

Exercices 1

Algorithmique et programmation en Python

Les bases : variables, types, opérateurs, conditions, boucles, fonctions

7 exercices · niveau ●○○ · durée conseillée : 1 h 30 · sans ordinateur

★ À RETENIR — Ce qu'il faut savoir faire à la fin de la feuille

- Suivre des affectations, échanger deux variables, reconnaître un type, utiliser les conversions et les opérateurs `//`, `%`, `**`, `and`, `or`, `not`.
- Exécuter à la main un `if` / `elif` / `else`, une boucle `for` ou `while`, compléter un tableau d'exécution, écrire une fonction avec `def` et `return`.

■ RAPPEL — Le formulaire du chapitre

- **Affectation, types** : `a = a + 1` calcule d'abord `a + 1`, puis range le résultat dans `a`. Types : entier (`int`), flottant (`float`), booléen (`bool` : `True/False`), chaîne (`str`); conversions `int(...)`, `float(...)`, `str(...)`.
- **Opérateurs** : `a // b` quotient entier, `a % b` reste, `a ** n` puissance; `a / b` renvoie *toujours* un flottant. Comparaisons `==`, `!=`, `<=`, `>=`; connecteurs `and`, `or`, `not`.
- **Boucles, fonctions** : `for i in range(a, b, p)` fait prendre à `i` les valeurs `a`, `a + p`, ... en s'arrêtant **avant** `b`; `while condition` : répète *tant que* la condition est vraie; dans `def f(x) : ... return` renvoie une valeur réutilisable, `print` se contente d'afficher.

Exercice 1 Affectations successives et échange

●○○

- 1.1 Compléter le tableau d'exécution du programme 1 : après chaque ligne, donner `a` et `b`. Que fait ce programme, quelles que soient les valeurs de départ ?

Ligne	1	2	3	4	5
<code>a</code>					
<code>b</code>					

```

1  a = 7
2  b = 3
3  a = a + b
4  b = a - b
5  a = a - b

```

Programme 1 — Que fait-il ?

- 1.2 Pour échanger deux variables, un élève écrit `u = v` puis `v = u`. En partant de `u = 12` et `v = 5`, donner les valeurs obtenues, expliquer l'erreur, puis corriger en utilisant une variable `t`.

Exercice 2 Types et conversions

●○○

- 2.1 Pour chaque expression, donner son **type** (entier, flottant, booléen ou chaîne) et sa valeur.
- a) `7.0` b) `"7"` c) `7 == 7` d) `6 / 3` e) `9 // 2` f) `"a" + "b"`
- 2.2 Donner ce qu'affichent `print(int(3.9))`, `print(float(5))`, `print(str(12) + "3")` et `print(int("8") + 1)`. Pourquoi `"3" + "4"` ne donne-t-il pas 7 ?

Exercice 3 Opérateurs et booléens

●○○

- 3.1 Calculer sans machine.

- a) `17 // 5` b) `17 % 5` c) `2 ** 5` d) `100 // 7`
e) `100 % 7` f) `20 % 4` g) `1 // 3` h) `7 % 7`

3.2 Donner la valeur (True ou False) de chaque expression.

a) $5 == 5.0$

b) $4 <= 4$

c) $\text{not } (2 > 1)$

d) $3 != 3$

e) $(3 > 2) \text{ and } (5 < 1)$

f) $(3 > 2) \text{ or } (5 < 1)$

3.3 Traduire en Python : « n est pair » ; « n est multiple de 3 mais pas de 5 » ; « x appartient à $[2; 7]$ ».

Exercice 4 Boucle bornée : compteur et accumulateur

●○○

4.1 Compléter le tableau d'exécution : pour chaque valeur de i, donner s à la fin du tour de boucle. Qu'affiche le programme ?

i	1	2	3	4	5
s					

```

1 s = 0
2 for i in range(1, 6):
3     s = s + i * i
4 print(s)

```

Programme 1 — Somme de carrés

4.2 Donner, dans l'ordre, les valeurs prises par i dans range(5), range(2, 7), range(0, 10, 3), range(10, 0, -2).

4.3 Que modifier pour obtenir $1^2 + 2^2 + \dots + 10^2$? Combien de tours de boucle effectue-t-on alors ?

Exercice 5 Un organigramme à traduire

●●○

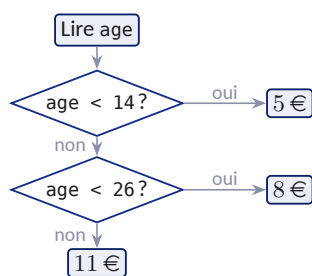


Figure 1 — Prix d'une place de cinéma

Un cinéma fixe le prix d'une place selon l'âge du spectateur (figure 1).

5.1 Donner le prix payé par un spectateur de 12 ans, de 14 ans, de 25 ans, de 26 ans, puis de 40 ans.

5.2 Traduire cet organigramme par une fonction prix(age) utilisant if, elif et else : le prix est rangé dans une variable p, renvoyée à la dernière ligne par return p.

5.3 Un élève remplace le elif par un second if indépendant (le else restant attaché à ce second if). Quel prix obtient alors un spectateur de 12 ans ? Expliquer l'erreur.

Exercice 6 Boucle non bornée

●●○

6.1 Compléter le tableau d'exécution : à chaque tour, donner n et p à la fin du tour (ajouter des colonnes si besoin). Qu'affiche le programme ?

Tour	1	2	3	4	...
n					
p					

```

1 n = 1
2 p = 1
3 while p < 100:
4     n = n + 1
5     p = p * n
6 print(n, p)

```

Programme 1 — Une boucle while

6.2 Pour la valeur finale de n, que vaut $1 \times 2 \times \dots \times n$? Que cherche ce programme ?

6.3 On supprime la ligne 4. Qu'observe-t-on à l'exécution ? Comment appelle-t-on ce phénomène ?

Exercice 7 return ou print ?

●●○

7.1 Que vaut carre(5) + 1 ? Que se passe-t-il avec affiche_carre(5) + 1 ? Expliquer la différence entre return et print.

7.2 Calculer carre(carre(2)), puis carre(3) + carre(4).

```

1 def carre(x):
2     return x * x
3
4 def affiche_carre(x):
5     print(x * x)

```

Programme 1 — Deux fonctions