CN1 Utilisation de matplotlib.pyplot : Tracé d'un graphe ou d'un histogramme, régression linéaire et barre d'incertitudes

1. Importation de la bibliothèque matplotlib.pyplot

On utilise l'alias plt pour appeler la bibliothèque.

```
import matplotlib.pyplot as plt # Pour tracer des graphiques
```

2. Tracé d'un graphe

Entrée des données expérimentales dans un tableau **array**, grâce à la bibliothèque **numpy** on crée un np.array à 1 dimension pour chacune des grandeurs (abscisses et ordonnées).

```
x = np.array([rentrer les valeurs de x séparées par des virgules])
#contient les valeurs x auxquelles ont été mesurées celles de y
y = np.array([rentrer les valeurs de y séparées par des virgules])
#ce sont les valeurs de y correspondantes
```

Il est important de rentrer le même nombre de valeurs de x et de y et dans le même ordre.

On utilise la fonction matplotlib.pyplot noté plt pour tracer le graphe, nommer les axes, l'accompagner d'un titre et d'une légende.

```
plt.plot(x,y,'b+')   #représenter y en fonction de x avec des croix bleues
plt.xlabel('légende à adapter') #légende de l'axe des abscisses
plt.ylabel('légende à adapter') #légende de l'axe des ordonnées
plt.title('titre à adapter') #titre
plt.legend() #afficher la légende (utile si plusieurs courbes sur le même graphique)
plt.grid() #afficher un quadrillage
plt.show()
```

Pour personnaliser la courbe on peut jouer (entre autres) sur le style de point ou de trait et sur sa couleur.

```
plt.plot(x,y,'couleur style_point et/ou style_trait')
```

avec
- `couleur` : première lettre anglaise de la couleur (b, g, y, c, r, m, w ou k pour le noir)
- `style_point` : (les principaux)

paramètre	x	X	+	P	.	o	*	d
marqueur	croix	croix plein	plus	plus plein	pixel	rond	étoile	carreau

- `style_trait` :

paramètre	:	-	-.	–	,	v
trait	pointillé	ligne continue	point tiret	ligne de tirets	pixel	triangle

exemple : `plt.plot(x,y,'g+-')` si on veut tracer la courbe ET les points, en vert ici...

remarque : `plt.plot(x,y,linestyle='none', color='black', marker='+')` a le même effet que `plt.plot(x,y,'k+')`

remarque : on peut aussi influer sur la taille des points avec `markersize=10` (par exemple) .

3. Tracé d'un histogramme

Pour tracer un histogramme avec Python, on utilise la fonction plt.hist

```
#Représentation d'un histogramme des effectifs
plt.hist(x, bins='rice') # Représentation de l'histogramme des valeurs x /
   avec optimisation du nombre de classes.
plt.title('Histogramme de x')
plt.xlabel("x (préciser l'unité)")
plt.ylabel('Effectifs')
plt.show()
```

4. Ajouter Les barres d'incertitude

Pour ajouter les barres d'incertitude-type sur un graphe, on utilise la fonction errorbar

```
# Au préalable renseigner les grandeurs x et y dans un array

#  Ajout des barres d'incertitude
 u_x= # valeur ou tableau contenant les incertitudes-types sur x
 u_y = # valeur ou tableau contenant les incertitudes-types sur y
 plt.errorbar(x, y, xerr=u_x, yerr=u_y, fmt='go') # Tracé du nuage de points en vert
 avec les barres d'incertitude en x et y.
```

5. Faire une régression linéaire

Il y a plusieurs possibilités pour faire une régression linéaire (qui permet de savoir si une relation linéaire de type y =ax+b existe entre 2 variables x et y, a est alors la pente de la courbe et b l'ordonnée à l'origine. Nous présentons ici le script permettent de faire une régression linéaire avec la fonction np.polyfit de la bibliothèque numpy.

```
# au préalable importer numpy et matplotlib.pyplot
# au préalable renseigner les grandeurs x et y dans des array et faire tracer le graphe.
# au préalable ajouter les barres d'incertitude.

# Régression linéaire de y en fonction de x : équation y = p[0]*x+p[1], qui est un
polynôme d'ordre 1 p[0]correspond à la pente et p[1]correspond à l'ordonnée à l'origine
 p=np.polyfit(x, y, 1)
 pente=p[0] # pente de la droite
 ord_origine=p[1] # ordonnée à l'origine
 plt.plot(x,np.polyval(p,x)) # Tracé de la droite de régression (p en fonction de x)
 plt.title('régression linéaire')
 plt.xlabel('à adapter')
 plt.ylabel('à adapter')
 print('la pente vaut =',round(pente, nombre de CS à adapter))
 print('l ordonnée à l origine vaut =',round (ord_origine, nombre de CS à adapter))
 plt.show()
```