Cours

2 Outils graphiques

Capacités exigibles

- Utiliser les fonctions de base de la bibliothèque **matplotlib** pour représenter un nuage de points
- Utiliser les fonctions de base de la bibliothèque **matplotlib** pour tracer la courbe représentative d'une fonction
- Utiliser les fonctions de base de la bibliothèque **matplotlib** pour tracer une courbe plane paramétrée

A Représentation graphique d'un nuage de points

Comment faire ?

Utiliser l'instruction `plt.scatter(X,Y,marker='+')` où les **listes `X`** et **`Y`** regroupent respectivement les valeurs d'abscisses et les valeurs d'ordonnées correspondantes classées dans le même ordre.

Il est bien évident que **ces listes doivent avoir la même taille** (même nombre d'éléments).

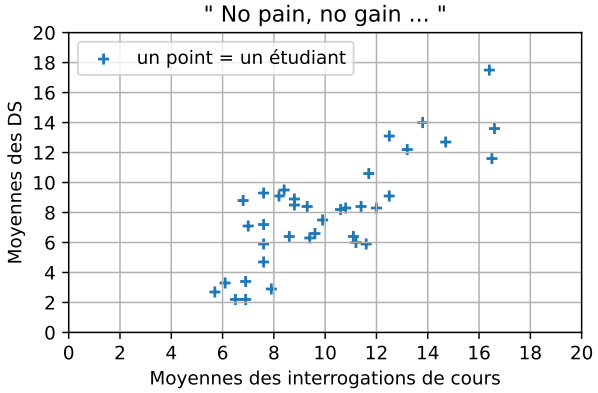
R *Plutôt que des listes, on peut aussi utiliser des tableaux NumPy à une dimension ...*

Exemple : corrélation entre interrogations de cours et réussite aux DS

À l'issue du 1er semestre, il peut être intéressant de représenter les moyennes individuelles des devoirs surveillés en fonction des moyennes individuelles obtenues en interrogation de cours :

```
# ===== IMPORTATION DES BIBLIOTHEQUES ==================
import matplotlib.pyplot as plt

# ===== SAISIE DES DONNEES ==================

# Moyennes des interrogations de cours
X = [5.7,7.6,7.6,9.3,13.2,11.7,6.9,9.4,11.1,11.2,6.8,9.6,11.4,
               8.8,13.8,12.0,8.8,7.0,6.1,8.2,14.7,6.5,7.6,16.6,6.9,12.5,
               8.6,12.5,8.4,11.6,16.5,7.6,16.4,7.9,9.9,10.6,10.8]

# Moyennes des devoirs surveillés
Y = [2.7,9.3,7.2,8.4,12.2,10.6,3.4,6.3,6.4,6.0,8.8,6.6,8.4,8.9,14.0,
            8.3,8.5,7.1,3.3,9.1,12.7,2.2,4.7,13.6,2.2,9.1,6.4,13.1,9.5,
            5.9,11.6,5.9,17.5,2.9,7.5,8.2,8.3]

# ===== TRACE DU NUAGE DE POINTS ==================

plt.scatter(X,Y,marker='+',label='un point = un étudiant')
plt.ylabel('Moyennes des DS')
plt.xlabel('Moyennes des interrogations de cours')
plt.legend() # pour afficher la légende
plt.xlim(0,20) # pour déterminer les valeurs extrêmes d'abscisses
plt.ylim(0,20)# pour déterminer les valeurs extrêmes d'ordonnées
plt.grid() # pour faire apparaître un quadrillage
plt.title('" No pain, no gain ... "')  # choix d'un titre
plt.show() # pour faire apparaître le graphique
```

On obtient la représentation graphique ci-contre.



Conclusion ? Si on apprend régulièrement son cours, on a plutôt tendance à mieux réussir en devoir surveillé.

*(Rappel : il n'est toutefois pas suffisant de bien apprendre son cours, **il faut aussi retravailler efficacement les exercices déjà effectués** ...)*

N. Gaudouen

B Tracé graphique d'une fonction

Comment faire ?

1. Définir la fonction **f** dont on souhaite donner une représentation graphique :

```
1 def f(x,...):
2     return    # préciser la valeur à renvoyer, dépendant de celle de x
```

R *Les pointillés ... représentent d'autres arguments éventuels de la fonction.*

2. Définir un tableau numpy **X** contenant des valeurs d'abscisses avec l'instruction :
X = np.linspace(*valeur de début, valeur de fin, nombre de valeurs***)**

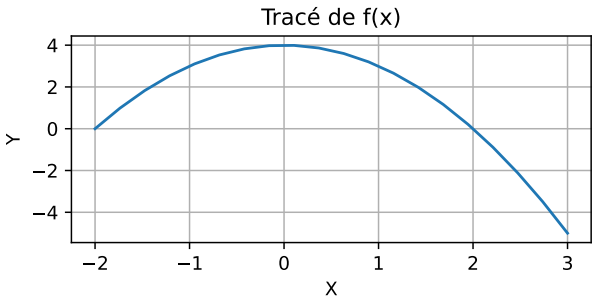
3. Appliquer la fonction **f** à **X** avec l'instruction **Y = f(X,...)**.
On obtient ainsi un tableau numpy **Y** regroupant les valeurs d'ordonnnées correspondant aux différentes abscisses et classées dans le même ordre.

4. Utiliser l'instruction **plt.plot(X,Y)** pour obtenir le tracé.

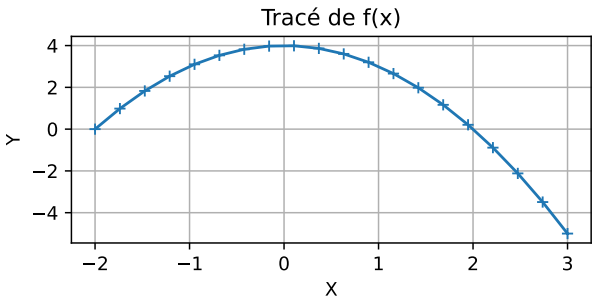
Exemple : tracé de $f(x) = 4 - x^2$

```
1  import numpy as np
2
3  # Définir la fonction à tracer
4  def f(x):
5      return 4-x**2
6
7  X = np.linspace(-2,3,100)  # 100 valeurs en abscisse entre -2 et 3
8  Y = f(X)  # les 100 valeurs en ordonnées correspondantes
9
10 # Tracé
11 plt.plot(X,Y)
12 plt.grid()
13 plt.xlabel("X")
14 plt.ylabel("Y")
15 plt.title("Tracé de f(x)")
16 plt.show()
```

On obtient la représentation graphique ci-contre.



Si on veut faire apparaître les points, on peut remplacer la ligne 11 par celle-ci : **plt.plot(X,Y,marker='+')**.



R *Au lieu d'utiliser* ***plt.plot()****, on utilisera* ***plt.semilogx()*** *pour une obtenir échelle logarithmique en abscisse, ou bien* ***plt.semilogy()*** *pour obtenir une échelle logarithmique en ordonnée ...*

C Tracé d'une courbe plane paramétrée

On parle de **courbe paramétrée** lorsque les abscisses et les ordonnées du tracé à effectuer dépendent de la valeur d'un *paramètre* t. En physique, très souvent, ce paramètre sera la variable de temps. Pour le tracé, on précision deux fonctions $x(t)$ et $y(t)$ représentant respectivement l'abscisse et l'ordonnée de la courbe :

$$\begin{cases} x(t) &= \dots \\ y(t) &= \dots \end{cases}$$

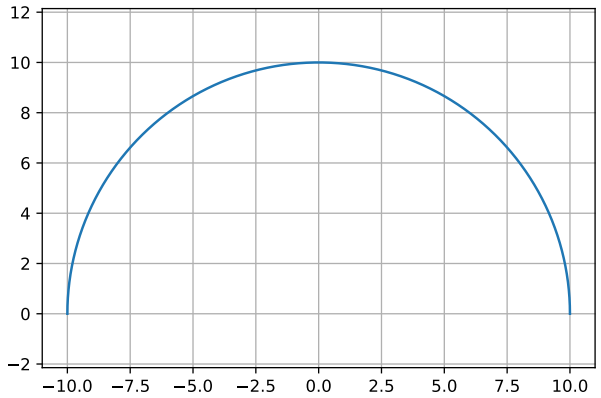
Comment faire ?

1. Définir un tableau numpy `t` contenant les valeurs prises successivement par le paramètre t.
   On se servira de l'instruction : **`t = np.linspace(`***valeur de début, valeur de fin, nombre de valeurs***`)`**
2. Calculer les valeurs de $x(t)$ et $y(t)$. Si besoin, définir des fonctions comme cela a été vu à en **B**.
3. Tracer y en fonction de x comme cela a été vu en **B**.

Exemple : tracé d'un demi-cercle

Ici, plutôt que le temps t, le paramètre est θ, position angulaire par rapport à l'axe Ox *(penser au cercle trigonométrique...)*. Pour tracer un demi-cercle, on peut faire varier θ de 0 à π par exemple :

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

# Rayon du cercle
R = 10 # par exemple ...

# Valeurs prises par le paramètre
theta = np.linspace(0, np.pi, 100)

# Calcul des abscisses et ordonnées
x = R*np.cos(theta)
y = R*np.sin(theta)

# Tracé
plt.plot(x,y)
plt.grid()
plt.axis("equal")
plt.show()
```



R *L'instruction* **`plt.axis("equal")`** *permet d'imposer la même échelle sur les deux axes. Autrement, le tracé aurait ressemblé à une demi-ellipse plutôt.*