Chapitre 4 :
Courbes et données numériques

I Représentation graphique

Pour faire des représentations graphiques, on utilise le sous-module pyplot du module matplotlib. On commencera donc par l'importer en abrégeant son nom, avec l'alias usuel plt.

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

1.1 Tracés de courbes

Pour effectuer un tracé de courbe, on utilise ensuite la fonction plot du module matplotlib.pyplot, qui prend en argument le tableau des abscisses et le tableau des ordonnées des points à tracer.

1. Pour tracer une courbe sur un intervalle $[a,b]$, on commence donc par définir un tableau x contenant les abscisses des points à tracer.

   - Utilisation de numpy :
     La fonction linspace du module numpy permet de créer des points uniformément répartis sur un intervalle. Plus précisément, pour créer un tableau contenant n points régulièrement répartis dans l'intervalle $[a,b]$, on utilise la commande numpy.linspace(a,b,n) (après avoir importé le module numpy).

     ```
     >>> import numpy as np
     >>> x=np.linspace(1,10,19)
     >>> print(x)
     [ 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. ,
            5.5, 6. , 6.5, 7. , 7.5, 8. , 8.5, 9. , 9.5, 10. ]
     ```

   - Utilisation des listes définies en compréhension :

     On n'utilise plus numpy mais les listes définies en compréhension.

     ```
     >>> x=[1+0.5*i for i in range(19)]
     >>> print(x)
     [ 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. ,
            5.5, 6. , 6.5, 7. , 7.5, 8. , 8.5, 9. , 9.5, 10. ]
     ```

2. On va construire maintenant le tableau y des ordonnées.

   - Utilisation de numpy :
     La commande $y = f(x)$ permet ensuite de créer le tableau y des ordonnées des points du graphe de f d'abscisses contenus dans x.

     La fonction f ne doit pas être définie avec les fonctions du module math car x a un type propre à numpy. Elle doit être définie à l'aide des fonctions du module numpy.

```
>>> x=np.linspace(-np.pi,np.pi,13)
>>> x
array([-3.14159265, -2.61799388, -2.0943951 , -1.57079633, -1.04719755,
       -0.52359878, 0. , 0.52359878, 1.04719755, 1.57079633,
       2.0943951 , 2.61799388, 3.14159265])
>>> y=np.sin(x)
>>> y
array([ -1.22464680e-16, -5.00000000e-01, -8.66025404e-01,
        -1.00000000e+00, -8.66025404e-01, -5.00000000e-01,
        0.00000000e+00, 5.00000000e-01, 8.66025404e-01,
        1.00000000e+00, 8.66025404e-01, 5.00000000e-01,
        1.22464680e-16])
>>> import math
>>> z=math.sin(x)
Traceback (most recent call last) :
  File "<console>", line 1, in <module>
TypeError : only length-1 arrays can be converted to Python scalars
```

- Utilisation des listes définies en compréhension :

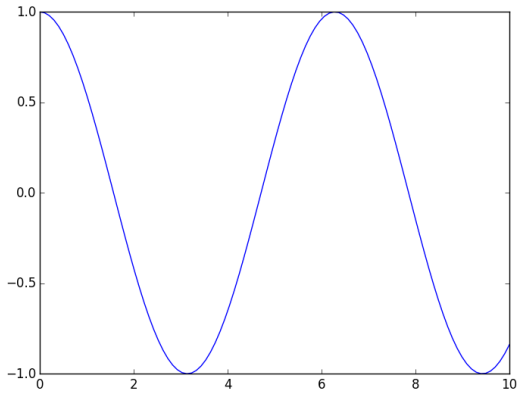
```
>>> from math import *
>>> x=np.linspace(-pi,pi,13)
>>> y=[cos(i) for i in x]
>>> y
[-1.0, -0.8660254037844387, -0.5000000000000002, 6.123233995736766e-17
, 0.4999999999999999, 0.8660254037844385, 1.0, 0.8660254037844387,
 0.5000000000000003, 6.123233995736766e-17, -0.4999999999999994,
 -0.8660254037844385, -1.0]
```

On peut également utiliser une fonction que l'on a programmée.

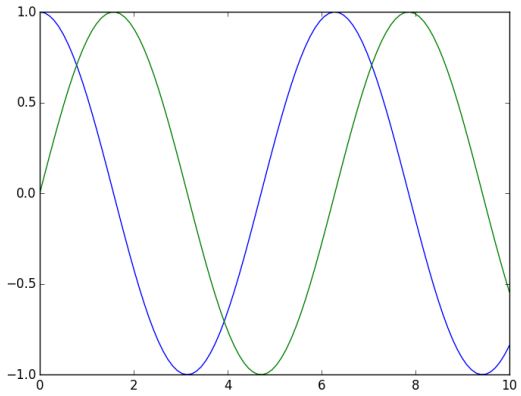
```
def f(x) :
    return x**2+x-1

>>> x=np.linspace(-1,1,20)
>>> y=[f(i) for i in x]
```

3. Le tracé de la fonction s'effectue alors avec la commande plt.plot(x,y).
On utilise la commande plt.show() pour afficher le graphique qui apparaîtra alors dans une fenêtre extérieure.

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> x=np.linspace(0,10,100)
>>> plt.plot(x,np.cos(x))
[<matplotlib.lines.Line2D object at 0x00000000053CD390>]
>>> plt.show()
```



```
>>> plt.plot(x,np.cos(x))
[<matplotlib.lines.Line2D object at 0x0000000008333B00>]
>>> plt.plot(x,np.sin(x))
[<matplotlib.lines.Line2D object at 0x0000000008339390>]
>>> plt.show()
```



4. Il existe de nombreuses options concernant les graphiques :
   - **plt.title('Titre') :** ajout d'un titre.
   - **plt.xlabel('axe des abscisses') :** ajout d'un nom à l'axe des abscisses.
   - **plt.ylabel('axe des ordonnées') :** ajout d'un nom à l'axe des ordonnées.
   - **plt.plot(x,y,label='nom')** suivi de **plt.legend() :** ajout d'une légende.
   - Les options suivantes doivent être remplacer la chaîne de caractères 'ici' :

plt.plot(x,y,'ici')

Style de ligne

-	ligne continue
- -	tirets
:	ligne en pointillé
-.	tirets points
	pas de ligne

Style de points

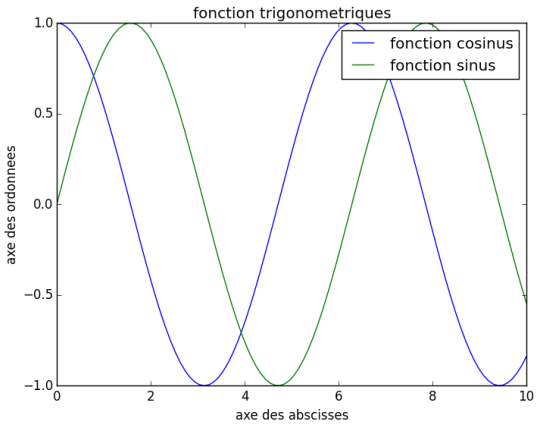
.	points
o	cercles
s	carrés (square)
x	croix
...	...

Couleurs

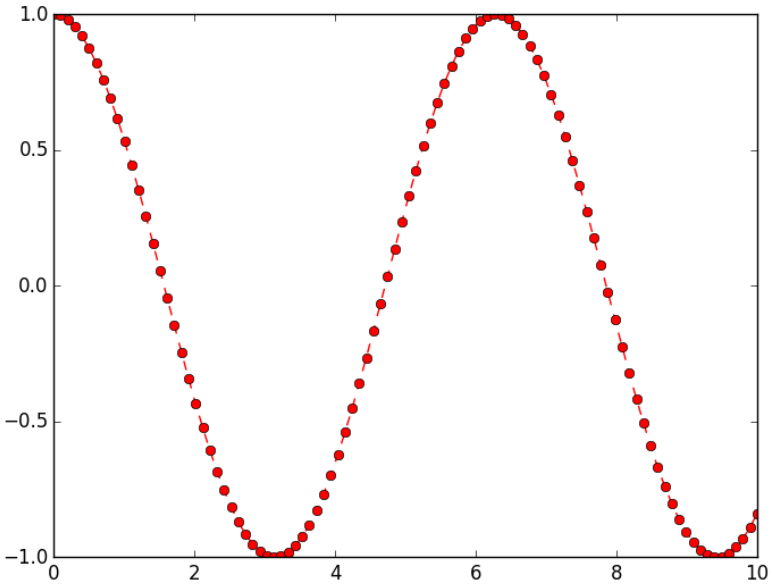
b	bleu
g	vert
r	rouge
k	noir
...	...

Ces instructions sont à placer avant plt.show().

```
>>> plt.plot(x,np.cos(x),label="fonction cosinus")
[<matplotlib.lines.Line2D object at 0x00000000057757F0>]
>>> plt.plot(x,np.sin(x),label="fonction sinus")
[<matplotlib.lines.Line2D object at 0x000000000551F358>]
>>> plt.legend()
<matplotlib.legend.Legend object at 0x000000000577D3C8>
>>> plt.title("fonction trigonometriques")
<matplotlib.text.Text object at 0x00000000057574E0>
>>> plt.xlabel("axe des abscisses")
<matplotlib.text.Text object at 0x00000000053B9898>
>>> plt.ylabel("axe des ordonnees")
<matplotlib.text.Text object at 0x000000000573EDD8>
>>> plt.show()
```



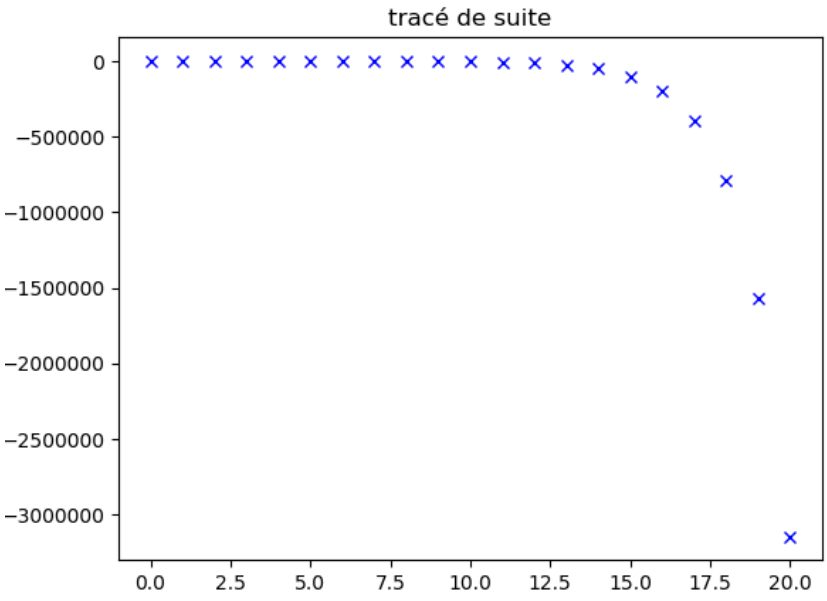

```
>>> plt.plot(x,np.cos(x),'r–o')
[<matplotlib.lines.Line2D at 0x9834d68>]
>>> plt.show()
```



1.2 Tracés de suites

Pour tracer des suites, on procède comme pour les fonctions mais on préfère utiliser une liste, plutôt qu'un range, pour définir les abscisses.

```
def liste_suite(n) :
    u=1
    l=[u]
    for i in range(1,n+1) :
        u=2*u-4
        l.append(u)
    return l

>>> x=[i for i in range(21)]
>>> y=liste_suite(20)
>>> plt.plot(x,y,'xb')
>>> plt.title('tracé de suite')
>>> plt.show()
```



1.3 Tracés d'histogrammes

On considère une série de données numériques stockées dans une liste l. On souhaite représenter cette série dans un histogramme. La commande est :

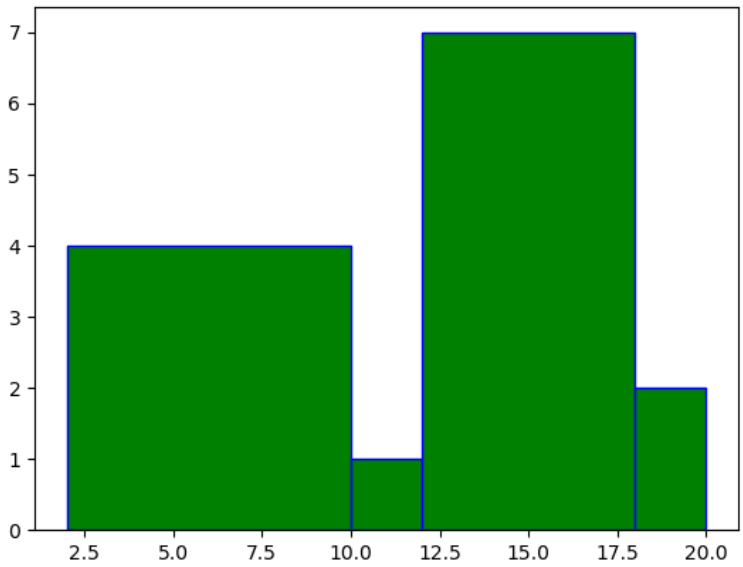
```
plt.hist(l,bins=[x1,x2,...,xn],color='.',edgecolor='.')
```

avec :

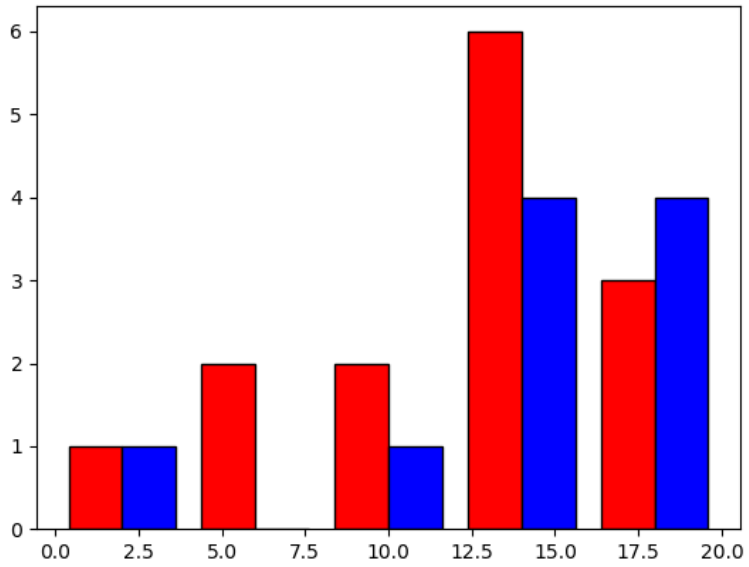
- l'argument de bins représente les classes de l'histogramme : bins=[x1,x2,...,xn] signifie que la première classe est $[x1, x2[$, la seconde est $[x2, x3[$, ...
- l'argument color correspond à la couleur des barres, sa valeur est identique à celle vue pour les courbes,
- l'argument edgecolor correspond à la couleur des bordures, sa valeur est identique à celle vue pour les courbes.

Pour tracer sur un même histogramme plusieurs séries de données, on utilise des listes.

```
>>> l1=[10,12,15,13.5,4,14,2,8,5,18,12,13,16,19]
>>> l2=[15,14,20,3,8,16,17,16,15,13]
>>> plt.hist(l1,bins=[2,10,12,18,20],color='g',edgecolor='b')
>>> plt.show()
```



```
>>> plt.hist([l1,l2],bins=[4*i for i in range(0,6)],color=['r','b'],edgecolor='k')
>>> plt.show()
```



II Manipulation de fichiers

On va travailler sur un fichier contenant des données numériques. On suppose que le fichier est placé dans un répertoire et on se placera dans ce répertoire. Pour cela, on utilise le File Browser de Pyzo : on choisit le répertoire dans lequel le fichier est stocké et on clique sur "Go to this directory in the current shell". Le fichier sera au format CSV (Comma Separated Values) qui est un format texte qui permet de représenter des données de tableur. Chaque ligne du texte correspond à une ligne du tableau et les colonnes sont séparées par des virgules.

On utilisera les commandes suivantes :

- open : ouverture du fichier. On précise un argument qui peut être :
  - 'r' : pour une ouverture en lecture (read).
  - 'w' : pour une ouverture en écriture (write). Dans ce cas, à chaque ouverture, le fichier est écrasé. Si le fichier n'existe pas, il sera automatiquement créé.
  - 'a' : pour une ouverture en mode ajout à la fin du fichier (append). Si le fichier n'existe pas, il sera automatiquement créé.

  On peut également préciser l'encodage afin de ne pas avoir de problème avec les caractères spéciaux.
- read : lecture du fichier. Les informations sont copiées dans une variable.
- close : fermeture du fichier. Cette commande permet de libérer les ressources dont on ne se sert plus.
- readline : lecture d'une ligne du fichier.
- readlines : création d'une liste contenant toutes les lignes du fichier.
- split : extraction des différents champs d'une ligne via des séparateurs.
- write : écriture à la fin d'un fichier.

On considère le fichier arrondissements.csv contenant des infirmations sur les arrondissements parisiens :

	A	B	C	D	E	F	G	H
1	n_sq_ar	c_ar	c_arinsee	l_ar	l_aroff	n_sq_co	surface	perimetre
2	750000005	5	75105	5ème Ardt	Panthéon	750001537	2539374.62284532	6239.19539614
3	750000003	3	75103	3ème Ardt	Temple	750001537	1170882.82818778	4519.26364836
4	750000010	10	75110	10ème Ardt	Entrepôt	750001537	2891739.44162064	6739.37505466
5	750000017	17	75117	17ème Ardt	Batignolles-Monceau	750001537	5668834.50445393	10775.579516
6	750000013	13	75113	13ème Ardt	Gobelins	750001537	7149311.09107136	11546.5465264
7	750000018	18	75118	18ème Ardt	Buttes-Montmartre	750001537	5996051.30811905	9916.46417634
8	750000008	8	75108	8ème Ardt	Élysée	750001537	3880036.39704363	7880.53326819
9	750000004	4	75104	4ème Ardt	Hôtel-de-Ville	750001537	1600585.63150251	5420.90843368
10	750000020	20	75120	20ème Ardt	Ménilmontant	750001537	5983446.03718297	10704.9404863
11	750000012	12	75112	12ème Ardt	Reuilly	750001537	16314782.6372674	24089.6662978
12	750000016	16	75116	16ème Ardt	Passy	750001537	16372542.1289739	17416.1096565
13	750000014	14	75114	14ème Ardt	Observatoire	750001537	5614877.30907921	10317.4833099
14	750000002	2	75102	2ème Ardt	Bourse	750001537	991153.74474596	4554.10435957
15	750000006	6	75106	6ème Ardt	Luxembourg	750001537	2153095.58639283	6483.68678565
16	750000009	9	75109	9ème Ardt	Opéra	750001537	2178303.27487137	6471.5882901
17	750000001	1	75101	1er Ardt	Louvre	750001537	1824612.86048666	6054.93686218
18	750000007	7	75107	7ème Ardt	Palais-Bourbon	750001537	4090057.18546976	8099.42488348
19	750000011	11	75111	11ème Ardt	Popincourt	750001537	3665441.55248808	8282.01188584
20	750000015	15	75115	15ème Ardt	Vaugirard	750001537	8494994.08101075	13678.7983149
21	750000019	19	75119	19ème Ardt	Buttes-Chaumont	750001537	6792651.12902648	11253.1824786
22								
23								

- On ouvre et on lit le fichier :

```
>>> fic=open('arrondissements.csv','r',encoding='utf-8')
>>> f=fic.read()
>>> f
'n _ sq_ar,c_ ar,c_ arinsee,l_ ar,l_ aroff,n_ sq_ co,surface,perimetre \ n
 750000005,5,7510 5,5ème Ardt,Panthéon,750001537,2539374.62284532,
6239.19539614 \ n 750000003,3,75103 ,3ème Ardt,Temple,750001537,
1170882.82818778,4519.26364836 \ n 750000010,10,75110,10ème Ardt,...
```

L'argument encoding='utf-8' donne l'encodage des données textuelles. Il permet de ne pas avoir de problème avec les caractères spéciaux comme les lettres accentuées.
La variable fic a été vidée lors de la lecture. Si on fait à nouveau une lecture, le résultat sera vide.
On remarque que le type de la variable f est une chaine de caractères (délimitée par ' ou "). Ce type sera étudié dans un autre chapitre.

- On extrait les données sous forme de listes :

```
>>> t=f.split('\n')
>>> t
['n _ sq_ar,c_ ar,c_ arinsee,l_ ar,l_ aroff,n_ sq_ co,surface,perimetre' ,
'750000005,5,7 5105,5ème Ardt,Panthéon,750001537,2539374.62284532,
6239.19539614', '750000003,3, 75103,3ème Ardt,Temple,750001537,
1170882.82818778,4519.26364836', '750000010,10, 75110,10ème Ardt,...
```

t est la liste dont les éléments sont les lignes de f sous forme de chaines de caractères. Le séparateur '\n' correspond au saut de ligne.

```
>>> L=[t[i].split(',') for i in range(1,len(t)-1)]
>>> L
[['750000005', '5', '75105', '5ème Ardt', 'Panthéon', '750001537',
 '2539374.6228 4532', '6239.19539614'], ['750000003', '3', '75103', '3ème Ardt',
 'Temple', '750 001537', '1170882.82818778', '4519.26364836'], ['750000010', '10',
 '75110', '10è me Ardt',...
```

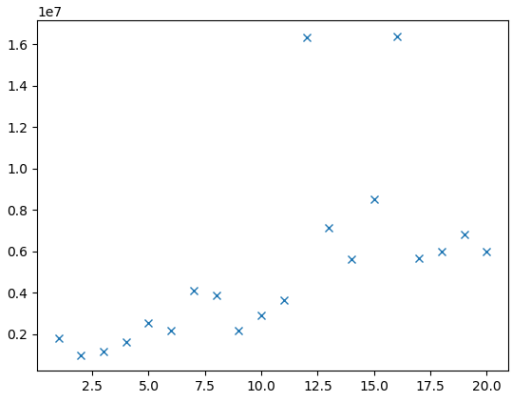
t est la liste dont les éléments sont les lignes de f sous forme de listes dont les éléments sont des chaines de caractères.

```
>>> surface=[float(L[i][6]) for i in range(len(L))]
>>> surface
[2539374.62284532, 1170882.82818778, 2891739.44162064, 5668834.50445393,
 7149311.09107136, 5996051.30811905, 3880036.39704363, 1600585.63150251,
 5983446.03718297, 16314782.6372674, 16372542.1289739, 5614877.30907921,
 991153.74474596, 2153095.58639283, 2178303.27487137, 1824612.86048666,
 4090057.18546976, 3665441.55248808, 8494994.08101075, 6792651.12902648]
>>> arrondissement=[int(L[i][1]) for i in range(len(L))]
>>> arrondissement
[5, 3, 10, 17, 13, 18, 8, 4, 20, 12, 16, 14, 2, 6, 9, 1, 7, 11, 15, 19]
```

On a créer deux listes de nombres contenant les informations de la 6ème colonne (surface de l'arrondissement) et de la 1ère colonne (numéro de l'arrondissement).

- On peut utiliser ces données pour tracer des graphiques comme vu précédemment.

```
>>> plt.plot(arrondissement,surface,'x')
>>> plt.show()
```




```
>>> plt.hist(surface,bins=[0,10**6,2*10**6,4*10**6,8*10**6, 10**7,2*10**7],
edgecolor='k')
>>> plt.show()
```