Notebook

April 4, 2019

1 Comment créer un graphique sous Python? (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:

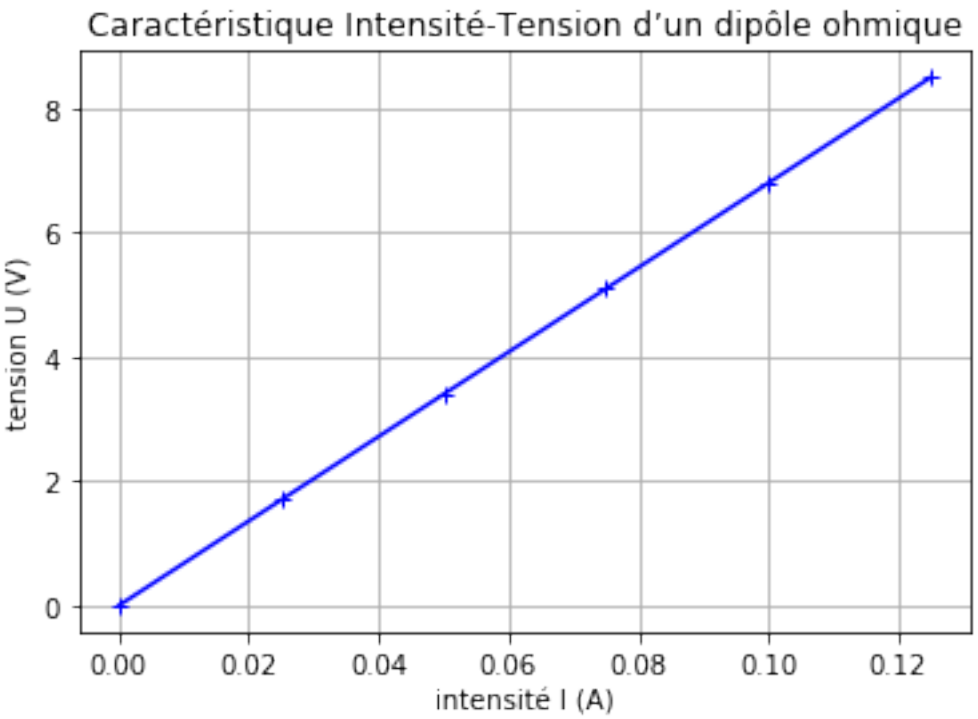
I(mA)	0	25	50	75	100	125
U(V)	0	1,7	3,4	5,1	6,8	8,5

Dans un premier temps, vous allez exécuter les deux programmes fournis en observant les différences notables. Si le graphique ne s'affiche pas à la première exécution, exécuter la cellule une deuxième fois.

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

Premier programme (le plus simple)

In [2]:
```
import matplotlib.pyplot as plt
I=[0,25e-3,50e-3,75e-3,100e-3,125e-3]
U=[0,1.7,3.4,5.1,6.8,8.5]
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 [ ]: import matplotlib.pyplot as plt
        from matplotlib.ticker import MultipleLocator
        I=[0,25e-3,50e-3,75e-3,100e-3,125e-3]
        U=[0,1.7,3.4,5.1,6.8,8.5]
        plt.figure("loi d'Ohm", figsize=(12,10), dpi=150)
        plt.plot(I,U,color='b', marker = '+',markersize = 12, linestyle='-.', linewidth = 2, la
        plt.legend(loc=2)
        plt.xlim(0,0.150)
        plt.ylim(0,10)
        plt.xlabel("intensité I (A)")
        plt.ylabel("tension U (V)")
        plt.axes().xaxis.set_major_locator(MultipleLocator(0.01))
        plt.axes().xaxis.set_minor_locator(MultipleLocator(0.001))
        plt.axes().yaxis.set_major_locator(MultipleLocator(1))
        plt.axes().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='mo
        ,color='m',backgroundcolor='y',horizontalalignment='center')
        plt.text(0.100,5,'résistance du dipôle égale à 68 Ohms',fontsize=8,family='cursive',fon
        ,color='m',backgroundcolor='c',alpha=1)
        plt.show()
```

1.1 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.

1.2 Import des bibliothèques utiles pour la création de graphiques

```
In [ ]: import matplotlib.pyplot as plt                # bibliothèque obligatoire
        import numpy as np                             # utile pour créer une liste de graduati
        from matplotlib.ticker import MultipleLocator # utile uniquement pour graduer les axes
```

1.3 Création 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 [ ]: I=[0,25e-3,50e-3,75e-3,100e-3,125e-3]      # création d'une liste pour la grandeur por
                                                    # ici l'intensité est convertie en Ampère
        U=[0,1.7,3.4,5.1,6.8,8.5]                  # création d'une liste pour la grandeur por
```

1.4 Comment 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. D'autres paramètres existent, pour plus d'informations : https://matplotlib.org/api/_as_gen/matplotlib.pyplot.figure.html#matplotlib.pyplot.figure

- 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 4 du premier programme et ligne 5 du second programme)

1.5 Comment 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 : 'b': bleu, 'g': vert, 'r': rouge, 'c': cyan, 'm': magenta, 'y': jaune, 'k': noir, 'w': blanc - marker : forme des points (marqueurs) - '.' marqueur de point - ',' marqueur de pixel - 'o' marqueur de cercle - '+' marqueur plus - 'x' marqueur x - 'v' marqueur triangle_down - '^' marqueur triangle_up - '<' marqueur triangle_left - '>' marqueur triangle_right - '1' marqueur tri_down - '2' marqueur tri_up - '3' marqueur tri_left - '4' marqueur tri_right - 's' marqueur carré - 'p' marqueur du pentagone - '*' marqueur étoile - 'h' marqueur hexagonal - 'H'

marqueur hexagonal - 'D' marqueur de diamant - 'd' marqueur mince - '|' marqueur vline - '_' marqueur hline - markersize : taille du marqueur - linestyle : style de la ligne - '-' style de trait plein - '–' style de ligne en pointillé - '-.' style de trait pointillé - ':' style en pointillé - linewidth : épaisseur de la ligne - label : permet de légender la courbe

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 le label à l'écran, il faut utiliser la méthode **plt.legend(loc)**. Le paramètre loc permet de choisir l'emplacement du label sur le graphique:

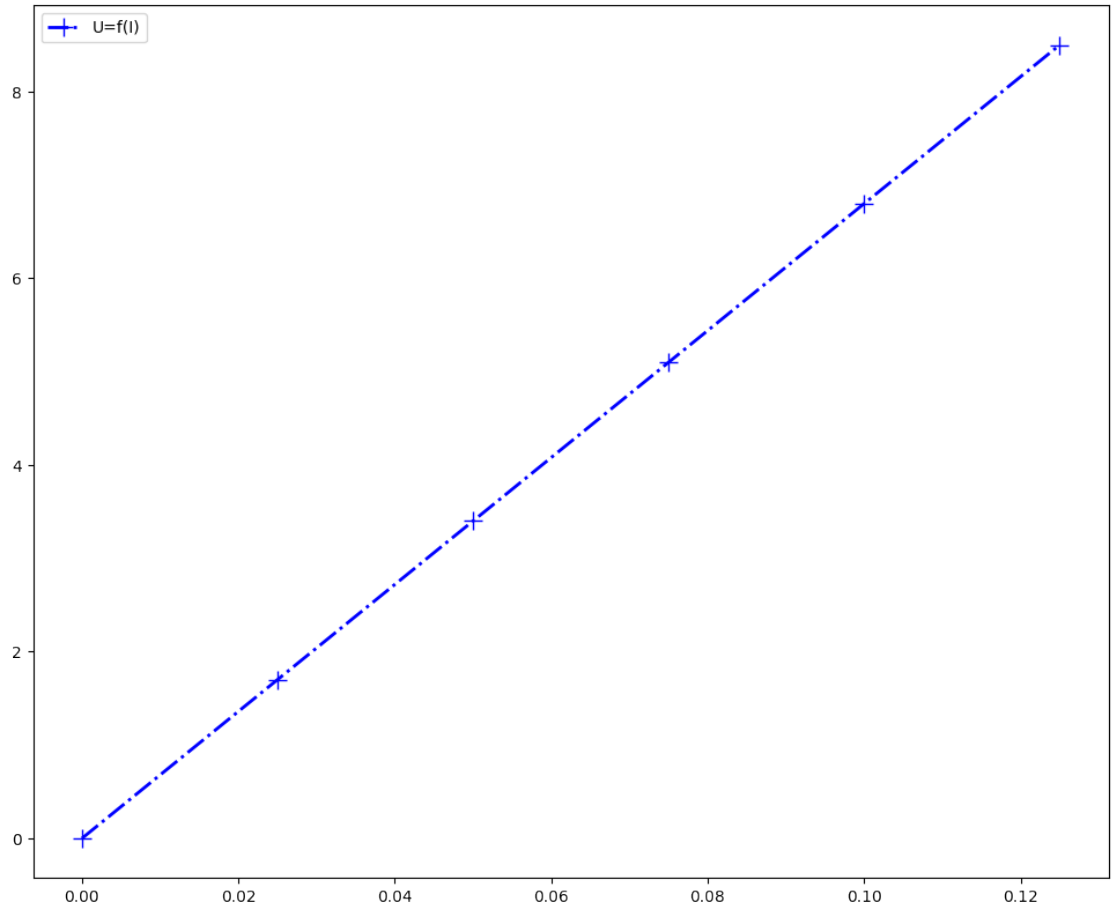
Emplacement et code d'emplacement - meilleur emplacement en fonction du graphique 0 - en haut à droite 1 - en haut à gauche 2 - en bas à gauche 3 - en bas à droite 4 - droite 5 - centre gauche 6 - centre droit 7 - centre bas 8 - centre supérieur 9 - centre 10

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

```
Out[7]: <matplotlib.legend.Legend at 0x7f75b6fa7240>
```



1.6 Comment 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 bibliothèque MultipleLocator : **from matplotlib.ticker import MultipleLocator**
   - utiliser les méthodes pour les graduations majeures :
     - **plt.axes().xaxis.set_major_locator(MultipleLocator(valeur d'une graduation))**
     - **plt.axes().yaxis.set_major_locator(MultipleLocator(valeur d'une graduation))**
   - utiliser les méthodes pour les graduations mineures :
     - **plt.axes().xaxis.set_minor_locator(MultipleLocator(valeur d'une graduation))**
     - **plt.axes().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 : 'b': bleu, 'g': vert, 'r': rouge, 'c': cyan, 'm': magenta, 'y': jaune, 'k': noir, 'w': blanc - linestyle : le style des lignes - '-' style de trait plein - '–' style de ligne en pointillé - '-.' style de trait pointillé - ':' style en pointillé - linewidth : l'épaisseur des lignes

D'autres paramètres existent. Pour plus d'informations :https://matplotlib.org/api/_as_gen/matplotlib.pyplot.grid.html

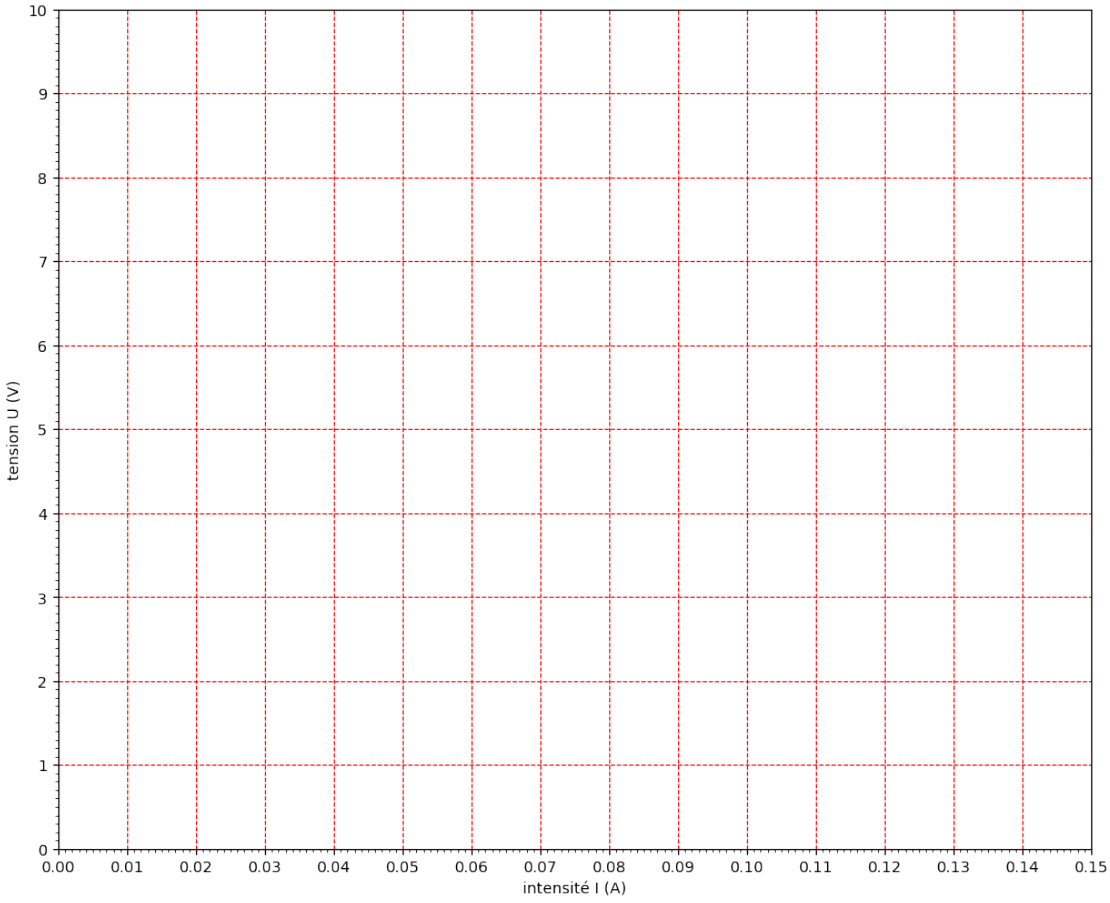
Pour plus d'informations sur les couleurs : https://matplotlib.org/tutorials/colors/colors.html#sphx-glr-tutorials-colors-colors-py

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

```
In [ ]: 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)
```

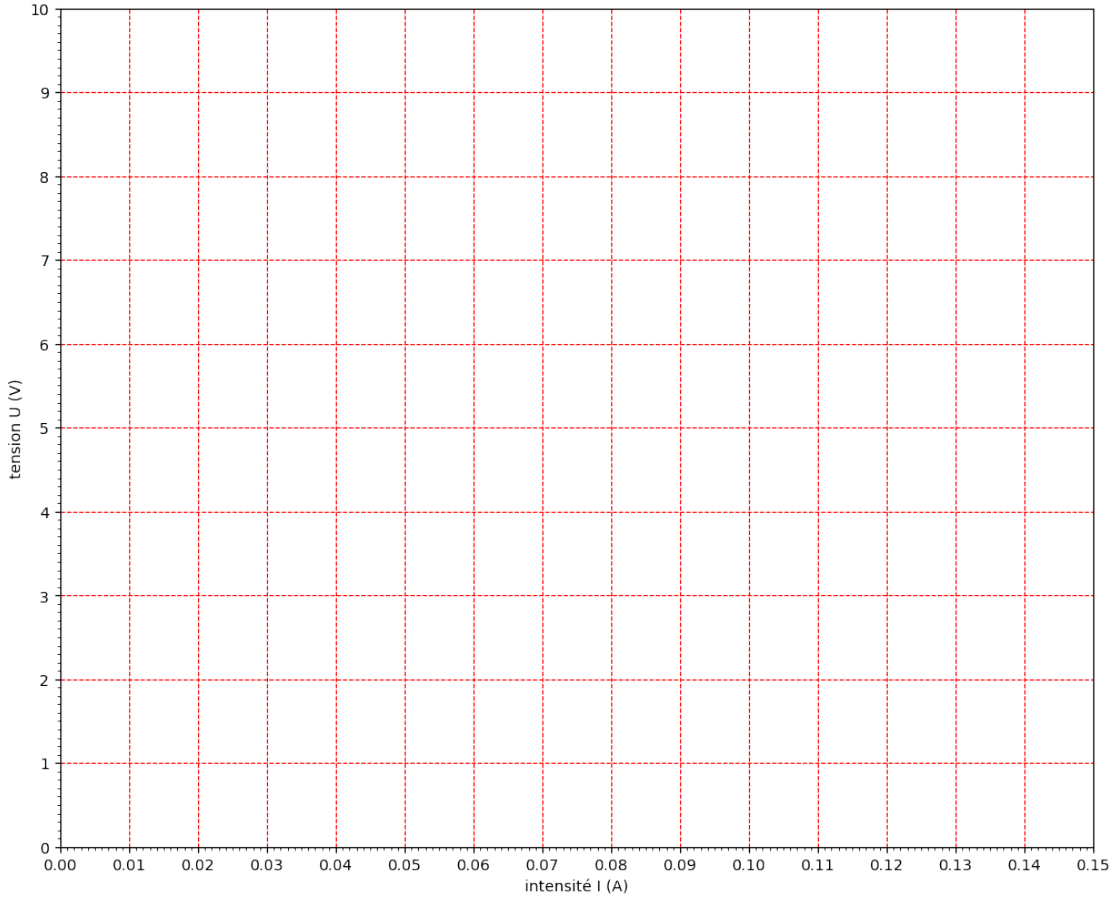
Deuxième exemple avec les méthodes plt.axes().xaxis.set_major_locator(MultipleLocator()) etc...

```
In [9]: 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.axes().xaxis.set_major_locator(MultipleLocator(0.01))
        plt.axes().xaxis.set_minor_locator(MultipleLocator(0.001))
        plt.axes().yaxis.set_major_locator(MultipleLocator(1))
        plt.axes().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 [10]: 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.axes().xaxis.set_major_locator(MultipleLocator(0.01))
         plt.axes().xaxis.set_minor_locator(MultipleLocator(0.001))
         plt.axes().yaxis.set_major_locator(MultipleLocator(1))
         plt.axes().yaxis.set_minor_locator(MultipleLocator(0.1))
         plt.grid(color='r', linestyle='--', linewidth=0.75)
```



1.7 Comment écrire 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', 'cursive', 'fantasy', '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 [ ]: plt.xlim(0,0.150)
        plt.ylim(0,10)

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

        plt.text(0.100,5,'résistance du dipôle égale à 68 Ohms',fontsize=8,family='cursive',fo
        ,color='m',backgroundcolor='g',alpha=1)
```

1.8 Comment 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.