Annexe TP 1 : Graphes et Python

1 Généralités

Le module `matplotlib` propose de nombreuses fonctionnalités en lien avec les graphes.
Ces graphes peuvent prendre différentes formes : Diagramme circulaire (camembert), Diagramme en bâtons, Diagramme en bâtons empilés, ...
L'ajout de texte est également possible.
Plusieurs graphes peuvent être réalisés sur une même figure.
L'export en png s'obtient avec `plt.savefig(fichier_png)`.
L'affichage de la figure avec `plt.show()`.

2 Simple diagramme circulaire (en camembert) avec matplotlib

Avec `matplotlib`, nous pouvons facilement tracer un simple diagramme circulaire (`Pie Chart`), exemple :

```
import matplotlib.pyplot as plt

labels = 'Allemagne', 'France', 'Belgique', 'Espagne'
sizes = [15, 80, 45, 40]
colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']

plt.pie(sizes, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')

plt.savefig('PieChart01.png')
plt.show()
```

Autre exemple en mettant en évidence une partie du diagramme avec `explode` :

```
import matplotlib.pyplot as plt

labels = 'Allemagne', 'France', 'Belgique', 'Espagne'
sizes = [15, 80, 45, 40]
colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']
explode = (0, 0.1, 0, 0)

plt.pie(sizes, explode=explode, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')

plt.savefig('PieChart02.png')
plt.show()
```

3 Comment tracer un diagramme en bâtons avec Matplotlib ?

Exemples de comment tracer un diagramme en bâtons avec `Matplotlib` et la fonction `bar` du module `pyplot`.
Tracer un diagramme en bâtons avec `Matplotlib`

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

fig = plt.figure()

x = [1,2,3,4,5,6,7,8,9,10]
height = [8,12,8,5,4,3,2,1,0,0]
width = 1.0

plt.bar(x, height, width, color='b' )

plt.savefig('SimpleBar.png')
plt.show(
```

Nous pouvons ensuite modifier le style.

```
width = 0.05 # modifier la largeur des bâtons
plt.xlim(0,11) # Modifier les limites sur x
plt.ylim(0,14) # Modifier les limites sur y
pylab.xticks(x, BarName, rotation=40) # ajouter des labels aux bâtons
plt.scatter([i+width/2.0 for i in x],height,color='k',s=40) # ajouter un cercle au sommet du bâton
etc
```

Comment tracer un diagramme en bâtons avec Matplotlib ?

```
import matplotlib.pyplot as plt
import numpy as np
import pylab

fig = plt.figure()

x = [1,2,3,4,5,6,7,8,9,10]
height = [8,12,8,5,4,3,2,1,2,4]
width = 0.05
BarName = ['a','b','c','d','e','f','g','h','i','j']

plt.bar(x, height, width, color=(0.65098041296005249, 0.80784314870834351, 0.89019608497619629, 1.0)
plt.scatter([i+width/2.0 for i in x],height,color='k',s=40)

plt.xlim(0,11)
plt.ylim(0,14)
plt.grid()

plt.ylabel('Counts')
plt.title('Diagramme en Batons !')

pylab.xticks(x, BarName, rotation=40)

plt.savefig('SimpleBar.png')
plt.show()
```

4 Empiler les diagrammes à barre Matplotlib

Nous générons des diagrammes à barres dans `Matplotlib` en utilisant la méthode `matplotlib.pyplot.bar()`.

Pour empiler le diagramme en barres d'un certain ensemble de données sur un autre, nous ajoutons tous les ensembles de données que nous devons empiler et nous passons la somme comme paramètre `bottom` à la méthode `bar()`.

```
import matplotlib.pyplot as plt
data1=[30,20,10,0,0]
data2=[20,20,20,20,0]
data3=[50,60,70,80,100]

year=["2015","2016","2017","2018","2019"]

fig,ax=plt.subplots(3,1,figsize=(10,8))

ax[0].bar(year,data1,color="red")
ax[0].legend(["C++"])
ax[1].bar(year,data2,color="yellow")
ax[1].legend(["JavaScript"])
ax[2].bar(year,data3,color="green")
ax[2].legend(["Python"])

plt.show()
```

Ici, nous avons trois histogrammes distincts qui représentent la préférence d'un langage de programmation pour les employés d'une entreprise sur cinq ans. Nous discuterons des moyens d'empiler les diagrammes à barres d'un langage sur un autre et étudierons le choix global des langages de programmation au fil des ans avec un seul diagramme à barres.

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

data1=[30,20,10,0,0]
data2=[20,20,20,20,0]
data3=[50,60,70,80,100]

year=["2015","2016","2017","2018","2019"]

plt.figure(figsize=(9,7))
plt.bar(year,data3,color="green",label="Python")
plt.bar(year,data2,color="yellow",bottom=np.array(data3),label="JavaScript")
plt.bar(year,data1,color="red",bottom=np.array(data3)+np.array(data2),label="C++")

plt.legend(loc="lower left",bbox_to_anchor=(0.8,1.0))
plt.show()
```

Il empile les différents bargraphes les uns sur les autres.

Dans le diagramme, nous traçons d'abord les `data3` comme des données Python, qui servent de base pour les autres barres, puis nous traçons la barre des `data2`, et la barre de base des `data3` comme base pour la barre des `data2`. Pour empiler la barre de `data2` sur `data3`, nous définissons `bottom=np.array(data3)`.

De même, en traçant la barre pour `data1`, nous utilisons le tracé de la barre de `data2` et `data3` comme base. Pour ce faire, nous définissons `bottom=np.array(data3)+np.array(data2)` tout en traçant la barre de `data1`.

Un point important à noter est que nous devons utiliser des tableaux `NumPy` pour ajouter les données pour le paramètre `bottom`. Si nous mettons `bottom=data3+data2`, il créera par liste en ajoutant les éléments de `data2` à la fin de la liste `data3`.

Si nous ne souhaitons pas utiliser les tableaux `NumPy`, nous pouvons utiliser la compréhension de liste pour ajouter les éléments correspondants de la liste.

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

data1=[30,20,10,0,0]
data2=[20,20,20,20,0]
data3=[50,60,70,80,100]
```

```
year=["2015","2016","2017","2018","2019"]

plt.figure(figsize=(9,7))
plt.bar(year,data3,color="green",label="Python")
plt.bar(year,data2,color="yellow",bottom=data3,label="JavaScript")
plt.bar(year,data1,color="red",bottom=[sum(data) for data in zip(data2,data3)],label="C++")

plt.legend(loc="lower left",bbox_to_anchor=(0.8,1.0))
plt.show()
```