

Programmation en Python

Corrigé des exercices - 2de

Corrigé détaillé des exercices du chapitre. Vérifie tes réponses étape par étape et corrige-toi.

1 Reconnaître les types ★★★

- a. 12 : **int** (entier).
- b. 3.0 : **float** (il y a une partie décimale écrite, même nulle).
- c. "12" : **str** (guillemets : c'est du texte).
- d. True : **bool** (booléen, sans guillemets).
- e. 7 / 2 : **float** (une division « / » donne toujours un float : 3.5).
- f. 4 ** 2 : **int** ($4^2 = 16$, entier).
- g. "Vrai" : **str** (guillemets : texte, pas un booléen).
- h. 10 / 5 : **float** (la division donne 2.0).

2 Échange de deux variables ★★★

Suivi des variables :

Ligne	a	b
1	4	—
2	4	9
3	13	9
4	13	4
5	9	4

Le programme affiche :

9 4

Il **échange** les valeurs de a et de b sans variable supplémentaire.

3 Quatre affichages ★★★

x vaut 5

x vaut 5

```
x * 2
10
```

Les guillemets signalent du texte à afficher tel quel : la ligne 4 affiche donc les caractères « x * 2 », alors que la ligne 5 calcule $5 \times 2 = 10$. Les lignes 2 et 3 donnent le même texte : la première utilise une virgule (Python insère un espace), la seconde colle des chaînes avec +, ce qui oblige à convertir x avec str.

4 Prix toutes taxes comprises ★★★

Ajouter 20 % revient à multiplier par 1,2. En Python, le point remplace la virgule.

```
prix_ht = float(input("Prix hors taxe : "))
prix_ttc = prix_ht * 1.2
print("Prix TTC :", prix_ttc)
```

Affichage obtenu :

```
Prix hors taxe : 50
Prix TTC : 60.0
```

Le prix TTC est de 60 €. On convertit l'entrée avec float, car input renvoie du texte.

5 Valeurs de conditions ★★★

- a. $x > 5$ and $x < 10$: **True**
- b. $x \leq 3$ or $x == 7$: **True**
- c. $x \neq 7$: **False**
- d. not ($x > 9$) : **True**
- e. $x < 7$ or $x > 7$: **False**

Justifications : a. $7 > 5$ et $7 < 10$ sont vrais. b. la deuxième partie $7 == 7$ est vraie. c. $7 \neq 7$ est faux. d. $7 > 9$ est faux, donc son contraire est vrai. e. $7 < 7$ et $7 > 7$ sont faux tous les deux.

6 Lire un if else ★★★

Pour 15 : la condition $15 < 18$ est vraie.

```
Age : 15
Mineur
```

```
Merci
```

Pour 18 : $18 < 18$ est faux, on passe au else.

```
Age : 18  
Majeur  
Merci
```

Pour 40 :

```
Age : 40  
Majeur  
Merci
```

La ligne « Merci » n'est pas indentée : elle est affichée dans tous les cas.

7 Appeler une fonction ★★★

$f(4) = 3 \times 4 - 7 = 5$; $f(-2) = 3 \times (-2) - 7 = -13$; $f(0) = 0 - 7 = -7$.

```
5 -13 -7
```

8 Deux boucles for ★★★

```
1  
4  
9  
16  
Top 0  
Top 1  
Top 2
```

`range(1, 5)` donne $i = 1, 2, 3, 4$: quatre tours (5 n'est pas atteint). `range(3)` donne $j = 0, 1, 2$: trois tours.

9 Tarif de piscine ★★★

On utilise `elif` : les cas sont testés dans l'ordre. Si $\text{age} < 12$ est faux, alors $\text{age} \geq 12$ et le second test suffit.

```
age = int(input("Age : "))
if age < 12:
    print("Tarif : 3 euros")
elif age < 18:
    print("Tarif : 4.5 euros")
else:
    print("Tarif : 6.5 euros")
```

Résultats :

âge 11 :

```
Age : 11
Tarif : 3 euros
```

âge 12 :

```
Age : 12
Tarif : 4.5 euros
```

âge 17 :

```
Age : 17
Tarif : 4.5 euros
```

âge 18 :

```
Age : 18
Tarif : 6.5 euros
```

Les valeurs 11, 12, 17 et 18 sont les valeurs « aux bords » : elles vérifient que les inégalités strictes sont bien placées.

10 Pair ou impair ★★★

Un entier est pair si le reste de sa division par 2 est nul : $n \% 2 == 0$.

```
n = int(input("Entier : "))
if n % 2 == 0:
    print(n, "est pair")
else:
```

```
print(n, "est impair")
```

```
Entier : 38  
38 est pair
```

```
Entier : 51  
51 est impair
```

En effet $38 = 2 \times 19 + 0$ et $51 = 2 \times 25 + 1$.

11 Fonctions du rectangle ★★★

```
def perimetre(L, l):  
    return 2 * (L + l)  
  
def aire(L, l):  
    return L * l  
  
print(perimetre(8, 3))  
print(aire(8, 3))
```

Les deux appels affichent :

```
22  
24
```

Vérification : $P = 2 \times (8 + 3) = 22$ cm et $A = 8 \times 3 = 24$ cm². Pour 12,5 et 4 : `aire(12.5, 4)` renvoie 50, soit 50 cm².

12 Chasse aux erreurs ★★★

- Ligne 2 : il faut `==` pour tester une égalité (et non `=`).
- Ligne 2 : il manque les deux-points à la fin du `if`.
- Ligne 3 : le `print` doit être **indenté** dans le bloc du `if`.
- Ligne 4 : il manque les deux-points après `else`.

Programme corrigé :

```
n = int(input("Nombre : "))  
if n == 10:
```

```
print("dix")
else:
    print("autre")
```

Avec 10 :

```
Nombre : 10
dix
```

Avec 3 :

```
Nombre : 3
autre
```

13 Somme des entiers ★★★

1. Le programme calcule $1 + 2 + \dots + 10$:

```
55
```

On retrouve la formule $10 \times 11 \div 2 = 55$.

2. Avec le pas 2, i prend les valeurs 1, 3, 5, 7, 9 :

```
s = 0
for i in range(1, 11, 2):
    s = s + i
print(s)
```

```
25
```

Vérification : $1 + 3 + 5 + 7 + 9 = 25 = 5^2$.

14 Table de multiplication ★★★

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

```
1 x 7 = 7
```

```
2 x 7 = 14
3 x 7 = 21
4 x 7 = 28
5 x 7 = 35
6 x 7 = 42
7 x 7 = 49
8 x 7 = 56
9 x 7 = 63
10 x 7 = 70
```

La boucle fait 10 tours : il y a 10 lignes, la dernière est « 10 x 7 = 70 ». Il faut `range(1, 11)` et non `range(1, 10)`.

15 Placement à 5 % ★★★

Chaque année, le capital est multiplié par 1,05.

```
capital = 1000
annees = 0
while capital < 1200:
    capital = capital * 1.05
    annees = annees + 1
print(annees, capital)
```

```
4 1215.5062500000001
```

Le capital dépasse 1 200 € au bout de **4 ans** (environ 1 215,51 €). Vérification : $1\,000 \times 1,05^3 \approx 1\,157,63$ (trop peu), $1\,000 \times 1,05^4 \approx 1\,215,51$.

16 Vrai ou faux ? ★★★

- a. **Faux.** Ce n'est pas une égalité mais une affectation : on calcule $x + 1$ puis on range le résultat dans x .
- b. **Faux.** `input` renvoie toujours un `str` : il faut écrire `int(input(...))`.
- c. **Faux.** `range(4)` donne 0, 1, 2, 3 (quatre valeurs, en partant de 0).
- d. **Faux.** Dès qu'une condition est vraie, son bloc est exécuté et les suivants sont ignorés.

17 Compte à rebours ★★★

10

8

6

4

2

0

Decollage !

`range(10, -1, -2)` part de 10 et recule de 2 tant qu'on n'a pas atteint -1 : i vaut 10, 8, 6, 4, 2, 0. Pour compter de 10 à 1 de 1 en 1 : `range(10, 0, -1)`.

18 Fonction booléenne ★★★

1. Cette condition est vraie pour $x \in [-1 ; 8[$.
2. -1 est dans l'intervalle (borne fermée), 8 ne l'est pas (borne ouverte), 3,5 y est, -4 n'y est pas :

True False True False

```
def hors_intervalle(x):  
    return x < -1 or x >= 8
```

On inverse chaque inégalité et on remplace and par or.

19 Tarif de parking ★★★

a. 1 h : 1,5 € ; 4 h : $3 + 2 \times 2 = 7$ € ; 10 h : $3 + 2 \times 8 = 19$ €, ce qui dépasse le plafond, donc 15 € ; 20 h : $3 + 2 \times 18 = 39$ €, donc 15 €.

b.

```
def parking(h):  
    if h <= 2:  
        cout = 1.5 * h  
    else:  
        cout = 3 + 2 * (h - 2)  
    if cout > 15:  
        cout = 15  
    return cout
```

```
print(parking(1), parking(4), parking(10), parking(20))
```

Le programme affiche :

```
1.5 7 15 15
```

Pour 10 h le calcul brut donne 19, mais la deuxième condition applique le plafond : 15.

20 Croissance d'une population ★★★

a. On continue tant que la population ne dépasse pas 6 000, donc la condition est *pop* ≤ 6000, et on multiplie par 1,04 à chaque tour.

```
pop = 5000
annees = 0
while pop <= 6000:
    pop = pop * 1.04
    annees = annees + 1
print(annees)
```

b. Affichage :

```
5
```

Vérification : $5\,000 \times 1,04^4 \approx 5849$ (encore inférieur à 6 000) et $5\,000 \times 1,04^5 \approx 6083$ (supérieur). Il faut donc **5 ans**.

21 Premier carré supérieur à 500 ★★★

```
n = 0
while n * n <= 500:
    n = n + 1
print(n)
```

```
23
```

On a $22^2 = 484 \leq 500$ et $23^2 = 529 > 500$: le plus petit entier est **23**. La condition du while est le contraire de ce qu'on cherche ($n^2 \leq 500$).

22 Compter les diviseurs ★★★

1. Pour chaque d de 1 à n , elle teste si $n \% d$ vaut 0, c'est-à-dire si d divise n , et compte ces d .
2. Diviseurs de 12 : 1, 2, 3, 4, 6, 12 (6 diviseurs) ; de 17 : 1 et 17 (2) ; de 36 : 1, 2, 3, 4, 6, 9, 12, 18, 36 (9).

```
6 2 9
```

3. Un entier $n \geq 2$ est premier exactement quand il a **deux** diviseurs : on teste `nb_diviseurs(n) == 2`. Ainsi 17 est premier, 12 et 36 ne le sont pas.

23 Factorielle ★★★

Le produit de 1 à n s'obtient avec k variant de 1 à n inclus, donc `range(1, n + 1)`, et on multiplie l'accumulateur (initialisé à 1, pas à 0) par k .

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

Affichage de `factorielle(0)`, `factorielle(5)` et `factorielle(8)` :

```
1 120 40320
```

Vérification : $5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$ et $8! = 120 \times 6 \times 7 \times 8 = 40\,320$. Pour $n = 0$, la boucle ne fait aucun tour et la fonction renvoie 1, ce qui est la bonne valeur.

24 Remboursement d'un prêt ★★★

- a. Après 1 mois : $2\,000 \times 1,01 - 150 = 2\,020 - 150 = 1\,870$ €. Après 2 mois : $1\,870 \times 1,01 - 150 = 1\,888,70 - 150 = 1\,738,70$ €.
- b. La dette est mise à jour avec `dette = dette * 1,01 - 150` :

```
dette = 2000  
mois = 0  
while dette > 0:  
    dette = dette * 1.01 - 150
```

```
mois = mois + 1
print(mois)
```

15

Il faut **15 mois** pour tout rembourser (le dernier versement est inférieur à 150 €).

25 Fonctions qui s'appellent ★★★

1.

```
25
True
False
```

somme_carres(3, 4) = $3^2 + 4^2 = 9 + 16 = 25$. Pour est_rectangle(3, 4, 5) : $25 == 5^2$ est vrai. Pour (5, 6, 8) : $25 + 36 = 61$ et $8^2 = 64$, donc faux.

2. Elle teste la **réciroque du théorème de Pythagore** avec c comme plus grand côté. Le triangle de côtés 5, 6, 8 n'est **pas** rectangle, car $5^2 + 6^2 = 61 \neq 64 = 8^2$.

26 Indice de masse corporelle ★★★

```
def classe(imc):
    if imc < 18.5:
        return "maigreur"
    elif imc < 25:
        return "normal"
    elif imc < 30:
        return "surpoids"
    else:
        return "obesite"

def imc(masse, taille):
    return masse / taille ** 2

print(classe(imc(70, 1.75)))
print(classe(imc(50, 1.80)))
print(classe(imc(95, 1.70)))
```

Calculs : $70 \div 1,75^2 \approx 22,9$; $50 \div 1,80^2 \approx 15,4$; $95 \div 1,70^2 \approx 32,9$. Affichage :

```
normal  
maigreur  
obesite
```

Comme les return sortent de la fonction, aucun else n'est nécessaire entre les cas, mais le dernier else récupère tout ce qui est ≥ 30 .