Algorithmique. Fiche d'exercices n°1

Initiation à Python

Les bases à connaître

À connaître — variables et affectation

Une **variable** est un nom qui désigne une valeur. Le signe = est une **affectation** : « la variable de gauche reçoit la valeur de droite » ; ce n'est pas une égalité mathématique, `x = x + 1` signifie « augmenter `x` de 1 ». Les valeurs ont un **type** : entier (`int`), décimal (`float`, le séparateur est le point), chaîne de caractères (`str`, entre guillemets), booléen (`bool` : `True` ou `False`).

Python
```
n = 12          # entier
x = 2.5         # décimal
nom = "Léa"     # chaîne de caractères
n = n + 3       # n vaut maintenant 15
print(n, x, nom)
```

affichage
```
15 2.5 Léa
```

À connaître — opérations

`+`, `-`, `*`, `/` sont les quatre opérations ; `**` est la puissance ; `//` est le **quotient entier** et `%` le **reste** de la division euclidienne. Les fonctions mathématiques s'importent : `from math import sqrt` (racine carrée), `from math import pi`.

Python
```
from math import sqrt
print(17 / 5, 17 // 5, 17 % 5)
print(2 ** 10, sqrt(16), 3 * (4 + 1))
```

affichage
```
3.4 3 2
1024 4.0 15
```

À connaître — afficher et saisir

`print(...)` affiche ses arguments séparés par une espace ; `print(x, end=" ")` affiche `x` sans aller à la ligne. `input("texte")` affiche le texte et renvoie ce que l'utilisateur tape, toujours sous forme de **chaîne** : pour obtenir un nombre on écrit `int(input(...))` (entier) ou `float(input(...))` (décimal).

Python
```
n = int(input("Un entier : "))
print("Le double de", n, "est", 2 * n)
```

affichage
```
Un entier : 8
Le double de 8 est 16
```

À connaître — tests et instruction conditionnelle

Une **condition** est une expression vraie ou fausse : comparaisons `==` (égal), `!=` (différent), `<`, `<=`, `>`, `>=` ; on les combine avec `and`, `or`, `not`. L'instruction `if condition:` exécute le bloc **indenté** (décalé de 4 espaces) qui suit lorsque la condition est vraie ; `else:` introduit le bloc exécuté sinon ; `elif` (« sinon si ») permet d'enchaîner plusieurs cas. Ne pas oublier les deux-points.

Python

```
x = 7
if x % 2 == 0:
    print(x, "est pair")
else:
    print(x, "est impair")
if x > 5 and x < 10:
    print("x est entre 5 et 10")
```

affichage

```
7 est impair
x est entre 5 et 10
```

À connaître — boucle bornée `for ... in range(...)`

`for i in range(n)` : répète le bloc indenté pour `i` valant successivement $0, 1, \ldots, n-1$ (donc n **fois**, la borne `n` est exclue). `range(a, b)` donne les entiers de `a` à `b - 1` ; `range(a, b, p)` avance de `p` en `p`. Pour calculer une somme ou un produit, on utilise une variable **accumulateur** initialisée avant la boucle.

Python

```
for i in range(1, 6):
    print(i, end=" ")
print()
S = 0
for k in range(1, 5):
    S = S + k ** 2
print("Somme des carrés :", S)
```

affichage

```
1 2 3 4 5
Somme des carrés : 30
```

À connaître — boucle non bornée `while`

`while condition:` répète le bloc indenté **tant que** la condition est vraie ; on l'utilise quand on ne connaît pas à l'avance le nombre de répétitions (recherche d'un seuil). Le bloc doit modifier la condition, sinon la boucle ne s'arrête jamais. Un **compteur** permet de connaître le nombre de passages.

Python

```
n = 0
while 2 ** n < 100:
    n = n + 1
print("Plus petit n :", n)
```

affichage

```
Plus petit n : 7
```

À connaître — fonctions

`def nom(paramètres):` définit une **fonction** ; le bloc indenté calcule et `return` **renvoie** le résultat (et termine la fonction). On **appelle** ensuite la fonction par `nom(valeurs)`, autant de fois que l'on veut. `return` renvoie une valeur réutilisable dans un calcul ; `print` ne fait qu'afficher.

Python

```
def f(x):
    return x ** 2 - 3 * x

def aire_disque(r):
    return 3.14159 * r ** 2

print(f(5), f(-1), f(1.5))
print(aire_disque(2))
```

affichage

```
10 4 -2.25
12.56636
```

Variables, calculs et affichage

Exercice 1 ⋆

Lire un programme.

a. Qu'affiche le programme ci-dessous ? Répondre sans ordinateur, puis vérifier.

```python
a = 7
b = 2
print(a + b, a - b, a * b)
print(a / b, a // b, a % b)
print(a ** b)
```

b. Donner la valeur de `x` après chaque ligne du programme suivant, puis ce qu'il affiche.

```python
x = 4
x = x + 3
x = 2 * x
x = x - 1
print(x)
```

Exercice 2 ⋆

Échanger deux variables. On veut échanger les valeurs des variables `a` et `b`.

a. Qu'affiche le programme ci-contre ? Expliquer pourquoi il ne réalise pas l'échange.

```python
a = 3
b = 8
a = b
b = a
print(a, b)
```

b. Corriger le programme en utilisant une troisième variable `c`.

Exercice 3 ⋆

Rectangle.

a. Écrire un programme qui demande la longueur et la largeur d'un rectangle (nombres décimaux) puis affiche son périmètre et son aire.

b. Le compléter pour qu'il affiche aussi la longueur de la diagonale, arrondie au centième (on pourra utiliser `round(d, 2)`).

Instructions conditionnelles : `if ... else`

Exercice 4 ⋆

Signe et parité.

a. Qu'affiche le programme ci-dessous ? Et si l'on remplace la première ligne par `x = 0` ?

```python
x = -3
if x >= 0:
    print("positif")
else:
    print("négatif")
```

b. Modifier le programme pour qu'il affiche `nul` lorsque `x` vaut 0.

c. Écrire un programme qui affiche si l'entier `n` est pair ou impair (on utilisera le reste `n % 2`).

Exercice 5 ⋆⋆

Résultat d'un examen.

a. Compléter le programme pour qu'il affiche `Admis` si la note est supérieure ou égale à 10 et `Refusé` sinon.

```python
note = float(input("Note sur 20 : "))
if ... :
    print(...)
else:
    print(...)
```

b. Modifier le programme avec `elif` pour qu'il affiche `Admis avec mention` à partir de 14. Que donne-t-il pour les notes 15 ; 12,5 ; 8 ?

Exercice 6 ⋆⋆

Valeur absolue et triangle rectangle.

a. Écrire un programme qui affiche la valeur absolue d'un nombre `x` sans utiliser la fonction `abs`.

b. Écrire un programme qui demande les longueurs entières `a`, `b`, `c` des côtés d'un triangle, `c` étant le plus grand côté, et qui affiche si le triangle est rectangle ou non. Tester avec 3, 4, 5 puis avec 4, 5, 6.

Exercice 7 ⋆⋆⋆

Équation du premier degré. Écrire un programme qui demande deux nombres `a` et `b` et qui résout dans $\mathbb{R}$ l'équation $ax + b = 0$: il affiche la solution, ou `Tout nombre est solution`, ou `Pas de solution` selon les cas. Tester avec $a = 2, b = -5$; $a = 0, b = 3$; $a = 0, b = 0$.

Boucle bornée : `for ... in range`

Exercice 8 ⋆

Lire une boucle. Qu'affiche chacun des programmes suivants ? Combien de fois le bloc indenté est-il exécuté ?

```python
for i in range(4):
    print(i, end=" ")
```

```python
for i in range(2, 7):
    print(i, end=" ")
```

```python
for i in range(10, 0, -3):
    print(i, end=" ")
```

```python
for i in range(3):
    print("Bonjour")
```

Exercice 9 ⋆

Tables.

a. Écrire une boucle qui affiche la table de multiplication de 7, de `7 x 1 = 7` à `7 x 10 = 70`.

b. Écrire un programme qui affiche, pour les entiers de 1 à 10, la ligne `n au carré vaut n²`, par exemple `4 au carré vaut 16`.

Exercice 10 ⋆⋆

Sommes. Écrire, à l'aide d'un accumulateur, un programme qui calcule et affiche :

a. $1 + 2 + 3 + \cdots + 100$;

b. $1^2 + 2^2 + \cdots + 10^2$;

c. la somme des entiers pairs de 2 à 100 (on pourra utiliser `range(2, 101, 2)`).

Exercice 11 ⋆⋆

Tableau de valeurs d'une fonction. Soit f la fonction définie sur $\mathbb{R}$ par $f(x) = x^2 - 4x + 1$.

a. Écrire un programme qui affiche, pour chaque entier x de -2 à 6, la ligne `x f(x)`.

b. `range` ne produit que des entiers. Compléter le programme ci-dessous pour afficher le tableau de valeurs de f sur $[0\,;3]$ avec un pas de 0,5.

```python
for k in range(0, ...):
    x = ...
    print(x, x ** 2 - 4 * x + 1)
```

c. Que peut-on conjecturer sur le minimum de f ?

Exercice 12 ⋆⋆⋆

Compter et sommer sous condition. Écrire un programme qui :

a. compte le nombre de multiples de 7 compris entre 1 et 100 ;

b. calcule la somme des entiers strictement inférieurs à 100 qui sont multiples de 3 ou de 5 ;

c. affiche tous les entiers n de 1 à 50 dont le carré se termine par le chiffre 6. Que remarque-t-on ?

Boucle non bornée : `while`

Exercice 13 ⋆

Lire une boucle `while`.

a. Recopier et compléter le tableau des valeurs successives de `n` pour le programme ci-dessous, puis donner ce qu'il affiche.

```python
n = 1
while n < 50:
    n = n * 2
print(n)
```

b. Ajouter un compteur `k` qui compte le nombre de passages dans la boucle et afficher `n` et `k`.

Exercice 14 ⋆⋆

Un seuil. Un capital de 1 000 € est placé à 3 % par an : chaque année il est multiplié par 1,03. Écrire un programme qui détermine au bout de combien d'années le capital dépasse 1 500 €, et qui affiche ce nombre d'années ainsi que le capital obtenu arrondi au centime.

Exercice 15 ⋆⋆

Chercher et deviner.

a. Écrire un programme qui détermine le plus petit entier n tel que $1 + 2 + \cdots + n \geqslant 1\,000$ et qui affiche n et la somme obtenue.

b. L'ordinateur a choisi le nombre 37. Qu'affiche le programme ci-dessous si l'utilisateur propose successivement 20, 40 puis 37 ?

```python
secret = 37
essai = 0
nombre = 0
while essai != secret:
    essai = int(input("Proposition : "))
    nombre = nombre + 1
    if essai < secret:
        print("Plus grand")
    elif essai > secret:
        print("Plus petit")
print("Gagné en", nombre, "essais")
```

Exercice 16 ⋆⋆⋆

Une boucle qui ne s'arrête pas, et l'algorithme d'Euclide.

a. Expliquer pourquoi le programme ci-dessous ne s'arrête jamais, puis corriger la condition pour qu'il s'arrête. Quelle est alors la dernière valeur de `n` ?

```python
n = 10
while n != 0:
    n = n - 3
print(n)
```

b. Faire fonctionner « à la main » le programme ci-dessous en dressant le tableau des valeurs de `a`, `b` et `r` à chaque passage. Qu'affiche-t-il ? Quel résultat du cours reconnaît-on ?

```python
a = 756
b = 360
while b != 0:
    r = a % b
    a = b
    b = r
print(a)
```

Fonctions

Exercice 17 ⋆

Image par une fonction.

a. Qu'affiche le programme ci-dessous ?

```python
def f(x):
    return 2 * x ** 2 - 3 * x + 1

print(f(0), f(1), f(-2), f(0.5))
```

b. Écrire une fonction `g` correspondant à $g(x) = (x-1)(x+3)$, puis afficher $g(1)$, $g(-3)$ et $g(2)$.

Exercice 18 ⋆⋆

Fonctions à plusieurs paramètres.

a. Écrire deux fonctions `perimetre(L, l)` et `aire(L, l)` qui renvoient le périmètre et l'aire d'un rectangle, puis les utiliser pour un rectangle de côtés 5 et 3.

b. Écrire une fonction `distance(xA, yA, xB, yB)` qui renvoie la distance AB dans un repère orthonormé. L'utiliser pour montrer que le triangle ABC avec $A(1\,;2)$, $B(4\,;6)$ et $C(-3\,;5)$ est isocèle en A.

Exercice 19 ⋆⋆

`return` ou `print` ?

a. Qu'affiche le programme ci-dessous ? Expliquer la seconde ligne affichée et corriger la fonction.

```python
def double(x):
    print(2 * x)

y = double(5)
print(y)
```

b. Écrire une fonction `est_pair(n)` qui renvoie `True` si `n` est pair et `False` sinon. Que renvoient `est_pair(12)` et `est_pair(7)` ?

c. Écrire une fonction `moyenne(a, b, c)` qui renvoie la moyenne de trois notes, et l'utiliser pour les notes 12, 15 et 9.

Exercice 20 ⋆⋆⋆

Synthèse.

a. À partir de l'exercice 16, écrire une fonction `pgcd(a, b)` qui renvoie le PGCD de deux entiers. Écrire ensuite une fonction `simplifie(p, q)` qui affiche la fraction $\frac{p}{q}$ sous forme irréductible. Tester avec $\frac{360}{756}$.

b. À partir de l'exercice 14, écrire une fonction `seuil(c)` qui renvoie le nombre d'années au bout duquel un capital de 1 000 € placé à 3 % dépasse `c` euros. Que renvoient `seuil(1500)` et `seuil(2000)` ?

c. Écrire un programme qui, pour la fonction $f(x) = x^2 - 4x + 1$ de l'exercice 11, parcourt les valeurs de x de -2 à 6 avec un pas de 0,1 et affiche la plus petite valeur de $f(x)$ rencontrée et la valeur de x correspondante.