

Introduction à Python Physique-Chimie – Terminale

Objectif : utiliser Python
comme outil de calcul et de
représentation graphique

Fiche méthode « Python » sur sciencespartout.fr



Livre Hatier Rabat II et III (à la fin du livre)



<https://trinket.io/python3>

Python 3 Trinkets

The easiest way to use the full power of Python 3.

Exécution du code

Saisie de code

Résultats

```
main.py
4 from scipy.stats import linregress
5
6 # Tableaux des valeurs expérimentales de la concentration en tBuCl au cours
7 C_tBuCl = [0.22, 0.46, 0.66, 0.82, 0.96, 1.07, 1.16, 1.24, 1.31, 1.36, 1.41, 1.45, 1.49]
8 t = [0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 10, 11, 12, 13, 14, 15]
9
10 # Calcul de v et création du graphique v = f(C_tBuCl)
11 v = [] # crée une liste vide nommée v
12 for i in range(1, len(t) - 1):
13     v.append((C_tBuCl[i+1] - C_tBuCl[i]) / (t[i+1] - t[i]))
14 del t[0] # enlève la première valeur de la liste t
15 del t[-1] # enlève la dernière valeur de la liste t
16 del C_tBuCl[0] # enlève la première valeur de la liste C_tBuCl
17 del C_tBuCl[-1] # enlève la dernière valeur de la liste C_tBuCl
18 #print(v) # affiche les valeurs de v
19
20 plt.plot(C_tBuCl, v, "bo")
21
22 a, b, r, p_value, std_err = linregress(C_tBuCl, v) # regression lineaire
23
24 plt.plot # droite de regression
25 [C_tBuCl[0], C_tBuCl[-1]], # valeurs de x
26 [a * C_tBuCl[0] + b, a * C_tBuCl[-1]] # valeurs de y
27 # couleur rouge avec un trait continu
28 #label="regression : "+str(round(a,3))+ "x" + "+str(round(b,3)) # les
29 Label="regression : "+str(round(a,3))+ "x" + "+str(round(b,3)) # les
30
31 plt.xlabel("C_tBuCl (mol/L)") # légende de l'axe des abscisses
32 plt.ylabel("vitesse de disparition de tBuCl (mol.L-1.min-1)")
33 plt.title("v=f(C_tBuCl)") # titre du graphique
34 plt.legend() # la légende
35 plt.grid() # affiche un quadrillage pour ce graphique
36 plt.show() # affiche les graphiques
```

Edit and run the code, then click Share. There's no simpler way to write & share Python 3 code.

Objectifs

- Comprendre la logique de base d'un programme Python
- Manipuler des variables et des listes
- Utiliser des boucles et des conditions
- Tracer une trajectoire et calculer des vitesses à partir de données expérimentales

Variables

Une variable permet de stocker une valeur.

Exemples

- `x = 5`
- `t = 0.20`
- `message = "Physique"`

Modifier une variable

On peut modifier une variable existante.

Exemple

- `a = 3`
- `a = a + 1` # a vaut maintenant 4
- `a += 1` # écriture équivalente

Listes

Une liste stocke plusieurs valeurs.

Exemples

- `t = [0.0, 0.1, 0.2]`
- `x = [0.00, 0.12, 0.25]`
- `Vide=[]`

! Important : le premier indice est 0.

Manipuler une liste

Accéder à un élément :

- `x[0]` → premier élément
- `x[2]` → troisième élément
- `t[:-1]` → toute la liste sans le dernier élément

Ajouter un élément (0.40) à la fin de la liste x :

- `x.append(0.40)`

Longueur de la liste :

- `len(x)`

Blocs d'instructions

En Python, les blocs sont définis par l'indentation et suivent le caractère « : »

Exemple

```
if v > 5:
```

```
    print("Rapide")
    print("et bien !")
```

bloc d'instructions exécuté lorsque la condition est vraie.

Indentation : décalage vers la droite

print() permet d'afficher à l'écran le contenu d'une variable ou un message.

Conditions : if / else

Permettent d'exécuter ou non un bloc.

Exemple

```
if v > 5:
```

```
    print("Rapide")
```

bloc d'instructions exécuté lorsque la condition est vraie.

```
else:
```

```
    print("Lent")
```

bloc d'instructions exécuté lorsque la condition est fausse.

Boucle for

Permet de répéter un bloc un nombre connu de fois.

Exemple

```
for i in range(5):
```

```
    print(i)
```

→ i prend les valeurs 0, 1, 2, 3, 4

La fonction range() : définir un intervalle

La fonction range() permet de définir une suite d'entiers, souvent utilisée dans une boucle for.

Syntaxe : range(début, fin, pas)

- début : valeur de départ (incluse)
- fin : valeur de fin (exclue)
- pas : incrément (optionnel, par défaut = 1)

Exemples

- range(5)
- range(1, 5)
- range(1, 10, 2)

Tracer un graphique

Avec matplotlib :

pyplot est un sous-module de la bibliothèque matplotlib, dédié au tracé de graphiques.

Importation du sous-module pyplot permettant de tracer des graphiques

▸ `import matplotlib.pyplot as plt`

▸ `plt.plot(x, y)`

▸ `plt.xlabel("x (m)")`

▸ `plt.ylabel("y (m)")`

▸ `plt.show()`

affiche la figure construite à l'écran, déclenche l'ouverture de la fenêtre graphique et termine le tracé

prépare le graphique, définit ce que l'on veut tracer (points, lignes, données) et ajoute une courbe à la figure

affiche les titres de l'axe des abscisses (axe x) et des ordonnées (axe y)

Correction – Application 1

```
main.py
1 t = [0, 0.1, 0.2, 0.3]
2 print(len(t))
3 print(t[1])
```

Powered by trinket

```
4
0.1
```



Application 1

- Créer une liste t contenant 0 ; 0,1 ; 0,2 ; 0,3
- Afficher la longueur de la liste
- Afficher le deuxième élément



Application 2

- On dispose des listes t et x (chronophotographie) :
 - `t = [0, 0.1, 0.2]`
 - `x = [0.00, 0.20, 0.54]`
- Calculer les vitesses moyennes suivant x entre deux points.

Correction – Application 2

```
main.py
1 t = [0, 0.1, 0.2]
2 x = [0.00, 0.20, 0.54]
3 vx = []
4 for i in range(len(t)-1):
5     vx.append((x[i+1]-x[i])/(t[i+1]-t[i]))
6 print(vx)
```

Powered by trinket [2.0, 3.4]

📸 Exemple final : chronophotographie

- Données : t, x, y issus d'un pointage.
 - ▶ t = [0.0, 0.1, 0.2, 0.3]
 - ▶ x = [0.00, 0.12, 0.25, 0.40]
 - ▶ y = [1.20, 1.15, 1.05, 0.90]
- Calculer vx, vy puis la vitesse v.
- Tracer la trajectoire : plot(x, y)
- Afficher les vitesses v

```
1 import matplotlib.pyplot as plt
2 import math
3
4 # Données
5 t = [0.0, 0.1, 0.2, 0.3]
6 x = [0.00, 0.12, 0.25, 0.40]
7 y = [1.20, 1.15, 1.05, 0.90]
8 v = []
9
10 # Calcul des vitesses
11 for i in range(len(t) - 1):
12     vx_i = (x[i+1] - x[i]) / (t[i+1] - t[i])
13     vy_i = (y[i+1] - y[i]) / (t[i+1] - t[i])
14     v.append(math.sqrt(vx_i**2 + vy_i**2))
15
16 # Tracé de la trajectoire
17 plt.plot(x, y, 'r', marker='o')
18 plt.xlabel("x (m)")
19 plt.ylabel("y (m)")
20 plt.title("Trajectoire du mobile")
21 plt.show()
22
23 # Afficher les vitesses
24 print("les valeurs des vitesses sont : ")
25 # Arrondir à deux chiffres après la virgule à l'affichage
26 for val in v:
27     print(f"{val:.2f}")
```

Correction