Plaquette 1 – Variables, boucles, conditions et fonctions

Algorithmique et Python — Seconde

Corrections détaillées

Boîte à outils

Notions et syntaxe utiles.

- **Affectation** : $x = 5$ met la valeur 5 dans la variable x. L'instruction $x = x + 1$ augmente x de 1.
- **Fonction** : définie par *def f(x):*, elle **renvoie** une valeur avec *return*. Sans *return*, elle renvoie *None*.
- *print* **affiche**, *return* **renvoie** : une fonction dont on veut réutiliser le résultat doit faire un *return*.
- **Boucle bornée** : *for i in range(n):* répète n fois, avec i prenant les valeurs $0, 1, \ldots, n-1$. *range(a, b)* parcourt $a, a+1, \ldots, b-1$.
- **Boucle non bornée** : *while condition:* répète **tant que** la condition est vraie.
- **Test** : *if condition: ... else: ...*. Comparaisons : == (égalité), *!=*, <=, >=.
- Opérations : // est le quotient entier, % le reste, ** la puissance. Pour des entiers a et $b > 0$, $q = a \,//\, b$ et $r = a \,\%\, b$ vérifient
$$a = b \times q + r \qquad \text{avec} \quad 0 \leq r < b$$
- **Accumulateur** : une variable initialisée avant la boucle ($s = 0$ pour une somme, $c = 0$ pour un compte) et mise à jour à chaque tour.
- Hasard : *randint(a, b)* renvoie un entier au hasard entre a et b inclus.

Correction de l'exercice 1 – Suivre des affectations

On exécute les instructions **dans l'ordre**, en notant la valeur de chaque variable.

Instruction	x	y
x = 3	3	—
y = 2 * x + 1	3	7
x = y - x	4	7

- Ligne 2 : $y = 2 \times 3 + 1 = 7$.
- Ligne 3 : on calcule d'abord $y - x$ avec les valeurs actuelles, **puis** on range le résultat dans x :
$$x \leftarrow y - x = 7 - 3 = 4$$
- À la fin, $x = 4$ et $y = 7$.

Piège. Dans une affectation, le membre de droite est calculé avec les **anciennes** valeurs.

Réponse. $x = 4$ (et $y = 7$).

Correction de l'exercice 2 – Dérouler une boucle for

range(5) fournit les valeurs 0, 1, 2, 3, 4 (cinq valeurs, en commençant à 0).

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

i	0	1	2	3	4
s après le tour	0	1	3	6	10

— On additionne :

$$s = 0 + 1 + 2 + 3 + 4 = 10$$

— Le programme affiche donc 10.

— Remarque : *range(1, 6)* aurait donné $1 + 2 + 3 + 4 + 5 = 15$.

Réponse. 10.

Correction de l’exercice 3 – print ou return

— *carreA* **affiche** le résultat mais ne le renvoie pas : elle renvoie *None*. L’expression *2 * carreA(3)* provoque une erreur, car on ne peut pas multiplier *None* par 2.

— *carreB* **renvoie** la valeur : *carreB(3)* vaut 9, et *2 * carreB(3)* vaut 18.

— C’est donc *carreB* qu’il faut utiliser.

À retenir. Une fonction que l’on veut réutiliser dans un calcul doit faire un *return*. Le *print* ne sert qu’à montrer quelque chose à l’écran.

Réponse. *carreB*, parce qu’elle renvoie la valeur (*return*) au lieu de seulement l’afficher.

Correction de l’exercice 4 – Fonction affine

— Traduction directe de la formule :

```
def f(x):
    return 3 * x - 5
```

— Pour $x = 4$: $3 \times 4 - 5 = 12 - 5 = 7$, donc *print(f(4))* affiche 7.

— Attention à la syntaxe : les deux-points en fin de ligne *def*, et l’indentation (quatre espaces) de la ligne *return*.

Réponse. *def f(x): return 3 * x - 5* et *print(f(4))* affiche 7.

Correction de l’exercice 5 – Condition : tester la parité

— *n % 2* est le **reste** de la division de *n* par 2 : il vaut 0 si *n* est pair, 1 sinon.

— Il faut donc compléter par 0 : la condition s’écrit *if n % 2 == 0:*.

— Pour $n = 7$: $7 = 2 \times 3 + 1$, le reste vaut 1, différent de 0 : la fonction renvoie "*impair*".

Piège. == teste l’égalité, = affecte une valeur. Écrire *if n % 2 = 0* provoque une erreur de syntaxe.

Réponse. Compléter par 0 ; *parite(7)* renvoie "*impair*".

Correction de l’exercice 6 – Boucle while : seuil à dépasser

On multiplie *u* par 1,2 jusqu’à dépasser 200 : on déroule les tours.

Tour	1	2	3	4
u	120	144	172,8	207,36

— Après le tour 3, $u = 172{,}8 < 200$: la boucle continue.

— Après le tour 4, $u \approx 207{,}36 \geq 200$: la condition devient fausse, la boucle s’arrête.

— La variable *n* a été augmentée quatre fois : le programme affiche 4. **Contrôle** :

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s’expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

$$100 \times 1{,}2^3 = 172{,}8 < 200 \qquad 100 \times 1{,}2^4 = 207{,}36 \geq 200$$

Méthode. Pour une boucle *while*, on remplit un tableau tour par tour et on teste la condition **avant** chaque nouveau tour.

Réponse. Le programme affiche 4 ($u \approx 207{,}36$).

Correction de l'exercice 7 – Somme des entiers de 1 à n

On utilise un **accumulateur** s, initialisé à 0, auquel on ajoute successivement chaque entier de 1 à n. Comme *range(1, n + 1)* parcourt les entiers de 1 à n inclus :

```
def somme(n):
    s = 0
    for k in range(1, n + 1):
        s = s + k
    return s
```

Pour $n = 10$, s prend les valeurs 1, 3, 6, 10, 15, 21, 28, 36, 45 puis 55 : *somme(10)* renvoie 55. Pour $n = 100$, la machine fait les 100 additions et renvoie 5 050.

Contrôle par l'astuce de Gauss : en associant le premier et le dernier terme, $1 + 100 = 2 + 99 = \cdots = 101$, on forme 50 paires de somme 101, d'où

$$1 + 2 + \cdots + 100 = 50 \times 101 = 5\,050$$

Réponse. *somme(10)* renvoie 55 et *somme(100)* renvoie 5 050.

Correction de l'exercice 8 – Compter avec une condition

On parcourt les entiers de -10 à 10 avec *range(-10, 11)*, la borne de droite étant exclue, et l'on augmente un compteur chaque fois que la condition est vraie :

```
def compte():
    c = 0
    for x in range(-10, 11):
        if 2 * x + 3 > 7:
            c = c + 1
    return c
```

La condition est vraie pour $x = 3, 4, \ldots, 10$, soit 8 entiers : la fonction renvoie 8.

Vérification. On résout l'inéquation :

$$2x + 3 > 7 \Leftrightarrow 2x > 4 \Leftrightarrow x > 2$$

Les entiers de -10 à 10 strictement supérieurs à 2 sont bien $3, 4, \ldots, 10$, au nombre de $10 - 3 + 1 = 8$. L'entier 2 ne convient pas, puisque $2 \times 2 + 3 = 7$ n'est pas strictement supérieur à 7.

Réponse. La fonction renvoie 8 (les entiers de 3 à 10).

Correction de l'exercice 9 – Simuler un lancer de dé

— *randint(1, 6)* simule un dé à six faces : il faut compléter par 6.
— La fonction doit renvoyer une **fréquence**, c'est-à-dire l'effectif divisé par le nombre de lancers : on complète par n, soit *return succes / n*.
— Pour n grand, la fréquence observée s'approche de la probabilité théorique

$$P(\text{obtenir 6}) = \frac{1}{6} \approx 0{,}167$$

— Chaque exécution donne un résultat différent : c'est la fluctuation d'échantillonnage.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Réponse. *randint(1, 6)* et *return succes / n* ; on doit approcher $\frac{1}{6} \approx 0{,}167$.

Correction de l’exercice 10 – Tableau de valeurs d’une fonction

On parcourt les entiers de 0 à 5 inclus avec *range(0, 6)*.

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

for x in range(0, 6):
    print(x, f(x))
```

— Les valeurs calculées :

x	**0**	**1**	**2**	**3**	**4**	**5**
$f(x)$	0	-2	-2	0	4	10

— Les images négatives correspondent à $x = 1$ et $x = 2$.
— On peut le retrouver par le calcul : $f(x) = x(x-3)$, produit négatif exactement lorsque $0 < x < 3$.

Piège. *range(0, 6)* s’arrête à 5 : la borne de droite est **exclue**.

Réponse. f vaut 0, -2, -2, 0, 4, 10 ; images négatives pour $x = 1$ et $x = 2$.

Correction de l’exercice 11 – Moyenne de trois notes

La moyenne de trois nombres est leur somme divisée par 3. Les parenthèses sont indispensables : sans elles, Python ne diviserait que c par 3, car la division est prioritaire sur l’addition.

```
def moyenne(a, b, c):
    return (a + b + c) / 3
```

Pour les notes 12, 9 et 15 :

$$\frac{12 + 9 + 15}{3} = \frac{36}{3} = 12$$

Pour 10, 10 et 11, on obtient $\frac{31}{3} \approx 10{,}33$: Python affiche 10.333333333333334, une valeur approchée. Avec l’écriture fautive *a + b + c / 3*, le premier calcul donnerait $12 + 9 + 5 = 26$, un résultat absurde pour une moyenne sur 20.

Réponse. *return (a + b + c) / 3* ; on obtient 12 et environ 10,33.

Correction de l’exercice 12 – Distance entre deux points

La racine carrée s’obtient avec *sqrt*, qu’il faut importer du module *math*.

```
from math import sqrt

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

Pour $A(1\,;2)$ et $B(4\,;6)$, les écarts de coordonnées valent $4 - 1 = 3$ et $6 - 2 = 4$:

$$AB = \sqrt{3^2 + 4^2} = \sqrt{9 + 16} = \sqrt{25} = 5$$

La fonction renvoie 5.0. **Contrôle** : l’ordre des points ne change rien, *distance(4, 6, 1, 2)* renvoie aussi 5.0, car $(-3)^2 = 3^2$ et $(-4)^2 = 4^2$.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s’expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Réponse. *return sqrt((xB - xA)**2 + (yB - yA)**2)* ; $AB = 5$.

Correction de l'exercice 13 – Un point est-il sur une droite ?

Le point $(x\,;y)$ est sur d exactement quand ses coordonnées vérifient l'équation, c'est-à-dire quand y est égal à $2x + 1$. Le test d'égalité == renvoie directement *True* ou *False*.

```
def surDroite(x, y):
    return y == 2 * x + 1
```

— Pour $(3\,;7)$: $2 \times 3 + 1 = 7$, égal à y, donc la fonction renvoie *True*.

— Pour $(-1\,;2)$: $2 \times (-1) + 1 = -1$, différent de 2, donc elle renvoie *False*.

$$2 \times 3 + 1 = 7 \qquad 2 \times (-1) + 1 = -1 \neq 2$$

Attention avec des décimaux : à cause des arrondis, Python trouve *0.1 + 0.2 == 0.3* faux. Avec des coordonnées décimales, on teste plutôt si l'écart *abs(y - (2 * x + 1))* est très petit.

Réponse. *return y == 2 * x + 1* ; *True* pour $(3\,;7)$ et *False* pour $(-1\,;2)$.

Correction de l'exercice 14 – Compter les multiples de 7

On parcourt les entiers de 1 à 100 avec *range(1, 101)* et l'on compte ceux dont le reste dans la division par 7 est nul.

```
c = 0
for n in range(1, 101):
    if n % 7 == 0:
        c = c + 1
print(c)
```

Les multiples de 7 sont 7, 14, 21, …, jusqu'au plus grand qui ne dépasse pas 100. Comme

$$100 = 7 \times 14 + 2$$

ce plus grand multiple est $7 \times 14 = 98$, et il y en a 14 : le programme affiche 14.
Contrôle : *100 // 7* vaut aussi 14. Avec *range(1, 100)*, le résultat serait le même, car 100 n'est pas multiple de 7 ; mais pour compter les multiples de 5, oublier 100 fausserait le résultat.

Réponse. 14 multiples (de 7 à 98), car $100 = 7 \times 14 + 2$.

Correction de l'exercice 15 – Synthèse : prix avec remise

a) Une remise de 15 % correspond à un coefficient multiplicateur de $1 - 0{,}15 = 0{,}85$.

```
def prixFinal(p):
    if p > 50:
        return p * 0.85
    else:
        return p
```

b) *prixFinal(40)* : $40 \leq 50$, aucune remise, la fonction renvoie 40 €.

— *prixFinal(80)* : $80 > 50$, donc

$$80 \times 0{,}85 = 68 \text{ €}$$

— Contrôle : la remise vaut $80 \times 0{,}15 = 12$ € et $80 - 12 = 68$ €. Cohérent.

Réponse. b) 40 € et 68 €.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).