Les graphiques (première partie)

Ceci est un petit tutoriel permettant de tracer des graphiques très simples mais suffisants dans le cadre du programme de physique-chimie. Pour plus d'informations, se référer au site : https://matplotlib.org/tutorials/index.html

Nous vous conseillons d'enregistrer ce fichier notebook sous un nom personnalisé afin de tester et modifier les lignes de codes à votre guise sans impacter la version originale.

Voici deux programmes (l'un très simple, l'autre plus complexe permettant d'afficher la caractéristique tension-intensité d'un conducteur ohmique à partir du tableau de valeurs suivantes:

I(mA)	0	25	50	75	100	125
U(V)	0	1,8	3,3	5,2	6,8	8,5

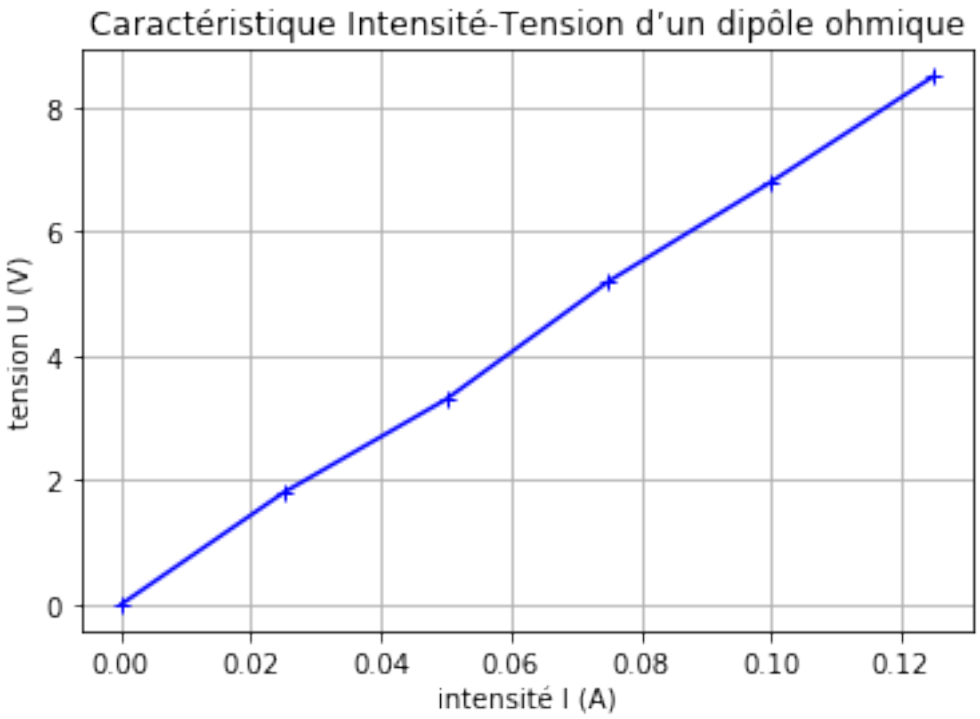
Dans un premier temps, vous allez exécuter les deux programmes fournis en observant les différences notables.

Dans un second temps, nous allons expliquer chacun de ces programmes

Premier programme (le plus simple)

In [1]:

```
import matplotlib.pyplot as plt
%matplotlib inline
I=[0,25e-3,50e-3,75e-3,100e-3,125e-3]
U=[0,1.8,3.3,5.2,6.8,8.5]
plt.figure()
plt.plot(I,U,color='b', marker = '+')
plt.xlabel("intensité I (A)")
plt.ylabel("tension U (V)")
plt.grid()
plt.title("Caractéristique Intensité-Tension dun dipôle"
          " ohmique")
plt.show()
```

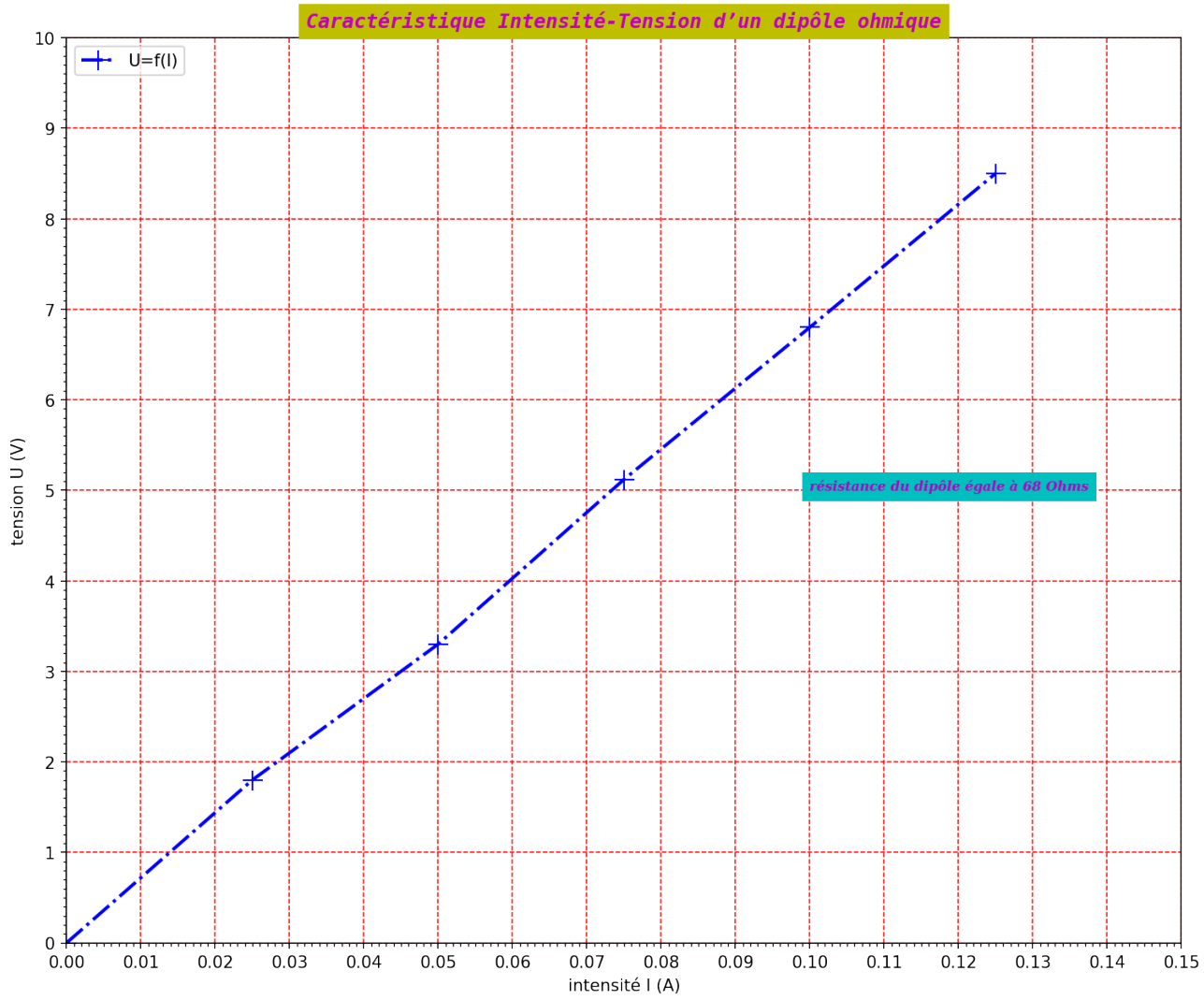


Deuxième programme (plus complexe)

In [2]:

```
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator
%matplotlib inline
I=[0,25e-3,50e-3,75e-3,100e-3,125e-3]
U=[0,1.8,3.3,5.12,6.8,8.5]
plt.figure("loi d'Ohm", figsize=(12,10), dpi=150)
plt.plot(I,U,color='b', marker = '+',markersize = 12,
         linestyle='-.', linewidth = 2, label='U=f(I)')
plt.legend(loc=2)
plt.xlim(0,0.150)
plt.ylim(0,10)
plt.xlabel("intensité I (A)")
plt.ylabel("tension U (V)")
plt.gca().xaxis.set_major_locator(MultipleLocator(0.01))
plt.gca().xaxis.set_minor_locator(MultipleLocator(0.001))
plt.gca().yaxis.set_major_locator(MultipleLocator(1))
plt.gca().yaxis.set_minor_locator(MultipleLocator(0.1))
plt.grid(color='r', linestyle='--', linewidth=0.75)
plt.title("Caractéristique Intensité-Tension dun dipôle ohmique",
          fontsize=12,family='monospace',fontweight='bold',
          style='italic',color='m', backgroundcolor='y',
          horizontalalignment='center')
plt.text(0.100,5,'résistance du dipôle égale à 68 Ohms',
         fontsize=8, family='serif',fontweight='heavy',
         style='oblique',color='m',backgroundcolor='c', alpha=1)
```

```
plt.show()
```



EXPLICATIONS DES PROGRAMMES

Attention à bien exécuter les cellules de code suivantes les unes après les autres. Il est normal que rien ne s'affiche lors de l'exécution de certaines cellules.

Import des bibliothèques utiles pour la création de graphiques

In [3]:

```
# bibliothèque matplotlib.pyplot obligatoire
# pour la création de graphiques

import matplotlib.pyplot as plt

# %matplotlib inline permet d'afficher les graphiques
# matplotlib sous les cellules de code du notebook
```

```
%matplotlib inline

# bibliothèque numpy utile pour créer une liste de graduations

import numpy as np

# bibliothèque matplotlib.ticker utile ici uniquement pour
# graduer les axes à sa convenance avec la fonction MultipleLocator

from matplotlib.ticker import MultipleLocator
```

Créer des listes contenant les valeurs du tableau

Pour créer un graphique sous Python, il faut tout d'abord créer un tableau de valeurs ou l'importer à partir d'un fichier .csv (voir tuto Comment importer les données numériques d'un tableur scientifique dans un programme python ?)

In [4]:

```
# création d'une liste pour la grandeur portée en abscisse
# ici l'intensité est convertie en Ampère

I=[0,25e-3,50e-3,75e-3,100e-3,125e-3]

# création d'une liste pour la grandeur portée en ordonnée
# ici la tension

U=[0,1.8,3.3,5.2,6.8,8.5]
```

Créer et paramétrer une fenêtre graphique

On utilise la méthode **plt.figure(num, figsize, dpi)** de la bibliothèque matplotlib.pyplot as plt.

Voici les principaux paramètres de cette méthode, il n'est pas obligatoire de tous les spécifier. Dans ce cas, ils prendront leur valeur par défaut.

- num : dénomination de la fenêtre graphique : nombre entier ou chaîne de caractères ; n'apparaît pas sur le graphique
- figsize =(x,y) : largeur x , hauteur y en pouces, valeur par défaut : [6.4, 4.8]
- dpi: résolution de la figure, par défaut : 100

(voir ligne 5 du premier programme et ligne 6 du second programme)

D'autres paramètres existent, pour plus d'informations :

https://matplotlib.org/api/_as_gen/matplotlib.pyplot.figure.html#matplotlib.pyplot.figure

Afficher les points ainsi que leur légende

Toujours à l'aide de la bibliothèque matplotlib.pyplot as plt :

1. Pour gérer l'affichage des points, on utilise la méthode **plt.plot(x, y, color, marker, markersize, linestyle, linewidth, label)**

Les paramètres présentés ici sont :

- x et y : grandeurs portées en abscisse et en ordonnée
- color : couleur des points et de la ligne les reliant

code de la couleur	couleur
'b'	bleu
'g'	vert
'r'	rouge
'c'	cyan
'm'	magenta
'y'	jaune
'k'	noir
'w'	blanc

Pour plus d'informations sur les couleurs :

https://matplotlib.org/tutorials/colors/colors.html#sphx-glr-tutorials-colors-colors-py

- marker : forme des points (marqueurs)

code du marqueur	forme du marqueur
'.'	point
','	pixel
'o'	cercle
'+'	plus
'x'	fois
'v'	triangle vers le bas
'^'	triangle vers le haut
'<'	triangle vers la gauche
'>'	triangle vers la droite
'1'	étoile à trois branches dont l'une pointe vers le bas
'2'	étoile à trois branches dont l'une pointe vers le haut
'3'	étoile à trois branches dont l'une pointe vers la gauche
'4'	étoile à trois branches dont l'une pointe vers la droite
's'	carré
'p'	pentagone
'*'	étoile
'h'	hexagone
'H'	hexagone
'D'	diamant
'd'	losange
'AltGr+6'	ligne verticale
'_'	ligne horizontale

- markersize : taille du marqueur
- linestyle : style de ligne

code du style	style de ligne
'-'	ligne en trait plein
'–'	ligne en pointillé long
'-.'	ligne en pointillé mixte
':'	ligne en pointillé court

- linewidth : épaisseur de la ligne
- label : permet de légender la courbe en créant une étiquette

D'autres paramètres existent. Pour plus d'informations :

https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot

2. Pour faire apparaître l'étiquette (label) sur le graphique, il faut utiliser la méthode **plt.legend(loc)**.

Le paramètre loc permet de choisir l'emplacement de l'étiquette sur le graphique :

Emplacement	Code d'emplacement
meilleur	0
en haut à droite	1
en haut à gauche	2
en bas à gauche	3
en bas à droite	4
droite	5
centre gauche	6
centre droit	7
centre bas	8
centre supérieur	9
centre	10

D'autres paramètres existent, notamment pour choisir la couleur , la police....Pour plus d'informations :

https://matplotlib.org/api/_as_gen/matplotlib.pyplot.legend.html

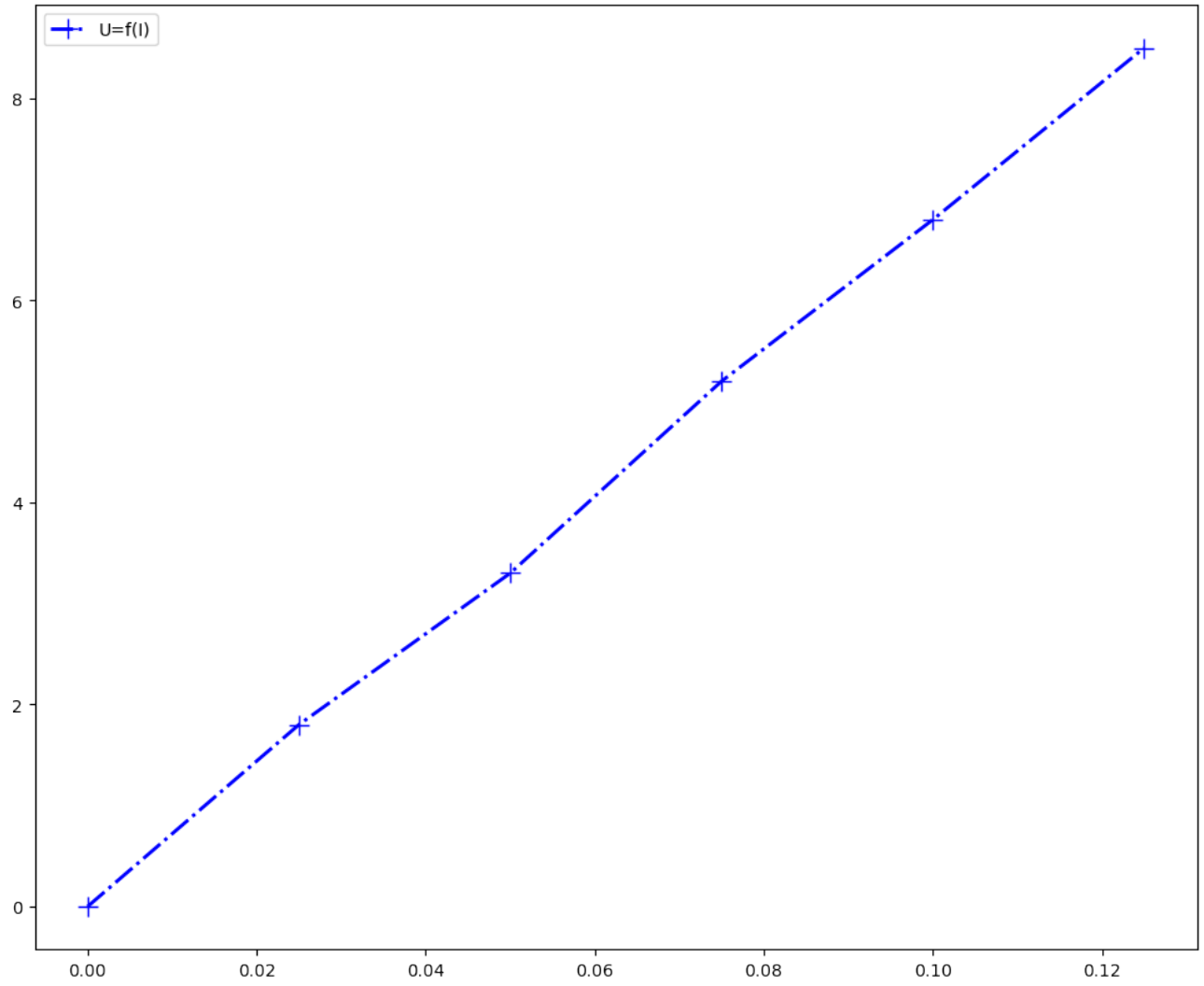
A vous maintenant de modifier les paramètres des trois lignes de code ci-dessous.

In [5]:

```
plt.figure("loi d'Ohm", figsize=(12,10), dpi=100)
plt.plot(I,U,color='b', marker = '+',markersize = 12,
         linestyle='-.', linewidth = 2, label='U=f(I)')
plt.legend(loc=2)
```

Out [5]:

```
<matplotlib.legend.Legend at 0x7fca18321320>
```



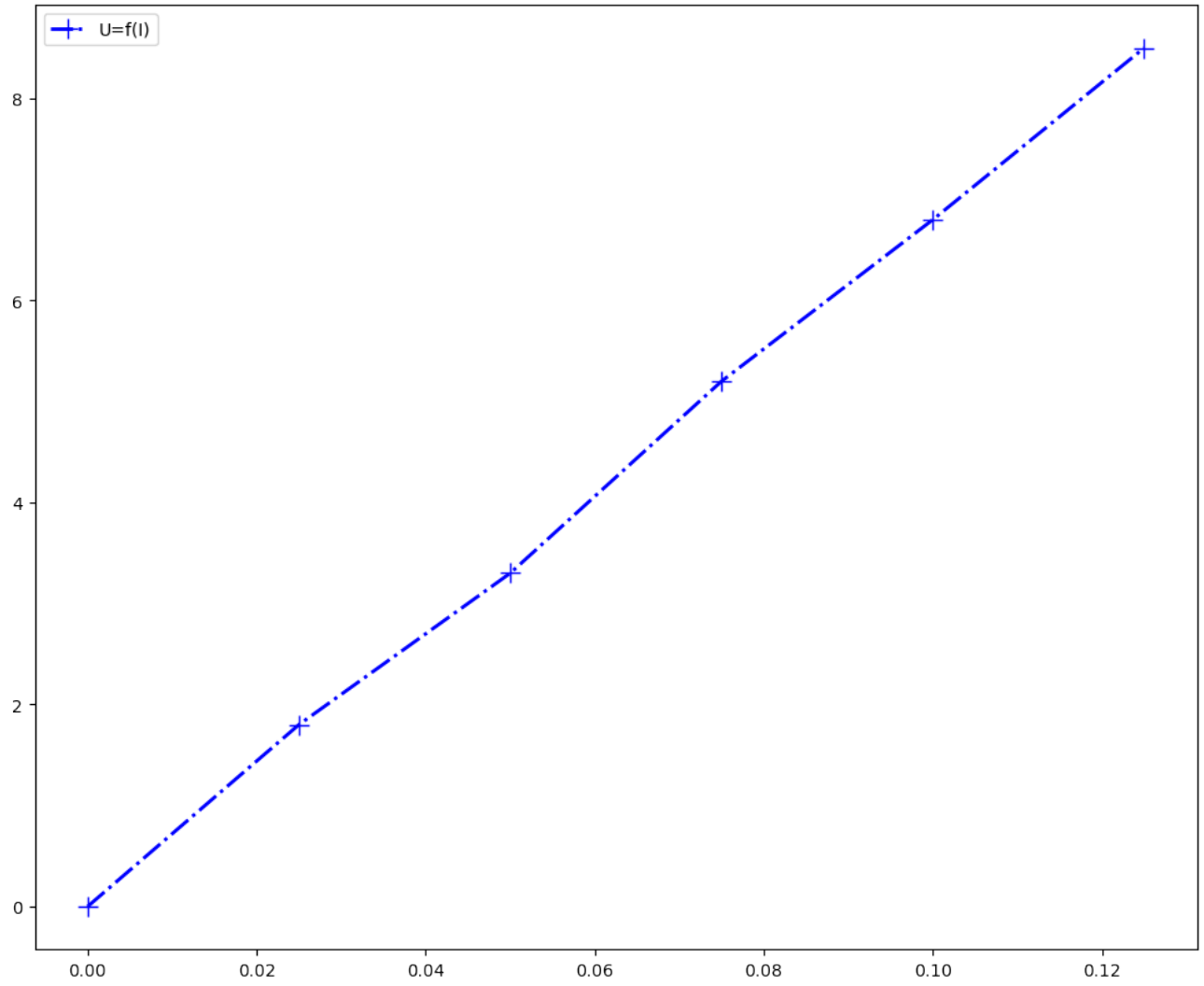
Une autre possibilité de codage plus simple pour un même résultat :

In [6]:

```
plt.figure("loi d'Ohm", figsize=(12,10), dpi=100)
plt.plot(I,U,'b+-.',markersize = 12, linewidth = 2, label='U=f(I)')
plt.legend(loc=2)
```

Out [6]:

```
<matplotlib.legend.Legend at 0x7fca18714320>
```

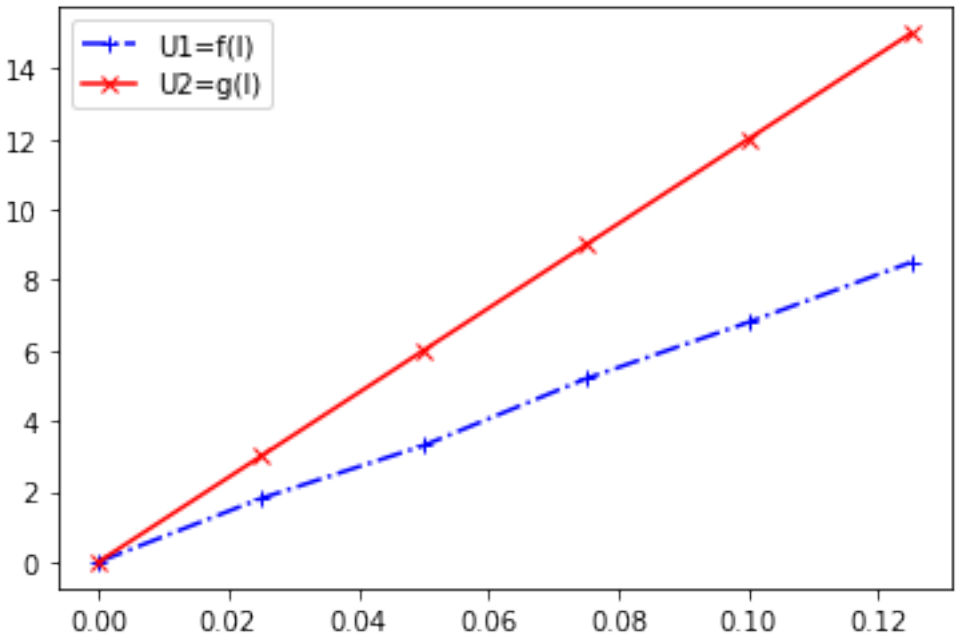


Et si l'on veut afficher deux courbes sur le même graphique?

Première possibilité :

In [7]:

```
I=[0,25e-3,50e-3,75e-3,100e-3,125e-3]
U1=[0,1.8,3.3,5.2,6.8,8.5]
U2 = [0,3,6,9,12,15]
plt.figure()
a=plt.plot(I,U1,'b+-.',label='U1=f(I)')
plt.plot(I,U2,'rx-',label='U2=g(I)')
plt.legend(loc=2)
plt.show()
```

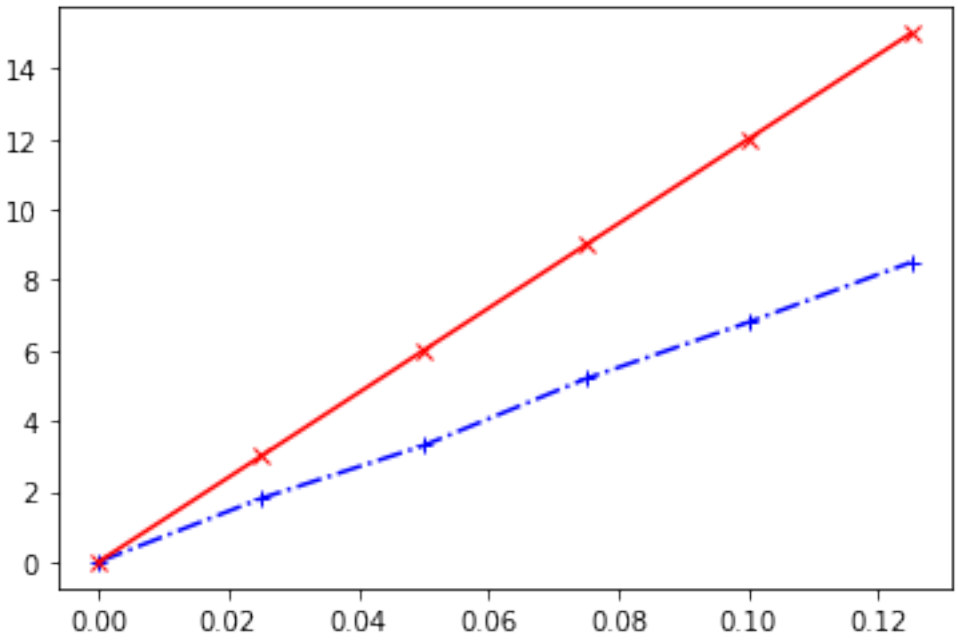


Deuxième possibilité : La méthode plt.plot() renvoie une liste de courbes. Il est donc possible d'afficher deux courbes simultanément avec un seul appel de la methode plt.plot() comme dans l'exemple ci-dessous.

Remarque utile pour comprendre le prochain tutoriel sur les animations de graphique : L'indice de la première courbe est 0, celui de la deuxième est 1...

In [8]:

```
I=[0,25e-3,50e-3,75e-3,100e-3,125e-3]
U1=[0,1.8,3.3,5.2,6.8,8.5]
U2 = [0,3,6,9,12,15]
plt.figure()
plt.plot(I,U1,'b+-.',I,U2,'rx-')
plt.show()
```



Configurer les axes du graphique et faire apparaître le quadrillage

Toujours à l'aide de la bibliothèque matplotlib.pyplot as plt :

1. Pour délimiter les valeurs minimales et maximales des axes, il y a deux possibilités :
   - soit on utilise la méthode : **plt.axis([xmin,xmax,ymin,ymax])**
   - soit on utilise les méthodes : **plt.xlim(xmin,xmax)** et **plt.ylim(ymin,ymax)**
2. Pour légender les axes, on utilise les méthodes : **plt.xlabel("légende1")** et **plt.ylabel("légende2")**
3. La graduation des axes se fait par défaut. Pour graduer les axes à votre convenance, voici deux méthodes :

   Première méthode :

   - appeler la bibliothèque numpy : **import numpy as np**
   - utiliser les méthodes :
     - **plt.xticks(np.arange(xmin, xmax, valeur d'une graduation))**
     - **plt.yticks(np.arange(ymin, ymax, valeur d'une graduation))**

   Deuxième méthode :

   - appeler la fonction MultipleLocator de la bibliothèque matplotlib.ticker : **from matplotlib.ticker import MultipleLocator**
   - utiliser les méthodes pour les graduations majeures :
     - **plt.gca().xaxis.set_major_locator(MultipleLocator(valeur d'une graduation))**
     - **plt.gca().yaxis.set_major_locator(MultipleLocator(valeur d'une graduation))**
   - utiliser les méthodes pour les graduations mineures :
     - **plt.gca().xaxis.set_minor_locator(MultipleLocator(valeur d'une graduation))**
     - **plt.gca().yaxis.set_minor_locator(MultipleLocator(valeur d'une graduation))**
4. Pour faire apparaître le quadrillage sur les graduations majeures ou par défaut, on utilise la méthode : **plt.grid(color, linestyle, linewidth)**.

Les paramètres du quadrillage mentionnés ci-dessus sont les suivants :

- color : la couleur des lignes (codes : cf. tableau ci-dessus)
- linestyle : le style des lignes (codes : cf. tableau ci-dessus)
- linewidth : l'épaisseur des lignes

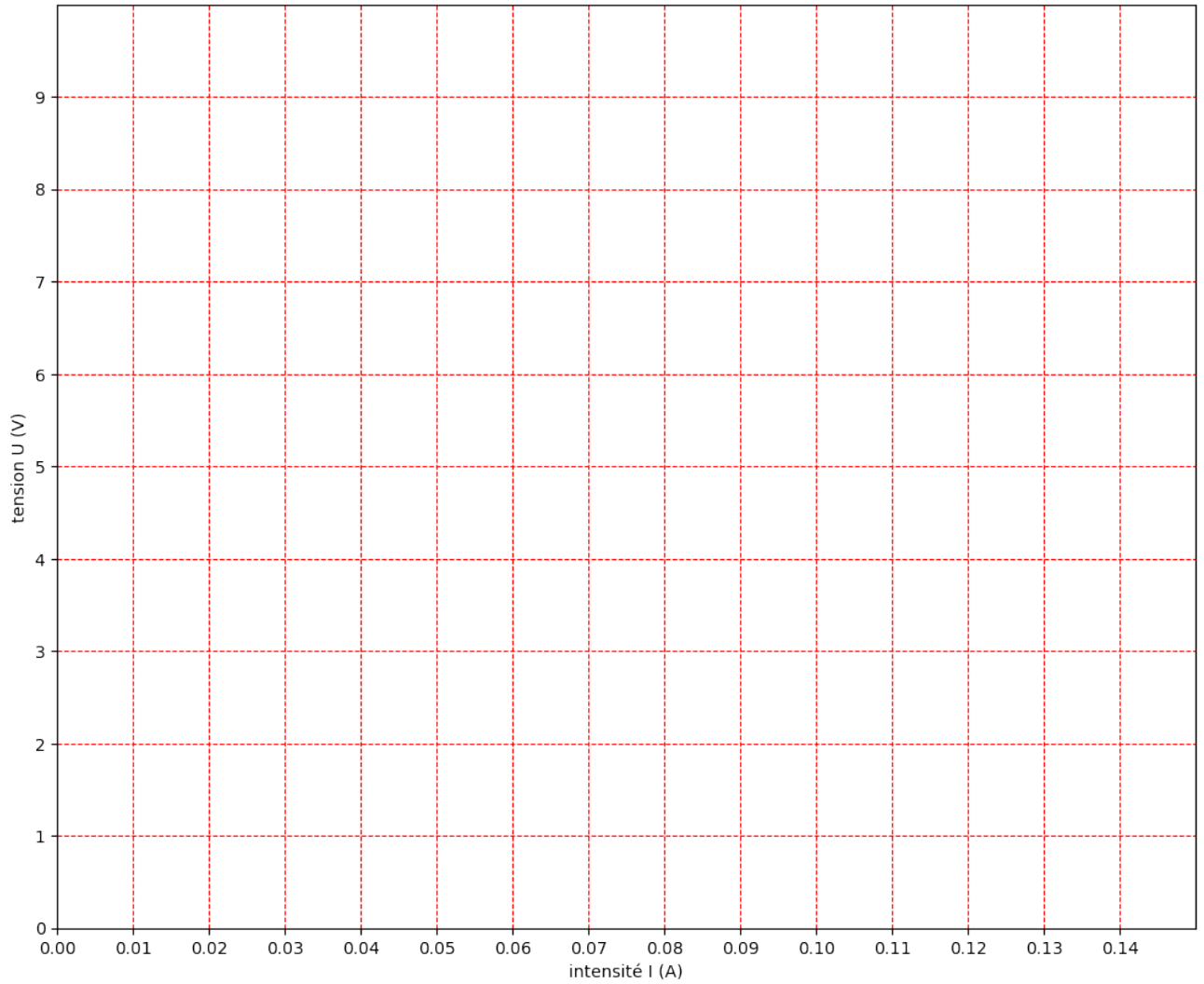
D'autres paramètres existent. Pour plus d'informations :

https://matplotlib.org/api/_as_gen/matplotlib.pyplot.grid.html

Premier exemple avec les méthodes plt.axis(), plt.xticks() et plt.yticks() :

In [9]:

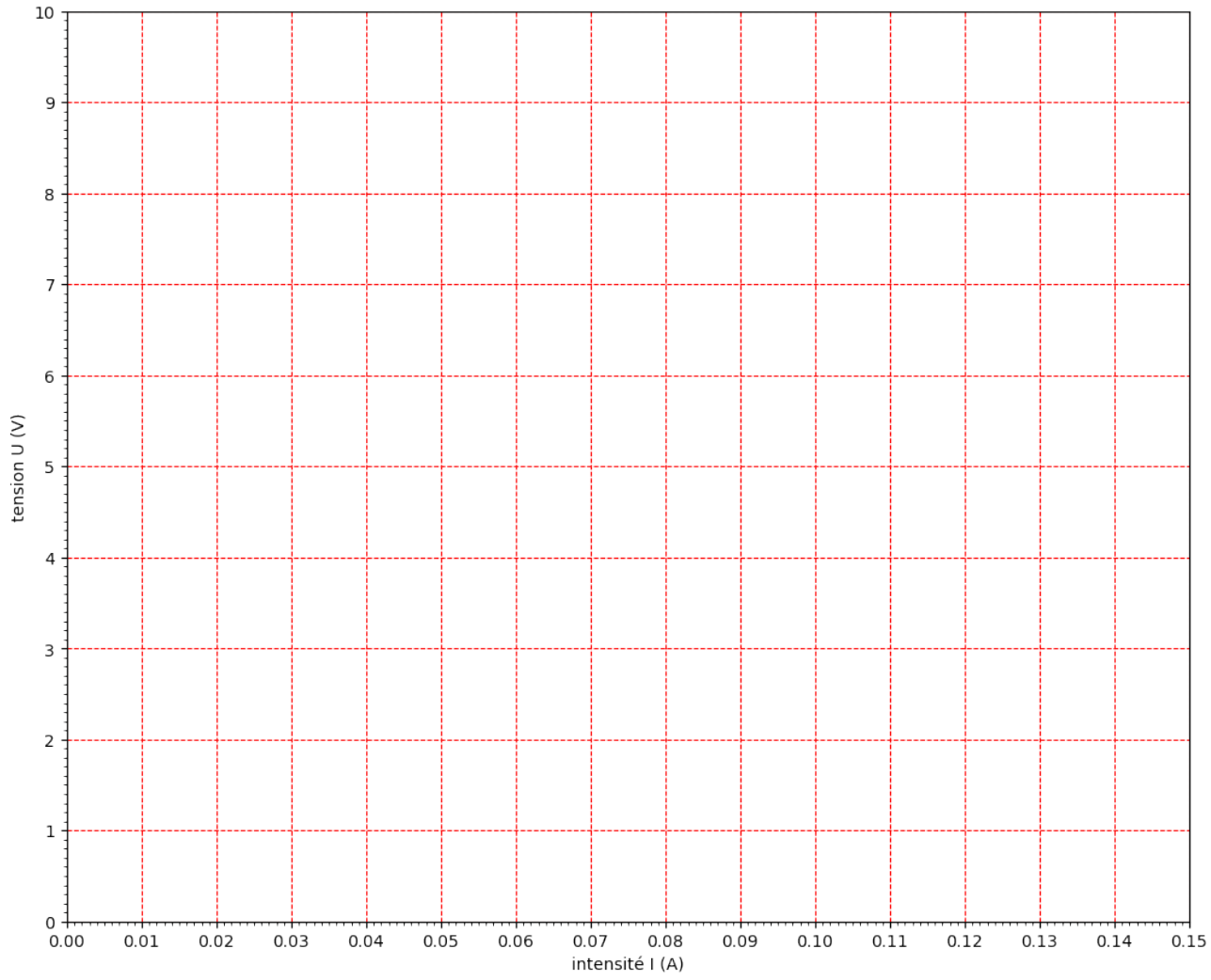
```
plt.figure(1, figsize=(12,10), dpi=100)
plt.axis([0,0.150,0,10])
plt.xlabel("intensité I (A)")
plt.ylabel("tension U (V)")
plt.yticks(np.arange(0,10,1))
plt.xticks(np.arange(0,0.150,0.01))
plt.grid(color='r', linestyle='--', linewidth=0.75)
```



Deuxième exemple avec les méthodes plt.gca().xaxis.set_major_locator(MultipleLocator()) etc. . .

In [10]:

```
plt.figure(2, figsize=(12,10), dpi=100)
plt.axis([0,0.150,0,10])
plt.xlabel("intensité I (A)")
plt.ylabel("tension U (V)")
plt.gca().xaxis.set_major_locator(MultipleLocator(0.01))
plt.gca().xaxis.set_minor_locator(MultipleLocator(0.001))
plt.gca().yaxis.set_major_locator(MultipleLocator(1))
plt.gca().yaxis.set_minor_locator(MultipleLocator(0.1))
plt.grid(color='r', linestyle='--', linewidth=0.75)
```

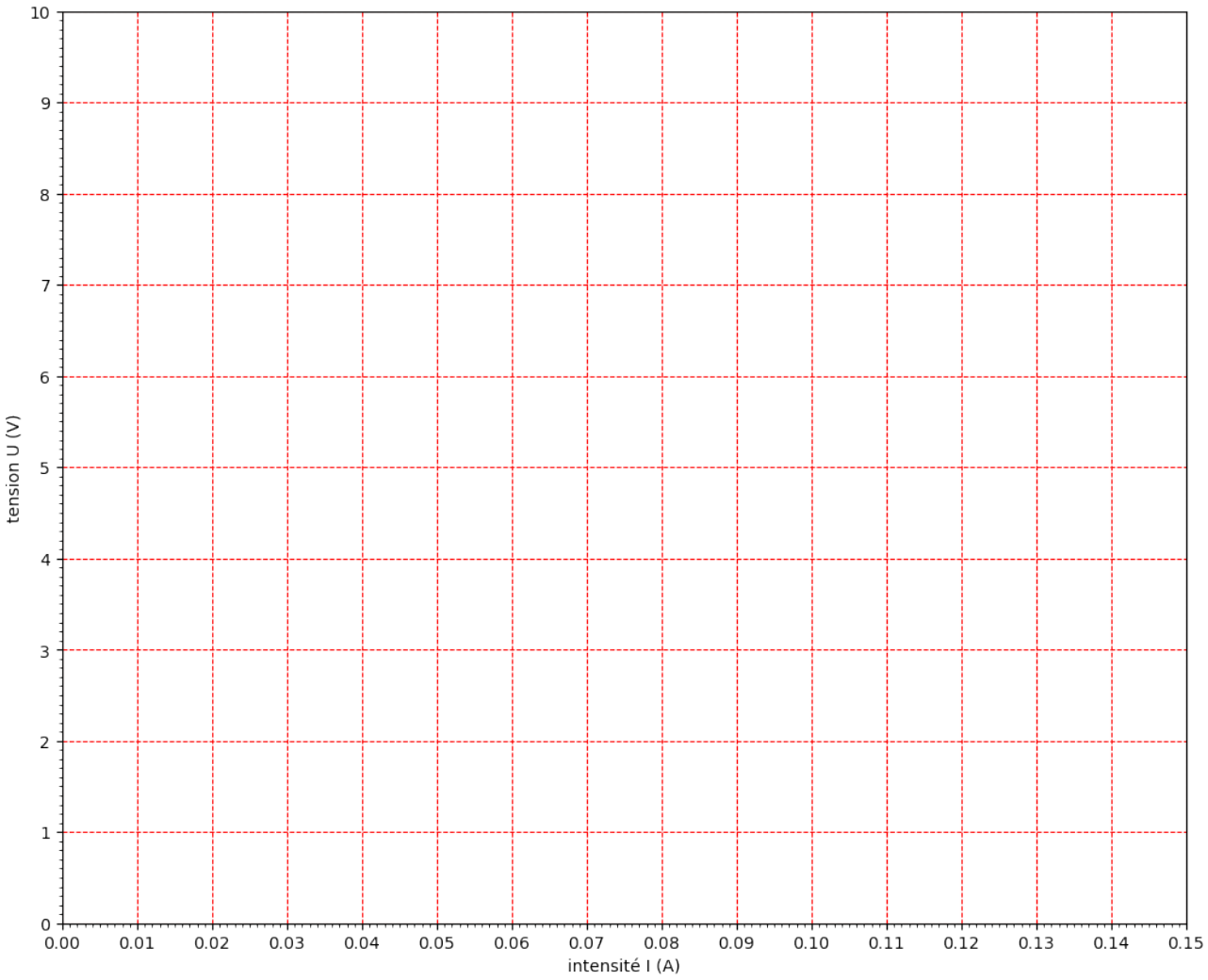


Troisième exemple avec les méthodes plt.xlim() et plt.ylim() qui donne un résultat identique au précédent :

In [11]:

```
plt.figure(3, figsize=(12,10), dpi=100)
plt.xlim(0,0.150)
plt.ylim(0,10)
```

```
plt.xlabel("intensité I (A)")
plt.ylabel("tension U (V)")
plt.gca().xaxis.set_major_locator(MultipleLocator(0.01))
plt.gca().xaxis.set_minor_locator(MultipleLocator(0.001))
plt.gca().yaxis.set_major_locator(MultipleLocator(1))
plt.gca().yaxis.set_minor_locator(MultipleLocator(0.1))
plt.grid(color='r', linestyle='--', linewidth=0.75)
```



Ecrire un titre et/ou un texte sur le graphique

On utilise la méthode **plt.title("Titre",...)** et/ou la méthode **plt.text(x,y,"Texte", ...)** de la bibliothèque matplotlib.pyplot as plt.

x, y sont les coordonnées du texte sur le graphique

Voici quelques paramètres pouvant être ajoutés pour modifier le texte affiché :

- fontsize: la taille de la police de caractères
- family: le type de police ('serif', 'sans-serif', 'monospace').
- fontweight : l'épaisseur de la police ('normal', 'bold', 'heavy', 'light', 'ultrabold', 'ultralight').

- style : le style de la police (‘normal’, ‘italic’, ‘oblique’).
- color : la couleur de la police.
- backgroundcolor : la couleur du fond.
- horizontalalignment : permet de centrer le texte (‘left’, ‘center’, ‘right’). Attention : ‘left’ veut dire que c’est la partie gauche du texte qui est positionnée au centre du graphique donc cette commande décale le texte vers la droite.
- alpha : permet de moduler la transparence du texte (0 : transparent ; 1 : opaque)

In [16]:

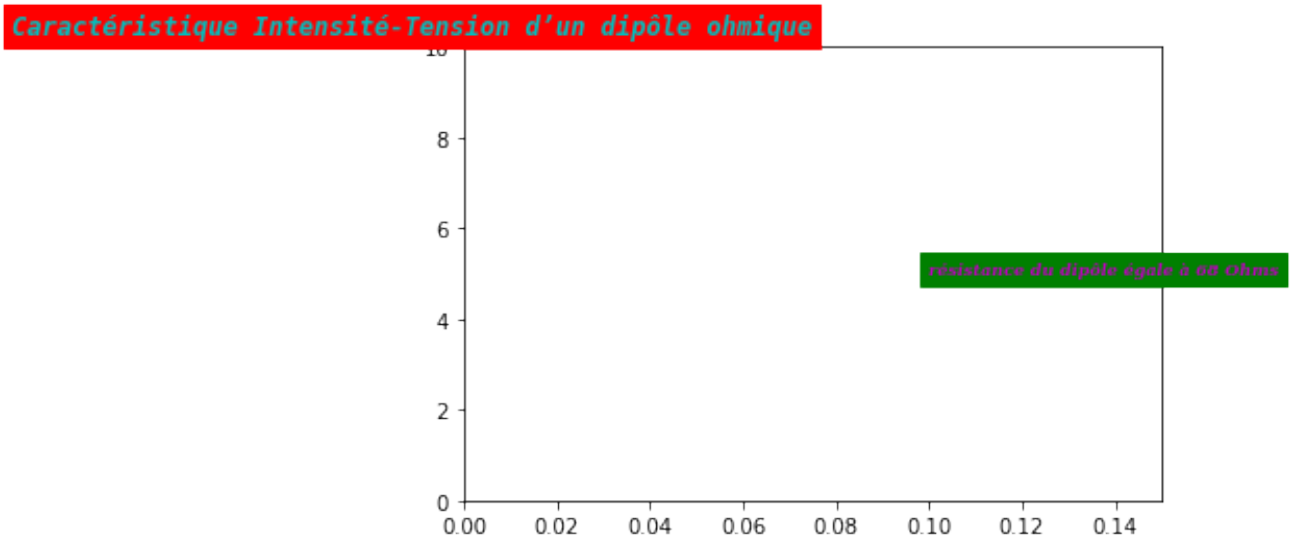
```
plt.xlim(0,0.150)
plt.ylim(0,10)

plt.title("Caractéristique Intensité-Tension dun dipôle ohmique",
          fontsize=12,family='monospace',fontweight='bold',
          style='italic',color='c',backgroundcolor='r',
          horizontalalignment='right')

plt.text(0.100,5,'résistance du dipôle égale à 68 Ohms',
         fontsize=8,family='serif',fontweight='heavy',
         style='oblique',color='m',backgroundcolor='g',alpha=1)
```

Out [16]:

```
Text(0.1, 5, 'résistance du dipôle égale à 68 Ohms')
```



Afficher la fenêtre graphique

On utilise la méthode **plt.show()** de la bibliothèque matplotlib.pyplot as plt en fin de programme afin d’afficher la fenêtre graphique réalisée précédemment.