ANALYSE NUMÉRIQUE - SAGEMATH/PYTHON

Ines Abdeljaoued Tej [1]

Travaux pratiques numéro 1

1 Introduction à Python

Créez un fichier dans votre espace de travail. Tous vos résultats doivent être enregistrés dans ce fichier. Dans cette séance, le but est de vous montrer les bases de la programmation avec Python et ces applications au calcul scientifique. Ce TP donne quelques idées sur les possiblilités de Python :

1.1 Utilisation de Python comme calculatrice

Par exemple, vous pouvez utiliser l'interpréteur comme une simple calculatrice de bureau.

```
1 + 1
1.5 - 2
2-9 #Les espaces sont optionnels
(7+3)*4
1 / 2
12+1j #les nombres complexes sont repréentés avec l'unité imaginaire j
1j*1j
5\%3, 3.5\%2.8 #Modulo
10**2, 10^2 #attention à la puissance
7/3 #Les divisions entières retournent des entiers
7.0/3 #Ca donne des divisions non entières en manimpulant des réels
x=4 #affectation des variables avec le signe "="
x*2, x**2 #x au carré
```

1.2 Opération mathématiques supplémentaires

Pour effectuer des opérations mathématique plus complexes, nous devons utiliser quelques fonctions supplémentaires : pour cela, nous commençons par charger/importer la bibliothèque avec les fonctions scientifiques appelée NumPy :

1. inestej@gmail.com, http://bit.ly/1JEWZiK

```
import numpy as np
np.sin(3)
np.pi
np.deg2rad(180)
np.rad2deg(np.pi)

np.ceil(6.03) #arrondi au plus petit entier supérieur
np.floor(4.99) #arrondi au plus grand entier inférieur
np.round(2.49) #arrondi au plus proche
np.round(2.50) #arrondi au plus proche

np.conj(5) #la valeur conjugué d'un nombre complexe
np.conj(5+1j)

dir() #affiche la liste avec toutes les variables actuellement défines

dir(np) #affiche la liste avec toutes les fonctions dfinies par np

np.abs(23)
np.abs(-23)
np.abs(1+1j)
```

Le module SciPy contient de nombreux algorithmes très utilisés par les numériciens : méthodes itératives ou directes pour résoudre des systèmes linéaires, intégration numérique... Dans toute la suite, nous utiliserons :

```
import numpy as np
import scipy as sp
import matplotlib as mpl
import matplotlib .pyplot as plt
```

Pour voir les différences (Numpy $\subset$ SciPy) essayez par exemple :

```
np.sqrt(-1.)
sp.sqrt(-1.)
```

Parmi les subpackages de SciPy, on peut citer :

1.3 Tableaux, vecteurs et matrices

Toutes les opérations précédentes peuvent être appliquées à plusieurs nombres à la fois. Ceci est possible avec l'utilisation des tableaux. EN Python il existe deux façons de déclarer des tableaux (array en anglais) :

Subpackage	Description
cluster	Clustering algorithms
constants	Physical and mathematical constants
fftpack	Fast Fourier Transform routines
integrate	Integration and ordinary differential equation solvers
interpolate	Interpolation and smoothing splines
io	Input and Output
linalg	Linear algebra
maxentropy	Maximum entropy methods
ndimage	N-dimensional image processing
odr	Orthogonal distance regression
optimize	Optimization and root-finding routines
signal	Signal processing
sparse	Sparse matrices and associated routines
spatial	Spatial data structures and algorithms
special	Special functions
stats	Statistical distributions and functions
weave	C/C++ integration

1.3.1 Listes en Python

```
x = [123, 1+2j, 13, -5, 0, 900.2]
x
len(x) #taille d'un tableau
x[0] #contenu de la case 0 : le premier indice étant 0
```

Qu'est-ce qui se passe quand vous essayez de lire le contenu d'une case qui n'existe pas ?

```
x[100]
```

```
x[0] = -2 #modification de la case
x
```

```
x.append(1) #ajouter un élément
x
```

```
x.remove(0) #on supprime un élément
```

Qu'est-ce qui se passe si on essaye de supprimer une valeur qui n'est pas présente dans la liste ?

```
y = ["un", "deux", 1,2,3] #Une liste peut mélanger plusieurs types de données
y
```

```
z = x+y #concaténation de deux listes
x*2
```

1.4 Tableaux NumPy

Les listes sont très utilisées en Python.

```
np.zeros(10) #Créer un vecteur contenant 10 zéros
np.ones(10) #Créer un vecteur contenant 10 un
np.arange(10) #Créer un vecteur contenant les entiers de 0 à 9
np.array([1,2,3,8,9,10]  #Créer la liste (1,2,3,8,9,10)

x = np.array([1,2,3,0.3,9])
x
x*2
x+1
np.sin(x)
np.ceil(x)

y = np.ones(5)
y

x+y
2*x + 3*y

x.sort() #trier les éléments
x

x.sum() #calculer la somme des éléments
x.var() #Trouver la variance des éléments
x.std() #trouver l'écart-type des éléments
x.max()
x.min

help(x.mean) #affiche l'aide pour la fonction mean de x
```

1.4.1 Tableaux NumPy multidimentionnels

Pour définir des matrices nous avons besoin de tableaux à deux dimensions. Par exemple,

```
x = [12, 1, 10]
y = [-3, 8, 20]
z = [13, 2, 8]

liste2d = [x,y,z]
liste2d

matrix = np.array(liste2d)
matrix

matrix[0,0]

np.diag([1,2,3,4,5]) #une matrice diagonale avec 1,2,3,4 et 5 dans la diagonale
```

```
np.identity(10) #matrice d'identité de taille 10

A = np.array([[1,2,3],[8,9,7],[6,5,4]])
A.shape #Les dimensions de la matrice A
np.transpose(A)
np.trace(A)
np.diag(A)

B = np.linalg.inv(A) #La matrice inverse de A
B

np.dot(A,B)
np.linalg.det(A) #le déterminant de A
```

2 Exercices

1. Résoudre le système $Ax = b$ avec $A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 5 & 1 \\ 2 & 3 & 8 \end{pmatrix}$ et $b = \begin{pmatrix} 10 \\ 8 \\ 3 \end{pmatrix}$.
2. Trouvez $I_1 = \int_0^5 (4x + 12)dx$ et $I_2 = \int_0^{+\infty} e^{-2x}dx$. Pour cela, utilisez les commandes suivantes :

   ```
   import scipy.integrate as si
   help(si) #voir les algorithmes d'intégration numérique et particulièrement quad
   f = lambda x: np.exp(-2*x)
   f #la fonction à intégrer
   np.Inf #l'infini
   ```

3. Importez pyplot de matplotlib (from matplotlib import pyplot). Générez aléatoirement 100 valeurs uniformément répartis dans [0,1] avec la commande np.random.uniform et affichez un nuage de points avec les commandes pyplot.scatter(x,y) et pyplot.show()
4. SUivez les commandes suivantes et expliquez ce que vous obtenez au fur et à mesure :

   ```
   import numpy as np
   import matplotlib.pyplot as plt
   x = np.linspace(0, 2*np.pi, 50)
   y = np.sin(x)
   y2 = y + 0.1 * np.random.normal(size=x.shape)
   fig, ax = plt.subplots()
   ax.plot(x, y, 'k--')
   ax.plot(x, y2, 'ro')
   # set ticks and tick labels
   ax.set_xlim((0, 2*np.pi))
   ax.set_xticks([0, np.pi, 2*np.pi])
   ax.set_xticklabels(['0', '$\pi$', '2$\pi$'])
   ax.set_ylim((-1.5, 1.5))
   ```

```
ax.set_yticks([-1, 0, 1])
# Only draw spine between the y-ticks
ax.spines['left'].set_bounds(-1, 1)
# Hide the right and top spines
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
# Only show ticks on the left and bottom spines
ax.yaxis.set_ticks_position('left')
ax.xaxis.set_ticks_position('bottom')
plt.show()

#http://matplotlib.org/examples/ticks_and_spines/spines_demo_bounds.html
```