UNIVERSITÉ Larbi Ben Mhidi

Faculté des Sciences de la Terre, de la Géographie et de l'Aménagement du Territoire

Département Géographie et aménagement du territoire

Matière : Programmation

Niveau : 1ère année Licence Géographie et aménagement du territoire — Semestre 2

Crédits : 2 | **Coefficient :** 1

Enseignant : Bezzaz Soumia

Année universitaire : 2025–2026

Les Bibliothèques NumPy et Pandas

Python est devenu le langage de référence pour l'analyse de données et le calcul scientifique grâce à ses bibliothèques puissantes. Les deux bibliothèques essentielles sont :

- **NumPy** (Numerical Python) : pour le calcul numérique et les tableaux multidimensionnels
- **Pandas** : pour la manipulation et l'analyse de données structurées

Pourquoi utiliser ces bibliothèques en géographie et aménagement du territoire?

- Analyser des données géographiques (populations, températures, précipitations)
- Traiter des fichiers CSV (bases de données géographiques)
- Effectuer des calculs statistiques sur des séries temporelles
- Visualiser des tendances et des corrélations

NumPy : Calcul Numérique

1.1. Installation et importation

Installation (si nécessaire) :

```
pip install numpy
```

Importation standard :

```
import numpy as np  # Convention : toujours utiliser 'np'
```

1.2. Les tableaux NumPy (arrays)

Un **array NumPy** est une structure de données qui stocke des éléments de même type de manière efficace. C'est la base de NumPy.

Créer un tableau 1D (vecteur)

```
import numpy as np

# Températures mensuelles moyennes (°C)
temperatures = np.array([12, 14, 18, 22, 26, 30, 33, 32, 28, 22, 16,
13])
print(temperatures)
print(type(temperatures))  # <class 'numpy.ndarray'>
```

Créer un tableau 2D (matrice)

```
# Données de population (villes × années)
populations = np.array([
    [3000000, 3100000, 3200000],  # Alger
    [800000, 850000, 900000],     # Oran
    [450000, 470000, 500000]      # Constantine
])

print(populations.shape)  # (3, 3) → 3 lignes, 3 colonnes
```

1.3. Opérations mathématiques

NumPy permet d'effectuer des opérations mathématiques sur des tableaux entiers, élément par élément.

Opérations de base

```
temperatures = np.array([20, 22, 25, 28, 30])

# Ajouter 5°C à toutes les températures
print(temperatures + 5)  # [25 27 30 33 35]

# Convertir en Fahrenheit : F = C × 9/5 + 32
fahrenheit = temperatures * 9/5 + 32
print(fahrenheit)  # [68. 71.6 77. 82.4 86.]
```

1.4. Fonctions utiles de NumPy

Fonction	Description
`np.arange()`	Crée un tableau avec une séquence de nombres
`np.linspace()`	Crée un tableau avec n valeurs espacées uniformément

`np.zeros()`	Crée un tableau rempli de zéros
`np.ones()`	Crée un tableau rempli de uns
`np.where()`	Filtre les éléments selon une condition

Utilisation de ces fonctions

```
# Créer une séquence de 0 à 10 (exclu)
seq = np.arange(10)
print(seq)  # [0 1 2 3 4 5 6 7 8 9]

# Créer 5 valeurs entre 0 et 100
vals = np.linspace(0, 100, 5)
print(vals)  # [  0.  25.  50.  75. 100.]

# Filtrer les températures > 25°C
temperatures = np.array([20, 22, 25, 28, 30])
chaudes = temperatures[temperatures > 25]
print(chaudes)  # [28 30]
```

Pandas : Analyse de Données

Pandas est une bibliothèque Python pour manipuler et analyser des données structurées. Elle est construite au-dessus de NumPy et offre des structures de données très pratiques.

Installation :

```
pip install pandas
```

Importation standard :

```
import pandas as pd  # Convention : toujours utiliser 'pd'
```

2.1. Les Series

Une **Series** est un tableau 1D avec des index étiquetés. C'est comme une colonne d'un tableau Excel.

Créer une Series

```
import pandas as pd

# Populations des wilayas
```

```
populations = pd.Series([3000000, 850000, 500000],
                        index=['Alger', 'Oran', 'Constantine'])
print(populations)
```

Sortie :

```
Alger          3000000
Oran            850000
Constantine     500000
dtype: int64
```

2.2. Les DataFrame (tableaux de données)

Un **DataFrame** est un tableau 2D avec des lignes et des colonnes étiquetées. C'est l'équivalent d'un tableau Excel ou d'une table de base de données.

Créer un DataFrame

```
# Données climatiques de trois villes
data = {
    'Ville': ['Alger', 'Oran', 'Constantine'],
    'Population': [3000000, 850000, 500000],
    'Temperature_Moyenne': [18, 19, 16],
    'Precipitation_Annuelle': [680, 400, 530]
}

df = pd.DataFrame(data)
print(df)
```

Sortie :

```
          Ville  Population  Temperature_Moyenne  Precipitation_Annuelle
0        Alger     3000000                   18                     680
1         Oran      850000                   19                     400
2  Constantine      500000                   16                     530
```

2.3. Lire des fichiers CSV

Pandas excelle dans la lecture de fichiers CSV (Comma-Separated Values), format très utilisé en géographie.

Lire un fichier CSV

```
# Lire le fichier
df = pd.read_csv('stations_meteo.csv')
```

```
# Afficher les 5 premières lignes
print(df.head())

# Informations sur le DataFrame
print(df.info())
```

2.4. Sélectionner des données

Sélection de colonnes et lignes

```
# Sélectionner une colonne
print(df['Population'])

# Sélectionner plusieurs colonnes
print(df[['Ville', 'Temperature_Moyenne']])

# Filtrer les lignes selon une condition
villes_chaudes = df[df['Temperature_Moyenne'] > 17]
print(villes_chaudes)
```

2.5. Opérations sur les colonnes

Calculs sur colonnes

```
# Ajouter une nouvelle colonne calculée
df['Densite'] = df['Population'] / 1000  # hab/km² (exemple)

# Statistiques sur une colonne
print(f"Population moyenne : {df['Population'].mean()}")
print(f"Température max : {df['Temperature_Moyenne'].max()}°C")
```

2.6. Sauvegarder dans un fichier CSV

Exporter les données

```
# Sauvegarder le DataFrame dans un CSV
df.to_csv('resultats_analyse.csv', index=False)
```

Exemple: analyse climatique

Analyse complète de données météorologiques

```
import pandas as pd
import numpy as np
```

```python
# Créer des données de température mensuelle
mois = ['Jan', 'Fev', 'Mar', 'Avr', 'Mai', 'Jun',
        'Jul', 'Aou', 'Sep', 'Oct', 'Nov', 'Dec']
temp_alger = [12, 13, 15, 18, 21, 25, 28, 29, 26, 22, 17, 13]
temp_oran = [13, 14, 16, 19, 22, 26, 29, 30, 27, 23, 18, 14]

# Créer le DataFrame
df_meteo = pd.DataFrame({
    'Mois': mois,
    'Alger': temp_alger,
    'Oran': temp_oran
})

# Calculer la différence de température
df_meteo['Difference'] = df_meteo['Oran'] - df_meteo['Alger']

# Statistiques
print("=== ANALYSE CLIMATIQUE ===")
print(f"Température moyenne Alger :
{df_meteo['Alger'].mean():.1f}°C")
print(f"Température moyenne Oran : {df_meteo['Oran'].mean():.1f}°C")
print(f"Mois le plus chaud à Alger :
{df_meteo.loc[df_meteo['Alger'].idxmax(), 'Mois']}")

# Sauvegarder
df_meteo.to_csv('analyse_climatique.csv', index=False)
```

EXERCICES

Exercice 1

Créer un tableau NumPy avec les précipitations mensuelles [45, 38, 52, 60, 25, 10, 2, 5, 15, 42, 55, 48] et calculer la moyenne, le minimum et le maximum.

Solution

```
import numpy as np

precipitations = np.array([45, 38, 52, 60, 25, 10, 2, 5, 15, 42, 55, 48])

print(f"Moyenne : {np.mean(precipitations):.2f} mm")
print(f"Minimum : {np.min(precipitations)} mm")
print(f"Maximum : {np.max(precipitations)} mm")
```

Exercice 2

Créer un DataFrame avec les colonnes 'Ville', 'Latitude', 'Longitude' pour Alger (36.75, 3.04), Oran (35.70, -0.63) et Constantine (36.37, 6.61).

Solution

```
import pandas as pd

data = {
    'Ville': ['Alger', 'Oran', 'Constantine'],
    'Latitude': [36.75, 35.70, 36.37],
    'Longitude': [3.04, -0.63, 6.61]
}

df = pd.DataFrame(data)
print(df)
```

Exercice 3

À partir du DataFrame de l'Exercice 2, afficher uniquement les villes dont la latitude est supérieure à 36.

Solution

```
villes_nord = df[df['Latitude'] > 36]
```

```
print(villes_nord)
```

Conclusion

NumPy et Pandas sont des outils essentiels pour tout travail avec des données en Python :

• **NumPy** : calcul numérique rapide et efficace

• **Pandas** : manipulation et analyse de données structurées

Ces bibliothèques sont largement utilisées en géographie pour analyser des données climatiques, démographiques, économiques et bien d'autres types de données spatiales.