Utilisation de listes ou de tableaux et tracé de courbes

1. Donnée en annexe d'un sujet de concours

Bibliothèque NUMPY

Dans les exemples ci-dessous, la bibliothèque `numpy` a préalablement été importée à l'aide de la commande : `import numpy as np`. On peut alors utiliser les fonctions de la bibliothèque, dont voici quelques exemples :

- **`np.linspace(start, stop, N_point)`** :
  - Description : renvoie un nombre d'échantillons espacés uniformément, calculés sur l'intervalle [start, stop] ;
  - Argument d'entrée : début, fin et nombre d'échantillons dans l'intervalle ;
  - Argument de sortie : un tableau.

Commande	Résultat
`np.linspace(1, 4, 5)`	`[1., 1,75, 2,5, 3,25, 4.]`

- **`np.zeros(i)`** :
  - Description : renvoie un tableau de taille i rempli de zéros ;
  - Argument d'entrée : un scalaire
  - Argument de sortie : un tableau.

Commande	Résultat
`np.zeros(5)`	`[0, 0, 0, 0, 0]`

- **`np.array(liste)`** :
  - Description : crée une matrice (de type tableau) à partir d'une liste.
  - Argument d'entrée : une liste définissant un tableau à 1 dimension (vecteur) ou 2 dimensions (matrice).
  - Argument de sortie : un tableau (matrice).

Commande	Résultat
`np.array([4, 3, 5])`	`[4, 3, 5]`

Bibliothèque `MATPLOTLIB.PYPLOT`

Cette bibliothèque permet de tracer des graphiques. Dans les exemples ci-dessous, la bibliothèque `matplotlib.pyplot` a préalablement été importée à l'aide de la commande : `import matplotlib.pyplot as plt`.

- Description : fonction permettant de tracer un graphique de n points dont les abscisses sont contenues dans le vecteur x et les ordonnées dans le vecteur y. Cette fonction doit être suivie de la fonction `plt.show()` pour que le graphique soit affiché.
- Argument d'entrée : un vecteur d'abscisses x (tableau de n éléments) et un vecteur d'ordonnées y (tableau de n éléments). La chaîne de caractères 'SC' précise le style et la couleur de la courbe tracée. Des valeurs possibles pour ces deux critères sont :

Valeurs possibles pour S (style) :

Description	Ligne continue	Ligne traitillée	Marqueur rond	Marqueur plus
Symbole S	-	--	o	+

Valeurs possibles pour C (couleur) :

Description	bleu	rouge	vert	noir
Symbole C	b	r	g	k

- Argument de sortie : un graphique.

```
x = np.linspace(3, 25, 5)
y = np.sin(x)
plt.plot(x,y,'-b') # tracé d'une ligne bleue continue
plt.title('titre_graphique') # titre du graphe
plt xlabel('x') # titre de l'axe des abscisses
plt ylabel('y') # titre de l'axe des ordonnées
plt.show()
```

2. Savoir faire 1 : utilisation de listes ou de tableaux

Syntaxe concernant les listes

Une liste l s'écrit de la forme l=[..,..,..]
len(l) : désigne le nombre de termes dans la liste
l[0] : désigne le 1er terme de la liste
l[1] : désigne le 2ième terme de la liste
l[2] : désigne le 3ième terme de la liste...
l.append(b) : sert à ajouter un terme dans la liste, ce terme a pour valeur b

Syntaxe concernant un tableau à une dimension

Un tableau à une dimension (ou vecteur) v se note v=np.array([..,..,..])
len(v) : désigne le nombre de termes dans le vecteur
v[0] : désigne le 1er terme du vecteur
v[1] : désigne le 2ième terme du vecteur
v[2] : désigne le 3ième terme du vecteur
v=np.zeros(N) : sert à créer un vecteur contenant N termes tous égaux à 0

Exemple 1:
l=[1,3,7,8,11]
len(l)= 5
l[2]= 7 l[0]=1
l.append(15) renvoie l= [1,3,7,8,11,15]

Exemple 2:
v=np.zeros(4) V = np.array([0,0,0,0]) len(V) = 4
for i in range(len(v)): i = 0,1,2,3
——v[i]=2*i V[0] = 0 V[1]= 2 V[2] = 4 V[3] = 6
L'exécution de ce code donne v= np.array([0,2,4,6])

Exemple 3:
l=[0] l[0]=0
for i in range(3): i = 0,1,2
——l.append(l[i]+2) pour i=0: l[0]+2 = 2 i=1 : l[1]+2 = 4 i = 2 : l[2]+2 = 6
L'exécution de ce code donne l= [0,2,4,6]

Exemple 4:
x=np.linspace(0,3,4) 4 termes x = np.array([0,1,2,3])
def f(a,b): f(a,b) = 3b² + a
——return 3*b**2+a
y=f(2,x) = 3x² + 2
L'exécution de ce code donne:
x= np.array([0,1,2,3])
y= np.array([2,5,14,29])

Exemple 5:

l=[1,3,7,8]

s=0

for i in range(len(l)):

——s=s+l[i]

L'exécution de ce code donne s= 19 : c'est le code pour calculer la somme des termes d'une liste ou d'un tableau

Remarque : intérêt d'une liste : on peut à tout moment ajouter des termes, c'est très utile lorsque l'on ne connaît pas à l'avance le nombre de termes de la liste.

Remarque : intérêt d'un tableau: on peut appliquer une fonction f aux termes d'un tableau (on ne peut pas le faire avec les termes d'une liste).

Exemple: v=np.array([0,2,4,6])

w=v**2 renvoie

def f(x):

——return 3*x+2

y=f(v) renvoie

3. Savoir faire 2: tracer une courbe

Cas 1: tracé de points de mesures expérimentaux:

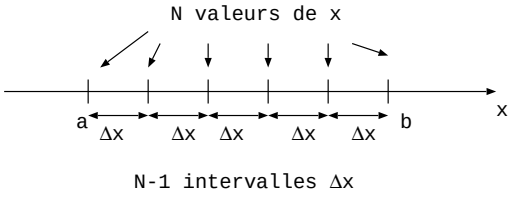
x=[x1,x2,...,xN] ou x=np.array([x1,x2,...,xN]) : on écrit les valeurs de x dans une liste ou un vecteur

y=[y1,y2,...,yN] ou y=np.array([y1,y2,...,yN]) : on écrit les valeurs de y dans une liste ou un vecteur

plt.plot(x,y,'*') : cela permet de tracer les points expérimentaux ici représentés par le symbole *

Cas 2: tracé d'une fonction f(x) sur un intervalle donné $[a, b]$:

x=np.linspace(a,b,N) : cela crée un vecteur contenant N valeurs régulièrement espacées entre a et b. L'intervalle Δx entre deux valeurs consécutives est $\Delta x = \dfrac{b-a}{N-1}$: on parle d'échantillonnage.

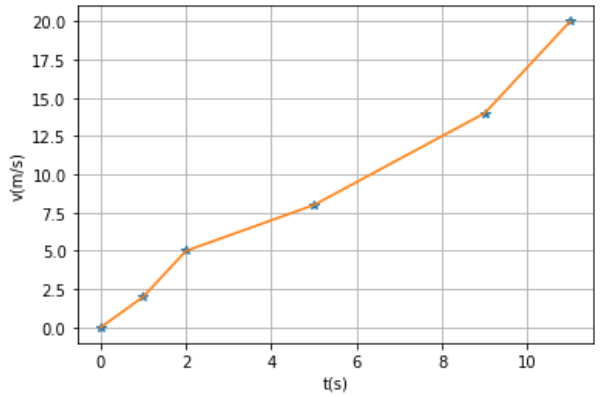


def f(x):

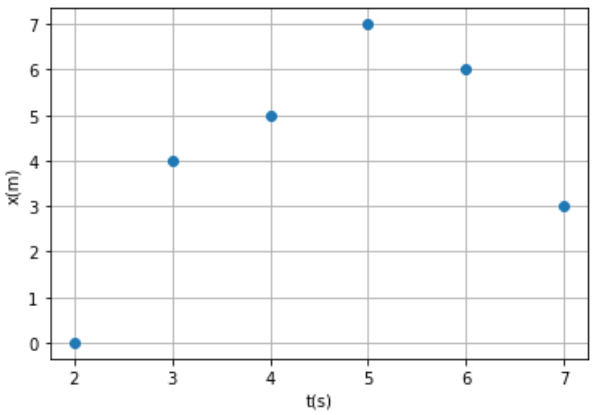
——return ..... : on définit la fonction f(x)

plt.plot(x,f(x)) : permet de tracer la courbe f(x), les points sont reliés entre eux sur cette courbe.

Exemple 1:

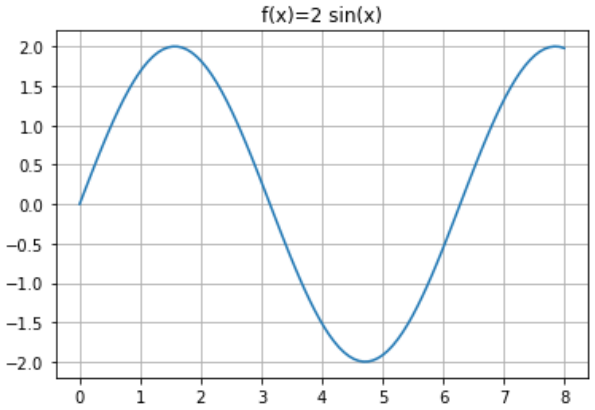


Exemple 2:



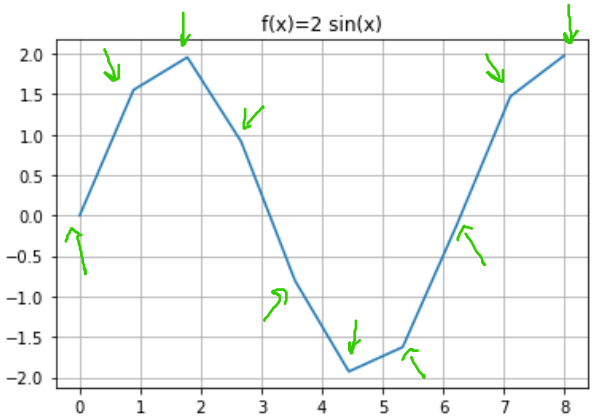

```
x = [2,3,4,5,6,7]
y = [0,4,5,7,6,3]
plt.plot(x,y,'o')
plt.grid()
plt.show()
```

Exemple 3:




```
x = np.linspace(0,8,100)   # grand nombre de points
y = 2*np.sin(x)
plt.plot(x,y)
plt.title('f(x) = 2 sin(x)')
plt.grid()
plt.show()
```

Exemple 4:



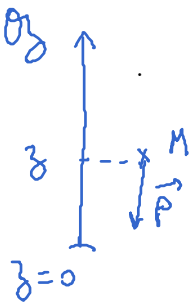

```
x = np.linspace(0,8,10)   # ici il y a 10 points
y = 2*np.sin(x)
...
```

l'échantillon (la distance entre 2 points) est trop grand car il n'y a pas assez de points. Il faudra donc prendre dans linspace un grand nbre de points pour une courbe correcte

4. Exercice

Soit un projectile que l'on lance vers le haut depuis une altitude $h = 1\ m$ avec une vitesse verticale $v_0 = 3\ m.s^{-1}$. On note Oz l'axe verticale ascendant et $z(t)$ l'altitude du projectile au cours du temps. On néglige tout frottement. Donnée: $g = 9,8\ m.s^{-2}$.

1- Etablir l'équation de la trajectoire $z(t)$.



M ne subit que son poids, la RFD s'écrit : $m\vec{a} = \vec{P} = -mg\vec{z}$

Sur (Oz) : $\ddot{z} = -g$ $\quad \dot{z} = -gt + v_0$ $\quad z = -\frac{gt^2}{2} + v_0 t + h$

C.I. $\dot{z}(t=0) = v_0$ $\quad$ C.I $z(t=0) = h$

On désire tracer la courbe donnant $z(t)$ jusqu'à ce que le projectile tombe sur le sol.

2- Dans un premier temps, on utilise sous python, une liste temps de la forme $t_i = idt$ où dt est le pas de temps choisi et une liste $z_i = z(t_i)$. Compléter le code suivant:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 #Données
5 g,h,v0=.9.8,.1.,.3.
6 #Lancer d'un projectile: utilisation de liste
7 lt,lz=[.0.],[.h.] #on initialise les listes t(temps) et z(hauteur)
8 dt=20e-3 #c'est le pas de temps
9 i=0
10 while ...lz[i]...>.0....: #condition d'arrêt du calcul de t et de z(t)
11     lt.append(..lt[i] + dt)
12     lz.append(..-g * lt[i+1]**2/2 + v0 * lt[i+1] + h)
13     i=i+1
14 plt.plot(.lt,lz.) #on trace z(t)
15 plt.grid()
16 plt.xlabel('.t.(s).')
17 plt.ylabel('.z.(m).')
18 plt.show()
```

ce terme correspond à t_{i+1}

ce terme correspond à $z(t_{i+1})$

on calcule $lz[i] = z(t_i)$ tant qu'il est > 0

$t_{i+1} = t_i + dt$

$z(t_i) = -\frac{gt_i^2}{2} + v_0 t_i + h$

$z(t_{i+1}) = -\frac{gt_{i+1}^2}{2} + v_0 t_{i+1} + h$

3- Déterminer, en fonction de v_0, g et h, l'expression théorique du temps t_f pour lequel le projectile tombe sur le sol. Dans un second temps, on utilise sous python, des vecteurs t pour le temps et z pour l'altitude du projectile. Compléter le code suivant:

```
20 #Lancer d'un projectile: utilisation d'un vecteur
21 tf=..(v0/g) + ( (v0/g)**2 - 2*h/g) ** 0.5
22 def z(t):
23     return ..-g.*t**2/2 + v0 * t + h
24 t=np.linspace(.0.,tf,1000) on trace z(t) pour t ∈ [0, tf], on utilise assez de points pour tracer la courbe bien qu'elle soit fidèle
25 plt.plot(.t.,z(t)) #on trace z(t)
26 plt.grid()
27 plt.xlabel('.t.(s).')
28 plt.ylabel('.z.(m).')
29 plt.show()
```

t_f vérifie : $z(t_f) = 0$ soit $-\frac{gt_f^2}{2} + v_0 t_f + h = 0$ ou $t_f^2 - \frac{2v_0}{g} t_f - \frac{2h}{g} = 0$

$\Delta = \frac{4v_0^2}{g^2} - \frac{8h}{g} > 0$ $\quad t_f = \frac{v_0}{g} + \frac{\sqrt{\Delta}}{2} = \frac{v_0}{g} + \sqrt{\frac{v_0^2}{g^2} - \frac{2h}{g}}$