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.
- **Liste** : *L = [3, 7, 2]*, on accède au premier élément par *L[0]*, on ajoute avec *L.append(v)*, on obtient la longueur avec *len(L)*.
- 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 $y - x = 7 - 3 = 4$, **puis** on range le résultat dans x.
- 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).

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

- On additionne : $0 + 1 + 2 + 3 + 4 = 10$.
- 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.

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 éléments d’une liste

On utilise un **accumulateur** initialisé à 0, puis une boucle qui parcourt la liste.

```
def somme(L):
    s = 0
```

```
    for v in L:
        s = s + v
    return s
```

— Déroulé avec *L = [3, 7, 2, 8]* : *s* prend les valeurs 0, puis 3, 10, 12, 20.
— La fonction renvoie 20, ce qui est bien 3 + 7 + 2 + 8.
— Variante avec les indices : *for i in range(len(L)): s = s + L[i].*

Réponse. *somme([3, 7, 2, 8])* renvoie 20.

Correction de l’exercice 8 – Compter avec une condition

On combine un compteur et un test dans la boucle.

```
def nbPositifs(L):
    compteur = 0
    for v in L:
        if v > 0:
            compteur = compteur + 1
    return compteur
```

— Avec *L = [-3, 5, 0, 8, -1, 2]* : seuls 5, 8 et 2 vérifient *v > 0* (0 n’est pas strictement positif).
— La fonction renvoie donc 3.

Réponse. 3.

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.

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 – 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 $80 \times 0{,}85 = 68$ €.

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

Réponse. b) 40 € et 68 €.