



# Prise en main de Pyplot



**PyPlot** est une sous-bibliothèque de **MatPlotLib** permettant de représenter graphiquement des données.



La bibliothèque MatPlotLib est intégrée à EduPython.

## Import de la bibliothèque

- `import matplotlib.pyplot` ou `import matplotlib.pyplot as plt`  
Allège le code... Par exemple `matplotlib.pyplot.plot(2,1,'ko')` devient `plt.plot(2,1,'ko')`



Dans tout ce document la bibliothèque **`matplotlib.pyplot`** est renommée **`plt`**.  
Instruction : `import matplotlib.pyplot as plt`.

## La fonction plot

Par défaut, le repère s'adapte à l'ensemble des points construits.

### Placer un point de coordonnées (x, y) dans un repère

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

Coordonnées  
du point

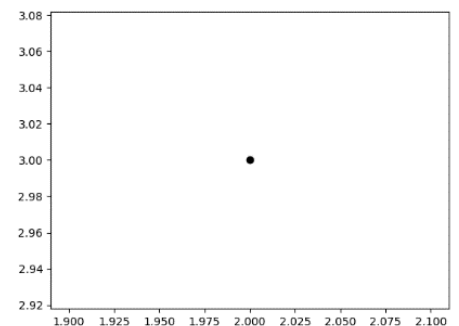
Couleur et style  
de marqueur

#### options :

- couleurs usuelles : **b** (bleu) , **g** (vert) , **r** (rouge) , **c** (cyan) , **m** (magenta) , **y** (jaune) , **k** (noir) , **w** (blanc).
- styles de marqueurs :

"."	","	"o"	"+"	"x"	" "	"_"	"X"	"*"	"v"	"^"	"<"	">"	"s"	"p"	"P"	"D"	"d"
•	.	●	+	×		—	✕	★	▼	▲	◀	▶	■	●	+	◆	◆

Exemple : 'ko' pour un cercle noir ● ou 'bs' pour un carré bleu ■.



### Tracer une courbe point par point

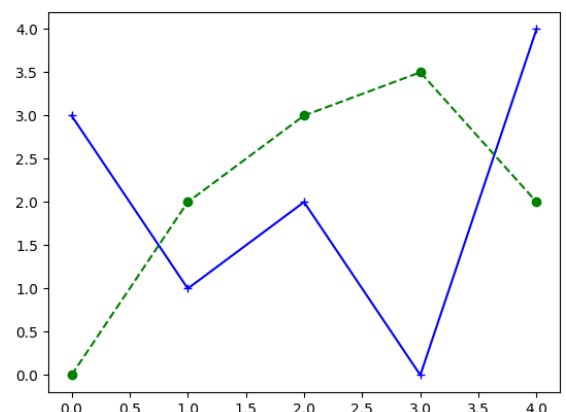
`plt.plot(x,y,options)`

Listes des  
coordonnées  
des points

Couleur et style  
de marqueur

#### Exemple :

```
plt.plot([0,1,2,3,4],[3,1,2,0,4],'b+-')
plt.plot([0,1,2,3,4],[0,2,3,3.5,2],'go--')
```



#### options :

- styles de traits:

'-' : —————

'.' : .....

--' : - - - - -

-. ' : - . . . .

'b+-'

couleur : b

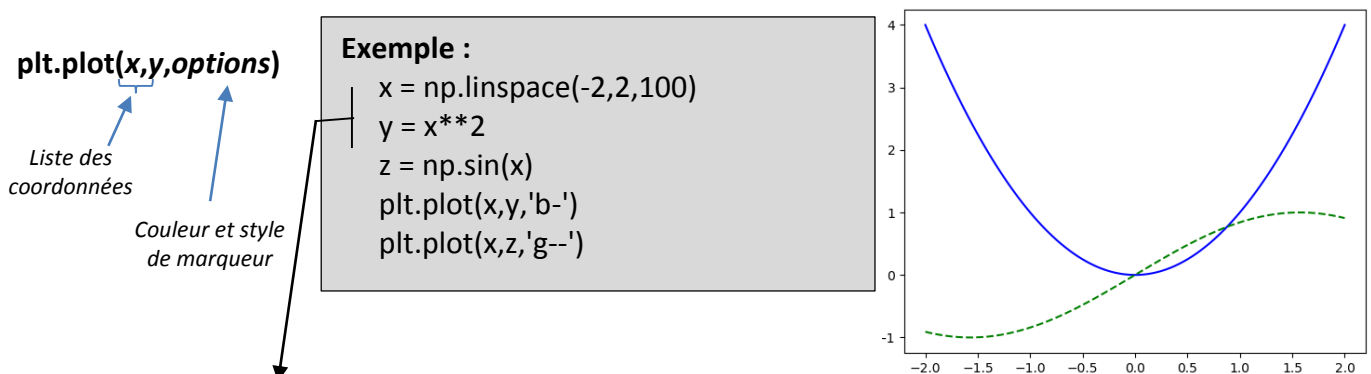
marqueur : +

trait : -

## Tracer la représentation graphique d'une fonction sur un intervalle

### Principe :

- Générer un tableau de nombres couvrant l'intervalle considéré (tableau des abscisses : x)
- Calculer les images des abscisses par la fonction à représenter (tableau des ordonnées : y)
- Placer l'ensemble de des points donc les coordonnées sont dans les deux tableaux précédents.

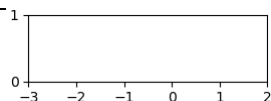


Les abscisses et ordonnées des points sont données dans des tableaux numpy (*numpy.array*).

- **np.linspace(-2,2,100)** : génère le tableau (de type *numpy.array*) des abscisses contenant 100 réels répartis dans l'intervalle  $[-2 ; 2]$  avec un espacement fixe.
- **$x**2$**  et **np.sin(x)** génère également des tableaux de nombres numpy.array. (on utilise les fonction numpy opérant sur les tableaux *numpy.array*).

**Rque :** **sin()** opère sur les réels, **np.sin()** opère sur les réels mais aussi sur les tableaux *numpy*.

### Paramétrer la figure et les axes

Paramétrer	Exemples
Ajouter un titre à la figure	<code>plt.title("Titre de la figure")</code>
Ajouter une légende à une courbe	<code>plt.plot( x , y ,label=" f(x) ")</code> <code>plt.legend()</code>
Modifier l'épaisseur de trait	<code>plt.plot( x , y , linewidth = 2)</code>
Ajouter un titre aux axes de coordonnées	<code>plt.xlabel('axe des abscisses')</code> <code>plt.ylabel('axe des ordonnées')</code>
Limiter les axes	<code>plt.xlim((x_min , x_max))</code> <code>plt.ylim((y_min , y_max))</code>
Travailler en repère orthonormé	<code>plt.axis('equal')</code>
Intersection des axes de coordonnées à l'origine du repère :	<code>axes = plt.gca()</code> <code>axes.spines['right'].set_color('none')</code> <code>axes.spines['top'].set_color('none')</code> <code>axes.spines['bottom'].set_position('zero')</code> <code>axes.spines['left'].set_position('zero')</code>
Ajouter du texte sur la figure	<code>plt.text(x_text,y_text, 'texte')</code>
Etiquettes des axes	<div style="display: flex; align-items: center;"> <div style="margin-right: 20px;"> <code>plt.xticks( range(-4,5) )</code>  <code>plt.yticks( [-1, 0,+1] )</code> </div>  </div> <code>plt.xticks([-np.pi, 0, np.pi], [r'\$-\pi\$', r'\$0\$', r'\$+\pi\$'])</code> <p>3 graduations <math>x=-\pi</math>, <math>x=0</math> et <math>x=\pi</math></p> <p>Les étiquettes respectives sont données en LaTeX</p>

## La fonction hist (histogramme)

### Histogramme simple :

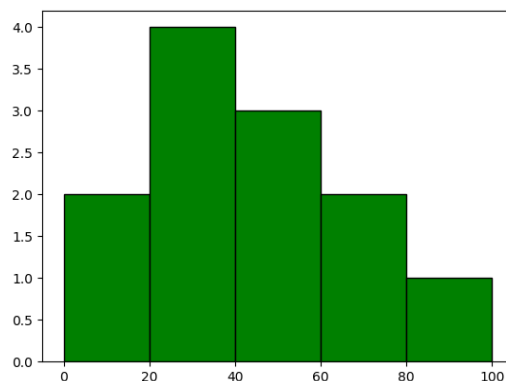
`plt.hist(valeurs, bins = nb_rect)`

Liste des valeurs  
à représenter

Nombre de rectangles  
(regroupement par classes)

#### Exemple :

```
plt.hist([8,15,22,27,32,37,47,51,58,62,71,99] , bins = 5 ,  
color = 'green' , edgecolor = 'black')
```



### Quelques paramètres pour les histogrammes :

	Exemples
Couleur des barres	<code>color = 'green'</code>
Couleur des contours des rectangles	<code>edgecolor = 'black'</code>
Tailles de classes inégales	<code>plt.hist(x, bins = [0, 30, 40, 50, 60, 70, 80, 100])</code>
Histogramme à valeurs pondérées	<code>plt.hist( x , weights=liste_ponderations)</code>
Histogramme des fréquences	<code>normed = True</code>
Diagramme en bâtons	<code>histtype = 'step'</code>
Histogramme horizontal	<code>orientation = 'horizontal'</code>
Réduire la largeur des rectangles	<code>rwidth = 0.5</code> (Largeur réduite à 50%).
Ajouter des hachures	<code>hatch = '/'</code> Valeurs possibles : <code>'/'</code> , <code>'\''</code> , <code>' '</code> , <code>'-'</code> , <code>'+'</code> , <code>'x'</code> , <code>'o'</code> , <code>'O'</code> , <code>':'</code> , <code>'*'</code>
Doubles séries	<code>plt.hist([x1, x2], color = ['blue', 'green'], label = ['x1', 'x2'],)</code>

### Remarque :

L'ajout de titres ou légendes est possible avec une syntaxe identique à celle des figures obtenues avec la fonction `plot`. Par exemples :

- `plt.xlabel('valeurs')`
- `plt.ylabel('effectifs')`
- `plt.title('HISTOGRAMME')`
- `plt.hist(x , bins=5 , label='série1')`  
`plt.legend()`