Exercices en Python : Utilisation de la boucle `for`

Pour Débutants

Introduction

Ces exercices sont conçus pour aider les débutants à pratiquer l'utilisation de la boucle `for` en Python. Chacun des exercices couvre une application de base, allant de l'itération simple à des applications un peu plus poussées de la boucle `for`.

Exercices boucle For – Niveau 1

1. **Exercice 1**
   Écrivez un programme qui affiche les nombres de 1 à 10 en utilisant une boucle `for`.

2. **Exercice 2**
   Écrivez un programme qui affiche tous les nombres pairs de 1 à 20.

3. **Exercice 3**
   Créez un programme qui affiche les carrés des nombres de 1 à 10.

4. **Exercice 4**
   Écrivez un programme qui affiche les éléments d'une liste de fruits `['pomme', 'banane', 'orange', 'raisin']`.

5. **Exercice 5**
   Écrivez un programme qui calcule et affiche la somme des nombres de 1 à 100.

6. **Exercice 6**
   Créez un programme qui affiche les lettres de l'alphabet en utilisant une boucle `for`.

7. **Exercice 7**
   Écrivez un programme qui prend une liste de nombres et affiche chaque nombre multiplié par 2.

8. **Exercice 8**
   Écrivez un programme qui affiche tous les éléments d'une liste de noms en commençant par la première lettre de chaque nom en majuscule.

9. **Exercice 9**
   Créez un programme qui affiche les 10 premiers multiples de 3.

10. **Exercice 10**
    Écrivez un programme qui affiche les nombres de 10 à 1 dans l'ordre décroissant.

11. **Exercice 11**
    Écrivez un programme qui affiche la table de multiplication de 7 (de 1 à 10).

12. **Exercice 12**
    Créez un programme qui demande à l'utilisateur un nombre entier, puis affiche les `n` premiers nombres impairs, où `n` est l'entier donné.

13. **Exercice 13**
    Écrivez un programme qui prend une phrase et affiche chaque mot de cette phrase séparément.

14. **Exercice 14**
    Créez un programme qui demande à l'utilisateur une chaîne de caractères et qui affiche chaque caractère de la chaîne individuellement.

15. **Exercice 15**
    Écrivez un programme qui utilise une boucle `for` pour compter le nombre de voyelles dans une phrase donnée.

16. **Exercice 16**
    Créez un programme qui utilise une boucle `for` pour calculer la factorielle d'un nombre donné par l'utilisateur.

17. **Exercice 17**
    Écrivez un programme qui prend une liste de notes (entiers) et calcule la moyenne de ces notes.

18. **Exercice 18**
    Créez un programme qui demande à l'utilisateur un nombre et affiche les diviseurs de ce nombre.

19. **Exercice 19**
    Écrivez un programme qui affiche les nombres de 1 à 50, mais pour les multiples de 3 affiche "Fizz", pour les multiples de 5 affiche "Buzz", et pour les multiples de 3 et 5 affiche "FizzBuzz".

20. **Exercice 20**
    Créez un programme qui affiche les nombres de la suite de Fibonacci jusqu'au 10^{e} terme.

Exercices boucle For – Niveau 2

1. **Exercice 21**
   Écrivez un programme qui prend une liste de mots et affiche le mot le plus long de cette liste. Utilisez une boucle `for` pour parcourir la liste.

2. **Exercice 22**
   Écrivez un programme qui affiche les nombres premiers de 1 à 100. Utilisez une boucle `for` imbriquée pour vérifier si chaque nombre est divisible uniquement par 1 et par lui-même.

3. **Exercice 23**
   Créez un programme qui prend un mot et vérifie s'il est un palindrome (mot qui se lit de la même manière à l'envers). Utilisez une boucle `for` pour comparer les caractères.

4. **Exercice 24**
   Écrivez un programme qui calcule le produit scalaire de deux listes d'entiers de même longueur. Par exemple, si `A = [1, 2, 3]` et `B = [4, 5, 6]`, le produit scalaire est `1*4 + 2*5 + 3*6 = 32`.

5. **Exercice 25**
   Créez un programme qui génère une pyramide de nombres. Par exemple, si l'utilisateur entre `4`, le programme doit afficher :

   ```
   1
   22
   333
   4444
   ```

6. **Exercice 26**
   Écrivez un programme qui utilise une boucle `for` pour trier une liste d'entiers en utilisant l'algorithme de tri par sélection. Ne pas utiliser de fonctions de tri intégrées.

7. **Exercice 27**
   Créez un programme qui prend une liste de phrases et affiche les mots de chaque phrase qui contiennent exactement 5 lettres. Utilisez une boucle `for` pour parcourir chaque mot.

8. **Exercice 28**
   Écrivez un programme qui utilise une boucle `for` pour compter le nombre de fois où chaque lettre apparaît dans une phrase donnée. Affichez le résultat sous forme de dictionnaire où chaque lettre est une clé.

9. **Exercice 29**
   Créez un programme qui prend une liste de listes (par exemple `[[1, 2], [3, 4], [5, 6]]`) et calcule la somme de tous les éléments. Utilisez des boucles `for` imbriquées pour parcourir la liste principale et les sous-listes.

10. **Exercice 30**
    Écrivez un programme qui génère les `n` premiers termes de la séquence de Fibonacci et les stocke dans une liste. Utilisez une boucle `for` pour générer les termes et afficher la liste finale.

Exercices – Niveau 3

1. **Exercice 31**
   Créez un programme qui prend une liste de mots et détermine, pour chaque mot, si celui-ci est un palindrome. Le programme doit afficher les mots qui sont des palindromes.

2. **Exercice 32**
   Écrivez un programme qui calcule la somme de tous les nombres premiers inférieurs à un nombre donné par l'utilisateur. Utilisez des boucles `for` imbriquées pour vérifier si chaque nombre est premier.

3. **Exercice 33**
   Créez un programme qui prend deux chaînes de caractères et trouve les lettres communes entre les deux chaînes, sans répétition. Utilisez une boucle `for` pour comparer les lettres et construisez le résultat dans une liste ou une chaîne.

4. **Exercice 34**
   Écrivez un programme qui génère la suite de Collatz pour un nombre donné. La suite de Collatz est définie par : si le nombre est pair, le diviser par 2 ; s'il est impair, le multiplier par 3 et ajouter 1. La suite continue jusqu'à ce que le nombre atteigne 1.

5. **Exercice 35**
   Créez un programme qui prend une liste de listes représentant une matrice carrée et calcule la somme des éléments au-dessus de la diagonale principale.

6. **Exercice 36**
   Écrivez un programme qui génère les premiers `n` nombres de la suite de Lucas, une variante de la suite de Fibonacci avec les termes initiaux `2` et `1`.

7. **Exercice 37**
   Créez un programme qui prend une phrase et affiche chaque mot avec ses lettres en ordre alphabétique. Par exemple, pour "bonjour tout le monde", le programme devrait retourner "bjnoru ottu el ddmeno".

8. **Exercice 38**
   Écrivez un programme qui trouve tous les nombres parfaits inférieurs à un nombre donné. Un nombre parfait est un nombre égal à la somme de ses diviseurs propres (comme 6, qui est 1 + 2 + 3).

9. **Exercice 39**
   Créez un programme qui prend une liste de nombres et calcule le produit de tous les éléments pairs, tout en additionnant les éléments impairs. Affichez le produit des pairs et la somme des impairs.

10. **Exercice 40**
    Écrivez un programme qui génère les `n` premières lignes du triangle de Pascal et les affiche sous forme de liste de listes. Utilisez une boucle `for` pour construire chaque ligne.

Exercices – Niveau 4

1. **Exercice 31**
   Écrivez un programme qui trie une liste d'entiers en utilisant l'algorithme de tri par insertion. Ne pas utiliser les fonctions de tri intégrées de Python, mais seulement des boucles `for` et des comparaisons.

2. **Exercice 32**
   Créez un programme qui prend une matrice (liste de listes) et calcule la somme des éléments de la diagonale principale (du coin supérieur gauche au coin inférieur droit). Utilisez une boucle `for` pour accéder aux éléments de la diagonale.

3. **Exercice 33**
   Écrivez un programme qui génère tous les sous-ensembles possibles d'une liste donnée, sans utiliser de bibliothèques ou de fonctions prédéfinies pour le faire. Par exemple, pour `[1, 2, 3]`, le programme devrait générer `[[], [1], [2], [3], [1, 2], [1, 3], [2, 3], [1, 2, 3]]`.

4. **Exercice 34**
   Créez un programme qui vérifie si deux chaînes de caractères sont des anagrammes (contiennent les mêmes lettres en quantités identiques, sans ordre particulier). Utilisez des boucles `for` pour compter et comparer les lettres de chaque chaîne.

5. **Exercice 35**
   Écrivez un programme qui implémente l'algorithme de tri rapide (quick sort) en utilisant des boucles `for` et des fonctions récursives. Ne pas utiliser les fonctions de tri intégrées.

6. **Exercice 36**
   Créez un programme qui prend une liste de chaînes de caractères et regroupe celles qui sont des anagrammes. Par exemple, pour `['rat',`

`'tar', 'art', 'bat', 'tab']`, le programme devrait retourner `[['rat', 'tar', 'art'], ['bat', 'tab']]`.

7. **Exercice 37**
   Écrivez un programme qui utilise une boucle `for` pour générer les nombres de la suite de Fibonacci jusqu'à ce qu'un terme dépasse une valeur donnée par l'utilisateur. Affichez chaque terme généré.

8. **Exercice 38**
   Créez un programme qui, donné un nombre entier positif, vérifie si ce nombre est un nombre parfait (c'est-à-dire égal à la somme de ses diviseurs propres). Par exemple, 6 est parfait car `1 + 2 + 3 = 6`.

9. **Exercice 39**
   Écrivez un programme qui prend une matrice carrée (liste de listes) et vérifie si elle est symétrique (égale à sa transposée). Utilisez une boucle `for` pour comparer les éléments de la matrice.

10. **Exercice 40**
    Créez un programme qui génère le triangle de Pascal jusqu'à une hauteur donnée. Utilisez une boucle `for` pour calculer chaque ligne du triangle. Par exemple, pour une hauteur de 5, le triangle doit ressembler à :

```
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```