Cours Python : Développer un Analyseur de Données Météorologiques

Objectifs du Cours

Le but de ce TP est d'apprendre les bases du langage python de programmation en créant un petit programme qui analyse des données météorologiques fictives.

À la fin de ce cours, vous devriez être capable de :

- Importer des modules en Python.
- Utiliser des variables pour stocker des données.
- Afficher des informations avec `print`.
- Manipuler des listes.
- Utiliser la bibliothèque `numpy` pour des calculs numériques.
- Créer et utiliser des boucles.
- Définir et appeler des fonctions.

Qu'est-ce qu'un module ?

Un module est un fichier contenant du code Python pré-écrit que vous pouvez utiliser dans vos propres programmes. Cela permet de réutiliser du code et d'accéder à des fonctionnalités avancées sans les écrire vous-même.

Exemple : Importer `numpy`

```
import numpy as np
```

Explications :

- `import` : Mot-clé utilisé pour importer un module.
- `numpy` : Un module puissant pour les calculs numériques.
- `as np` : Permet d'utiliser un alias (`np`) pour simplifier l'écriture des commandes de `numpy` .

Exercice :

Essayez d'importer le module `math` et utilisez-le pour calculer la racine carrée de 16.

Les Variables :

Les variables sont des espaces de stockage pour conserver des données. Vous pouvez les nommer et leur attribuer des valeurs.

Exemple :

```
# Définir une variable contenant une liste de températures
temperatures = [22, 21, 23, 20, 19, 24, 22]

# Afficher le contenu de la variable
print("Températures enregistrées :", temperatures)
```

Explications :

- `temperatures` : Nom de la variable.
- `[22, 21, 23, 20, 19, 24, 22]` : Liste de valeurs stockées dans la variable.
- `print` : Fonction qui affiche le contenu des variables à l'écran.

Exercice :

Créez une variable `jours` contenant les jours de la semaine et affichez-la.

Les Listes :

Les listes sont des structures de données qui permettent de stocker plusieurs éléments dans un ordre spécifique. Chaque élément peut être accédé via son indice, qui commence à 0.

Exemple :

```
# Ajouter une température pour le 8ème jour
temperatures.append(25)
print("Températures après ajout :", temperatures)

# Accéder à la température du premier jour
premier_jour = temperatures[0]
print("Température du premier jour :", premier_jour, "°C")
```

Explications :

- `.append(25)` : Ajoute l'élément `25` à la fin de la liste `temperatures`.
- `temperatures[0]` : Accède au premier élément de la liste (indice 0).

Exercice :

Ajoutez la température `18` au début de la liste `temperatures` et affichez la liste mise à jour.

Pourquoi utiliser `numpy` ?

`numpy` est une bibliothèque puissante qui facilite les calculs numériques, particulièrement avec des tableaux de données.

Exemple :

```
# Convertir la liste en tableau numpy
temp_array = np.array(temperatures)

# Calculer la moyenne
moyenne = np.mean(temp_array)
print("Température moyenne :", moyenne, "°C")

# Calculer la température maximale et minimale
max_temp = np.max(temp_array)
min_temp = np.min(temp_array)
print("Température maximale :", max_temp, "°C")
print("Température minimale :", min_temp, "°C")
```

Explications :

- `np.array(temperatures)` : Convertit la liste `temperatures` en un tableau `numpy`.
- `np.mean()` : Calcule la moyenne des éléments du tableau.
- `np.max()` et `np.min()` : Trouvent respectivement les valeurs maximale et minimale.

Exercice :

Calculez la médiane des températures en utilisant `numpy`.

Les Boucles :

Les boucles permettent de répéter des actions pour chaque élément d'une liste ou d'un autre type de collection.

Exemple :

```
# Liste des jours correspondants
jours = ["Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi", "Dimanche", "Lundi suivant"]

# Afficher chaque température avec le jour correspondant
for i in range(len(temperatures)):
    print(f"{jours[i]} : {temperatures[i]} °C")
```

Explications :

- `for i in range(len(temperatures))` : Boucle qui itère sur les indices de la liste `temperatures`.
- `f"{jours[i]} : {temperatures[i]} °C"` : Utilise une f-string pour formater la chaîne de caractères avec les valeurs des listes.

Exercice :

Modifiez la boucle pour afficher uniquement les températures supérieures à 20°C.

Les Fonctions :

Les fonctions sont des blocs de code réutilisables qui effectuent une tâche spécifique.
Elles permettent d'organiser le code de manière modulaire et lisible.

Exemple :

```
def afficher_statistiques(temp_list):
    temp_np = np.array(temp_list)
    moyenne = np.mean(temp_np)
    max_temp = np.max(temp_np)
    min_temp = np.min(temp_np)
    print("\nStatistiques des températures :")
    print(f"Moyenne : {moyenne} °C")
    print(f"Maximale : {max_temp} °C")
    print(f"Minimale : {min_temp} °C")

# Appeler la fonction avec la liste des températures
afficher_statistiques(temperatures)
```

Explications :

- `def afficher_statistiques(temp_list):` : Déclare une fonction nommée `afficher_statistiques` qui prend `temp_list` en paramètre.
- À l'intérieur de la fonction, nous calculons et affichons la moyenne, la température maximale et minimale.

Exercice :

Créez une fonction `ajouter_temperature` qui prend une liste de températures et une nouvelle température en paramètre, ajoute la température à la liste et affiche la liste mise à jour.

Script Complet

```
import numpy as np

# Variables initiales
temperatures = [22, 21, 23, 20, 19, 24, 22]
jours = ["Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi", "Dimanche", "Lundi suivant"]

# Fonctions
def afficher_temperatures(jours, temp_list):
    print("\nTempératures enregistrées :")
    for i in range(len(temp_list)):
        print(f"{jours[i]} : {temp_list[i]} °C")

def afficher_statistiques(temp_list):
    temp_np = np.array(temp_list)
    moyenne = np.mean(temp_np)
    max_temp = np.max(temp_np)
    min_temp = np.min(temp_np)
    print("\nStatistiques des températures :")
    print(f"Moyenne : {moyenne} °C")
    print(f"Maximale : {max_temp} °C")
    print(f"Minimale : {min_temp} °C")

def ajouter_temperature(temp_list, nouvelle_temp):
    temp_list.append(nouvelle_temp)
    print("\nTempératures après ajout :", temp_list)

# Ajout d'une nouvelle température
ajouter_temperature(temperatures, 25)

# Afficher les températures
afficher_temperatures(jours, temperatures)

# Afficher les statistiques
afficher_statistiques(temperatures)
```

Explications :

1. **Importation :** Nous importons `numpy` pour les calculs numériques.
2. **Variables :** Définition des listes `temperatures` et `jours`.
3. **Fonctions :**
   - `afficher_temperatures` : Affiche chaque température avec le jour correspondant.
   - `afficher_statistiques` : Calcule et affiche les statistiques des températures.
   - `ajouter_temperature` : Ajoute une nouvelle température à la liste et affiche la liste mise à jour.
4. **Exécution :**
   - Ajout d'une température.
   - Affichage des températures.
   - Affichage des statistiques.

Exercice Final :

Ajoutez une fonctionnalité pour calculer et afficher la température médiane dans la fonction `afficher_statistiques`.

Prochaines Étapes

1. **Exercices Supplémentaires :**
    - Ajoutez plus de jours et de températures à la liste.
    - Implémentez une fonctionnalité pour entrer les températures via le clavier.
    - Calculez d'autres statistiques, comme l'écart-type.
2. **Explorer d'Autres Bibliothèques :**
    - Apprenez à utiliser `matplotlib` pour créer des graphiques de vos données météorologiques.
3. **Projets Plus Avancés :**
    - Développez une application complète d'analyse de données météorologiques avec des fonctionnalités avancées, comme la lecture de données depuis un fichier.

Ressources Utiles

- **Documentation Python :** https://docs.python.org/3/
- **Tutoriels `numpy` :** https://numpy.org/doc/stable/
- **Communauté Python :** Rejoignez des forums comme Stack Overflow pour poser vos questions.