

Devoir d'entraînement

Première spécialité NSI

Bases de la programmation en Python : variables, types, conditions, boucles

Durée : 1 h

Calculatrice et ordinateur interdits

Nom : _____

Prénom : _____

Classe : _____

Consignes

- Les exercices 1 et 2 se font **directement sur le sujet**, les exercices 3 et 4 sur copie.
- Le code écrit doit être correctement **indenté** : l'indentation fait partie du programme.
- Les exercices sont indépendants et peuvent être traités dans l'ordre de votre choix.
- Toute trace de recherche pertinente sera valorisée.

Barème

- Exercice 1 : .../ 5
- Exercice 2 : .../ 5
- Exercice 3 : .../ 4
- Exercice 4 : .../ 6

Remarque

Rappel : pour deux entiers `a` et `b`, `a // b` donne le quotient entier et `a % b` le reste de la division euclidienne de `a` par `b`.

Exercice 1 – QCM (5 points)

Pour chaque question, une seule réponse est correcte. Reporter la lettre choisie dans le tableau en fin d'exercice. Une bonne réponse rapporte 0,5 point, une réponse fausse ou une absence de réponse n'enlève aucun point.

1. Quel est le type de la valeur de l'expression `3 * 2.0` ?
a) `float` b) `int` c) `str` d) `bool`
2. Que vaut l'expression `int("12") + 3` ?
a) `"123"` b) `15` c) `"15"` d) une erreur
3. Quelles valeurs prend la variable `i` dans la boucle `for i in range(5):` ?
a) 1, 2, 3, 4, 5 b) 0, 1, 2, 3, 4, 5 c) 0, 1, 2, 3, 4 d) 1, 2, 3, 4
4. Qu'affiche le programme suivant ?

```
1  x = 5
2  x = x * 2
3  x = x - 3
4  print(x)
```

- a) `10` b) `7` c) `5` d) `13`

5. Qu'affiche l'instruction `print(20 % 6)` ?

- a) 3 b) 3.33 c) 2 d) 0

6. Combien de fois le programme suivant affiche-t-il `NSI` ?

```
1 i = 1
2 while i < 10:
3     print("NSI")
4     i = i + 2
```

- a) 4 b) 9 c) 10 d) 5

7. Pour une variable `x` de type `int`, quelle condition est équivalente à `not (x > 3)` ?

- a) `x < 3` b) `x <= 3` c) `x >= 3` d) `x != 3`

8. Que vaut l'expression `"ab" * 3` ?

- a) `"ababab"` b) `"ab3"` c) `"aaabbbb"` d) une erreur

9. Quelle boucle, suivie de l'instruction `print(i)` dans son corps, affiche les nombres 1, 2 et 3 (un par ligne) ?

- a) `for i in range(1, 3):` b) `for i in range(3):`
 c) `for i in range(4):` d) `for i in range(1, 4):`

10. On exécute `n = int(input("Nombre ? "))` et l'utilisateur tape `3.5`. Que se passe-t-il ?

- a) `n` vaut 3 b) `n` vaut 4 c) `n` vaut 3.5 d) une erreur

Question	1	2	3	4	5	6	7	8	9	10
Réponse										

Exercice 2 – Suivre l'évolution des variables (5 points)

Partie A

On considère le programme suivant :

```
1 x = 3
2 for i in range(4):
3     x = 2 * x - 1
4 print(x)
```

1. Compléter le tableau ci-dessous avec les valeurs de `i` et de `x` à la fin de chaque tour de boucle.

Tour	i	x
1		
2		
3		
4		

2. Qu'affiche ce programme ?
3. On remplace `range(4)` par `range(2, 10)`. Combien de fois la ligne 3 est-elle alors exécutée ? Justifier.

Partie B

On considère le programme suivant :

```

1  somme = 0
2  n = 1
3  while somme < 20:
4      somme = somme + n
5      n = n + 2
6  print(n, somme)

```

4. Compléter le tableau ci-dessous avec les valeurs de `somme` et de `n` à la fin de chaque tour de boucle (toutes les lignes ne seront pas forcément utilisées).

	somme	n
Avant la boucle	0	1
Tour 1		
Tour 2		
Tour 3		
Tour 4		
Tour 5		
Tour 6		
Tour 7		

5. Qu'affiche ce programme ?
6. On supprime la ligne 5 (`n = n + 2`). La boucle s'arrête-t-elle encore ? Si oui, qu'affiche alors le programme ? Justifier.

Exercice 3 – Trouver et corriger les erreurs (4 points)

Le programme suivant devrait demander 10 nombres entiers à l'utilisateur, puis afficher, **une seule fois** à la fin, combien d'entre eux sont pairs. Il contient **quatre erreurs**.

```

1  compteur == 0
2  for i in range(10):
3      nombre = int(input("Nombre : "))
4      if nombre % 2 = 0:
5          compteur = compteur + 1
6      print("Nombres pairs : " + compteur)

```

1. Pour chacune des quatre erreurs, donner le numéro de la ligne concernée, expliquer le problème en une phrase, et proposer une correction.
2. Parmi ces quatre erreurs, l'une ne provoque aucun message d'erreur de Python (une fois les trois autres corrigées), mais le programme ne fait pas ce qui est demandé. Laquelle ? Que verrait-on alors à l'écran ?

Exercice 4 – Écrire des programmes (6 points)

Les trois questions sont indépendantes. Pour chacune, écrire le programme Python demandé.

1. (2 points) Écrire un programme qui demande un entier n positif, puis affiche un compte à rebours de n jusqu'à 1 (un nombre par ligne), suivi du message « Décollage! ». Exemple : si l'utilisateur tape **3**, le programme affiche **3, 2, 1**, puis **Décollage !**.
2. (2 points) Écrire un programme qui demande un mot, puis une lettre, et affiche le mot dans lequel chaque apparition de cette lettre est remplacée par une étoile *****. Exemple : avec le mot **banane** et la lettre **a**, le programme affiche **b*n*ne**.
3. (2 points) On place 1000 euros sur un compte rémunéré. Chaque année, le capital augmente d'un certain pourcentage, appelé taux. Écrire un programme qui demande le taux (par exemple **5** pour 5%), puis affiche le nombre d'années nécessaires pour que le capital atteigne ou dépasse 2000 euros.
*Rappel : augmenter une quantité **c** de **t** % revient à la multiplier par $1 + t / 100$.*
 Exemple : avec un taux de **5**, le programme affiche **15**.

Correction de l'exercice 1

Question	1	2	3	4	5	6	7	8	9	10
Réponse	a	b	c	b	c	d	b	a	d	d

1. Une opération entre un **int** et un **float** donne un **float** : **3 * 2.0** vaut **6.0**.
2. **int("12")** convertit la chaîne en l'entier 12, auquel on ajoute 3.
3. **range(5)** commence à 0 et s'arrête strictement avant 5 : il y a bien 5 valeurs.
4. **x** vaut successivement 5, 10, puis 7.
5. $20 = 6 \times 3 + 2$: le reste vaut 2.
6. **i** prend les valeurs 1, 3, 5, 7, 9 (5 affichages), puis 11 : la condition **i < 10** devient fausse.
7. Le contraire de « strictement supérieur à 3 » est « inférieur ou égal à 3 ».
8. ***** entre une chaîne et un entier répète la chaîne.

9. `range(1, 4)` donne 1, 2, 3 (la borne 4 est exclue).
10. `"3.5"` n'est pas l'écriture d'un entier : `int("3.5")` provoque une erreur (`ValueError`). Il faudrait utiliser `float`.

Correction de l'exercice 2

Partie A

1.

Tour	i	x
1	0	5
2	1	9
3	2	17
4	3	33

2. Le programme affiche `33`.
3. `range(2, 10)` produit les entiers de 2 à 9, soit $10 - 2 = 8$ valeurs : la ligne 3 est exécutée 8 fois.

Partie B

4.

	somme	n
Avant la boucle	0	1
Tour 1	1	3
Tour 2	4	5
Tour 3	9	7
Tour 4	16	9
Tour 5	25	11

Après le tour 5, `somme` vaut 25 : la condition `somme < 20` est fausse, la boucle s'arrête. Les lignes 6 et 7 restent vides.

5. Le programme affiche `11 25`.
6. Oui : `n` reste égal à 1, mais `somme` augmente toujours de 1 à chaque tour. Elle finit donc par atteindre 20, et la condition `somme < 20` devient fausse (après 20 tours). Le programme affiche `1 20`.

Remarque

Pour aller plus loin avec la classe : en partie B, `somme` prend les valeurs 1, 4, 9, 16, 25, qui sont les carrés des entiers. La somme des k premiers nombres impairs vaut k^2 .

Correction de l'exercice 3

1. Les quatre erreurs :
 - **Ligne 1** : `==` est un test d'égalité, pas une affectation ; `compteur` n'existe pas encore, d'où une erreur. Correction : `compteur = 0`
 - **Ligne 4** : dans une condition, on teste l'égalité avec `==`, pas avec `=`. Correction : `if nombre % 2 == 0:`
 - **Ligne 6** : `+` ne peut pas coller une chaîne et un entier. Correction :

```
print("Nombres pairs :", compteur) (ou str(compteur)).
```

- **Ligne 6** : cette ligne est indentée, donc exécutée à chaque tour de boucle. Correction : la désindenter pour la placer après la boucle.

Programme corrigé :

```
1  compteur = 0
2  for i in range(10):
3      nombre = int(input("Nombre : "))
4      if nombre % 2 == 0:
5          compteur = compteur + 1
6  print("Nombres pairs :", compteur)
```

2. C'est l'erreur d'indentation de la ligne 6 : c'est une erreur de **logique**, que Python ne signale pas. Le message « Nombres pairs : ... » s'afficherait 10 fois, après chaque saisie, avec le nombre de pairs trouvés jusque-là, au lieu d'une seule fois à la fin.

Correction de l'exercice 4

```
1. n = int(input("n = "))
2  for i in range(n, 0, -1):
3      print(i)
4  print("Décollage !")
```

Accepter aussi une boucle `while` :

```
1  n = int(input("n = "))
2  while n > 0:
3      print(n)
4      n = n - 1
5  print("Décollage !")
```

Points attendus : conversion avec `int`, bornes correctes (on affiche n et 1, pas 0), message affiché une seule fois après la boucle.

```
2. mot = input("Mot : ")
2  lettre_cachee = input("Lettre : ")
3  resultat = ""
4  for lettre in mot:
5      if lettre == lettre_cachee:
6          resultat = resultat + "*"
7      else:
8          resultat = resultat + lettre
9  print(resultat)
```

Points attendus : chaîne résultat initialisée à `""` avant la boucle, construite par concaténation, affichée après la boucle.

```
3. taux = float(input("Taux : "))
2  capital = 1000
3  annees = 0
4  while capital < 2000:
5      capital = capital * (1 + taux / 100)
```

```
6     annees = annees + 1
7     print(annees)
```

Points attendus : boucle `while` (on ne connaît pas le nombre d'années à l'avance), condition `capital < 2000` (on s'arrête dès que 2000 est atteint ou dépassé), compteur d'années mis à jour à chaque tour. Avec un taux de 5, on obtient bien 15 années ($1000 \times 1,05^{14} \approx 1980$ et $1000 \times 1,05^{15} \approx 2079$).