Python: Parcourir une Liste en Python

Parcourir une liste en Python est une tâche fondamentale dans le développement de nombreux programmes. Que ce soit pour effectuer des calculs, rechercher des éléments spécifiques, ou simplement afficher le contenu, savoir comment parcourir efficacement une liste est essentiel. Dans cet article, nous explorerons les différentes méthodes pour parcourir une liste en Python, avec des exemples concrets à chaque étape.

5GMS 3TTR Exploration

En ce qui concerne la vidéo: 😜😂😂😂😂

Parcours avec une Boucle `for`

La méthode la plus courante pour parcourir une liste en Python est d'utiliser une boucle `for`. Voici comment cela fonctionne :

```python
ma_liste = [1, 2, 3, 4, 5]

for element in ma_liste:
    print(element)
```

Cette boucle `for` itère sur chaque élément de la liste `ma_liste` et affiche chaque élément à l'écran. Vous pouvez ensuite effectuer des opérations sur chaque élément à l'intérieur de la boucle.

Accès à l'Indice et à l'Élément en Parallèle

Parfois, vous avez besoin d'accéder à la fois à l'indice et à l'élément de la liste pendant le parcours. Vous pouvez le faire en utilisant la fonction `enumerate()` en combinaison avec une boucle `for`. Voici un exemple :

```python
ma_liste = ['a', 'b', 'c', 'd', 'e']

for index, element in enumerate(ma_liste):
    print(f"L'élément à l'indice {index} est {element}")
```

Cette boucle `for` itère sur chaque élément de la liste `ma_liste` tout en fournissant à la fois l'indice et l'élément à chaque itération.

Parcours avec une Boucle `while`

Bien que moins courante, vous pouvez également parcourir une liste en utilisant une boucle `while` en combinant cela avec un compteur d'index. Voici un exemple :

```
ma_liste = [1, 2, 3, 4, 5]
index = 0

while index < len(ma_liste):
    print(ma_liste[index])
    index += 1
```

Cette boucle `while` utilise un index pour accéder à chaque élément de la liste `ma_liste` et les affiche à l'écran. Elle s'exécute tant que l'index est inférieur à la longueur de la liste.

Conclusion

Parcourir une liste en Python est une opération fondamentale dans de nombreux programmes. Que ce soit avec une boucle `for`, une boucle `while`, en accédant à l'indice et à l'élément en parallèle, vous avez maintenant les outils nécessaires pour parcourir efficacement des listes dans vos programmes Python. En comprenant ces différentes méthodes, vous serez en mesure de manipuler et d'exploiter les données de manière efficace dans vos projets.

Exercices

01 Afficher les nombres

Nom du fichier: `listes_for_01.py` Créez une liste qui contient les 9 premiers nombres premiers et écrivez un programme Python qui parcourt cette liste pour afficher chaque nombre.

```
nombres = [1,3,5,7,11,13,17,19,23]
```

Résultat attendu:

```
1
3
```

```
5
7
11
13
17
19
23
```

02 Jours de la semaine

Nom du fichier: `listes_for_02.py` Créez une liste contenant les jours de la semaine et affichez-la.

Résultat attendu:

```
Lundi
Mardi
Mercredi
Jeudi
Vendredi
Samedi
Dimanche
```

03 Recette de cuisine

Nom du fichier: `listes_for_03.py` Créez une liste d'ingrédients et écrivez un programme qui **affiche** chaque élément de la liste et **compte** le nombre d'éléments de cette liste (comptage en utilisant une variable intermédiaire et sans utiliser la fonction `len`).

Exemple de résultat attendu:

```
Farine
Beurre
Sucre
Chocolat

La liste contient 4 ingrédients.
```

04 Somme

Nom du fichier: `listes_for_04.py` Écrivez un programme Python qui parcourt une liste de nombres et affiche la somme de tous les nombres.

05 Taille

Nom du fichier: `listes_for_05.py` Écrivez un programme Python qui parcourt une liste de chaînes de caractères (noms de chiens) et **affiche la longueur** de chaque chaîne.

Exemple de résultat:

```
médor: 5
snoopy: 6
foulkan: 7
```

06 Pairs

Nom du fichier: `listes_for_06.py` Écrivez un programme Python qui parcourt une liste de nombres et affiche uniquement les nombres pairs.

Exemple de résultat:

```
2
8
12
100
250
```

07 Jours de la semaine

Nom du fichier: `listes_for_07.py` Créez une liste contenant les jours de la semaine et affichez-la en indiquant aussi le numéro de jour.

Résultat attendu:

```
1: Lundi
2: Mardi
3: Mercredi
4: Jeudi
5: Vendredi
6: Samedi
7: Dimanche
```

08 Liste de courses

Nom du fichier: `listes_for_08.py` Créez une liste de courses, afficher cette liste avec le n° de l'aliment, demandez à l'utilisateur le n° de l'élément à supprimer, supprimez l'élément et affichez la liste à nouveau.

Exemple de résultat attendu:

```
Liste:
1. Beurre
2. Pain
3. Sel
4. Salade
5. Eau

Élément à supprimer de la liste? 3

Liste:
1. Beurre
2. Pain
3. Sel
4. Eau
```

09 Sous-marin

Nom du fichier: `listes_for_09.py` Lorsque un sous-marin descend sous la surface de l'océan, il effectue automatiquement un balayage par sonar du fond marin à proximité. Il obtient des valeurs qui sont des mesures de la profondeur du fond marin. Ces valeurs sont enregistrées dans une liste [199, 200, 208, 210, 200, 207, 240, 269, 260, 263].

Combien de valeurs sont **supérieures** à une valeur seuil ? Demander la valeur à l'utilisateur.

Exemples de résultat attendu:

```
Mesures: [199, 200, 208, 210, 200, 207, 240, 269, 260, 263]

Seuil? 210

4 mesures sont supérieures au seuil de 210.

Mesures: [199, 200, 208, 210, 200, 207, 240, 269, 260, 263]

Seuil? 264

1 mesure est supérieure au seuil de 264.


Mesures: [199, 200, 208, 210, 200, 207, 240, 269, 260, 263]

Seuil? 270

Aucune mesure n'est supérieure au seuil de 270.
```

10 Sous-marin mode Kung-Fu panda

Nom du fichier: `listes_for_10.py` Lorsque un sous-marin descend sous la surface de l'océan, il effectue automatiquement un balayage par sonar du fond marin à proximité. Il obtient des valeurs qui sont des mesures de la profondeur du fond marin. Ces valeurs sont enregistrées dans une liste [199, 200, 208, 210, 200, 207, 240, 269, 260, 263].

Combien de valeurs sont **plus grandes que la valeurs précédente** ?

Résultat attendu:

```
Il y a 7 augmentations.
```

En effet :

199, 200 augmentation 200, 208 augmentation 208, 210 augmentation 210, 200 pas d'augmentation 200, 207 augmentation 207, 240 augmentation 240, 269 augmentation 269, 260 pas d'augmentation 260, 263 augmentation

11 Calculer la moyenne "à la main"

Nom du fichier: `listes_for_11.py` Étant donnée une liste de valeurs, calculer la moyenne des éléments de cette liste à l'aide d'une boucle `for`.

Exemple de résultat attendu:

```
Liste: [4, 8, 16, 32, 64]
Moyenne: 24.8
```

12 Élément du milieu

Nom du fichier: `listes_for_12.py` Étant donnée une liste de valeurs, affichez l'élément qui se trouve **exactement** au milieu de la liste. Pour cela, la liste doit contenir un nombre impair de valeurs. Si ce n'est pas le cas, affichez le message "`Pas de milieu... pour l'instant ;)`".

Exemples de résultats attendus:

```
Liste: ["Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi", "Dimanche"]
Milieu: Jeudi

Liste: [1,2,3,4,5,6]
Pas de milieu... pour l'instant ;)
```

Indice: pour trouver le milieu, utilisez `//`.

13 Élément*S* du milieu

Nom du fichier: `listes_for_13.py` Étant donnée une liste de valeurs, affichez le.s élément.s qui se trouve.nt au milieu de la liste.

Si la liste contient un nombre impair de valeurs, affichez la valeur du milieu, sinon affichez les éléments qui se trouvent de part et d'autre du milieu.

Exemples de résultats attendus:

```
Liste: ["Janvier", "Février", "Mars", "Avril", "Mai", "Juin", "Juillet", "Août", "Septembre", "Octobre",
Milieu: Juin / Juillet

Liste: [1,2,3,4,5,6]
Milieu: 3 / 4
```

Indice: pour trouver le milieu, utilisez `//`.

14 Calculer la médiane d'une liste de valeurs

Nom du fichier: `listes_for_14.py` La médiane d'une liste de valeurs est la valeur qui divise la liste en deux parties égales lorsque les valeurs sont triées dans l'ordre croissant ou décroissant. Si la liste a un nombre impair de valeurs, la médiane est simplement la valeur au milieu de la liste. Si la liste a un nombre pair de valeurs, la médiane est la moyenne des deux valeurs du milieu.

Instructions :

1. Ecrivez une fonction nommée `calculer_mediane` qui prend en entrée une liste de nombres et retourne la médiane de la liste.
2. Si la liste a un nombre **impair** de valeurs, la fonction doit retourner la valeur au milieu de la liste après avoir trié les valeurs.
3. Si la liste a un nombre **pair** de valeurs, la fonction doit retourner la moyenne des deux valeurs du milieu après avoir trié les valeurs.
4. Testez votre fonction avec différentes listes de valeurs pour vous assurer qu'elle calcule correctement la médiane dans tous les cas possibles.

Rappel : Pour trier une liste de valeurs en Python, vous pouvez utiliser la méthode `sort()` ou la fonction `sorted()`. La méthode `sort()` modifie la liste directement, tandis que la fonction `sorted()` retourne une nouvelle liste triée sans modifier l'originale.

Exemple de résultat attendu:

```
Liste: [99,86,87,88,111,86,103,87,94,78,77,85,86]
Médiane: 87
```

Vidéo à la une à regarder sur Youtube: http://youtu.be/d584hPPW-x8