Corrigé et barème – Algorithmique et programmation

Variables, instructions conditionnelles, boucles et fonctions : les bases de l'algorithmique au programme de 2nde générale

Durée : 90 min – Calculatrice interdite – Sur 20 points · Mathématiques 2nde

Exercice 1 – Exercice 1 — Suivre un script *(4 pts)*

a) Après `x = 4` : x = 4. Après `y = 9` : x = 4, y = 9. Après `x = y - x` : x = 9 − 4 = 5, y = 9. Après `y = y + x` : x = 5, y = 9 + 5 = 14. (1)

b) Le script affiche **5 14**. (1)

c) Avec l'ordre inversé : `y = y + x` donne y = 9 + 4 = 13, puis `x = y - x` donne x = 13 − 4 = 9. L'affichage est **9 13**. (1)

d) Chaque affectation utilise les valeurs courantes des variables ; modifier y avant de calculer x change la donnée utilisée dans le second calcul. (1)

Le point clé — Une affectation écrase la valeur précédente : on suit toujours les instructions dans l'ordre, ligne par ligne.

Exercice 2 – Exercice 2 — Tarif d'un parking *(4 pts)*

a)

```
def prix(h):
    if h > 8:
        return 14
    else:
        return 2.4 + 1.6 * (h - 1)
```

(1)

b) prix(1) = 2,40 € ; prix(5) = 2,4 + 1,6 × 4 = 2,4 + 6,4 = **8,80 €** ; prix(9) = **14 €** (forfait). (1)

c) Pour 8 heures : 2,4 + 1,6 × 7 = 2,4 + 11,2 = 13,60 €, soit moins que le forfait. Le forfait n'est avantageux qu'à partir de 9 heures de stationnement. (1)

d) Le test $h >= 1$ est vrai pour toutes les valeurs autorisées : il est exécuté en premier et le forfait n'est jamais appliqué. Il faut tester d'abord le cas le plus restrictif, $h > 8$. (1)

Erreur fréquente — Dans un enchaînement de tests, la condition la plus restrictive doit toujours être écrite en premier.

Exercice 3 – Exercice 3 — Deux boucles *(4 pts)*

a)

```
S = 0
for k in range(1, 100, 2):
    S = S + k
print(S)
```

Le pas 2 permet de ne parcourir que les impairs 1, 3, 5, …, 99. (1)

b) Les termes sont 1, 3, …, 99, soit 50 nombres. Leur somme vaut $50 \times (1 + 99) \div 2 = 50 \times 50 = \mathbf{2\,500}$. (1)

c)

```
S = 0
n = 0
while S <= 1000:
    n = n + 1
    S = S + n
print(n)
```

(1)

d) Pour n = 44 : $44 \times 45 \div 2 = 990$, qui n'est pas supérieur à 1 000. Pour n = 45 : $45 \times 46 \div 2 = 1\,035 > 1\,000$. Donc $\mathbf{n = 45}$. (1)

Repère — Nombre de répétitions connu à l'avance : boucle for. Condition d'arrêt à atteindre : boucle while.

Exercice 4 – Exercice 4 — Fonction Python et images *(4 pts)*

a)

```
def g(x):
    return 2 * x ** 2 - 7 * x + 3
```

(1)

b) $g(0) = 3$; $g(3) = 2 \times 9 - 21 + 3 = 18 - 21 + 3 = \mathbf{0}$; $g(-2) = 2 \times 4 + 14 + 3 = \mathbf{25}$. (1)

c)

```
for x in range(-3, 7):
    if g(x) <= 0:
        print(x)
```

Images : $g(1) = -2$, $g(2) = -3$, $g(3) = 0$, les autres étant positives. Le script affiche **1, 2 et 3**. (1)

d) Avec `print`, la fonction affiche la valeur mais ne renvoie rien ($None$). La comparaison $g(x) <= 0$ porte alors sur $None$ et provoque une erreur de type. (1)

À retenir — Une fonction dont on veut réutiliser le résultat dans un test ou un calcul doit se terminer par return.

Exercice 5 – Exercice 5 — Une roue de loterie *(4 pts)*

a)

```
from random import randint

def gagne():
    return randint(1, 12) <= 3
```

(1)

b)

```
def frequence(n):
    c = 0
    for i in range(n):
        if gagne():
            c = c + 1
    return c / n
```

(1)

c) Les 12 secteurs étant identiques, la situation est équiprobable : la probabilité de gagner vaut $3/12 = \mathbf{0{,}25}$. Pour 10 000 tours, on attend une fréquence proche de 0,25. (1)

d) Une fréquence observée fluctue autour de la probabilité : les deux valeurs encadrent 0,25 avec des écarts d'environ 0,012 et 0,011, tout à fait compatibles avec le hasard. Une simulation ne prouve jamais qu'une roue est truquée. (1)

Attention — Fréquence et probabilité ne se confondent pas : la première s'observe et fluctue, la seconde se calcule.