I_{09} Numpy et matplotlib : présentation rapide de deux bibliothèques scientifiques incontournables.

PCSI 2020 – 2021

I Numpy

Numpy est une bibliothèque scientifique qui permet de faire du calcul numérique de façon plus rapide et pratique (par rapport aux fonctionnalités de base de python). On supposera pour tout les exemples que la bibliothèque numpy a été importé de la façon suivante : `import numpy as np`.

1. Type de base

Numpy introduit essentiellement un nouveau type : le type `ndarray` (pour tableau à n dimension) .
Il existe plusieurs moyen de créer un tel tableau, en particulier :

```
x=np.array([1,3,6,10])#array :conversion depuis une liste ou un tuple
x=np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])#tableau à deux
   dimensions
x=np.linspace(1,4,10)#crée un tableau de 10 points équi-répartis allant
    de exactement 1 à exactement 4
x=np.arange(1,4,0.3)#équivalent de range : tableau qui part de 1 (
   inclus) qui va jusqu'à 4 (exclus) par pas de 0.3
x=np.zeros( (5,3) )# crée un tableau de 0 de 5 lignes et 3 colonnes
```

On accède aux éléments du tableau de manière similaire aux listes :

```
N=np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])#tableau à deux
   dimensions
print(N[1,0])#renvoie 5 on accède comme aux matrices
print(N[0,2])#renvoie 3 (mais en commençant les indices à 0 et non pas
   à 1)
```

2. Changement de forme et de taille

Si l'on souhaite changer un tableau de « forme » (passer d'un tableau de 1 ligne 12 colonnes à un tableau de 2 lignes 6 colonnes par exemple) on peut utiliser la méthode ou la fonction reshape . Pour changer la taille, on peut utiliser la fonction resize .

```
M=np.linspace(1,12,12)#tableau de 1 lignes, 12 colonnes
N=M.reshape(2,6)#N reçoit un tableau contenant les éléments de M sur 2
   lignes et 6 colonnes. M n'est pas modifié
N=M.reshape(2,-1)#N reçoit un tableau contenant les éléments de M sur 2
    lignes et 'le bon nombre de colonnes'. M n'est pas modifié
N=np.reshape(M,(-1,4))#idem avec 4 colonnes et 'le bon nombre de ligne'
    (3)
N=np.resize(M,(1,24))#N reçoit un tableau de une ligne et 24 colonnes.
   Les éléments de N répliquent les éléments déjà existant dans M.
```

On peut aussi concaténer 2 tableau horizontalement avec `vstack` ou horizontalement avec `hstack`.

3. Calcul sur les tableaux

Contrairement aux listes, l'opération + représente l'addition terme à terme et non pas la concaténation. De même pour la multiplication et les autres opérations :

```
t1=np.linspace(0,1,5)# => array([ 0.  ,  0.25,  0.5 ,  0.75,  1.  ])
t2=np.linspace(1,3,5)# => array([ 1. ,  1.5,  2. ,  2.5,  3. ])
t1+t2# => array([ 1.  ,  1.75,  2.5 ,  3.25,  4.  ])
4*t1# => array([ 0.,  1.,  2.,  3.,  4.])
4*t1*t2# => array([  0. ,   1.5,   4. ,   7.5,  12. ])
1+t1# => array([ 1.  ,  1.25,  1.5 ,  1.75,  2.  ]) considère que le
   scalaire 1 représente un tableau de 1
```

De même les fonctions mathématiques de numpy peuvent s'appliquer terme à terme à des tableaux ou à des listes qui seront automatiquement converties en tableaux (contrairement aux fonction de la bibliothèque math qui ne s'applique qu'à des types simples).

```
t=np.linspace(0,np.pi,5)
f=np.cos(t)#calcul "vectoriellement" : sur tout le tableau, avec le
   cosinus du module math, il faudrait faire une boucle
```

4. Produit matriciel

L'opérateur `'*'` représente donc la multiplication terme à terme. Pour accéder au produit matriciel, il faut utiliser la fonction `dot` :

```
M=np.array([[1,0],[1,1]])
N=M.transpose()#enregistre dans n la transposé de N
np.dot(N,M)# => array([[2, 1],[1, 1]])
np.dot(M,N)# => array([[1, 1],[1, 2]])
```

II Matplotlib

Matplotlib est une bibliothèque graphique qui permet de tracer des courbes et des graphiques. Elle permet aussi de faire des animations, d'afficher des images etc...
On importera le sous module pyplot de la façon suivante : `import matplotlib.pyplot as plt.`
3 fonctions de bases sont à connaitre :

- `plt.figure()` : crée une nouvelle figure.
- `plt.plot(x,y)` : x et y sont des tableaux numpy à 1D (ou des listes de nombres), trace dans une figure les couples de points x[i],y[i] et les relie par un trait.
- `plt.show()` : affiche toutes les figures crées. Devrait être la dernière instruction du programme.

Toutes ces fonctions peuvent prendre énormément d'arguments optionnels. Il existe aussi BEAUCOUP d'autres fonctions. Il convient donc de regarder la documentation et/ou les exemples lorsque l'on veut faire quelque chose de plus évolué (par exemple pour les TIPE).
Voici quelques exemples d'utilisation :

```
fig=plt.figure()
x=np.linspace(-2,2,200)
y=np.exp(-x**2)#courbe gaussienne
plt.plot(x,y,'-k')#trace les points et les relie par des traits noirs (
   blacK)
```

```
fig.savefig('figure_test.jpg')#sauvegarde la figure
#On peut rajouter des options
plt.axis([-2,2,0,1.1])#limite les axes à -2; 2 en abscisse et 0; 1.1 en
    ordonnée
plt.xlabel('$x$ (cm)')#Donne un label à l'axe des x
plt.ylabel('$P(x)$ (cm$^{-1}$)')#idem en y
plt.show()#déclenche l'affichage
```

```
plt.figure()
plt.suptitle('Un joli cercle et une belle spirale')#ajoute un titre
x=np.linspace(0,2*np.pi,200)
plt.plot(np.cos(2*x),np.sin(2*x),'--')#cercle en pointillé
plt.plot(3*np.cos(5*x)/(3+5*x),3*np.sin(5*x)/(3+5*x),'-o')#spirale en
   faisant apparaitre les points et les traits qui les relient
plt.axis('equal')#même échelle en x et en y
plt.show()#déclenche l'affichage
```

Table des matières