Chapitre 1
Programmation en langage Python

Objectifs

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

Culture scientifique

Considérée comme la première algorithmicienne, **Ada Lovelace (1815-1852)** a collaboré à la conception de la machine analytique. Plus d'un siècle avant l'apparition des premiers ordinateurs, elle conceptualise la notion de programme informatique. Elle est la première à utiliser le terme algorithme dans son travail. Un langage de programmation, portant son nom, a été développé par l'armée américaine dans les années 1980.

Et sinon, dans la vraie vie ?

Les algorithmes d'optimisation de trajet permettent de définir et d'évaluer un grand nombre d'itinéraires possibles. Ils s'appuient sur des données cartographiques, des informations sur les réseaux de transport et des données sur le trafic en temps réel.

Ils constituent un outil puissant pour améliorer l'efficacité et l'expérience des usagers des transports en commun : gain de temps, données précises, meilleure gestion des réseaux de transport.

Cours et applications directes

Compléments numériques
↘ *Sur Sacado via votre ENT*
À consulter dans "Livre numérique" en indiquant le numéro de page : **10**

A Variables et affectation

↘ Définition

Une **variable** (zone mémoire étiquetée) sert à stocker une information qui peut être sous la forme d'un nombre, une phrase, une liste, une fonction...

Vocabulaire des variables

Type	*entier*	*flottant*	*chaîne de caractères*	*booléen*
Python	`int`	`float`	`str`	`bool`
Exemple	3	3.2	"Bonjour !"	True

Remarques

· L'affectation des variables dans Python se fait avec le symbole =.
Ainsi **a=1** correspond à l'instruction stocker la valeur **1** dans la variable **a**.
Autrement dit, la variable **a** prend la valeur **1**, ce que l'on écrit de façon synthétique $a \leftarrow 1$.

· Le nom d'une variable ne peut ni être un nombre ni certains mots réservés.
Il est fortement recommandé de donner des noms explicites à tous les objets que l'on crée.

B Les fonctions `print` et `input`

↘ Définition

· Pour afficher une variable, un résultat ou un message, Python dispose de la fonction : `print`

```
1 print(a)
2 print("4")
3 print("Coucou")
```

→ affiche le contenu de la variable a.
→ affiche 4, 4 est une chaîne de caractères.
→ affiche Coucou.

· Pour permettre à un utilisateur de donner une valeur à une variable, Python utilise la fonction : `input()`.
Généralement, on indique à l'utilisateur ce que l'on attend de lui par un message dans les parenthèses.

```
1 nom = input("Quel est ton nom ?")
```

La variable `nom` prend la valeur donnée par l'utilisateur.

Remarques

· La fonction `input` renvoie une chaîne de caractères.
· On utilise `int` pour changer le type de la chaîne de caractères en un **nombre entier**, ou `float` pour la changer en un **nombre non entier** (dit flottant).

Exemples

```
1 age = int(input("Quel est ton âge en années ?"))
2 taille = float(input("Quelle est ta taille en mètre ?"))
```

On souhaite afficher le carré d'un nombre entier saisi par un utilisateur.

```
1 a = int(input("Saisir un nombre entier "))
2 print("le carré du nombre", a," est ", a**2)
```

Compléments numériques

↘ *Sur Sacado via votre ENT*
À consulter dans "Livre numérique"
en indiquant le numéro de page : 11

1 Voici un algorithme incomplet écrit en Python qui calcule la somme de deux nombres entiers **a** et **b** donnés par l'utilisateur. Compléter ce script.

```
1 a=..........(input("Entrer un nombre "))
2 b=..........(input("Entrer un nombre "))
3 print(a,"+",b,"=",..........)
```

2 a. Écrire une ligne de commande qui demande à un utilisateur de choisir un nombre entier et qui le stocke dans la variable **a**.

b. Écrire une ligne de commande qui affiche la chaîne : **"Tu as choisi le nombre : x"** où **x** est la valeur de la variable **a** saisie par l'utilisateur.

3 Traduire l'algorithme de langage naturel suivant en Python :

- demander à l'utilisateur un nombre ;
- prendre son carré ;
- ajouter 3 ;
- afficher le résultat.

4 Lors de l'exécution du programme suivant :

```
1 nombre=input("Entrez un nombre : ")
2 print("Vous avez saisi le nombre",nombre)
```

L'utilisateur a saisi **2+3**.

a. Que va afficher ce programme ? Quel est le type de la variable nombre ?

b. On exécute les lignes suivantes. Laquelle provoque une erreur ? Que contiennent les différentes variables ?

```
a = "2*3"
b = 2*3
c = 2*"3"
d = 2 + "3"
e = 5 < 4
```

5 On dispose de deux variables **a** et **b**. Écrire une séquence d'affectations qui échange les contenus des deux variables (la nouvelle valeur de **a** doit être l'ancienne valeur de **b**, et la nouvelle valeur de **b** doit être l'ancienne valeur de **a**).

Cours et applications directes

Compléments numériques

↘ *Sur Sacado via votre ENT*
À consulter dans "Livre numérique"
en indiquant le numéro de page : 12

C Variable booléenne

↘ Définition

Un test est une **expression booléenne**, une expression qui ne prend que deux valeurs : `Vrai` ou `Faux`.
En Python, les valeurs sont `True` et `False`.

D Instruction conditionnelle

↘ Définition

Une **instruction conditionnelle** est une instruction qui exécute une action parmi deux options, suivant le résultat d'un test.

En langage naturel

Si condition alors instruction1 sinon instruction2.

Script Python

```
if condition :
    instruction1
else :
    instruction2
```

Remarque

Les "`:`" en fin de ligne du `if` sont obligatoires. Ne pas oublier l'**indentation** (décalage) des blocs d'instructions.

Exemple

Algorithme

Saisir un entier n
Si n est pair alors afficher : n est un nombre pair
Sinon afficher : n est un nombre impair

Script Python

```
n = int(input("Entrer un entier : "))
if n % 2 == 0 :
    parite = "pair"
else :
    parite = "impair"
print(n, " est un nombre ", parite)
```

E Comparaison en Python

↘ Définition

Plusieurs symboles de comparaison entre deux nombres sont utilisés en Python :

Langage naturel	*égal à*	*différent de*	*strictement supérieur*	*strictement inférieur*	*supérieur ou égal*	*inférieur ou égal*
Python	==	!=	>	<	>=	<=

F Boucle conditionnelle

↘ Définition

Une **boucle conditionnelle** est une boucle qui répète une suite d'instructions (appelée corps de la boucle) tant qu'une condition est satisfaite.

Langage naturel	*Tant que condition faire : instructions*
Python	`while condition : instructions`

Exemple

Algorithme

$p \longleftarrow 0$
$i \longleftarrow 0$
tant que $p < 100$:
    $p \longleftarrow 4 * i$
    $i \longleftarrow i + 1$
afficher (p)

Script Python

```
p = 0
i = 0
while p < 100:
    p = 4 * i
    i = i + 1
print(p)
```

Compléments numériques
↘ *Sur Sacado via votre ENT*
À consulter dans "Livre numérique"
en indiquant le numéro de page : 13

6 Les variables **a** et **b** contiennent des nombres.
Écrire un programme qui affecte à une variable **m** le maximum de **a** et **b** sans utiliser la fonction native `max()`.

7 a, b et c sont les longueurs respectives entières des côtés BC, AC et AB d'un triangle ABC.
Écrire un programme demandant à l'utilisateur de saisir les longueurs des trois côtés et qui détermine si le triangle est rectangle (en quel sommet ?) ou non.

8 On considère le jeu suivant : on lance deux dés, si on obtient un double, on gagne 5 €,
si la somme des deux dés est égale à 7, on gagne 4 € et sinon, on perd 1 € (on gagne -1 €).
Écrire les tests qui, en fonction des valeurs **d1** et **d2** des deux dés, affectent le gain (en €) à une variable **g**.

9 Écrire un programme qui affiche les carrés des entiers, en partant de 0, jusqu'à ce que ce carré soit strictement supérieur à $1\,000$.

Cours et applications directes

Compléments numériques

↘ *Sur Sacado via votre ENT*
À consulter dans "Livre numérique"
en indiquant le numéro de page : 14

G Boucle itérative

↘ Définition

Une **boucle itérative** est une boucle dont on connaît le nombre de répétitions *a priori*.
Elle utilise une variable de boucle dont la valeur parcourt un intervalle d'entiers.

En Python, pour définir un intervalle d'entiers, on utilise la fonction `range`.

- `range(5,10)` définit les entiers entre 5 inclus et 10 exclu.
- `range(20)` définit les entiers entre 0 inclus et 20 exclu, comme le ferait `range(0,20)`.

Langage naturel	Pour i allant de à 0 à 9 : instructions
Python	`for i in range(10) : instructions`

Méthode

Langage naturel	i allant de 0 à 4	i allant de 1 à 10	i allant de 1 à 11 de 2 en 2
Python	`i in range(5)`	`i in range(1,11)`	`i in range(1,13,2)`

Exemple

Algorithme
Affichage des n premiers naturels :
Pour i allant de 0 à n :
    afficher (i)
Fin boucle pour

Script Python
Affichage des n premiers naturels :

```
1 for i in range(n):
2     print(i)
```

H Boucles imbriquées

↘ Définition

On peut **imbriquer** une boucle, c'est-à-dire placer une boucle dans le corps d'une autre boucle.

I Les fonctions

↘ Définition

Une **fonction** est une partie de code qui effectue une ou plusieurs opérations.

- Elle reçoit en entrée 0, 1 ou plusieurs paramètres et peut retourner éventuellement une valeur.
- Les fonctions commencent toujours par le mot clé `def` suivi du nom de la fonction et renvoient leur résultat par `return`.
- Le corps du code de la fonction doit être indenté.
- Les variables utilisées par la fonction ne sont pas reconnues à l'extérieur de la fonction. On dit que ce sont des variables locales.

Exemple
La fonction suivante admet deux paramètres **a** et **b** et renvoie leur somme et leur produit.

```
1 def somme (a,b): # Somme de 2 nombres
2     return a+b
```

```
1 def produit (a,b): # Produit de 2 nombres
2     return a*b
```

J Appel de fonction

↘ Définition

Pour appeler une fonction, on l'appelle directement dans le corps du programme.

```
1 def carre(n) :
2     return n**2
3 a = float(input('Saisir un nombre'))
4 print(carre(a))
```

Compléments numériques

↘ *Sur Sacado via votre ENT*
À consulter dans "Livre numérique"
en indiquant le numéro de page : 15

10 Écrire un programme Python qui affiche le premier entier naturel n pour lequel $n^3 + n > 1\,000$.

11 Écrire deux programmes qui affichent :

a. Les entiers de 10 à 20 inclus.

b. Les doubles des entiers de la question précédente.

12 Écrire un programme qui demande à l'utilisateur un entier strictement positif n et affiche tous les diviseurs positifs de n.

13 Un triplet $(a ; b ; c)$ d'entiers naturels non nuls est pythagoricien lorsque $a^2 + b^2 = c^2$.
Compléter le programme suivant pour qu'il affiche tous les triplets pythagoriciens avec $c < 100$:

```
for a in range(1 , .....) :
    for b in range(.........) :
        for ........................
            if .......... == ........... :
                print(a, "²+", b, "²=", c, "²")
```

14 Compléter la fonction qui renvoie le minimum entre deux nombres entiers passés en paramètre.

```
def minimum (.....,.....) :
    if a<b :
        m= ...........
    else :
        m= ...........
    return m
```

Exercices | Parcours 1

Compléments numériques

↘ *Sur Sacado via votre ENT*
À consulter dans "Livre numérique" en indiquant le numéro de page : 16

1 Voici un programme codé en Python.

```
n = int (input("Saisir un nombre entier"))
u = 0.6931
for i in range(1,n+1) :
        u = 1/i - u
print(u)
```

À l'aide de ce programme, on a obtenu le tableau de valeurs suivant :

n	0	1	2	3	4	5	10	100
Résultat	0,6931	0,3069	0,1931	0,1402	0,1098	0,0902	0,0475	0,0049

Écrire ce programme dans une console Python et retrouver ces résultats.

2 Compléter la fonction distance qui calcule la distance entre deux points du plan.

```
from math import sqrt
# importation de la racine carrée (square root en anglais) du module math
def distance (a, b, c, d):
        # a,b sont les coordonnées du premier point
        # c,d sont les coordonnées du second point
        return   sqrt(.....)
```

3 Voici un programme écrit en Python.

```
n=0
u = int(input("Saisir M ="))
while u > 50 :
        u = 0.9*u
        n=n+1
print(n)
```

a. Compléter le tableau ci-dessous en testant ce programme à la main (en débranché).

Étapes	u	n	Conditions
0	80	0	Vraie
1			
2			
3			
4			
5			

b. Écrire ce programme sur une console Python et retrouver les valeurs pour `M = 80`.

c. Expliquer le rôle de ce programme.

.....

.....

Exercices | Parcours 2

Compléments numériques

↘ *Sur Sacado via votre ENT*
À consulter dans "Livre numérique"
en indiquant le numéro de page : 17

1 a. Écrire l'algorithme suivant en Python.

```
u ⟵ 0
n ⟵ 0
Tant que u < 100
    u ⟵ 2n + 1
    n ⟵ n + 1
Afficher n
```

b. Que fait cet algorithme ?

c. Modifier ce programme pour qu'il affiche toutes les valeurs de u.

2 Créer une fonction qui prend comme paramètre un nombre entier naturel n et renvoie le produit des entiers naturels non nuls inférieurs ou égaux à n. On appelle ce nombre « factorielle n » et on l'écrit « $n!$ ».

3 Écrire un programme Python qui demande un multiple de 7 à l'utilisateur. Tant que l'utilisateur n'a pas donné un multiple de 7, le programme le lui demande à nouveau.
Aide : pour tester si un entier n est un multiple de 7, il suffit de calculer le reste de la division euclidienne de n par 7, ce qui en Python s'écrit **n%7**, *et de tester si ce reste est nul.*

Exercices | Parcours 3

Compléments numériques

↘ *Sur Sacado via votre ENT*
À consulter dans "Livre numérique" en indiquant le numéro de page : 18

1 On souhaite déterminer les images des entiers compris entre 1 à 5 par la fonction f définie par $f(x) = x^2$.

a. Écrire un algorithme.

b. Coder cet algorithme en Python.

2 Soit a un entier compris entre 0 et 100 donné par un utilisateur. Écrire un programme de calcul qui détermine la solution de l'équation $x^2 = a$ par balayage avec une précision de 10^{-2} près.

3 Écrire un programme en Python qui génère tous les nombres entiers supérieurs à 8 en tant que combinaisons linéaires des nombres 5 et 3.

Autoévaluation

Corrigé QCM
Sur Sacado via votre ENT
À consulter dans "Livre numérique" en indiquant le numéro de page : 19

QCM · Pour chacune des 10 questions suivantes, une seule des trois réponses proposées est exacte.

Ma note / 10

1 Lorsqu'on tape dans une console Python `2+"3"`, on obtient :

a. ☐ 5
b. ☐ 23
c. ☐ une erreur

2 Pour `a = 4`, que renvoie le code suivant :

```
if a < 12 :
    m = "matin"
else :
    m = "après-midi"
print(m)
```

a. ☐ matin
b. ☐ une erreur
c. ☐ après-midi

3 Lorsque `a = 3`, `print("a")` renvoie :

a. ☐ 3
b. ☐ a
c. ☐ une erreur

4 Une expression booléenne ne prend que deux valeurs qui sont, en Python :

a. ☐ 1 ou 2
b. ☐ `False` ou `True`
c. ☐ Oui ou Non

5 Que renvoie le code suivant :

```
a=0
while a <= 20 :
    a = a + 1
print(a)
```

a. ☐ une erreur
b. ☐ 20
c. ☐ 21

6 Que renvoie le code suivant :

```
for i in range (10) :
    print(i)
```

a. ☐
0
1
2
3
4
5
6
7
8
9

b. ☐
0
1
2
3
4
5
6
7
8
9
10

c. ☐ 110

7 Que renvoie le code suivant :

```
for i in range (12) :
    m=2*i+1
print(m)
```

a. ☐
1
3
5
7
11
13
15
17
18

b. ☐ 18

c. ☐ 19

8 L'instruction algorithmique $a \longleftarrow b$ signifie :

a. ☐ a est stocké dans b
b. ☐ b est stocké dans a
c. ☐ cette instruction n'existe pas

9 Par l'affectation suivante, `a = input("Renseigner un nombre")`, la variable a est :

a. ☐ un nombre
b. ☐ une chaîne de caractères
c. ☐ ça dépend de ce que tape l'utilisateur

10 Une fonction est définie par le mot clé :

a. ☐ `def`
b. ☐ `return`
c. ☐ `function`