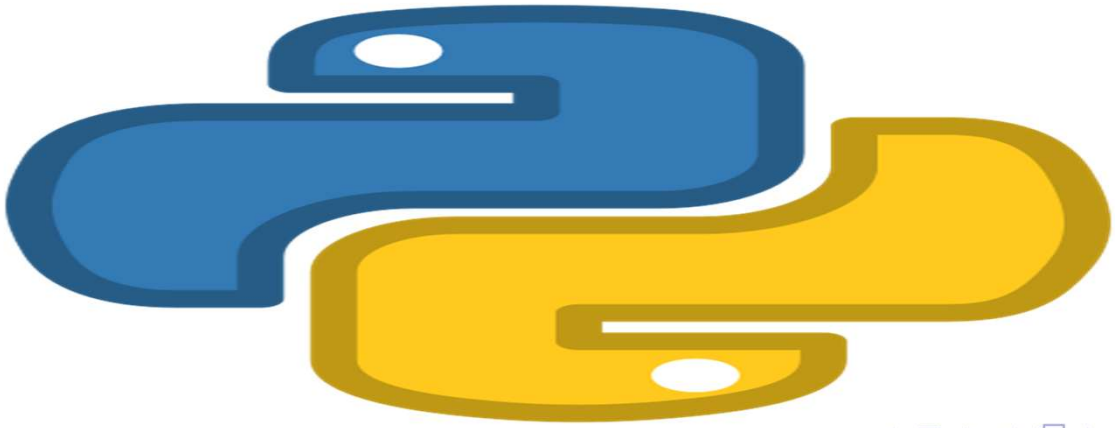
Python : Matplotlib

Mohammed OUANAN

m.ouanan@umi.ac.ma



Plan

Plan

Python

Matplotlib

- bibliothèque **Python** de visualisation de données
- son nom est dérivé de **MATLAB**, **plot** (en référence aux graphiques) **library**
- principalement utilisée pour créer des graphiques 2D variés, tels que :
  - Les histogrammes,
  - Les courbes,
  - Les graphiques en barres,
  - Les camemberts,
  - Les nuages de points,
  - ...

Python

Documentation officielle

```
https://matplotlib.org/
```

Python

Démarche

- Créez un répertoire `cours-matplotlib` dans votre espace de travail
- Lancez **VSC** et allez dans `File > Open Folder...` et choisissez `cours-pandas`
- Dans `cours-matplotlib`, créez un fichier `main.py`

Python

Pour installer matplotlib, lancez la commande

```
pip install numpy matplotlib
```

Python

Pour utiliser `matplotlib`, il faut importer `pyplot` (ici sous l'alias `plt`)

```
from matplotlib import pyplot as plt
```

Python

Pour utiliser `matplotlib`, il faut importer `pyplot` (ici sous l'alias `plt`)

```
from matplotlib import pyplot as plt
```

Python

Pour utiliser `matplotlib`, il faut importer `pyplot` (ici sous l'alias `plt`)

```
from matplotlib import pyplot as plt
```

Ou

```
import matplotlib.pyplot as plt
```

Python

Préparons les listes pour les deux axes (des abscisses et des ordonnées)

```
x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]
```

Python

Préparons les listes pour les deux axes (des abscisses et des ordonnées)

```
x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]
```

Pour créer le graphique

```
plt.plot(x, y)
```

Python

Préparons les listes pour les deux axes (des abscisses et des ordonnées)

```
x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]
```

Pour créer le graphique

```
plt.plot(x, y)
```

Pour afficher le graphique

```
plt.show()
```

Python

Commençons par importer `numpy`

```
import numpy as np
```

Python

Commençons par importer numpy

```
import numpy as np
```

Préparons les tableaux numpry

```
x = np.array([1, 2, 3, 4, 5])
y = np.array([10, 15, 7, 10, 20])
```

Python

Commençons par importer numpy

```
import numpy as np
```

Préparons les tableaux numpry

```
x = np.array([1, 2, 3, 4, 5])
y = np.array([10, 15, 7, 10, 20])
```

Même syntaxe pour créer et afficher le graphique

```
plt.plot(x, y)
plt.show()
```

Python

Pour ajouter un titre

```
plt.title('Graphique de ligne')
```

Python

Pour afficher plusieurs courbes dans le même graphique

```
from matplotlib import pyplot as plt
import numpy as np

x1 = [1, 2, 3, 4, 5]
y1 = [10, 15, 7, 10, 20]

x2 = np.array([ 1, 2.5, 3, 3.5, 4])
y2 = np.array([6, 2, 7, 11, 6])

plt.plot(x1, y1, x2, y2)
plt.title("Grphique")

plt.show()
```

Python

Pour définir plusieurs sous-graphiques

```
#plot 1 :
x1 = [1, 2, 3, 4, 5]
y1 = [10, 15, 7, 10, 20]

plt.subplot(1, 2, 1)
plt.plot(x1, y1)
plt.title("Graphique 1")

#plot 2 :
x2 = np.array([ 1, 2.5, 3, 3.5, 4])
y2 = np.array([6, 2, 7, 11, 6])

plt.subplot(1, 2, 2)
plt.plot(x2, y2)
plt.title("Grphique 2")

plt.show()
```

Python

Explication

- `plt.subplot(1, 2, 1)` signifie une figure divisée en 1 ligne et 2 colonnes, et sélectionne le premier emplacement pour `Graphique 1`.
- `plt.subplot(1, 2, 2)` sélectionne le deuxième emplacement dans cette même grille pour `Graphique 2`.

Python

Pour définir un super-titre pour tous les sous-graphiques

```
#plot 1 :
x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]

plt.subplot(1, 2, 1)
plt.plot(x,y)
plt.title("Graphique 1")

#plot 2 :
x = np.array([1, 2, 3, 4, 5])
y = np.array([10, 15, 7, 10, 20])

plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.title("Grphique 2")

plt.suptitle("Mon étude")
plt.show()
```

Python

Pour associer un label à chaque axe

```
from matplotlib import pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]

plt.plot(x, y)

plt.title('Graphique de ligne')


plt.xlabel('Nombre d\'étudiant')
plt.ylabel('Moyenne')

plt.show()
```

Python

Pour faciliter la lecture de la courbe, on peut ajouter une grille

```
from matplotlib import pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]

plt.plot(x, y)

plt.title('Graphique de ligne')


plt.xlabel('Nombre d\'étudiant')
plt.ylabel('Moyenne')

plt.grid()

plt.show()
```

Python

Pour choisir l'axe de la grille

```
from matplotlib import pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]

plt.plot(x, y)

plt.title('Graphique de ligne')


plt.xlabel('Nombre d\'étudiant')
plt.ylabel('Moyenne')

plt.grid(axis='x')

plt.show()
```

Python

Par défaut, la courbe est représentée par une ligne continue, mais on peut le changer

```
from matplotlib import pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]

plt.plot(x, y, linestyle = 'dotted')

plt.title('Graphique de ligne')


plt.xlabel('Nombre d\'étudiant')
plt.ylabel('Moyenne')

plt.grid()

plt.show()
```

Python

Ou les raccourcis

```
from matplotlib import pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]

plt.plot(x, y, ls=':')

plt.title('Graphique de ligne')


plt.xlabel('Nombre d\'étudiant')
plt.ylabel('Moyenne')

plt.grid()

plt.show()
```

Python

Autres options

- `dashed` ou `--`
- `solid` (par défaut) ou `-`
- `dashdot` ou `-.`
- `None` ou `''` ou `' '`

Python

Pour changer la couleur

```
from matplotlib import pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]

plt.plot(x, y, linestyle='dotted', color='red')

plt.title('Graphique de ligne')


plt.xlabel('Nombre d\'étudiant')
plt.ylabel('Moyenne')

plt.grid()

plt.show()
```

Python

Ou les raccourcis

```
from matplotlib import pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]

plt.plot(x, y, ls = ':', c='r')

plt.title('Graphique de ligne')


plt.xlabel('Nombre d\'étudiant')
plt.ylabel('Moyenne')

plt.grid()

plt.show()
```

Python

Options possibles pour la couleur

- 140 noms en anglais
  (Liste compléte : https://www.w3schools.com/colors/colors_names.asp)
- code hexadécimal

Python

Pour certaines couleurs, des raccourcis sont disponibles

- `r` pour `Red`
- `g` pour `Green`
- `b` pour `Blue`
- `c` pour `Cyan`
- `m` pour `Magenta`
- `y` pour `Yellow`
- `k` pour `Black`
- `w` pour `White`

Python

Pour changer l'épaisseur de la ligne de la courbe (l'unité est le `point`)

```
from matplotlib import pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]

plt.plot(x, y, linewidth='5.5')

plt.title('Graphique de ligne')


plt.xlabel('Nombre d\'étudiant')
plt.ylabel('Moyenne')

plt.grid()

plt.show()
```

Python

Ou le raccourci

```
from matplotlib import pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]

plt.plot(x, y, lw='5.5')

plt.title('Graphique de ligne')


plt.xlabel('Nombre d\'étudiant')
plt.ylabel('Moyenne')

plt.grid()

plt.show()
```

Python

Pour marquer les points de la courbe

```
from matplotlib import pyplot as plt
import numpy as np



x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]

plt.plot(x, y, marker='o')

plt.title('Graphique de ligne')


plt.xlabel('Nombre d\'étudiant')
plt.ylabel('Moyenne')

plt.grid()

plt.show()
```

Python

Autres marqueurs

- `s` pour `Square`
- `D` pour `Diamond`
- `p` pour `Pentagon`
- `.` pour `Point`
- `v` pour `Triangle`
- `*`
- `+`
- ...

Python

Pour modifier la taille du marqueur, on utilise `ms` (markersize)

```
from matplotlib import pyplot as plt
import numpy as np



x = [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 20]

plt.plot(x, y, marker='*', ms="20")

plt.title('Graphique de ligne')


plt.xlabel('Nombre d\'étudiant')
plt.ylabel('Moyenne')

plt.grid()

plt.show()
```

Python

Pour afficher que les points sans les lignes, on utilise `scatter` à la place de `plot`

```
from matplotlib import pyplot as plt
import numpy as np

x1 = [1, 2, 3, 4, 5]
y1 = [10, 15, 7, 10, 20]
plt.scatter(x1, y1)

plt.show()
```

Python

Pour changer les couleurs par défaut

```
from matplotlib import pyplot as plt
import numpy as np

x1 = [1, 2, 3, 4, 5]
y1 = [10, 15, 7, 10, 20]
plt.scatter(x1, y1, color="pink")

plt.show()
```

Python

Pour appliquer une couleur différente pour chaque point

```
from matplotlib import pyplot as plt
import numpy as np

x1 = [1, 2, 3, 4, 5]
y1 = [10, 15, 7, 10, 20]

colors = ["teal", "gold", "tomato", "skyblue", "hotpink"]

plt.scatter(x1, y1, color=colors)


plt.show()
```

Python

Pour représenter les données sous forme de barres

```
from matplotlib import pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([5, 6, 2, 10])

plt.bar(x, y)
plt.show()
```

Python

Pour afficher les barres horizontalement

```
from matplotlib import pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([5, 6, 2, 10])

plt.barh(x, y)
plt.show()
```

Python

Pour modifier la couleur

```
from matplotlib import pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([5, 6, 2, 10])

plt.barh(x, y, color="teal")
plt.show()
```

Python

Pour modifier la largeur d'une barre verticale

```
from matplotlib import pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([5, 6, 2, 10])

plt.bar(x, y, color="teal", width=0.3)
plt.show()
```

Python

Pour modifier la hauteur d'une barre horizontale

```
from matplotlib import pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([5, 6, 2, 10])

plt.barh(x, y, color="teal", height=0.3)
plt.show()
```

Python

Pour représenter les données sous forme d'un camembert

```
from matplotlib import pyplot as plt
import numpy as np

data = np.array([10, 15, 25, 20])

plt.pie(data)  # Créer le camembert
plt.show()
```

le terme **"camembert"** fait référence à un **graphique en secteurs** (ou **diagramme circulaire**), connu en anglais sous le nom de **"pie chart"**. Ce type de graphique est utilisé pour représenter des proportions ou des pourcentages sous forme de parts d'un cercle, comme les parts d'un camembert.

Python

Pour ajouter des labels aux différents fragments

```
from matplotlib import pyplot as plt
import numpy as np

data = np.array([10, 15, 25, 20])
strings = ["Fraises", "Bananes", "Mangues", "Tomates"]

plt.pie(data, labels=strings)
plt.show()
```

Python

Pour modifier les couleurs par défaut

```
from matplotlib import pyplot as plt
import numpy as np

data = np.array([10, 15, 25, 20])
strings = ["Fraises", "Bananes", "Mangues", "Tomates"]
couleurs = ["tomato", "gold", "teal", "#000000"]

plt.pie(data, labels=strings, colors=couleurs)
plt.show()
```

Python

Pour ajouter une légende

```
from matplotlib import pyplot as plt
import numpy as np

data = np.array([10, 15, 25, 20])
strings = ["Fraises", "Bananes", "Mangues", "Tomates"]
couleurs = ["tomato", "gold", "teal", "#000000"]

plt.pie(data, labels=strings, colors=couleurs)
plt.legend()
plt.show()
```

Python

Pour ajouter un titre à la légende

```
from matplotlib import pyplot as plt
import numpy as np

data = np.array([10, 15, 25, 20])
strings = ["Fraises", "Bananes", "Mangues", "Tomates"]
couleurs = ["tomato", "gold", "teal", "#000000"]

plt.pie(data, labels=strings, colors=couleurs)
plt.legend(title="Fruits")
plt.show()
```

Python

Pour ajouter de l'espace entre les différents fragments

```
from matplotlib import pyplot as plt
import numpy as np

data = np.array([10, 15, 25, 20])
strings = ["Fraises", "Bananes", "Mangues", "Tomates"]
couleurs = ["tomato", "gold", "teal", "#000000"]
explodes = [0.1, 0.2, 0, 0.1]

plt.pie(data, labels=strings, colors=couleurs, explode=explodes)
plt.legend(title="Fruits")
plt.show()
```

Python

Pour ajouter de l'ombre

```
from matplotlib import pyplot as plt
import numpy as np

data = np.array([10, 15, 25, 20])
strings = ["Fraises", "Bananes", "Mangues", "Tomates"]
couleurs = ["tomato", "gold", "teal", "#000000"]
explodes = [0.1, 0.2, 0, 0.1]

plt.pie(data, labels=strings, colors=couleurs, explode=explodes, shadow=True)
plt.legend(title="Fruits")
plt.show()
```

Python

Pour changer la position de la légende de l'ombre

```
from matplotlib import pyplot as plt
import numpy as np

data = np.array([10, 15, 25, 20])
strings = ["Fraises", "Bananes", "Mangues", "Tomates"]
couleurs = ["tomato", "gold", "teal", "#000000"]
explodes = [0.1, 0.2, 0, 0.1]

plt.pie(data, labels=strings, colors=couleurs, explode=explodes, shadow=True)
plt.legend(title="Fruits",  bbox_to_anchor=(1.3, 0.5))
plt.show()
```

Python

Pour changer la position de la le´gende de l'ombre

```
from matplotlib import pyplot as plt
import numpy as np

data = np.array([10, 15, 25, 20])
strings = ["Fraises", "Bananes", "Mangues", "Tomates"]
couleurs = ["tomato", "gold", "teal", "#000000"]
explodes = [0.1, 0.2, 0, 0.1]

plt.pie(data, labels=strings, colors=couleurs, explode=explodes, shadow=True)
plt.legend(title="Fruits",  bbox_to_anchor=(1.3, 0.5))
plt.show()
```

Informations sur le repère

- `(0, 0)` correspond au coin inférieur gauche de la zone de l'axe.
- `(1, 1)` correspond au coin supérieur droit de la zone de l'axe.

Python

Par défaut, l'angle de démarrage du camembert est l'axe des abscisses. Pour changer, on utilise `startangle`

```
from matplotlib import pyplot as plt
import numpy as np

data = np.array([10, 15, 25, 20])
strings = ["Fraises", "Bananes", "Mangues", "Tomates"]
couleurs = ["tomato", "gold", "teal", "#000000"]

plt.pie(data, labels=strings, colors=couleurs, startangle = 90)
plt.legend(title="Fruits",  bbox_to_anchor=(1.3, 0.5))

plt.show()
```

Python

Remarques

- Lorsqu'on utilise simplement `plt.show()`, **Matplotlib** ne génère pas de fichier.
- `plt.show()` ouvre une fenêtre d'affichage temporaire (ou affiche dans un **notebook Jupyter** si on est dans cet environnement) pour visualiser le graphique sans le sauvegarder.
- Pour sauvegarder les graphiques, on peut utiliser la fonction `savefig()`.

Python

Pour générer un fichier `JPEG` (Le type de fichier par défaut généré par plt.savefig() dans Matplotlib est un `PNG`)

```
from matplotlib import pyplot as plt
import numpy as np

data = np.array([10, 15, 25, 20])
strings = ["Fraises", "Bananes", "Mangues", "Tomates"]
couleurs = ["tomato", "gold", "teal", "#000000"]
explodes = [0.1, 0.2, 0, 0.1]

plt.pie(data, labels=strings, colors=couleurs, explode=explodes, shadow=True)
plt.legend(title="Fruits")
plt.savefig("mon_graphique.jpeg")
```

Python

Pour générer un fichier `PDF`

```
from matplotlib import pyplot as plt
import numpy as np

data = np.array([10, 15, 25, 20])
strings = ["Fraises", "Bananes", "Mangues", "Tomates"]
couleurs = ["tomato", "gold", "teal", "#000000"]
explodes = [0.1, 0.2, 0, 0.1]

plt.pie(data, labels=strings, colors=couleurs, explode=explodes, shadow=True)
plt.legend(title="Fruits")
plt.savefig("mon_graphique.PDF")
```

Python

Matplotlib supporte plusieurs extensions de fichier

- PNG (.png)
- PDF (.pdf)
- SVG (.svg)
- EPS (.eps)
- JPEG (.jpg ou .jpeg)
- TIFF (.tiff)
- RAW (.raw)

Différences clés

Critère	Matplotlib	Seaborn
Niveau d'abstraction	Bas niveau (plus de contrôle)	Haut niveau (plus simple et rapide)
Personnalisation	Très flexible, mais verbeux	Moins flexible, mais plus intuitif
Style par défaut	Style basique	Style moderne et attrayant
Intégration Pandas	Nécessite plus de code pour utiliser	Intégration native avec les DataFrames
Fonctionnalités	Graphiques généraux	Graphiques statistiques avancés
Apprentissage	Courbe d'apprentissage plus raide	Plus facile à apprendre et à utiliser

Python

```
.import seaborn as sns
import matplotlib.pyplot as plt

# Données
data = sns.load_dataset("tips")

# Graphique avec Seaborn
sns.boxplot(data=data, x="day", y="total_bill")

# Personnalisation avec Matplotlib
plt.title("Répartition des notes par jour")
plt.xlabel("Jour")
plt.ylabel("Montant total ($)")
plt.show()
```