Python: L'opérateur 'f' et les f-strings

Lorsque vous programmez en Python, vous rencontrerez probablement l'opérateur 'f' dans le contexte des chaînes de caractères formatées. Cet opérateur, également appelé 'f-string', est un outil puissant et pratique pour formater les chaînes de caractères de manière claire et concise. Dans cet article, nous allons explorer en profondeur l'utilisation et les avantages de l'opérateur 'f' en Python.

3TTR Exploration

Qu'est-ce qu'un f-string ?

Un f-string est une **syntaxe spéciale de formatage de chaînes de caractères** introduite dans Python 3.6. Il permet d'incorporer des expressions Python directement dans les chaînes de caractères, ce qui simplifie grandement le processus de création de chaînes formatées. Pour créer un f-string, **vous ajoutez simplement un "`f`" ou "`F`" avant le début de la chaîne** et vous utilisez des **accolades** "`{}`" pour insérer des expressions Python à l'intérieur de la chaîne.

Voici un exemple simple d'utilisation de l'opérateur "f" :

```python
name = "Alice"
age = 30
print(f"Bonjour, je m'appelle {name} et j'ai {age} ans.")
```

On peut aussi stocker le résultat dans une variable. Voici l'exemple précédent adapté:

```python
name = "Alice"
age = 30

salution = f"Bonjour, je m'appelle {name} et j'ai {age} ans."

print(salutation)
```

Dans cet exemple, les variables `name` et `age` sont directement incorporées dans la chaîne de caractères à l'aide de l'opérateur "f". Lorsque vous exécutez ce code, vous obtenez la sortie suivante :

```
Bonjour, je m'appelle Alice et j'ai 30 ans.
```

Comme vous pouvez le voir, les valeurs des variables sont automatiquement substituées dans la chaîne formatée.

Formatage des nombres avec l'opérateur "f"

En plus de formater les chaînes de caractères, l'opérateur "`f`" peut également être utilisé **pour formater des nombres de manière précise**. Vous pouvez spécifier la précision des nombres flottants, le nombre de chiffres après la virgule, et même le format des nombres entiers.

Voici quelques exemples de formatage de nombres avec l'opérateur "f" :

```
pi = 3.141592653589793
print(f"La valeur de pi avec 2 décimales : {pi:.2f}")
# Sortie : La valeur de pi avec 2 décimales : 3.14

percentage = 0.45678
print(f"Le pourcentage avec 1 décimale : {percentage:.1%}")
# Sortie : Le pourcentage avec 1 décimale : 45.7%

large_number = 1000000
print(f"Le grand nombre formaté avec séparateur de milliers : {large_number:,}")
# Sortie : Le grand nombre formaté avec séparateur de milliers : 1,000,000
```

Dans ces exemples, nous utilisons des spécificateurs de formatage après les deux-points ":" pour définir le format des nombres. Par exemple, ".2f" spécifie que nous voulons afficher le nombre avec deux décimales, "%.1f" spécifie une décimale pourcentage, et "," spécifie l'utilisation du séparateur de milliers pour les grands nombres entiers.

Spécifier le nombre de décimales des floats

Pour afficher une variable de type `float`, vous pouvez utiliser le spécificateur `.xf`, où `f` signifie '`float` et `x` représente le nombre de décimales à afficher. Notez que le résulat est **arrondi**!

```
pi = 3.141592653589793
print(f"La valeur de pi avec 2 décimales : {pi:.2f}")
# Sortie : La valeur de pi avec 2 décimales : 3.14

print(f"La valeur de pi avec 2 décimales : {pi:.4f}")
# Sortie : La valeur de pi avec 2 décimales : 3.1416
```

Utilisation des pourcentages avec les f-strings

L'opérateur "f" en Python offre une manière simple et efficace d'afficher des pourcentages dans vos chaînes de caractères formatées. Vous pouvez formater les pourcentages avec précision en utilisant des spécificateurs de formatage appropriés dans les f-strings.

Affichage d'un pourcentage avec une décimale

```python
percentage = 0.45678
print(f"Le pourcentage avec 1 décimale : {percentage:.1%}")
# Sortie : Le pourcentage avec 1 décimale : 45.7%
```

Dans cet exemple, le spécificateur de formatage ".1%" indique à Python d'afficher le pourcentage avec une seule décimale.

Affichage d'un pourcentage avec deux décimales

```python
percentage = 0.789
print(f"Le pourcentage avec 2 décimales : {percentage:.2%}")
# Sortie : Le pourcentage avec 2 décimales : 78.90%
```

Ici, le spécificateur ".2%" est utilisé pour afficher le pourcentage avec deux décimales.

Affichage d'un pourcentage sans décimales

```python
percentage = 0.25
print(f"Le pourcentage sans décimales : {percentage:.0%}")
# Sortie : Le pourcentage sans décimales : 25%
```

En spécifiant ".0%", vous pouvez afficher le pourcentage sans décimales.

Utilisation de pourcentages dans des calculs

Les f-strings permettent également d'incorporer des pourcentages dans des calculs et d'afficher les résultats formatés :

```python
total = 1500
percentage_discount = 0.20
discount_amount = total * percentage_discount
print(f"Vous avez droit à une réduction de {percentage_discount:.0%}, ce qui correspond à {dis
# Sortie : Vous avez droit à une réduction de 20%, ce qui correspond à 300 euros.
```

Dans cet exemple, nous calculons le montant de la réduction en multipliant le total par le pourcentage de réduction, puis affichons le pourcentage formaté dans la chaîne de caractères.

En utilisant les spécificateurs de formatage appropriés avec les f-strings, vous pouvez facilement manipuler et afficher des pourcentages de manière précise dans vos programmes Python.

Avantages de l'opérateur "f"

L'utilisation de l'opérateur "f" présente plusieurs avantages :

1. Lisibilité améliorée

Les f-strings rendent le code plus lisible en permettant l'insertion directe de variables et d'expressions Python dans les chaînes de caractères. Cela rend le code plus concis et plus facile à comprendre.

2. Plus de clarté

En utilisant des f-strings, vous pouvez éviter les constructions de chaînes de caractères compliquées avec des opérations de concaténation. Cela rend votre code plus clair et moins sujet aux erreurs.

3. Performances améliorées

Les f-strings sont généralement plus performants que d'autres méthodes de formatage de chaînes de caractères, comme les méthodes `format()` ou `%`. Cela est dû à leur implémentation optimisée dans Python.

Bonnes pratiques

Lorsque vous utilisez des f-strings, gardez à l'esprit les bonnes pratiques suivantes :

- Utilisez des noms de variables descriptifs pour améliorer la lisibilité de votre code.
- Évitez d'incorporer des expressions trop complexes dans les f-strings, car cela peut rendre le code difficile à comprendre.
- Assurez-vous que les expressions incorporées sont sûres et ne présentent pas de risques de sécurité, surtout si elles proviennent d'une source externe.

Conclusion

En résumé, l'opérateur "f" en Python est un outil extrêmement utile pour formater les chaînes de caractères de manière claire et concise, ainsi que pour formater des nombres avec précision. En utilisant des f-strings, vous pouvez améliorer la lisibilité de votre code, éviter les erreurs de formatage et améliorer les performances de votre application. En incorporant les bonnes pratiques, vous pouvez tirer pleinement parti de la puissance de l'opérateur "f" dans vos projets Python.

Exercices

Enregistrez vos fichiers dans `Python \ Exercices \ f-strings`.

Le gardien de phare

Reprenez l'exercice du Gardien de phare et réécrivez la `#sortie` en utilisant une f-string.

Nom du fichier: `gardien-fstring-[prenom].py`

Aire et circonférence du cercle

Reprenez l'exercice sur le calcul de l'aire et de la circonférence du cercle.

Utilisez la valeur de `pi` offerte par le module `math` (`from math import pi`).

Réécrivez la `#sortie` en utilisant une f-string pour obtenir le résultat suivant (valeurs arrondies à 2 décimales):

```
Rayon? 10
Circonférence: 62.83
Aire: 314.16
```

Nom du fichier: `cercle-fstring-[prenom].py`

Calcul de la TVA

Ecrivez un programme qui demande un prix unitaire hors TVA et le nombre d'unités à l'utilisateur et permet de calculer une remise, le montant de la TVA et le montant TVA comprise.

```
1  taux_remise = 0.03
2  taux_tva = 0.21
3
4  # Votre code
5
```

Résultat attendu:

```
Prix unitaire HTVA? 100
Nombre de pièces? 10

Montant: 1000€
Remise 3%: 30€

Montant HTVA: 970€
TVA 21%: 203.70€

Montant TVAC: 1173.70€
```

Remarquez comment les taux de remise et de TVA ainsi que les montants sont affichés: `3%`, `21%`, `203.70€` (2 chiffres après la virgule).

Nom du fichier: `tva-fstring-[prenom].py`

Vidéo à la une à regarder sur Youtube: http://youtu.be/RgzX4qrLmPM