Python: Les fonctions

En Python, comme dans d'autres langages de programmation, une fonction est une suite d'instructions que l'on peut appeler avec un nom. C'est un moyen de regrouper des opérations pour les réutiliser efficacement. Cela permet de rendre ton code plus clair, plus organisé et plus facile à maintenir.

5GMS Exploration

Qu'est-ce qu'une fonction?

Une fonction est une **séquence d'instructions** que Python effectue lorsqu'elle est appelée. Elle peut prendre des **paramètres** et peut **renvoyer une valeur**. Les fonctions sont définies avec le mot-clé `def` suivi du nom de la fonction et de parenthèses qui *peuvent* contenir des paramètres.

Syntaxe de base

```
def nom_de_la_fonction(paramètre1, paramètre2, ...):
    # Bloc d'instructions
    return valeur  # Facultatif
```

Pourquoi utiliser des fonctions?

- **Réutilisation de code**: Écris une fois, utilise plusieurs fois.
- **Organisation**: Divise ton programme en petites parties gérables.
- **Maintenance**: Corrige ou améliore une partie sans toucher au reste.
- **Clarté**: Rend ton code plus lisible et compréhensible.

Créer une fonction simple

Exemple:

```
def saluer(nom):
    print(f"Bonjour, {nom}!")
```

Appel de la fonction:

```
saluer("Alice")
```

Résultat:

```
Bonjour, Alice!
```

Paramètres et arguments

- **Paramètres**: Variables listées dans la définition de la fonction.
- **Arguments**: Valeurs fournies lors de l'appel de la fonction.

Types d'arguments

- **Positionnels**: Doivent être passés dans l'ordre.
- **Nominaux (mot-clé)**: Peuvent être passés dans n'importe quel ordre en spécifiant leur nom.

Exemple avec arguments nominaux:

```
def afficher_info(nom, age):
    print(f"Nom: {nom}, Âge: {age}")

afficher_info(age=30, nom="Alice")
```

Valeurs de retour

Utilise `return` pour renvoyer une valeur à l'endroit où la fonction est appelée.

Exemple:

```
def somme(a, b):
    return a + b

resultat = somme(5, 3)
print(resultat)  # Affichera 8
```

Tu peux faire ce que tu veux avec la valeur d'une fonction: l'afficher, l'utiliser dans un calcul, la stocker dans une variable (comme dans l'exemple précédent)... ou encore... ne rien faire:

```python
print(resultat(2,5)) # Affichera 7
print(2 * resultat(5,5)) # Affichera 20
resultat(7,7) # Pas très utile, mais fonctionne
```

Portée des variables

Les variables en Python peuvent être soit locales, soit globales, et la différence entre les deux est cruciale pour comprendre comment Python gère l'espace de noms des variables

- **Locale**: Existe uniquement à l'intérieur de la fonction.
- **Globale**: Accessible partout dans le code.

Une variable **locale** est définie **à l'intérieur** d'une fonction et n'existe que dans l'espace local de cette fonction. Elle est **inaccessible depuis l'extérieur de la fonction**, ce qui signifie que toute tentative d'y accéder en dehors de la fonction entraînera une erreur. Cela permet de s'assurer que la fonction ne dépend pas de, ou n'interfère pas avec, le reste du programme.

À l'inverse, une variable **globale** est définie en dehors de toutes les fonctions et est **accessible depuis n'importe quelle partie du programme**, y compris à l'intérieur des fonctions. Cependant, si tu souhaites modifier une variable globale à l'intérieur d'une fonction, tu dois d'abord la déclarer comme `global` dans la fonction; sinon, Python créera une nouvelle variable locale du même nom. Il est généralement conseillé d'utiliser des variables locales autant que possible pour éviter des effets secondaires inattendus et rendre le code plus facile à comprendre et à maintenir.

Exemple de variable locale:

```python
def exemple():
    x = 5  # x est local à exemple

exemple()
print(x)
```

Bonnes pratiques

- **Nom explicite**: Choisir des noms de fonctions clairs et descriptifs.
- **Une tâche**: Chaque fonction doit avoir une seule responsabilité.
- **Courte**: Si une fonction devient trop longue, envisage de la diviser.
- **Documentation**: Utilise des docstrings pour expliquer ce que fait la fonction.

Exemple de docstring:

```python
def somme(a, b):
    """
    Retourne la somme de deux nombres.
```

```
    """
3       return a + b
```

Conclusion

Les fonctions sont un outil puissant en Python qui t'aide à écrire des codes clairs, organisés et réutilisables. En les utilisant judicieusement, tu peux rendre la programmation plus agréable et maintenir tes projets plus facilement. Pense à les utiliser chaque fois que tu remarques une répétition ou que tu souhaites clarifier ton code!

Vidéo à la une à regarder sur Youtube: http://youtu.be/lp5KTIduKJw