

Corrigé Python

Utiliser des variables, des types et des affectations

Exercice 1. Types et affectations

1. Donner le type de chacune des valeurs : 12; 12.0; "12"; 12 > 5; [1, 2]; 7 / 2.

int; float; str; bool (qui vaut True); list; float (qui vaut 3.5).

2. Donner la valeur des variables à la fin du programme :

```
a = 4
b = 3 * a - 1
a = b - a
b = a * b
```

$a = 4$ puis $b = 11$ puis $a = 11 - 4 = 7$ puis $b = 7 \times 11 = 77$. À la fin : $a = 7$ et $b = 77$.

3. Que valent 17 // 5 et 17 % 5?

17 // 5 vaut 3 (quotient) et 17 % 5 vaut 2 (reste) car $17 = 3 \times 5 + 2$.

4. Écrire un programme qui échange les valeurs de deux variables x et y.

```
temp = x
x = y
y = temp
```

5. Que contient la variable c après l'instruction $c = "3" + "4"$? et après $c = 3 + 4$? et après $c = \text{int}("3") * 4$?

"34" (concaténation de chaînes); 7; 12.

6. Écrire un programme qui demande le rayon r d'un disque puis affiche son aire (on utilisera pi du module math).

```
from math import pi
r = float(input("Rayon : "))
print(pi * r**2)
```

7. On exécute $x = 5$, puis $x = x + 2$, puis $x = x * x$. Que vaut x?

$x = 5$, puis 7, puis 49.

Écrire des instructions conditionnelles

Exercice 2. Instructions conditionnelles

1. Qu'affiche le programme ci-dessous pour $x = 3$, puis pour $x = -2$, puis pour $x = 0$?

```
if x > 0:
    print("positif")
elif x == 0:
    print("nul")
else:
    print("négatif")
```

positif;négatif;nul.

2. Écrire une fonction `tarif(age)` qui renvoie le prix d'une place de cinéma : 5 € pour les moins de 14 ans, 7,50 € de 14 à 25 ans inclus, 11 € au-delà.

```
def tarif(age):
    if age < 14:
        return 5
    elif age <= 25:
        return 7.5
    else:
        return 11
```

3. Écrire une fonction `est_pair(n)` qui renvoie `True` si l'entier n est pair et `False` sinon.

```
def est_pair(n):
    return n % 2 == 0
```

4. Écrire une fonction `maximum(a, b)` qui renvoie le plus grand des deux nombres a et b .

```
def maximum(a, b):
    if a >= b:
        return a
    else:
        return b
```

5. Écrire une fonction `bissextile(a)` qui renvoie `True` si l'année a est bissextile : divisible par 4 mais pas par 100, ou divisible par 400.

```
def bissextile(a):
    return (a % 4 == 0 and a % 100 != 0) or a % 400 == 0
```

bissextile(2024) vaut True, bissextile(1900) vaut False.

6. Écrire une fonction `mention(note)` qui renvoie « TB » pour une note d'au moins 16, « B » d'au moins 14, « AB » d'au moins 12, « Passable » d'au moins 10 et « Non admis » sinon.

```
def mention(note):
    if note >= 16:
        return "TB"
    elif note >= 14:
        return "B"
    elif note >= 12:
        return "AB"
    elif note >= 10:
        return "Passable"
    else:
        return "Non admis"
```

Utiliser des boucles bornées et non bornées

Exercice 3. Boucles bornées

1. Qu'affiche ce programme?

```
for i in range(4):
    print(2 * i + 1)
```

Il affiche 1, 3, 5, 7 (les valeurs de $2i + 1$ pour $i = 0, 1, 2, 3$).

2. Écrire un programme qui calcule la somme $1^2 + 2^2 + \dots + 20^2$.

```
S = 0
for k in range(1, 21):
    S = S + k**2
print(S)          # affiche 2870
```

3. Écrire une fonction `moyenne(L)` qui renvoie la moyenne des nombres d'une liste L, à l'aide d'une boucle.

```
def moyenne(L):
    S = 0
    for x in L:
        S = S + x
    return S / len(L)
```

4. Écrire un programme qui affiche la table de multiplication de 7 (de 7×1 à 7×10).

```
for k in range(1, 11):  
    print(7, "x", k, "=", 7 * k)
```

5. Écrire une fonction `factorielle(n)` qui renvoie le produit $1 \times 2 \times \dots \times n$. Que renvoie `factorielle(6)` ?

```
def factorielle(n):  
    P = 1  
    for k in range(1, n + 1):  
        P = P * k  
    return P
```

`factorielle(6)` renvoie 720.

6. Écrire une fonction `compte_pairs(L)` qui renvoie le nombre d'entiers pairs de la liste `L`.

```
def compte_pairs(L):  
    c = 0  
    for x in L:  
        if x % 2 == 0:  
            c = c + 1  
    return c
```

Exercice 4. Boucles non bornées

1. Une population de bactéries double toutes les heures. Il y en a 500 au départ. Écrire un programme qui détermine au bout de combien d'heures elle dépasse 1 million.

```
P = 500  
h = 0  
while P <= 1000000:  
    P = 2 * P  
    h = h + 1  
print(h)          # affiche 11
```

En effet $500 \times 2^{10} = 512\,000$ et $500 \times 2^{11} = 1\,024\,000$.

2. Que fait le programme suivant? Que renvoie `mystere(50)` ?

```
def mystere(N):  
    n = 0  
    while n**2 <= N:
```

```
n = n + 1
return n - 1
```

Il renvoie le plus grand entier dont le carré est inférieur ou égal à N , c'est-à-dire la partie entière de $\sqrt{N}$. `mystere(50)` renvoie 7 ($7^2 = 49 \leq 50 < 64$).

3. Un capital de 1 000 € est placé à 3 % par an. Écrire une fonction `doublement()` qui renvoie le nombre d'années nécessaires pour que le capital double.

```
def doublement():
    C = 1000
    n = 0
    while C < 2000:
        C = C * 1.03
        n = n + 1
    return n          # renvoie 24
```

4. Que se passe-t-il si l'on écrit `while P > 0:` dans le programme des bactéries?

La condition reste toujours vraie car P ne fait qu'augmenter : la boucle ne s'arrête jamais (boucle infinie).

5. Écrire une fonction `premier_multiple(a, M)` qui renvoie le plus petit multiple de a strictement supérieur à M , à l'aide d'une boucle `while`.

```
def premier_multiple(a, M):
    m = a
    while m <= M:
        m = m + a
    return m
```

Exercice 5. Fonctions et mathématiques

1. Écrire une fonction $f(x)$ qui renvoie l'image de x par $f(x) = x^2 - 4x + 1$. Que renvoie $f(3)$?

```
def f(x):  
    return x**2 - 4*x + 1
```

$f(3)$ renvoie $9 - 12 + 1 = -2$.

2. Écrire une fonction $\text{distance}(x_A, y_A, x_B, y_B)$ qui renvoie la distance entre deux points d'un repère orthonormé.

```
from math import sqrt  
def distance(xA, yA, xB, yB):  
    return sqrt((xB - xA)**2 + (yB - yA)**2)
```

3. Écrire une fonction $\text{milieu}(x_A, y_A, x_B, y_B)$ qui renvoie les coordonnées du milieu d'un segment.

```
def milieu(xA, yA, xB, yB):  
    return (xA + xB) / 2, (yA + yB) / 2
```

4. Écrire une fonction $\text{colineaires}(x, y, x_p, y_p)$ qui renvoie True si les vecteurs $\vec{u} \begin{pmatrix} x \\ y \end{pmatrix}$ et $\vec{v} \begin{pmatrix} x' \\ y' \end{pmatrix}$ sont colinéaires.

```
def colineaires(x, y, xp, yp):  
    return x * yp - y * xp == 0
```

5. Écrire une fonction $\text{pente}(x_A, y_A, x_B, y_B)$ qui renvoie le coefficient directeur de la droite (AB) , ou None si la droite est parallèle à l'axe des ordonnées.

```
def pente(xA, yA, xB, yB):  
    if xA == xB:  
        return None  
    return (yB - yA) / (xB - xA)
```

6. Écrire une fonction $\text{diviseurs}(n)$ qui renvoie la liste des diviseurs positifs de n , puis une fonction $\text{est_premier}(n)$.

```
def diviseurs(n):  
    L = []  
    for d in range(1, n + 1):
```

```
        if n % d == 0:
            L.append(d)
    return L

def est_premier(n):
    return len(diviseurs(n)) == 2
```

7. Écrire une fonction `tableau_valeurs(a, b, pas)` qui affiche les valeurs de $f(x)$ pour x allant de a à b avec le pas donné.

```
def tableau_valeurs(a, b, pas):
    x = a
    while x <= b:
        print(x, f(x))
        x = x + pas
```