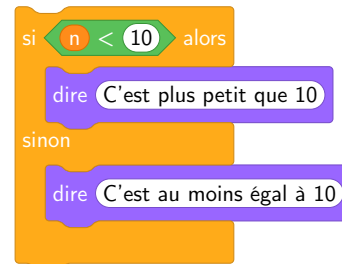
Python

```

1  if n < 10:
2    print("C'est plus petit que 10")
3  else:
4    print("C'est au moins égal à 10")

```

Scratch



La structure conditionnelle avec plusieurs tests se construit avec le mot-clé `elif` :

Exemple 11

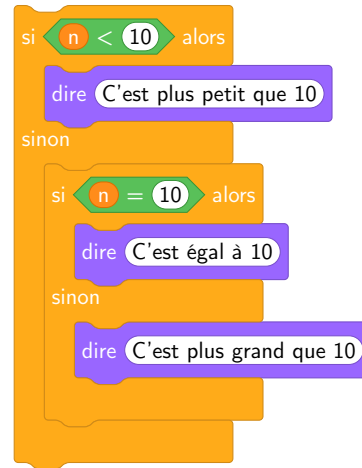
Langage naturel

```
1  SI n < 10:  
2    AFFICHER "C'est plus petit que 10"  
3  SINON SI n = 10:  
4    AFFICHER "C'est égal à 10"  
5  SINON:  
6    AFFICHER "C'est plus grand que 10"  
7  FIN SI
```

Python

```
1  if n < 10:  
2      print("C'est plus petit que 10")  
3  elif: n == 10:  
4      print("C'est égal à 10")  
5  else:  
6      print("C'est plus grand que 10")
```

Scratch



Exercices

Boucle bornée

Exercice 12

Les instructions `avancer()`, `droite()`, ... s'écrivent de la même manière en Python et en langage naturel. Traduire les programmes suivants en langage Python :

```
1 droite()
2 POUR i ALLANT DE 1 A 5:
3     avancer()
4 FIN POUR
```

```
1 gauche()
2 POUR k ALLANT DE 1 A 12:
3     sauter()
4     avancer()
5     coup()
6 FIN POUR
7 ouvrir()
```

```
1 POUR i ALLANT DE 1 A 5:
2     POUR j ALLANT DE 1 A 6:
3         avancer()
4     FIN POUR
5     tourner()
6 FIN POUR
```

Exercice 13

Pour chaque question, choisir la ou les bonnes réponses.

- 1► Dans la boucle `for k in range(2,7)`, la variable `k` prend les valeurs :
 - a► 2 et 7
 - b► 2, 3, 4, 5 et 6
 - c► 2, 3, 4, 5, 6 et 7
 - d► 2, 3, 4, 5, 6, 7 et 8
- 2► Dans la boucle `for k in range(5)`, la variable `k` prend les valeurs :
 - a► 0, 1, 2, 3 et 4
 - b► 0, 1, 2, 3, 4 et 5
 - c► 1, 2, 3 et 4
 - d► 1, 2, 3, 4 et 5
- 3► Lorsqu'on veut qu'une variable `k` prenne successivement toutes les valeurs entières de 0 à 33, on peut utiliser l'instruction `for k in ...` :

- a► `range(0,33)`
- b► `range(0,34)`
- c► `range(33)`
- d► `range(34)`

- 4► Dans la boucle `for k in range(2,29)`, combien `k` prend-elle de valeurs ?

- a► 29
- b► 28
- c► 27
- d► 26

Exercice 14

Déterminer la valeur de la variable `d` après l'exécution du programme dans chaque cas ci-dessous.

```
1 a, b, c, d = 1, 1, 1, 0
2 for k in range(2):
3     a = a + b
4     b = b + a
5     c = c + b
6 d = d + c
```

```
1 a, b, c, d = 1, 1, 1, 0
2 for k in range(2):
3     a = a + b
4     b = b + a
5 c = c + b
6 d = d + c
```

```
1 a, b, c, d = 1, 1, 1, 0
2 for k in range(2):
3     a = a + b
4 b = b + a
5 c = c + b
6 d = d + c
```

Exercice 15

Alice verse 20 000€ sur son livret d'épargne. Celui-ci lui rapporte 100€ par an. Écrire une fonction `capital` qui prend comme argument une variable `n`, et renvoie le capital d'Alice au bout de `n` années. On pourra s'appuyer sur le squelette d'algorithme suivant :

```
1 C ← ...
2 POUR i ALLANT DE 1 A ...:
3     C ←
4 FIN POUR
```

Exercice 16

Bob s'entraîne pour une course de vélo. Il établit un programme sur 12 semaines : la première il parcourt 25km, et augmente la distance de 11km chaque semaine.

- 1▶ Déterminer la distance D parcourue la deuxième semaine, et la distance totale cumulée T parcourue au bout de 2 semaines.
 - 2▶ Écrire une fonction distance qui ne prend aucun argument et renvoie la distance totale cumulée parcourue au bout de 12 semaines.
- On s'appuiera sur le squelette d'algorithme suivant :

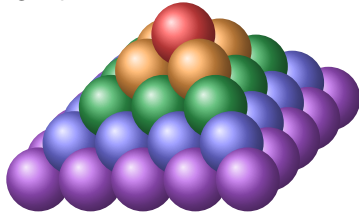
```

1  D ← ...
2  T ← ...
3  POUR i ALLANT DE 2 A 12:
4      D ← ...
5      T ← ...
6  FIN POUR

```

Problème 17

On empile des boules, formant une pyramide à base carrée comme illustré ci-dessous. On a coloré les différents étages pour une meilleure lisibilité.



- 1▶ Combien de boules faut-il pour faire une pyramide à 2 étages ?
- 2▶ Combien de boules faut-il pour faire une pyramide à 5 étages ?
- 3▶ Sachant qu'il faut 140 boules pour faire une pyramide à 7 étages, combien de boules faut-il pour faire une pyramide à 8 étages ?
- 4▶ Compléter la fonction ci-dessous qui permet de calculer le nombre de boules nécessaires pour construire une pyramide à n étages.

```

1  def nb_boules(...):
2      ...
3      for i in range(1,...):
4          ...
5      return ...

```

Boucle non bornée**Exercice 18**

Les instructions avancer(), droite(), ... s'écrivent de la même manière en Python et en langage naturel. Traduire les programmes suivants en langage Python :

```

1  TANT QUE peut_avancer():
2      avancer()
3  FIN TANT QUE

```

```

1  TANT QUE obstacle_devant():
2      coup()
3      sauter()
4      avancer()
5  FIN TANT QUE
6  avancer()
7  ouvrir()

```

```

1  TANT QUE cible_devant():
2      POUR j ALLANT DE 1 A 6:
3          tirer()
4      FIN POUR
5      tourner()
6  FIN TANT QUE

```

Exercice 19

On donne trois version d'une fonction calc :

```

1  def calc(a,b):
2      n = 0
3      while n < 20:
4          n = a + b
5          a = 2*a
6          b = b*3
7          n = n*b
8      return n

```

```

1  def calc(a,b):
2      n = 0
3      while n < 20:
4          n = a + b
5          a = 2*a
6          b = b*3
7      n = n*b
8      return n

```

```

1 def calc(a,b):
2     n = 0
3     while n < 20:
4         n = a + b
5         a = 2*a
6         b = b*3
7         n = n*b
8     return n

```

Pour chaque version de la fonction, déterminer le renvoi de `calc(5,1)`. On dressera un tableau d'avancement dans chaque cas.

Exercice 22

Dans une reprographie, le coût de l'impression en fonction du nombre de pages est donné par l'algorithme suivant :

```

1 if nbPages <= 50:
2     prix = 0.15*nbPages
3 else:
4     prix = 0.10*nbPages

```

Combien coûte l'impression d'un document de 25 pages ? De 75 pages ? De 50 pages ?

Structure conditionnelle

Exercice 20

Les commandes `avancer()`, `droite()`, ... s'écrivent de la même manière en Python et en langage naturel. Traduire les programmes suivants en langage Python :

```

1 SI obstacle_devant():
2     coup()
3 SINON:
4     avancer()
5 FIN SI

```

```

1 SI hauteur > 2:
2     tourner()
3 SINON SI hauteur = 2:
4     sauter_haut()
5 SINON:
6     sauter()
7 FIN SI

```

Exercice 21 (Déclic 2022)

On considère la fonction `mystere` ci-contre.

- 1► Quels sont les paramètres de cette fonction ?
- 2► Quels sont renvois des appels suivants ?

a► `mystere(3,4)` b► `mystere(12,5)`

```

1 def mystere(x, y):
2     if x < 10:
3         return 3*x - y
4     else:
5         return x + y

```

Exercice 23

Déterminer le renvoi le fonction `abs` ci-dessous pour $x = -2$, $x = 3$, $x = -5.8$ et $x = 0$.

```

1 def abs(x):
2     if x >= 0:
3         return x
4     else:
5         return -x

```

Exercice 24

Écrire une fonction `mini` qui renvoie le plus petit de ses deux arguments.

Exercice 25

Un manège fait payer son entrée en fonction de la taille et l'âge du client, selon l'algorithme suivant :

```

1 def prix(taille, age):
2     if taille <= 90:
3         entree = 0
4     else:
5         if 90 <= taille < 130:
6             entree = age*0.5
7         else:
8             entree = age*1.1
9     return entree

```

Combien doit payer un client de 18 ans mesurant 1,82m ? Et un enfant de 8 ans mesurant 1,20m ?

Exercice 26

```

1  def comp(a,b):
2      if a < b:
3          c = 0
4      else:
5          if b < a:
6              c = 1
7          else:
8              c = 2
9      return c

```

- 1► Déterminer le renvoi de comp dans les cas suivants :
 - a► $a = 6$ et $b = -3$
 - b► $a = 4$ et $b = 7$
 - c► $a = -7.12$ et $b = -7.1$
 - d► $a = 29/6$ et $b = 82/17$
- 2► Dans quel(s) cas la fonction comp renvoie-t-elle 2 ?

Exercice 27

Jouez au jeu PyRates disponible en ligne à l'adresse <https://py-rates.fr>.

Problème 28 (Suite de Syracuse)

- 1► Réaliser l'exercice 40 du chapitre d'algorithmique en langage Python, dans une fonction syracuse.
- 2► Modifier cette fonction pour qu'elle renvoie le nombre de tours de boucle avant que la suite n'arrive à 1.
- 3► Que fait l'algorithme suivant ?

```

1  n = 1
2  while syracuse(n) <= 100:
3      n + n + 1

```

Problème 29

Un camping propose deux formules de location :

- Réservation à 250€ + location à 10€ par jour
- Réservation à 300€ + location à 5€ par jour

Parmi les fonctions suivantes, laquelle ou lesquelles permet(tent) de déterminer la formule la plus rentable en fonction du nombre de jours ? Justifier en expliquant chaque algorithme en détail (ligne par ligne).

```

1  def algo1(n):
2      A = 250 + 10*n
3      B = 300 + 5*n
4      if A < B:
5          return "B"
6      else:
7          return "A"

```

```

1  def algo2(n):
2      A = 250 + 10*n
3      B = 300 + 5*n
4      if A < B:
5          return "A"
6      else:
7          return "B"

```

```

1  def algo3(n):
2      D = 5*n - 50
3      if D > 0:
4          return "B"
5      else:
6          return "A"

```

6 Python : les listes

7 Les langages du web

Thème d'étude III

Projets de fin d'année
