TP d'informatique n°05
utilisation de modules/bibliothèques

PCSI 2023 – 2024

I Rappels

Un module est (essentiellement) un ensemble de fonction et de constante déjà programmé par des professionnels. Une fois un module installé, pour l'utiliser en python il suffit de faire **import** `nomDuModule as nomCourt` et s'utilise ensuite en appelant les fonctions en faisant `nomCourt.nomFonctions(arguments)`.

Une quantité colossale de modules possédant de très nombreuses fonctions existe en python. Pour s'en sortir et les utiliser correctement, il convient de **regarder la documentation** ainsi que de **test et adapter** les exemples disponibles. Pour illustrer cela, nous allons utiliser les modules numpy et matplotlib.

II Documentation succincte

A. 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 et fonctionnalité de la bibliothèque, dont quelques exemples sont présents ci-dessous.

Remarque importante : un tableau numpy est une structure dans laquelle l'accès se fait de façon similaire à une liste, mais les opérations sont effectuées terme à terme. Par exemple `tab1 + tab2` nécessiterait deux tableau de même taille et renverrai un tableau de même taille contenant l'addition terme à terme des deux premiers tableaux. De même on peut appliquer directement les fonctions mathématiques de numpy à des tableaux ou à des listes pour calculer la fonction pour chacun des termes. Par exemple : `np.cos( [0, np.pi/2, np.pi])->array([1., 0., -1. ])`.

- `A[i];A[i,j]`
  - → *Entrée :* un objet de type `np.array` (`A` ici) et un entier ou un tuple d'entier (`i` ou `i,j` ici).
  - → *Sortie :* un nombre ou un tableau numpy.
  - → *Description :* syntaxe permettant de récupérer l'élément indicé par `i` d'un objet de type tableau ou l'élément indicé par `i` et `j` dans le cas d'un tableau à 2 dimensions (généralisable à n dimensions).
  - → **Exemples :** `A = np.array([[3,4,10],[1,8,7]])`
    - ◇ `A[0,2]` $\Longrightarrow$ `10` ; `A[:,2]` $\Longrightarrow$ `[10 7]` ; `A[1,:]` $\Longrightarrow$ `[1 8 7]`

- `np.zeros(dim)`
  - → *Entrée :* un objet de type entier ou un tuple d'entier
  - → *Sortie :* un tableau numpy d'éléments nul de dimension le nombre d'élément de `dim`. Dans le cas où `dim` est un tuple, `dim[i]` est le nombre d'éléments de la i^{e} dimension.
  - → *Description :* fonction permettant de créer un objet de type tableau de taille prédéfinie et ne contenant que des 0.
  - → **Exemples :**
    - ◇ `np.zeros(4)` $\Longrightarrow$ `[0 0 0 0]` ; `np.zeros( (2,3))` $\Longrightarrow$
```
[[0 0 0]
 [0 0 0]]
```

- `np.linspace(Min, Max, nbElements)`
  - → *Entrée :* deux nombres et un entier
  - → *Sortie :* un tableau numpy à une dimension
  - → *Description :* fonction permettant de créer tableau de `nbElements` nombres équirépartis entre `Min` et `Max` (inclus).

→ **Exemples :**

◇ `np.linspace(3,25,5)` $\Longrightarrow$ `[3. 8.5 14. 19.5 25.]`

◇ `np.linspace(1,0,3)` $\Longrightarrow$ `[1. 0.5 0.]`

- `np.loadtxt(nom_fichier)`
  - → *Entrée :* un objet de type **str** contenant le nom du fichier (et éventuellement le chemin relatif ou absolu dans le cas où le fichier n'est pas dans le répertoire d'exécution du programme).
  - → *Sortie :* un tableau numpy d'éléments de dimension a priori 2 contenant les nombres écrit dans le fichier.
  - → *Description :* fonction permettant de charger les données contenues dans un fichier texte.
  - → **Exemple :** Si le fichier nommé `"data.txt"` contient

$$\begin{matrix} 1.0 & 3.0 & 4.0 \\ 2.5 & 3.1 & 7.3 \end{matrix}$$

◇ `np.loadtxt("data.txt")` $\Longrightarrow$

```
[[1.0  3.0  4.0]
 [2.5  3.1  7.3]]
```

B. Matplotlib

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

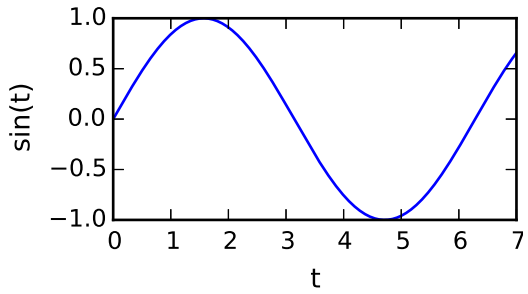
- `plt.plot(x,y)`
  - → *Entrée :* deux objets de type itérables de même dimension n (tableau numpy ou liste en particulier).
  - → *Description :* fonction permettant de tracer sur un graphique les n couples de points (x_i,y_i) et de les relier par un trait.

    Un appel de la fonction show (`plt.show()`) doit terminer le programme pour que le graphique soit effectivement affiché à l'écran.

    Si plusieurs plot se suivent, alors les différentes courbes se superposent sur la même figure.
  - → **Exemple :**

```
x = np.linspace(0,7,200)
y = np.sin(x)
plt.figure()
plt.plot(x,y)
plt.xlabel('t')
plt.ylabel('sin(t)')
plt.show()
```

$\Longrightarrow$



- `plt.show()`
  - → *Entrée :* **None**
  - → *Description :* fonction permettant d'afficher les graphiques précedemment définis.

- `plt.figure()`
  - → *Entrée :* **None**
  - → *Description :* fonction permettant de créer une nouvelle fenêtre graphique vierge.

III Utilisation basique

Des fonctions utiles de numpy et matplotlib sont indiquées sous les premiers exercices.

Exercice n°1 *Sans utiliser numpy*, créez une liste de nombre avec 10 points équirépartis (c'est-à-dire à la même "distance les uns des autres") entre 0 et 3 (**inclus**). Faites de même avec 100 points.
Attention à l'écart entre deux points, il est envisageable de dessiner un schéma sur une feuille de brouillon pour ne pas se tromper. Pensez à vérifier votre résultat (nombre de points, 0 et 3 inclus dans la liste).

Faites de même *en utilisant numpy* (fonction utile rappelée ci-dessous).

fonctions utiles : `linspace` (pensez à regarder la documentation et des exemples si besoin)

Exercice n°2 Créer une liste de 30000 nombres $L_i = \cos(0{,}01 \times i)$ pour $0 \leq i < 30000$ de deux façons :

- sans utiliser numpy (et en faisant une boucle ou une liste en compréhension) ;
- en utilisant numpy (sans faire de boucle)

Quelle méthode est la plus rapide (à la fois à coder et à s'exécuter) ?
fonctions utiles : `np.cos` ; `np.linspace`

IV Tracé de courbes

Exercice n°3 Tracez la courbe $y(x) = \left(\frac{\sin x}{x}\right)^2$ sur l'intervalle [-10 ; 10] en prenant un nombre pair de points. (Par exemple 10 points dans un premier temps, puis 1000 ensuite)
Ajoutez des labels aux axes.

fonctions utiles : `np.linspace` ; module `matplotlib.pyplot` ; `plt.plot` ; `plt.xlabel` ; `plt.ylabel`.

Remarque : cette courbe est la courbe théorique de l'intensité lumineuse lors de la diffraction par une fente rectangulaire.

Exercice n°4 On considère la courbe paramétrée suivante $\begin{cases} x &= 2\sin^2(t)\cos(t) \\ y &= 2\cos^2(t)\sin(t) \end{cases} \qquad t \in [0{,}2\pi]$
Créer un tableau t contenant 2000 nombres équirépartis entre 0 et 2π. Calculer les tableaux x et y correspondant. Tracer la courbe définie ci-dessus (y en fonction de x)

Exercice n°5 On considère la courbe paramétrée suivante en coordonnée polaire (r,θ)

$$r = 1 + \cos(4\theta) + \sin^2(4\theta) \qquad \theta \in [0{,}2\pi]$$

Créer un tableau $theta$ contenant 2000 nombres équirépartis entre 0 et 2π. Calculer le tableaux r correspondant. Projeter en coordonnées cartésiennes puis faites le tracé (rappel : $x = r\cos\theta$ et $y = r\sin\theta$).

V Ouverture de fichier et remplissage de matrice

Un fichier de mesure est généralement présenté sous la forme de plusieurs colonnes, chacune représentant une grandeur. On peut voir un exemple ci-contre à droite.
On se propose dans cet exercice de créer une fonction `charge_fichier` qui prendra en argument le nom d'un fichier et renverra un tableau de nombre contenant les données du fichier. On va écrire pour cela le script petit à petit :

t	u	i
0	0	2
0.1	3	1
0.2	4.5	0.5
0.3	5.25	0.25
⋮	⋮	⋮

L'écriture et la lecture d'un fichier se font à l'aide des fonctions **open**, `close`, `readlines`, `write`.

Exemple d'écriture.

```
f = open("nom_du_fichier","w")
#attention EFFACE le fichier
#w pour write
f.write("1")
f.write("\n") #retour à la ligne
f.write("2\n ligne3")
f.close()#toujours après un open
```

Exemple de lecture :

```
f2 = open("nom_du_fichier","r")
#r pour read
donnees = f2.readlines()
f2.close()
print(type(donnees))
print(donnees)
```

1. Testez l'exemple en écriture ci-dessus, **ne prenez pas** test1.out ou test2.out comme nom de fichier. Après avoir exécuté le programme, allez voir dans le dossier où est votre programme pour constater l'apparition d'un nouveau fichier. Vous pouvez regarder le contenu en faisant clic-droit → edit with notepad++.
2. Testez l'exemple en lecture ci-dessus. Quel type d'objet est contenu dans donnees ? Que représente chaque élément de cet objet ?
3. Téléchargez sur internet (grenard.dyndns.org/info.html)les deux fichiers pour ce TP (test1.out et test2.out) et sauvegardez les dans le même dossier que votre programme.
4. Modifier les lignes de codes précédentes pour détecter le nombre de lignes de données (on enlèvera une ligne pour l'entête) dans le fichier et l'enregistrer dans une variable.
5. De même, détecter le nombre de colonnes (les colonnes sont séparées par des tabulations, on pourra utiliser split (regardez la documentation sur internet)) et l'enregistrer dans une variable.
6. Créer une matrice de la bonne taille (On suggère d'utiliser `np.zeros`)
7. Remplir la matrice (2 boucles imbriquées, une sur les lignes, puis pour chaque ligne une sur les colonnes. On pourra commencer par ne remplir que la première ligne, pour ensuite rajouter la boucle sur les différentes lignes.)
8. Mettre en forme (si ce n'est pas déjà fait) le code de façon à former la fonction souhaitée.

Exercice n°6 Essayer de faire pareil en utilisant `np.loadtxt`, on pourra utiliser l'argument optionnel `skiprows`

Exercice n°7 À l'aide des fonctions précédentes, tracer pour les deux fichiers de test disponibles sur internet la dernière colonne en ordonnée et l'avant dernière en abscisse.

VI À faire pour la prochaine fois

Exercice n°8 Quelle est la différence majeure entre le cosinus de la bibliothèque math et celui de la bibliothèque numpy ?

Exercice n°9 Définir un tableau de 2000 points équirépartis entre 0 et 20.

Exercice n°10 Écrire quelques lignes de python permettant de tracer la courbe représentative de la fonction $t \mapsto \cos(t)\exp(-0.2 \times t)$ sur l'intervalle $[0,20]$ avec 2000 points.

Exercice n°11 On considère la courbe paramétrée suivante

$$\begin{cases} x &= \cos(t) \\ y &= \sin(t) + \sqrt{|\cos(t)|} \end{cases} \qquad t \in [0,2\pi]$$

Créer un tableau t contenant 2000 nombres équirépartis entre 0 et 2π. Calculer les tableaux x et y correspondant. Tracer la courbe définie ci-dessus (y en fonction de x) et expliquer pourquoi elle vous fait penser à votre cours de physique.

Exercice n°12 Orbite de Vénus vue depuis la Terre.
Depuis la Terre, l'orbite de Vénus a approximativement l'allure de la courbe paramétrée suivante :

$$z(t) = e^{2i\pi(t+t_0)} + 0{,}725 \cdot e^{2i\pi\left(\frac{13}{8}t+t_1\right)}, t \in [0,8],$$

où $2\pi t_0 \simeq \pi^2/2$ et $2\pi t_1 \simeq \pi^2/4$.
Il faut ensuite tracer z dans le plan complexe.
fonctions utiles : `np.real`; `np.imag`; `plt.axis("equal")`