Introduction aux graphiques en Python avec matplotlib.pyplot

source : https://zestedesavoir.com/tutoriels/469/introduction-aux-graphiques-en-python-avec-matplotlib-pyplot/ (https://zestedesavoir.com/tutoriels/469/introduction-aux-graphiques-en-python-avec-matplotlib-pyplot/)

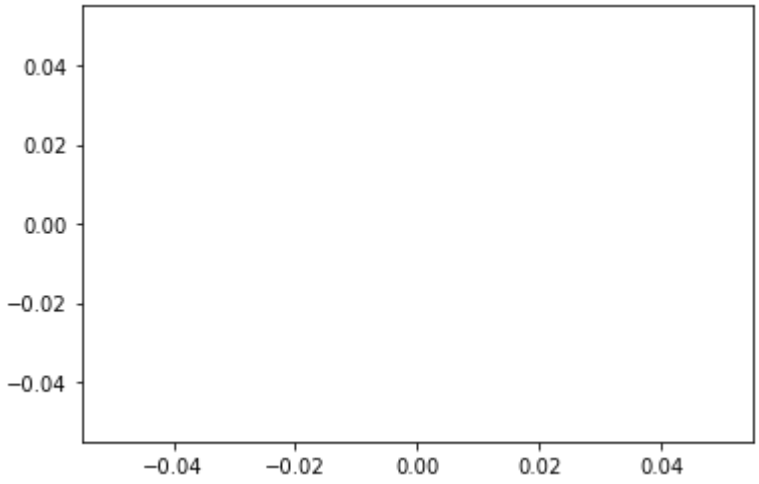
Entrée [ ]:

```
# importation
import matplotlib.pyplot as plt
```

Afficher un graphique vide

Entrée [4]:

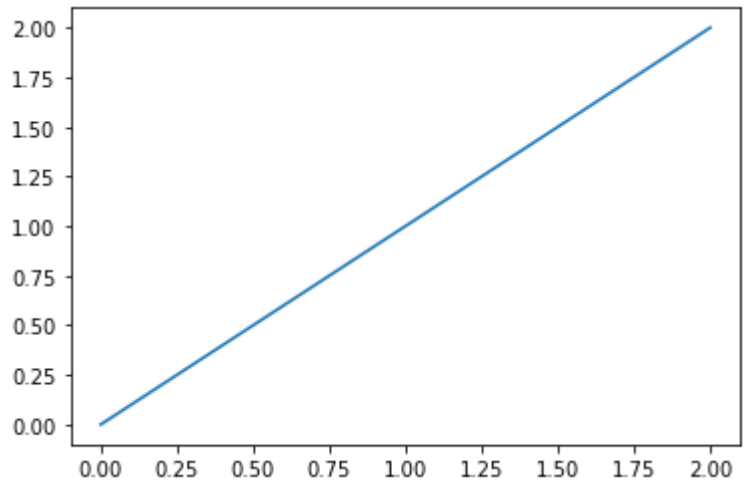
```
plt.plot()

# ouvre la fenêtre
plt.show()

# ferme la fenêtre
plt.close()
```



Tracer des lignes brisées

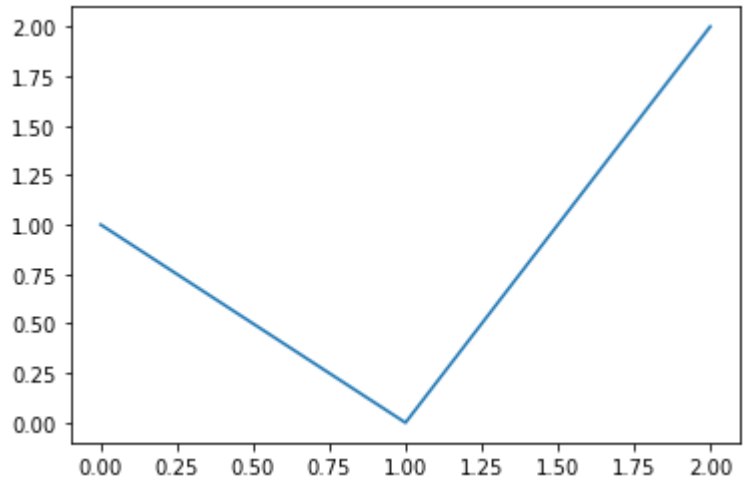
Entrée [5]:

```
plt.plot([0, 1, 2])
# 3 points : A(0, 0), B(1, 1) et C(2, 2)
plt.show()
plt.close()
```



Entrée [7]:

```
plt.plot([1, 0, 2])
# 3 points : A(1, 1), B(0, 0) et C(2, 2)
plt.show()
plt.close()
```



Enregistrement dans un fichier avec `savefig`

Entrée [8]:

```
plt.savefig("graph_test_tuto.png")
```

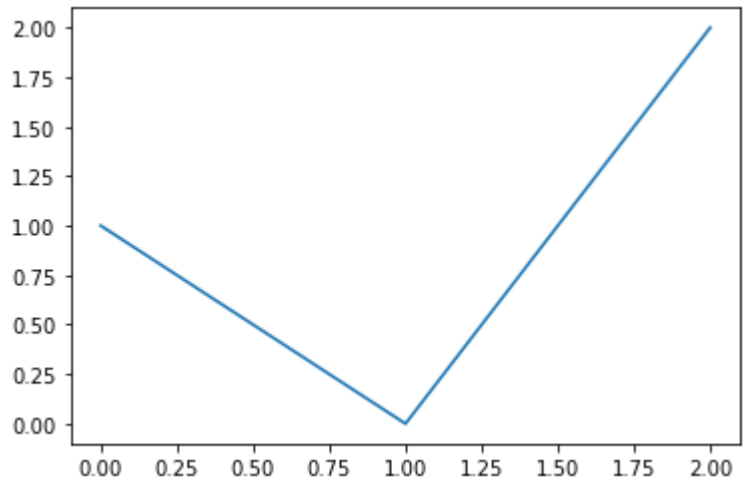
```
<Figure size 432x288 with 0 Axes>
```

Tracer une figure

Nous pouvons aussi passer deux listes en arguments à `plot` . La première liste correspondra à **la liste des abscisses** des points que nous voulons relier et la seconde à la **liste de leurs ordonnées**.

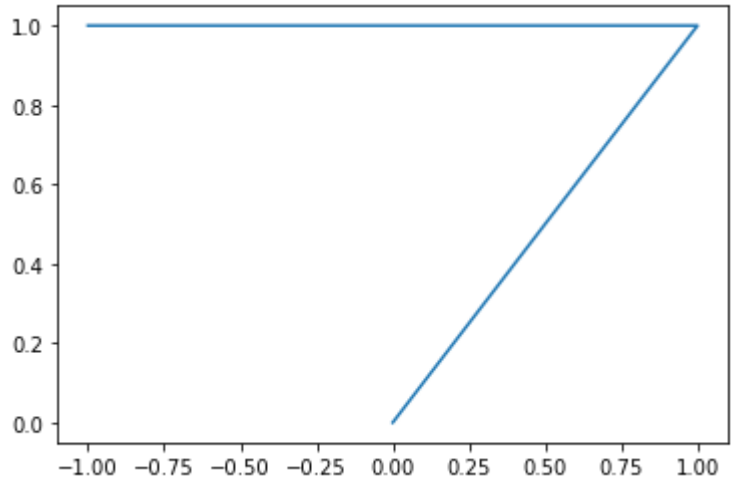
Entrée [9]:

```
x = [0, 1, 2]
y = [1, 0, 2]

plt.plot(x, y)

plt.show()
plt.close()
```



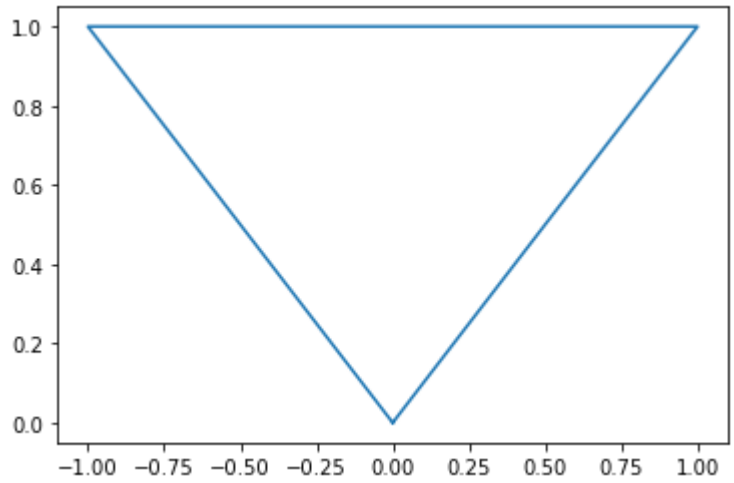
Entrée [10]:

```
x = [0, 1, -1]
y = [0, 1, 1]

plt.plot(x, y)

plt.show()
plt.close()
```



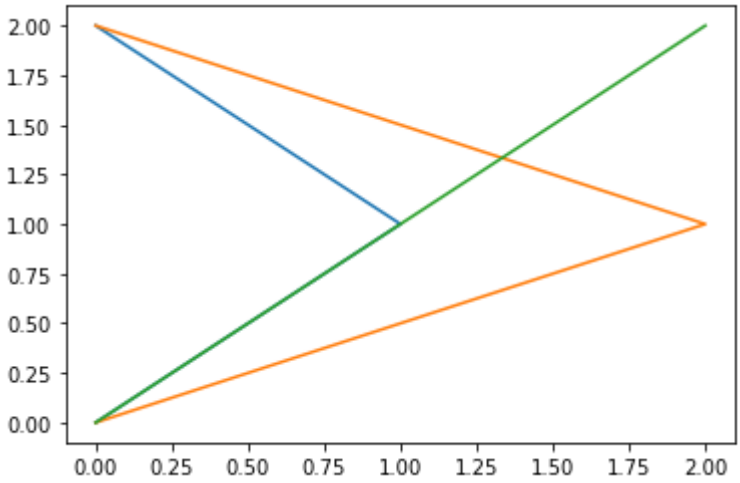
Entrée [11]:

```
x = [0, 1, -1, 0]
y = [0, 1, 1, 0]

plt.plot(x, y)

plt.show()
plt.close()
```



Entrée [12]:

```
x = [0, 1, 0]
y = [0, 1, 2]

x1 = [0, 2, 0]
y1 = [2, 1, 0]

x2 = [0, 1, 2]
y2 = [0, 1, 2]

plt.plot(x, y, x1, y1, x2, y2)
plt.show()
plt.close()
```

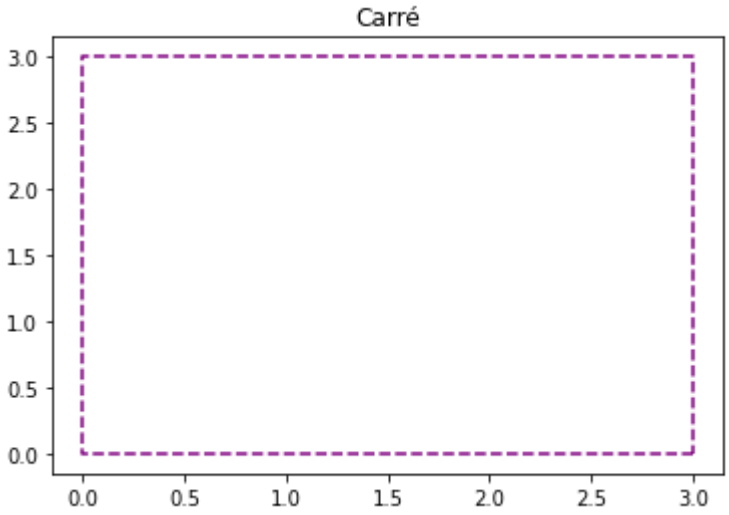


Entrée [46]:

```
x = [3, 0, 0, 3, 3]
y = [0, 0, 3, 3, 0]
plt.plot(x, y, color = "purple", label='test', linestyle='--')
plt.title("Carré")
```

Out[46]:

```
Text(0.5, 1.0, 'Carré')
```



Paramètre `color`

Le paramètre `color` permet de changer la couleur du tracé. Cette couleur peut être donnée sous plusieurs formes.

- Sous forme de chaîne de caractères représentant les noms (ou abréviations) pour les couleurs primaires, le noir et le blanc : b ou blue, g ou green, r ou red, c ou cyan, m ou magenta, y ou yellow, k ou black, w ou white. C'est quand même assez explicite, il suffit d'écrire les noms en anglais.
- Sous la forme d'un tuple correspondant aux valeurs RGB de la couleur. Cependant, ce tuple doit contenir des valeurs entre 0 et 1 (il suffit alors de diviser les valeurs RGB par 255.0). Ainsi, ce sera `color = (255 / 255.0, 0, 0)` pour obtenir du rouge. Notons que nous pouvons ajouter une valeur (toujours entre 0 et 1) à ce tuple pour représenter la transparence alpha.
- Sous la forme de chaîne de caractères représentant la couleur en notation hexadécimale. On aura donc `color = '#00FF00'` pour obtenir du vert.
- Et les adeptes des nuances de gris pourront donner en paramètre une chaîne de caractères correspondant à l'intensité en gris. Par exemple `color = '0.8'` permet d'obtenir un gris pâle.

Le style de ligne `linestyle`

Nous pouvons également changer le style des lignes en passant à la commande plot une chaîne de caractères. Les caractères acceptés et leur signification sont disponibles sur la documentation de la commande plot. Tous ces styles ne relient pas les points entre eux, certains ne font qu'afficher le signe à l'endroit où se situe le point. Présentons quelques caractères :

- `-` est le style par défaut, il correspond à une ligne pleine ;
- `--` correspond à une ligne en pointillés ;
- `:` correspond à une ligne formée de points ;

- `-.` correspond à une ligne formée d'une suite de points et de tirets.

Ces caractères correspondent au paramètre linestyle. Nous pouvons aussi ajouter des marqueurs avec le paramètre marker qui rajoute alors un marqueur pour chaque point de votre graphique. Ce paramètre est aussi une chaîne de caractères. Voici quelques marqueurs : `*` , `+` , `o` .

La grille `grid`

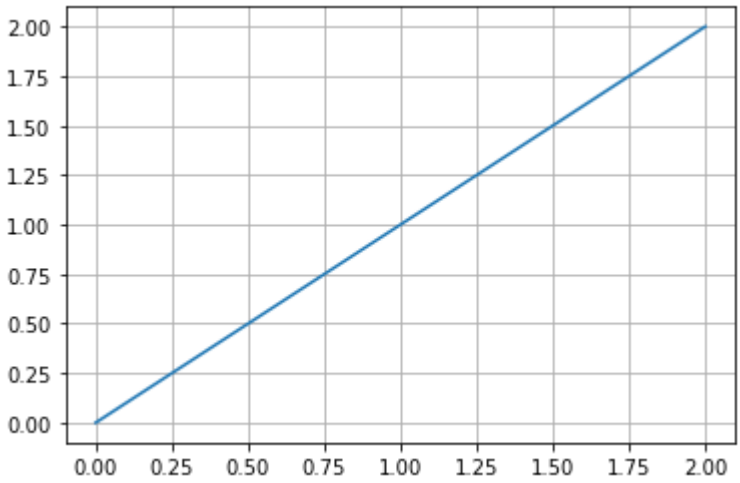
Nous pouvons ajouter une grille avec la fonction `grid` qui affiche un quadrillage en pointillés. Nous pouvons bien sûr changer le style de ce quadrillage.

- Le paramètre `axis` nous permet de choisir quels axes doivent être quadrillés. Il peut prendre les valeurs both (les deux), x ou y.
- Le paramètre `color` nous permet de choisir la couleur de l'axe. Il fonctionne de la même manière que le paramètre color de la fonction plot.
- Le paramètre `linewidth` permet de choisir l'épaisseur des traits.
- Le paramètre `linestyle` permet de choisir le style de quadrillage. Il peut prendre comme valeur (tout comme plot) `-` , `--` , `:` , `-.` .

Afficher une grille

Entrée [55]:

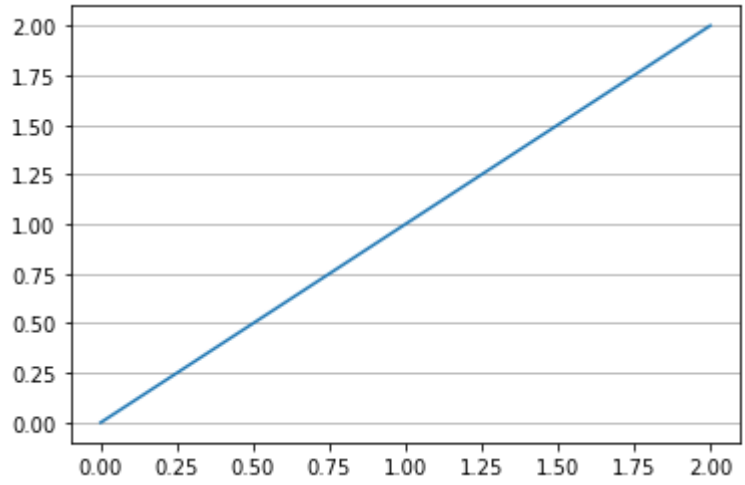
```
x = [0, 1, 2]
plt.plot(x)

# afficher la grille
plt.grid()
# montre les deux axes par défaut
# pour préciser : plt.grid(axis='both')
```



Afficher seulement l'axe 'y' sur la grille

Entrée [56]:

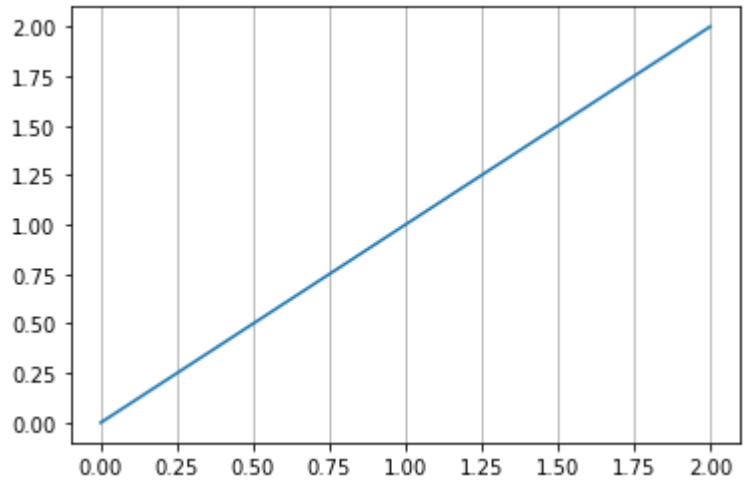
```
x = [0, 1, 2]
plt.plot(x)

# afficher la grille
plt.grid(axis='y')
```



Afficher seulement l'axe 'x' sur la grille

Entrée [57]:

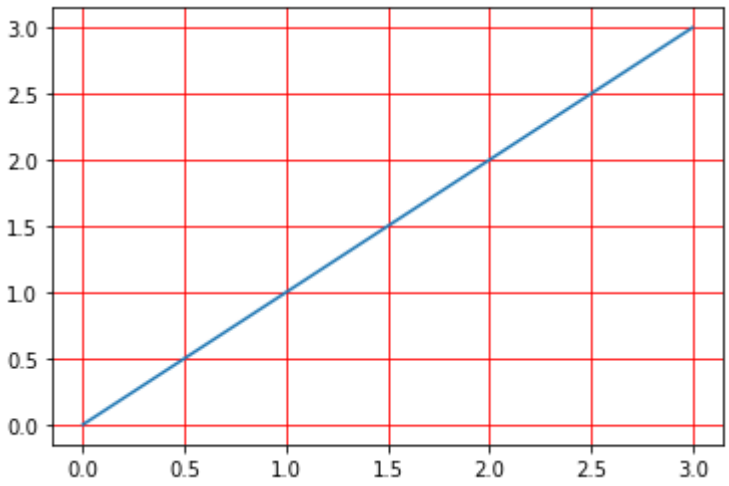
```
x = [0, 1, 2]
plt.plot(x)

# afficher la grille
plt.grid(axis='x')
```



Modifier la couleur de la grille

Entrée [65]:

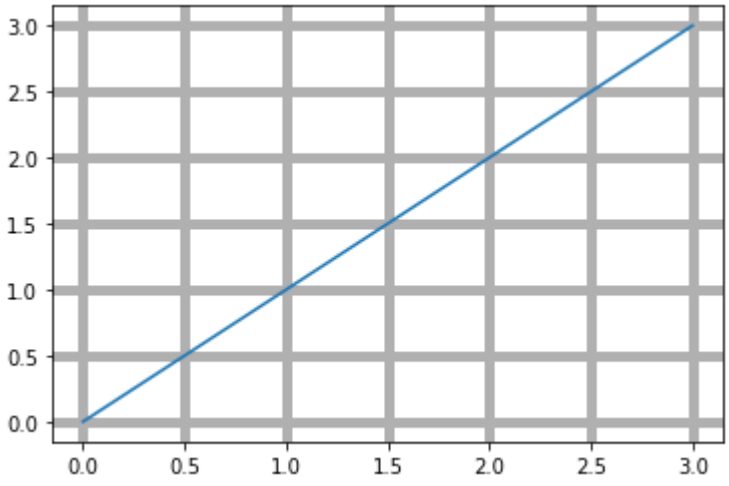
```
x = [0, 1, 2, 3]

plt.plot(x)
plt.grid(color='r')
```



Modifer l'épaisseur des traits de la grille

Entrée [68]:

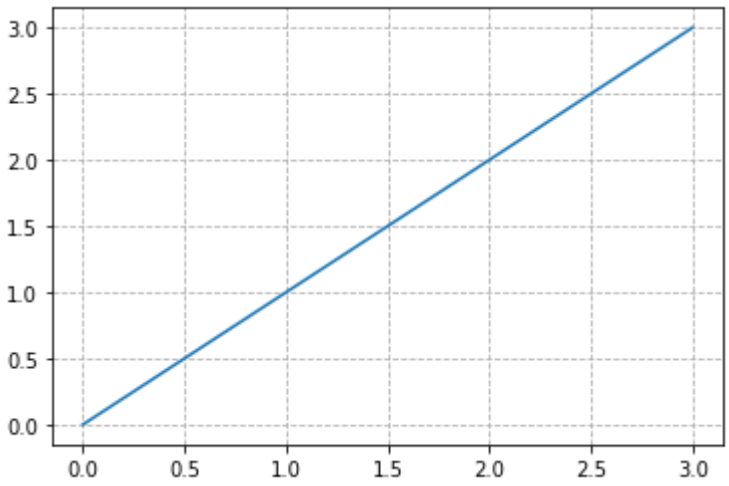
```
x = [0, 1, 2, 3]

plt.plot(x)
plt.grid(linewidth=5)
```



Modifier le style de quadrillage

Entrée [74]:

```
x = [0, 1, 2, 3]

plt.plot(x)
plt.grid(linestyle='--')
```



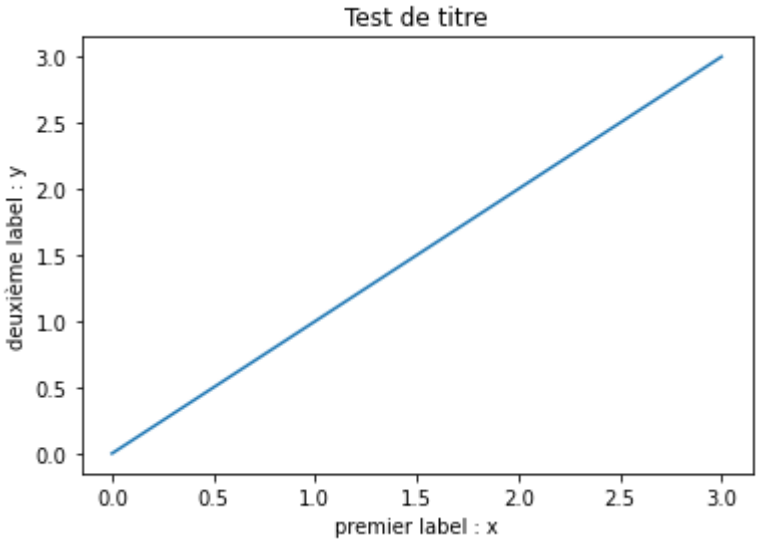
Labels sur les axes

Entrée [77]:

```
x = [0, 1, 2, 3]
plt.plot(x)

# Labels
plt.title("Test de titre")
plt.xlabel("premier label : x")
plt.ylabel("deuxième label : y")
```

Out[77]:

```
Text(0, 0.5, 'deuxième label : y')
```



Police, taille, couleur d'écriture

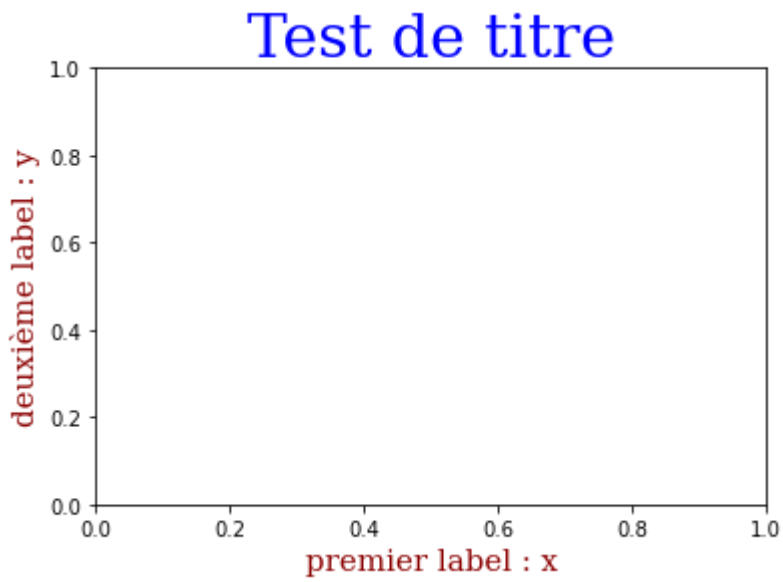
You can use the `fontdict` parameter in `xlabel()`, `ylabel()`, and `title()` to set font properties for the title and labels.

Entrée [79]:

```
font1 = {'family':'serif','color':'blue','size':30}
font2 = {'family':'serif','color':'darkred','size':15}

plt.title("Test de titre", fontdict = font1)
plt.xlabel("premier label : x", fontdict = font2)
plt.ylabel("deuxième label : y", fontdict = font2)
```

Out[79]:

```
Text(0, 0.5, 'deuxième label : y')
```



Position des labels

You can use the loc parameter in `title()` to position the title.

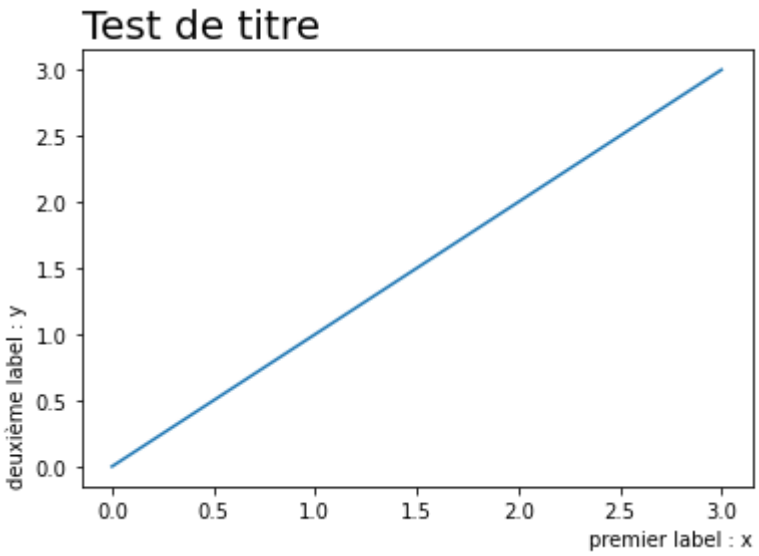
Legal values are: 'left', 'right', and 'center'. Default value is 'center'.

Entrée [87]:

```
x = [0, 1, 2, 3]
plt.plot(x)

# Labels
plt.title("Test de titre", loc='left', size='20')
plt.xlabel("premier label : x", loc='right')
plt.ylabel("deuxième label : y", loc='bottom')
```

Out[87]:

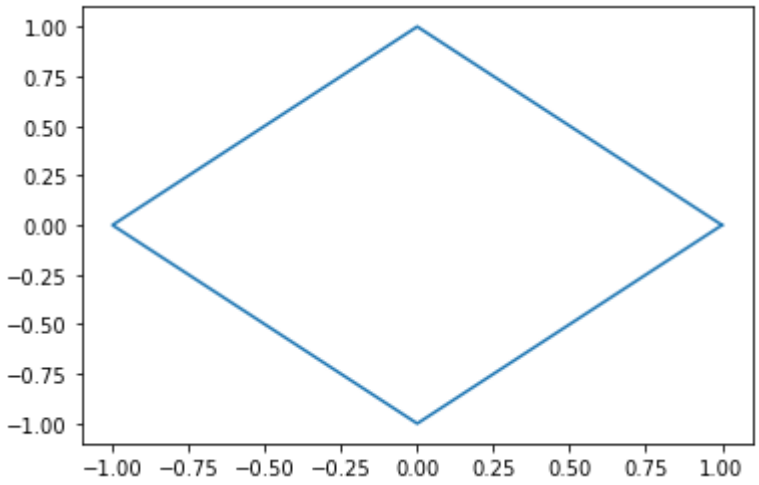
```
Text(0, 0, 'deuxième label : y')
```



Modifier l'échelle

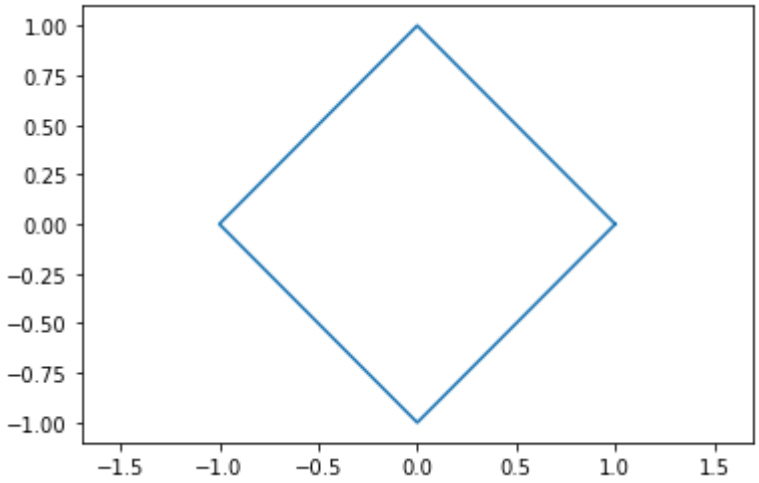
Entrée [89]:

```
# Exemple :


import matplotlib.pyplot as plt

x = [1, 0, -1, 0, 1]
y = [0, 1, 0, -1, 0]
plt.plot(x, y)
plt.show()
plt.close()
```



Entrée [90]:

```
# Modification :


import matplotlib.pyplot as plt

x = [1, 0, -1, 0, 1]
y = [0, 1, 0, -1, 0]
plt.plot(x, y)

# ligne qui change :
plt.axis('equal')
plt.show()

plt.close()
```



Exemple final de code :

Entrée [91]:

```
x = [0.25, 0.25, 1.25, 0.5, 1, 0.25, 0.6, 0, -0.6, -0.25, -1, -0.5, -1.25, -0.25, -0.25, 0.
y = [0, 0.5, 0.5, 1, 1, 1.5, 1.5, 2, 1.5 , 1.5, 1, 1, 0.5, 0.5, 0, 0]

plt.plot(x, y, '-.', color = "green", lw = 2)

plt.title("Mon beau sapin")

plt.axis('equal')

plt.xlabel("C'est Noel")
plt.ylabel("Vive le vent")

plt.show()
plt.close()
```