# Python pour la physique chimie

*Ce qu'il faut retenir de la seconde*

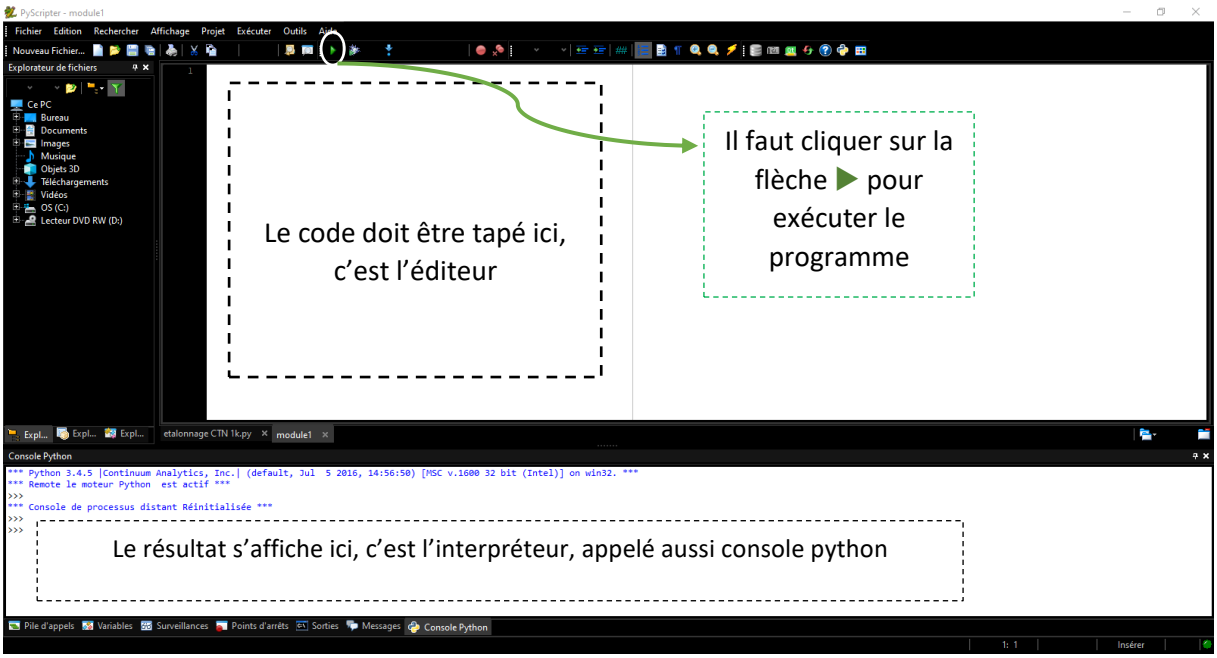
## I) Présentation

Le langage python permet de **faire des calculs** mais la plupart du temps en physique chimie de **représenter des graphiques** :

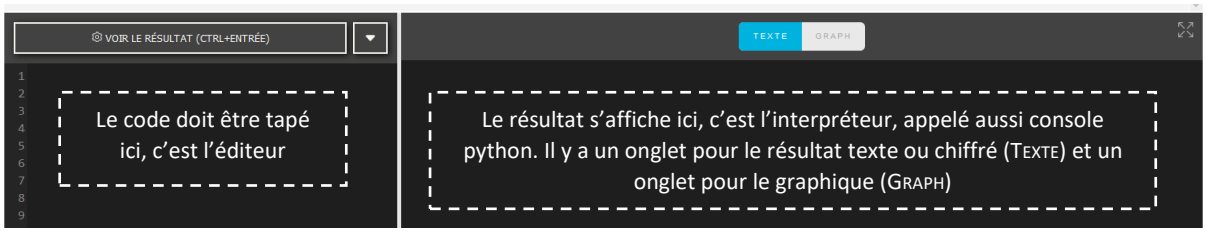
- **Nuages de points à modéliser (droite affine ou linéaire)**
- **Histogrammes**

Au lycée nous utilisons Edupython, mais vous pouvez utiliser la console du livre scolaire en ligne pour vous entrainer à la maison : lien : https://www.lelivrescolaire.fr/console-python

➔ **Interface sous Edupython :**




➔ **Interface du livre scolaire :**




Python est un langage interprété, les instructions du programme sont lues au fur et à mesure de l'exécution du programme.

## II) Les variables

### 1) Qu'est-ce qu'une variable ?

En informatique, une **variable** est une donnée que l'ordinateur stocke dans un espace mémoire.
**L'affectation** est le fait de donner une « valeur » à une variable.

Exemple : X prend la valeur 2, s'écrit en python :

```
X = 2
```

### 2) Quels sont les différents types de variables ?

- Les entiers appelés **int.**
- Les chiffres à virgules appelés **float** (attention : on utilise un point et pas une virgule pour écrire ces chiffres).
- Les chaines de caractères (mots) appelés **str**, ils sont notés entre guillemets simples **'……'** ou doubles **" ….. "**.
- Les listes appelées **list** de nombres entiers ou à virgules ou de mots et notées entre crochets [……] séparés par des **virgules**.
- Les booléens notés **bool** qui sont de deux types : **true** ou **false**.

*Remarque : attention aux guillemets à bien les retaper dans python car souvent les copier-coller à partir d'un fichier gardent la mise en page et le guillemet n'est pas reconnu par python.*

**Exercice :**

Déterminer le type de variable dans les cas suivants :

a) 2
b) 4.5
c) 'bonjour'
d) [1,2,3,4,5]

Réponses : a) int b) float c) str d) list

### 3) Les opérations usuelles sur les variables

| Addition | Soustraction | Multiplication | Division | Puissance |
|---|---|---|---|---|
| + | - | * | / | ** |

**Exercice :** Prévoir le résultat des calculs suivants et indiquer le type de variable renvoyé.

a) 5 + 2
b) 'cou' + 'cou'
c) 10 – 7
d) 4.5**2
e) 2 * 4
f) 'cou' * 2
g) 1 / 3
h) 2**3

Réponses :
a) 7 (int) b) coucou (str) c) 3 (int) d) 20.25 (float) e) 8 (int) f) coucou (str)
g) 0.33333 (float) h) 8 (int)

**Exercice :** Compléter par le résultat du calcul écrit en Python

a) 3 + 4*5
b) 6**2
c) 5*(2**3)
d)
```
X = 2
X = X +1
```
Que vaut X ?

e)
```
L = 2.5
l = 1.5
A = L*l
```
Que vaut A ?

Réponses : a) 23 b) 36 c) 40 d) X = 3 e) A = 3.75

## III) Les objets et les modules

Pour travailler avec certains objets moins courants, comme π, la racine carrée, le cosinus, le sinus, tracer des graphiques, modéliser des courbes … il faut aller chercher ces « objets » dans des « modules » : on dit qu'on les **importe**.

**▶ Les modules fréquemment utilisés en physique chimie sont :**

- **math**
- **matplotlib**
- **scipy**
- **numpy**
- **random**

## ▶ On importera les modules des deux façons suivantes :

**import module as *alias***

On importe toutes les commandes d'un module en lui donnant un alias. L'alias permet de raccourcir le nom du module importé puisque le nom de la commande sera précédé du nom du module.

**Exemple :** on veut calculer 2 x π x 6 et faire afficher le résultat (cela se fait grâce à **print()**).
Les lignes de commentaires sont précédées d'un **#**, que la machine ne prend pas en compte.

```
import math as mat
C = 2*mat.pi*6
print(C)
```

```
# on importe toutes les commandes du module math et on lui donne un alias qui est mat
# on souhaite calculer la circonférence d'un cercle de rayon 6 m. Pour cela on a besoin de π
# que l'on va chercher dans le module math la commande pi, ce que l'on note mat.pi
# print(C) affichera dans la console python la valeur calculée pour C = 2 x π x 6
```

**from module import***

On importe toutes les commandes d'un module. On peut ensuite utiliser n'importe laquelle dans la suite du code.

**Exemple :** on veut calculer $\sqrt{2 \times \pi \times 6}$ et faire afficher le résultat. La racine carrée se note sqrt.

```
from math import*
L = sqrt(2*pi*6)
print(L)
```

```
# on importe toutes les commandes du module math
# on affecte à L la valeur du calcul
# on affiche la valeur calculée pour L dans la console python
```

**Exercice :** aller sur la console python du livre scolaire ou utiliser Edupython et tester ces deux codes. Noter le résultat renvoyé dans la console python.


## ▶ Comment connaitre le contenu d'un module ?

**Pour obtenir le contenu d'un module, on exécute les instructions suivantes :**

```
import math
print(dir(math))
```

On obtient dans la console python :

```
>>>
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan',
'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial',
'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'hypot', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10',
'log1p', 'log2', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
>>>
```

On voit qu'entre autres dans le module math, on trouve cos, sin, tan, sqrt et pi.

**Pour avoir une description plus détaillée d'un module, on exécute les instructions suivantes :**

```
import math
help(math)
```

Dans la console python, on obtient entre autres pour le cosinus :

```
cos(...)
    cos(x)

    Return the cosine of x (measured in radians).
```

On peut convertir un angle de radians en degrés avec la fonction degrees() :

```
degrees(...)
    degrees(x)

    Convert angle x from radians to degrees.
```

## IV) Les fonctions

**Une fonction** est une portion de code que l'on peut appeler au besoin.
Les fonctions permettent d'éviter les répétitions dans le code.

Synthaxe :

```
def nom de la fonction (argument1, argument2, ….) :
        Séquence d'instructions éventuelle
        return résultat1, résultat2, …
```

**Exemple 1 :** on définit la fonction poids qui permet de calculer le poids d'un objet connaissant sa masse m.

```
def poids(m) :
        return m*9.81
```

L'appel poids(60) renvoie 588.6, c'est-à-dire la valeur du poids d'un objet de masse 60 kg.

**Exemple 2 :** on définit la fonction Ec qui permet de calculer l'énergie cinétique d'un système connaissant sa masse m et sa vitesse v.

```
def Ec (m,v)
        Ec = 0.5*m*v**2
        return Ec
```

L'appel Ec(150,14) renvoie 14700, c'est-à-dire la valeur de l'énergie cinétique Ec (en joule) pour un système de masse 150 kg et de vitesse 14 $m.s^{-1}$.

# V) Les listes

## 1) Présentation

**Une liste** est une collection ordonnée (on dit aussi une séquence) d'éléments **séparés par des virgules**, le tout étant **entre crochets**.

**Exemples :**

```
L = []                  # La liste L est vide
L = ['sos', 17, 6.8]    # la liste L contient du texte (str), un entier (int) et un chiffre à virgule (float).
L = list(range(5))      # crée une liste contant 5 valeurs en partant de 0 et non pas de 1 : [0,1,2,3,4]
```

## 2) Indexation

La collection est ordonnée : on accède à un élément d'une liste à l'aide de son index (c'est-à-dire sa position - 1 dans la liste).

Extraction de données :
- L[-1] renvoie la dernière donnée de L
- L[-2] renvoie l'avant-dernière donnée de L

**Exercice :**
On crée la liste suivante :

```
L = ['sos', 17, 6.8]
```

Qu'obtient-on si on tape print(L[2]) ? print(L[0]) ? print(L[-1]) ?

Réponses : a) 6.8 b) sos c) 6.8

## 3) Calculs sur les listes

▶ Utilisation de la boucle bornée : **for** i **in range** (m, n)
*instructions*

pour i allant de m à n-1, on répète les instructions qui figurent dans le bloc indenté (décalé vers la droite)

**Exemple :** On souhaite créer une liste 2 x V à partir d'une liste contenant 5 valeurs de volumes V.

```
volume = [25,30,35,40,45]          # la liste volume contient les 5 valeurs de volumes
doubleVolume = []                  # on crée une liste vide nommée doubleVolume

for i in range (0,5) :             # pour i allant de 0 à 4 :
    doubleVolume.append(2*volume[i])   # calculer pour chaque i : 2 x V et rajouter la valeur à la liste doubleVolume
print(doubleVolume)                # afficher la liste doubleVolume
```

Aller sur la console python du livre scolaire ou utiliser Edupython et tester ce code. Noter le résultat renvoyé dans la console python.

Résultat :

[50, 60, 70, 80, 90]

Pour ne pas avoir tout simplement tapé les instructions ci-dessous ?

```
volume = [25,30,35,40,45]
doubleVolume = volume*2
print(doubleVolume)
```

Répondre à la question puis tester ce code sur la console python du livre scolaire.

Résultat : on obtient 2 fois la liste volume écrite. Il n'y a pas d'opération sur le contenu de la liste

[25, 30, 35, 40, 45, 25, 30, 35, 40, 45]

## ▶ Utilisation de la fonction len() :

On va maintenant remplacer *for i in range (0, 5)*, par *for i in range(**len**(L))*

```
volume = [25,30,35,40,45]
doubleVolume = []

for i in range (len(volume)) :
    doubleVolume.append(2*volume[i])
print(doubleVolume)
```

Exécuter ce code dans la console python. Commenter le résultat et en déduire l'action de la fonction len()

Résultat : on obtient le même résultat dans les deux cas.

[50, 60, 70, 80, 90]

La fonction len() renvoie la longueur de la liste. Ici for i in range(len(L)) fait varier i de 0 au nombre de valeurs -1 de la liste, soit de 0 à 4, ce qui fait bien 5 valeurs.

## ▶ Une dernière méthode pour faire des calculs sur une liste :

```
volume = [25,30,35,40,45]
doubleVolume = [2*V for V in volume]
print(doubleVolume)
```

Exécuter le code dans la console python et vérifier que l'on trouve bien le même résultat.

# VI) Les graphiques

Pour tracer un graphique en langage python il faut importer le module **matplotlib.pyplot**, l'alias donné est souvent **plt**.

La syntaxe générale est :

```
import matplotlib.pyplot as plt
instructions pour les tracés
plt.show()
```

ou

```
from matplotlib import pyplot as plt
instructions pour les tracés
plt.show()
```

Les commandes utilisées doivent être précédées de plt (l'alias).

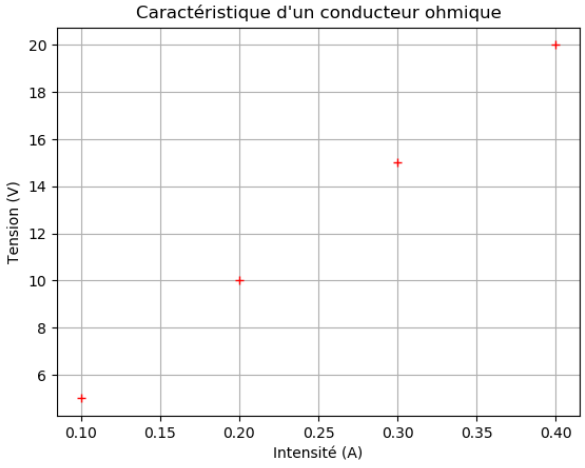
## 1) Tracer un nuage de points à partir de deux listes

**Etude d'un exemple :** on souhaite tracer la caractéristique tension-intensité (U en fonction de I) pour in conducteur ohmique de résistance R.
On a à disposition deux **listes** contenant les différentes valeurs de U et de I.

Exécuter le code suivant dans la console python :

```
1  import matplotlib.pyplot as plt          # on importe le module souhaité
2
3  U = [5.0,10.0,15.0,20.0] # liste contenant les valeurs de la tension aux bornes du conducteur ohmique
4  I = [0.100,0.200,0.300,0.400] # liste contenant les valeurs correspondantes de l'intensité du courant
5
6  plt.plot(I,U,'r+')                       # on trace des points rouges représentés par des +
7  plt.xlabel("Intensité (A)")              # on légende l'axe des abscisses
8  plt.ylabel("Tension (V)") # on légende l'axe des ordonnées
9  plt.title("Caractéristique d'un conducteur ohmique") # on légende le graphique
10 plt.grid()                               # on affiche un quadrillage
11 plt.show()                               # on affiche le graphique
```

Une fenêtre s'affiche automatiquement dans Edupython, sinon sur la console du livre scolaire, cliquer sur l'onglet GRAPH.



**Les commandes principalement utilisées sont :**

| | |
|---|---|
| **plt.plot(x,y, 'r+')**<br><br>  *Tester les changements possibles* | Place les points de coordonnées y en fonction de x en rouge et les points sont représentés par des +<br><br>*Changements possibles :*<br>▶ pour changer de couleur : r pour rouge, g pour vert, b pour bleu, y pour jaune, **k** pour noir.<br>▶ pour changer de forme : **o** pour des ronds, **+** pour des plus, ***** pour des croix, **s** pour des carrés et **^** pour des triangles<br>▶ pour relier les points « à la main » en pointillés : rajouter **linestyle = ':'** ou '-' pour un train plein.<br>Exemple :<br>plt.plot(x,y,linestyle = ':', linewidth = 2, color = 'k', marker = '+') |
| Equivalents<br>**plt.scatter(x, y, c='r', marker='+', label = 'points expérimentaux')** | Place les points de coordonnées y en fonction de x, le rouge, les points sont représentés par des + et on légende le nuage de points en l'appelant « points expérimentaux ». |
| plt.xlabel("Ecrire la légende")<br>plt.ylabel("Ecrire la légende") | Permet de légender l'axe des abscisses<br>Permet de légender l'axe des ordonnées |
| plt.title("Titre du graphique") | Permet de légender le graphique |
| plt.grid() | Affiche un quadrillage |
| plt.axis([a,b,c,d])<br><br>  *Tester différentes dimensions des axes* | Définit les dimensions des axes :<br>$a \leq x \leq b$ et $c \leq y \leq d$ |

## 2) Tracer un nuage de points à partir de deux tableaux

**Etude du même exemple :** on souhaite tracer la caractéristique tension-intensité (U en fonction de I) pour un conducteur ohmique de résistance R.
On a à disposition deux **tableaux** contenant les différentes valeurs de U et de I.

```
1  import matplotlib.pyplot as plt
2  import numpy as np                  # pour se servir des tableaux il faut importer numpy
3
4  U = np.array([5.0,10.0,15.0,20.0]) # tableau contenant les valeurs de la tension aux bornes du conducteur
5  ohmique
6  I = np.array([0.100,0.200,0.300,0.400]) # tableau contenant les valeurs correspondantes de l'intensité du
7  courant
8
9  plt.plot(I, U,'r+')                 # on trace la courbe U en fonction de I, les points sont des + rouges
10 plt.xlabel("Intensité (A)")          # on légende l'axe des abscisses
11 plt.ylabel("Tension (V)")            # on légende l'axe des ordonnées
12 plt.title("Caractéristique d'un conducteur ohmique") # on légende le graphique
13 plt.grid()                          # on affiche un quadrillage
   plt.show()                          # on affiche le graphique
```

Exécuter le code précédent dans la console python : le même graphique que précédemment s'affiche.

### 3) Modéliser un nuage de points

Pour trouver l'équation de la droite modélisant le nuage de points précédent on peut utiliser :
coeff=np.polyfit(x,y,1) ou droite=sc.linregress(x,y)

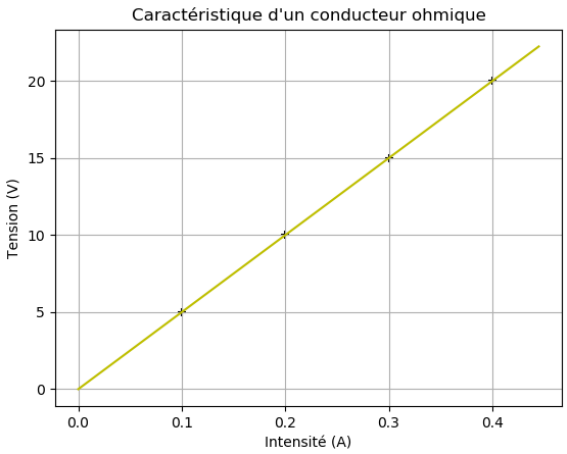
#### A) Première méthode

```
1  import matplotlib.pyplot as plt
2  import numpy as np    # pour se servir de polyfit il faut importer numpy
3
4  # Création des listes
5  U = [5.0,10.0,15.0,20.0] # liste contenant les valeurs de la tension aux bornes du conducteur
6  ohmique
7  I = [0.100,0.200,0.300,0.400] # liste contenant les valeurs correspondantes de l'intensité du
8  courant
9
10 # affichage du nuage de points :
11 plt.scatter(I, U, c='k', marker='+') # on affiche les points expérimentaux sous forme de + noirs
12 plt.xlabel("Intensité (A)") # on légende l'axe des abscisses
13 plt.ylabel("Tension (V)") # on légende l'axe des ordonnées
14 plt.title("Caractéristique d'un conducteur ohmique") # on légende le graphique
15 plt.grid() # on affiche un quadrillage
16
17 # partie régression linéaire :
18 coeff=np.polyfit(I,U,1) # on cherche une équation du type y=ax+b du 1er ordre
19 f=np.poly1d(coeff) # on définit la fonction f comme correspondant à l'équation trouvée f(x) =
20 ax + b
21 print (f) # on affiche l'expression de f
22
23 # on va tracer la droite du modèle trouvé
24 listeX=[] # on crée une liste vide pour les abscisses
25 listeY=[] # on crée une liste vide pour les ordonnées
26 for i in np.arange(0,0.45,0.005) :  # pour i variant de 0 à 0,45 exclu et par pas de 0,005 :
27     listeX.append(i) # on ajoute dans la liste des abscisses la valeur de i
28     listeY.append(f(i)) # et on ajoute dans la liste des abscisses la valeur de f(i), c'est à dire ai +b
29 plt.plot(listeX,listeY,'y') # on trace la droite modèle en jaune à partir des deux listes X et Y
30
31 plt.show() # on affiche la courbe
```

**Exercice :** exécuter ce code python et vérifier que vous retrouvez les résultats ci-après.
Quelle est l'équation de la droite U = f(I) ?

Résultats :

Fenêtre graphique :




Console python :

```
50 x
>>>
```

Solution : f = 50 x, d'où par identification : U = 50 x I

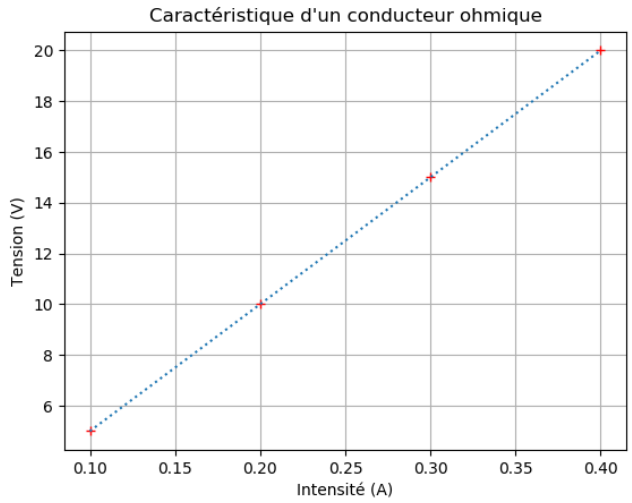
## B) Deuxième méthode

```
1  import matplotlib.pyplot as plt
2  import numpy as np    # pour se servir des tableaux il faut importer numpy
3  import scipy.stats as sc # pour se servir de la régression linéaire il faut importer scipy
4
5  # création des tableaux de valeurs :
6  U = np.array([5.0,10.0,15.0,20.0]) # tableau contenant les valeurs de la tension aux bornes du
7  conducteur ohmique
8  I = np.array([0.100,0.200,0.300,0.400]) # tableau contenant les valeurs correspondantes de
9  l'intensité du courant
10
11 # affichage du nuage de points :
12 plt.plot(I, U,'r+') # on trace la courbe U en fonction de I
13 plt.xlabel("Intensité (A)") # on légende l'axe des abscisses
14 plt.ylabel("Tension (V)") # on légende l'axe des ordonnées
1  plt.title("Caractéristique d'un conducteur ohmique") # on légende le graphique
16 plt.grid() # on affiche un quadrillage
17
18 # partie régression linéaire :
19 droite=sc.linregress(I,U) # effectue une régression linéaire
20 coef=droite.slope # attribue à la variable coef la valeur du coefficient directeur obtenu à la
21 régression linéaire
22 print("Le coefficient directeur est : ", coef) # permet d'afficher la valeur du coefficient directeur
23 appelé coef
24
25 # on va tracer la droite du modèle ayant pour coefficient directeur coef
26 Ureg=coef*I        # on calcule le U théorique connaissant I et le coef
27 plt.plot(I,Ureg,':') # construit la droite de régression linéaire
28
29 plt.show()         # affiche les courbes
```

**Exercice :** exécuter ce code python et vérifier que vous retrouvez les résultats ci-après.
Que vaut le coefficient directeur de la droite ? En déduire la valeur de la résistance R du conducteur ohmique (loi d'Ohm).

Résultats :

Fenêtre graphique :




Console python :

```
>>>
Le coefficient directeur est :  50.0
```

Solution :

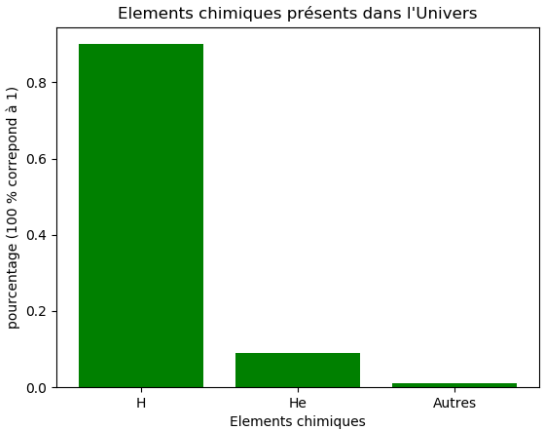
On a tracé U = f(I), or le coefficient directeur de cette droite affine vaut 50, on en déduit que U = 50 x I, soit comme U = R x I (loi d'Ohm), alors par identification il vient que R = 50 Ω

## VII) Les histogrammes (diagrammes bâtons)

**Etude d'un exemple :** l'Univers est constitué de 90 % d'hydrogène, 9 % d'hélium et de 1 % d'autres éléments. On souhaite construire le diagramme bâtons correspondant.

```
1  import matplotlib.pyplot as plt
2
3  x=["H","He","Autres"] # on crée une liste correspondant aux abscisses
4  hauteur=[0.9, 0.09,0.01] # on crée une liste correspondant aux ordonnées (pourcentages)
5  width = 1.0 # épaisseur des bâtons
6  plt.bar(x,hauteur,color='g') # on construit le diagramme bâtons qui sera vert
7  plt.title("Elements chimiques présents dans l'Univers") # on légende le diagramme bâtons
8  plt.xlabel("Elements chimiques") # on légende l'axe des abscisses
9  plt.ylabel("pourcentage (100 % correspond à 1)") # on légende l'axe des ordonnées
10 plt.show() # on affiche le diagramme bâtons
```

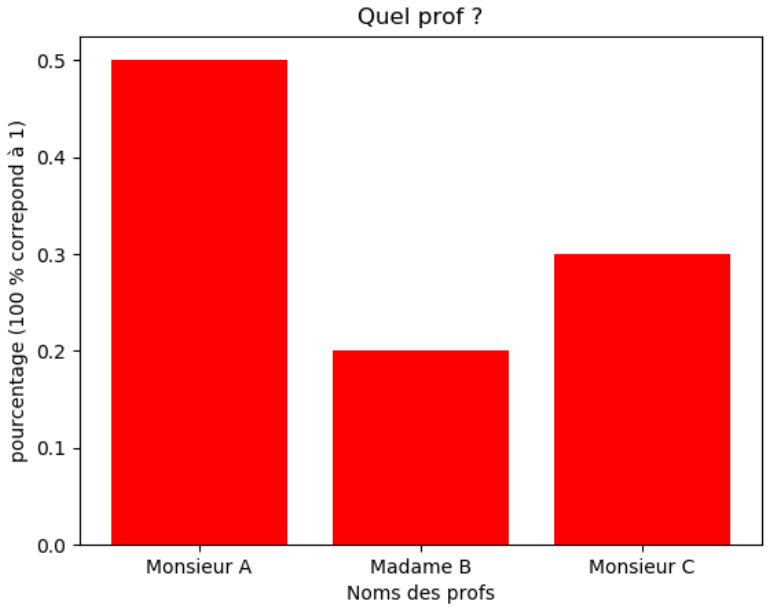
**Exercice :** exécuter ce programme.



**Exercice :** A l'aide du modèle précédent, modifier le programme pour qu'il affiche en rouge le diagramme bâtons correspondant à la situation suivante : Un élève a 50 % de chances de tomber sur Monsieur A l'an prochain, 20 % de chances de tomber sur Madame B et 30 % de chances de tomber sur Monsieur C.

Solution :

```
1  import matplotlib.pyplot as plt
2
3  x=["Monsieur A","Madame B","Monsieur C"] # on crée une liste correspondant aux abscisses
4  hauteur=[0.5, 0.2,0.3] # on crée une liste correspondant aux ordonnées (pourcentages)
5  width = 1.0 # épaisseur des bâtons
6  plt.bar(x,hauteur,color='r') # on construit le diagramme bâtons qui sera rouge
7  plt.title("Quel prof ?") # on légende le diagramme bâtons
8  plt.xlabel("Noms des profs") # on légende l'axe des abscisses
9  plt.ylabel("pourcentage (100 % correpond à 1)") # on légende l'axe des ordonnées
10 plt.show() # on affiche le diagramme bâtons
```



# Lexique Python

## 1) Affichage de texte, suite d'instructions

- print("texte") → permet d'afficher du texte et d'aller à la ligne.
  Exemple : print("Bonjour") permet d'afficher Bonjour
- print(nombre) → permet d'afficher un nombre
- print(calcul) → permet d'afficher le résultat d'un calcul
- print("J'ai", âge, "ans et je programme depuis", tempsProgrammation, "ans")

## 2) Répétitions d'instructions

- for <loop> in range(n): → boucle bornée qui permet de répéter n fois une instruction
  <instruction>

On décale l'instruction à répéter vers la droite avec des espaces, pour indiquer qu'elle « appartient » à l'instruction précédente. Effectuer des décalages comme ceci est très courant en programmation : on appelle ça l'**indentation**.

- print("texte", end = "") → permet d'afficher du texte sans retour à la ligne
- print() → permet d'aller à la ligne
- # une ligne de texte → permet d'insérer une ligne de commentaire qui sera ignorée par l'ordinateur
- """
  paragraphe → Permet d'insérer un paragraphe de commentaires qui sera ignoré par l'ordinateur.
  """

## 3) Lecture de l'entrée

- nbSachets = int(input()) → lit un nombre entier
- nbSachets = int(input("Nombre de sachets ? ")) → lit un nombre entier après avoir affiché du texte
- input()→ lit du texte
- float(input()) → lit un nombre à virgule

## 4) Tests et conditions

- if <condition 1> : → il s'agit d'une structure conditionnelle où le test d'une condition conduit à 2
  <instruction 1> instructions différentes selon si cette condition est vérifiée ou non.
  else :
  <instruction 2>

- = sert à affecter une valeur à une variable ;
- == sert à tester l'égalité de deux valeurs.
- Lorsqu'on veut uniquement tester si deux valeurs sont différentes, on utilise l'opérateur !=, qui se lit « *différent de* »