Atelier 1 : Introduction à Python et Concepts de Base pour le Machine Learning

Objectif de l'atelier :

Cet atelier vise à vous initier aux concepts fondamentaux de Python tout en montrant comment ces bases peuvent être appliquées à des tâches spécifiques de Machine Learning, comme la manipulation des données, leur visualisation et l'entraînement de modèles simples. Le but étant de vous montrer comment les concepts de base en Python (variables, structures conditionnelles, fonctions) peuvent être appliqués directement à des tâches spécifiques de Machine Learning, comme l'exploration de données, leur visualisation et l'entraînement de modèles simples.

Partie 1 : Structures de base

Exercice 1 (Les variables) :

Déclarez une variable `age` et assignez-lui votre âge, puis affichez-la.

Solution :

```
âge = 25

print("Mon âge est :", âge)
```

Exercice 2 (Les commentaires):

Ajoutez des commentaires à ce programme pour expliquer chaque étape.

Solution :

```
# Demander à l'utilisateur de saisir deux nombres

a = float(input("Entrez le premier nombre : "))

b = float(input("Entrez le deuxième nombre : "))

# Calculer la somme des deux nombres

somme = a + b

# Afficher le résultat

print("La somme est :", somme)
```

Exercice 3 (L'addition) :

Écrivez un programme Python qui demande à l'utilisateur de saisir deux nombres et affiche leur somme.

Solution :

```
# Demander à l'utilisateur de saisir trois nombres

nombre1 = float(input("Entrez le premier nombre : "))

nombre2 = float(input("Entrez le deuxième nombre : "))

nombre3 = float(input("Entrez le troisième nombre : "))

# Calculer le produit des trois nombres

produit = nombre1 * nombre2 * nombre3

# Afficher le résultat

print("Le produit est :", produit)
```

Exercice 4 (La déclaration des chiffres) :

Déclarez deux variables `x` et `y`, assignez-leur des valeurs, puis affichez-les.

Solution :

```
X = int(input("Entrez une premiére valeur : "))

Y = int(input("Entrez une deuxiéme valeur : "))
```

Exercice 5 (L'opérateur +) :

Écrivez un programme Python qui demande à l'utilisateur de saisir deux chaînes de caractères et les concatènes, puis affiche le résultat.

Solution :

```
chaine1 = input("Entrez la première chaîne : ")

chaine2 = input("Entrez la deuxième chaîne : ")

concaténation = chaine1 + chaine2

print("La concaténation des deux chaînes est :", concaténation)
```

Exercice 6 (Le typecasting(conversion de type)) :

Écrivez un programme Python qui convertit un nombre flottant en entier, puis affiche le résultat.

Solution :

```
# Demander à l'utilisateur de saisir un nombre flottant

nombre_flottant = float(input("Entrez un nombre flottant : "))

# Convertir le nombre flottant en entier

nombre_entier = int(nombre_flottant)

# Afficher le résultat

print("Le nombre flottant converti en entier est :", nombre_entier)
```

Exercice 7:

Écrivez un programme Python qui compte le nombre de caractères dans une chaîne donnée par l'utilisateur et affiche le résultat.

Solution :

```
chaine = input("Entrez une chaîne de caractères : ")

nombre_caracteres = len(chaine)

print("Le nombre de caractères dans la chaîne est :", nombre_caracteres)
```

Exercice 8 (La manipulation des listes) :

Écrivez un programme Python qui prend une liste de nombres donnée par l'utilisateur, calcule leur somme et affiche le résultat.

Solution :

```
liste_nombres = input("Entrez une liste de nombres séparés par des espaces : ").split()

liste_nombres = [int(nombre) for nombre in liste_nombres]

somme = sum(liste_nombres)

print("La somme des nombres est :", somme)
```

Partie 2 : Structures conditionnelles

Exercice 1:

Écrivez un programme Python qui demande à l'utilisateur de saisir un nombre et vérifie s'il est positif, négatif ou nul.

Solution :

```
nombre = float(input("Entrez un nombre : "))

if nombre > 0:

print("Le nombre est positif.")
```

```
elif nombre < 0:

print("Le nombre est négatif.")

else:

print("Le nombre est nul.")
```

Exercice 2 :

Écrivez un programme Python qui vérifie si un nombre donné par l'utilisateur est pair ou impair.

Solution :

```
nombre = int(input("Entrez un nombre : "))

if nombre % 2 == 0:

print("Le nombre est pair.")

else:

print("Le nombre est impair.")
```

Partie 3 : Déclarations de fonctions

Exercice 1:

Écrivez une fonction Python appelée `calcul_moyenne` qui prend deux nombres en argument et retourne leur moyenne.

Solution:

```
def calcul_moyenne(a, b):

return (a + b) / 2

print("La moyenne est :", calcul_moyenne(5, 8))
```

Exercice 2:

Écrivez une fonction Python appelée `diviser` qui prend deux nombres en argument et retourne le résultat de la division. Si le dénominateur est égal à zéro, affichez un message d'erreur.

Solution :

```
def diviser(dividende, diviseur):

if diviseur != 0:

return dividende / diviseur

else:

return "Erreur : division par zéro."

print(diviser(10, 2))

print(diviser(8, 0))
```

Partie 4 : Déclarations des variables

Exercice 1: Déclarez une variable `nom` et assignez-lui votre prénom, puis affichez-la.

Solution :

```
nom = "Alice"

print("Bonjour,", nom)
```

Exercice 2:

Déclarez deux variables `x` et `y`, assignez-leur des valeurs, puis échangez les valeurs des deux variables et affichez-les.

Solution :

```
x = 5

y = 10

# Échange des valeurs des variables
```

```
x, y = y, x

print("x est maintenant :", x)

print("y est maintenant :", y)
```

Partie 5 : Opérateurs

Exercice 1 : Écrivez un programme Python qui demande à l'utilisateur de saisir deux nombres et affiche leur somme.

Solution :

```
a = float(input("Entrez le premier nombre : "))

b = float(input("Entrez le deuxième nombre : "))

somme = a + b

print("La somme est :", somme)
```

Exercice 2 : Écrivez un programme Python qui vérifie si un nombre donné par l'utilisateur est pair ou impair.

Solution :

```
nombre = int(input("Entrez un nombre : "))

if nombre % 2 == 0:

print("Le nombre est pair.")

else: print("Le nombre est impair.")
```

Partie 6 : Conversion de types de données

Exercice 1 : Écrivez un programme Python qui demande à l'utilisateur de saisir un nombre entier et le convertit en flottant, puis affiche le résultat

Solution 1:

```
entier = int(input("Entrez un nombre entier : "))

flottant = float(entier)

print("Le nombre en flottant est :", flottant)
```

Exercice 2 : Écrivez un programme Python qui convertit une liste de nombres en une chaîne de caractères et affiche le résultat.

Solution 2 :

```
liste_nombres = [1, 2, 3, 4, 5]

chaine_nombres = ''.join(map(str, liste_nombres))

print("La liste convertie en chaîne est :", chaine_nombres)
```

Définition :

- Sépale : C'est une partie extérieure de la fleur qui entoure et protège les pétales avant l'épanouissement. Les sépales font généralement partie du calice de la fleur.
- Longueur du sépale : C'est une mesure de la longueur d'un sépale, exprimée en centimètres dans le dataset Iris.

Partie 7 : Manipulation des données

Exercice 1 (Variables et types de données): Déclarez une variable feature contenant une valeur numérique représentant une caractéristique d'un jeu de données (exemple : longueur_sépale = 5.1).

Solution :

```
longueur_sépale = 5.1
print("La longueur du sépale est :", longueur_sépale)
```

Exercice 2 (Listes et DataFrames): Créez une liste Python pour stocker les longueurs des sépales des cinq premières fleurs dans le dataset Iris.

Solution :

```
longueurs_sépales = [5.1, 4.9, 4.7, 4.6, 5.0]
print("Longueurs des sépales :", longueurs_sépales
```

Exercice 3 (Conversion en DataFrame Pandas): Transformez cette liste en un DataFrame Pandas avec une colonne nommée Longueur_Sépale.

Solution :

```
import pandas as pd
df = pd.DataFrame(longueurs_sépales, columns=["Longueur_Sépale"])
print(df)
```

Partie 8 : Explorations des structures conditionnelles

Exercice 1 (Conditions sur des caractéristiques) : Écrivez un programme Python qui vérifie si une valeur de longueur de sépale est supérieure à 5 cm.

Solution :

```
longueur_sépale = 5.1
if longueur_sépale > 5:
print("Le sépale est long.")
else:
print("Le sépale est court.")
```

Exercice 2 (Filtrage de données dans un DataFrame) : Filtrez les fleurs du dataset Iris dont la longueur de sépale est supérieure à 5 cm.

Solution :

```
from sklearn.datasets import load_iris
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
print(df[df["sepal length (cm)"] > 5])
```

Partie 9 : Visualisation des données

Exercice 1 (Création de graphiques simples avec Matplotlib) : Représentez graphiquement la relation entre la longueur et la largeur des sépales dans le dataset Iris.

Solution :

```
import matplotlib.pyplot as plt
plt.scatter(df["sepal length (cm)"], df["sepal width (cm)"])
plt.xlabel("Longueur des sépales")
plt.ylabel("Largeur des sépales")
plt.title("Relation entre longueur et largeur des sépales")
plt.show()
```

Exercice 2 (Ajout de couleurs pour visualiser les classes) : Coloriez les points du graphique précédent selon les espèces de fleurs (classes dans Iris).

Solution :

```
plt.scatter(df["sepal length (cm)"], df["sepal width (cm)"], c=iris.target, cmap='viridis')

plt.xlabel("Longueur des sépales")
plt.ylabel("Largeur des sépales")
plt.title("Séparation des espèces dans Iris")
plt.show()
```

Partie 10 : Utilisation de fonctions

Exercice 1 (Calcul de moyenne) :

Écrivez une fonction appelée calculer_moyenne qui prend une liste de valeurs en entrée et retourne leur moyenne.

Solution :

```
def calculer_moyenne(liste):
    return sum(liste) / len(liste)
print("Moyenne des longueurs :", calculer_moyenne(longueurs_sépales))
```

Exercice 2 (Standardisation d'une caractéristique) :

Écrivez une fonction Python pour standardiser une caractéristique en utilisant la formule : $z=x-\mu\sigma$, où μ est la moyenne et σ l'écart-type.

Solution :

```
def standardiser(valeurs):
    moyenne = sum(valeurs) / len(valeurs)
    ecart_type = (sum([(x - moyenne)**2 for x in valeurs]) / len(valeurs))**0.5
    return [(x - moyenne) / ecart_type for x in valeurs]
print("Longueurs standardisées :", standardiser(longueurs_sépales))
```

Partie 11 : Applications spécifiques au Machine Learning

Exercice 1 (Utilisation d'un modèle simple) :

Divisez les données en un jeu d'entraînement et de test, puis entraînez un modèle de classification simple sur le dataset Iris.

Solution :

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2,
random_state=42)
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
print("Précision :", model.score(X_test, y_test))
```
