Fiche pour les débutants en Python

1 Création et Exécution d'un Fichier Python

Pour créer un fichier Python, il suffit de créer un fichier texte avec l'extension `.py`. Par exemple, vous pouvez créer un fichier nommé `mon_programme.py` et y écrire votre code Python. Pour exécuter ce fichier, ouvrez un terminal, naviguez jusqu'au répertoire contenant le fichier et tapez :

```
python mon_programme.py
```

2 Votre Premier Programme Python

Pour commencer, écrivez un programme simple qui affiche un message à l'écran. Cela vous permettra de vous familiariser avec la syntaxe de base de Python.

```
print("Hello world!")
```

Ce code affiche "Hello world!" à l'écran. Félicitations, vous venez d'écrire votre premier programme en Python !

3 Les Variables et les Types de Données

Les variables sont des espaces de stockage pour des données qui peuvent être utilisées et manipulées dans votre programme. Elles peuvent contenir différents types de données tels que des nombres, des chaînes de caractères, des listes, etc.

3.1 Nombres (int, float)

Les nombres sont utilisés pour représenter des valeurs numériques. Les nombres entiers (`int`) sont des nombres sans décimales, tandis que les nombres à virgule flottante (`float`) sont des nombres avec décimales.

```
age = 25  # nombre entier
temperature = 36.6  # nombre flottant
print(age)  # Affiche: 25
print(temperature)  # Affiche: 36.6
```

3.2 Chaînes de caractères (str)

Les chaînes de caractères sont utilisées pour représenter du texte. Elles sont entourées de guillemets simples ou doubles.

```
nom = "Alice"
print(nom)  # Affiche: Alice
```

3.3 Listes

Les listes sont des collections ordonnées d'éléments, pouvant contenir des types de données différents. Elles sont très utiles pour stocker des séries de valeurs.

```
fruits = ["pomme", "banane", "cerise"]
print(fruits)  # Affiche: ['pomme', 'banane', 'cerise']
```

3.4 Dictionnaires

Les dictionnaires sont des collections non ordonnées de paires clé-valeur, où chaque clé est unique. Ils permettent d'associer des valeurs à des clés, facilitant ainsi l'accès et la manipulation des données.

```
etudiant = {"nom": "Alice", "age": 25, "cours": "Mathematiques"}
print(etudiant)  # Affiche: {'nom': 'Alice', 'age': 25, 'cours': '
    Mathematiques'}
```

3.5 Booléens

Les booléens sont des valeurs logiques qui peuvent être `True` (vrai) ou `False` (faux). Ils sont souvent utilisés dans les conditions pour contrôler le flux d'exécution du code.

```
est_etudiant = True
print(est_etudiant)  # Affiche: True
```

Les variables en Python sont dynamiquement typées, ce qui signifie que vous pouvez changer leur type en cours de route en leur assignant une nouvelle valeur d'un type différent.

4 Les Opérations de Base

Les opérations arithmétiques de base sont essentielles pour effectuer des calculs dans vos programmes. Python supporte plusieurs opérations arithmétiques, telles que l'addition, la soustraction, la multiplication, la division et le modulo.

4.1 Addition

Ajoute deux nombres.

```
somme = 5 + 3
print(somme)  # Affiche: 8
```

4.2 Soustraction

Soustrait un nombre d'un autre.

```
difference = 10 - 2
print(difference)  # Affiche: 8
```

4.3 Multiplication

Multiplie deux nombres.

```
produit = 4 * 2
print(produit)  # Affiche: 8
```

4.4 Division

Divise un nombre par un autre.

```
quotient = 16 / 4
print(quotient)  # Affiche: 4.0
```

4.5 Modulo

Renvoie le reste d'une division.

```
reste = 10 % 3
print(reste)  # Affiche: 1
```

Ces opérations peuvent être combinées et utilisées pour des calculs plus complexes.

5 Structures de Contrôle

Les structures de contrôle permettent de contrôler le flux d'exécution de votre code. Elles sont essentielles pour prendre des décisions et répéter des actions.

5.1 Conditions (if, elif, else)

Les conditions permettent d'exécuter du code uniquement si certaines conditions sont remplies. Elles sont très utiles pour prendre des décisions basées sur des valeurs dynamiques.

```
age = 20
if age < 18:
    print("Mineur")
elif age == 18:
    print("A peine majeur")
else:
    print("Majeur")
```

Dans cet exemple, le code vérifie l'âge et affiche un message différent selon que l'âge est inférieur à 18, égal à 18 ou supérieur à 18.

5.2 Boucles (for, while)

Les boucles permettent de répéter des actions plusieurs fois.

5.2.1 Boucle for

Utilisée pour itérer sur une séquence d'éléments (liste, tuple, dictionnaire, etc.).

```
for i in range(5):
    print(i)  # Affiche les nombres de 0 a 4
```

Dans cet exemple, la boucle `for` itère cinq fois, de 0 à 4, et imprime chaque valeur.

5.2.2 Boucle while

Continue de s'exécuter tant qu'une condition est vraie.

```
compte = 0
while compte < 5:
    print(compte)  # Affiche les nombres de 0 a 4
    compte += 1
```

Ici, la boucle `while` continue d'exécuter le bloc de code jusqu'à ce que `compte` atteigne 5.

Les structures de contrôle sont fondamentales pour créer des programmes logiques et dynamiques.

6 Fonctions

Les fonctions sont des blocs de code réutilisables qui effectuent une tâche spécifique. Elles permettent de structurer votre programme et de réduire la répétition de code.

```
def saluer(nom):
    return "Bonjour " + nom + " !"

message = saluer("Alice")
print(message)  # Affiche: Bonjour, Alice !
```

Dans cet exemple, la fonction `saluer` prend un paramètre `nom` et retourne une chaîne de caractères personnalisée. Les fonctions sont définies avec le mot-clé `def`, suivi du nom de la fonction et des parenthèses. L'utilisation de fonctions vous permet de diviser votre code en segments logiques, facilitant ainsi la lecture, la maintenance et le débogage.

7 Les Listes et les Dictionnaires

Les listes et les dictionnaires sont des structures de données fondamentales en Python.

7.1 Listes

Une liste est une collection ordonnée d'éléments. Les éléments d'une liste peuvent être modifiés, ajoutés ou supprimés. Les listes sont très utiles pour stocker des séries de valeurs et les manipuler.

```
fruits = ["pomme", "banane", "cerise"]
fruits.append("orange")
print(fruits)  # Affiche: ['pomme', 'banane', 'cerise', 'orange']
```

Dans cet exemple, nous créons une liste de fruits et ajoutons un nouvel élément à la liste avec la méthode `append`.

7.2 Dictionnaires

Un dictionnaire est une collection non ordonnée de paires clé-valeur. Chaque clé doit être unique dans un dictionnaire. Les dictionnaires sont idéaux pour stocker des données associatives.

```
etudiant = {"nom": "Alice", " ge ": 25, "cours": "Math matiques"}
etudiant["note"] = "A"
print(etudiant)  # Affiche: {'nom': 'Alice', ' ge ': 25, 'cours': '
    Math matiques', 'note': 'A'}
```

Ici, nous créons un dictionnaire pour un étudiant et ajoutons une nouvelle paire clé-valeur pour la note de l'étudiant. Les dictionnaires permettent un accès rapide et efficace aux données.

8 Modules et Bibliothèques

Python dispose de nombreux modules et bibliothèques qui étendent ses fonctionnalités. Un module est un fichier contenant du code Python, tandis qu'une bibliothèque est un ensemble de modules. Vous pouvez importer et utiliser des modules pour accéder à des fonctionnalités supplémentaires.

```
import math

print(math.sqrt(16))  # Affiche: 4.0
```

Dans cet exemple, nous utilisons le module `math` pour calculer la racine carrée de 16. Le mot-clé `import` est utilisé pour inclure un module dans votre programme. L'utilisation de modules vous permet de réutiliser du code existant et de simplifier votre propre code en vous appuyant sur des fonctionnalités éprouvées.

9 Exercices Pratiques

1. Écrivez un programme qui demande à l'utilisateur son nom et son âge, puis affiche un message de salutation.

2. Créez une liste de nombres de 1 à 10 et affichez leur carré.

3. Créez un dictionnaire avec des informations sur un livre (titre, auteur, année) et affichez ces informations.

10 Conclusion

Cette fiche vous a présenté les bases de Python. Pour aller plus loin, n'hésitez pas à explorer la documentation officielle et à pratiquer en résolvant des exercices et en travaillant sur des projets personnels ou à poser des questions à vos magnifiques assistants. Bonne programmation !