

Stage CPP Python

Python et calcul scientifique

Jean-Baptiste Durand (Jean-Baptiste.Durand@imag.fr)

Contributions de Benjamin Wack, Matthieu Moy

Grenoble INP / Ensimag

Août 2014



Jean-Baptiste Durand (Ensimag)

Python et calcul scientifique

Août 2014 < 1 / 20 >

Un point fort des IDE : débogueur

- Pas à pas et explorateur de variables
- Points d'arrêt
- Appels de fonctions

Exemple : suite récurrente

$$U_0 = 0, 1.$$

$$\forall n \in \mathbb{N}, U_{n+1} = \exp\left(-\frac{1}{U_n^2}\right)$$



Jean-Baptiste Durand (Ensimag)

Python et calcul scientifique

Août 2014 < 4 / 20 >

Aide embarquée

- Un doute sur une fonction ?

```
>>> help(math.floor)
Help on built-in function floor in module math:
```

```
floor(...)
floor(x)
```

```
Return the floor of x as a float.
This is the largest integral value <= x.
```

- On peut même le faire directement sur le module pour avoir toute l'aide :

```
>>> help(math)
```



Jean-Baptiste Durand (Ensimag)

Python et calcul scientifique

Août 2014 < 7 / 20 >

numpy

```
from numpy import *
```

```
>>> exp(range(0, 3))
array([ 1.          ,  2.71828183,  7.3890561 ])
```

```
ou bien import numpy as np
```

```
>>> np.exp(range(0, 3)) # noter le np.exp
array([ 1.          ,  2.71828183,  7.3890561 ])
```



Jean-Baptiste Durand (Ensimag)

Python et calcul scientifique

Août 2014 < 10 / 20 >

Environnement de développement intégré (IDE)

- Python
 - ▶ un simple interpréteur de commandes
 - ▶ pas d'éditeur
 - ▶ pas de rappel et complétion de commandes
- Spyder
 - ▶ inclut toutes les bibliothèques intéressantes
 - ▶ éditeur de texte adapté à python
 - ▶ facilite la compréhension de programmes à l'exécution



Jean-Baptiste Durand (Ensimag)

Python et calcul scientifique

Août 2014 < 3 / 20 >

Listing des fonctions et constantes

- Pour accéder au contenu d'un module :

```
>>> import math
>>> dir(math)
['_doc_', '__file__', '__name__', '__package__',
'acos', 'acosh', 'asinh', 'atan'...]
```

- Mais aussi :

```
>>> l = [42, True, 3]
[42, True, 3]

>>> dir(l)
# renvoie la liste des methodes et attributs de l
```



Jean-Baptiste Durand (Ensimag)

Python et calcul scientifique

Août 2014 < 6 / 20 >

Aide embarquée (suite)

- Et de même :

```
>>> help(l)
# la documentation de toutes les methodes de l
```

```
>>> help(l.append)
# la documentation d'une methode
```

- Ce n'est pas réservé aux modules/fonctions fournis.

```
>>> def f(x):
    "Voici une fonction peu utile."
    return x
```

```
>>> help(f)
Help on function f in module __main__:
```

```
f(x)
Voici une fonction peu utile.
```



Jean-Baptiste Durand (Ensimag)

Python et calcul scientifique

Août 2014 < 8 / 20 >

numpy.array

Objet central : numpy.array

- tableau homogène multidimensionnel

```
>>> a = zeros( (2,3) )
array([[0.,  0.,  0.],
       [0.,  0.,  0.]])
```

- Opérations naturelles sur les matrices

```
>>> 2*[1, 2] + [3, 4]
[1, 2, 1, 2, 3, 4]
>>> 2*array([1, 2]) + array([3, 4])
array([5, 8])
```

- Traitement (un peu) simplifié du partage
- a[m:n] est une vue d'une partie de a
- a.copy() est totalement disjoint de a



Jean-Baptiste Durand (Ensimag)

Python et calcul scientifique

Août 2014 < 11 / 20 >

numpy.array (suite)

Utiliser numpy.array

- Création : un peu comme une liste de listes...

```
>>> A = np.array([[1, 2], [3, 4], [5, 6]])
>>> A
array([[1, 2],
       [3, 4],
       [5, 6]])
```

- ... mais l'indexation comme une matrice est possible

```
>>> A[0,1]
2
>>> A[0][1]
2
```

- Voir aussi : ex_numpy.py

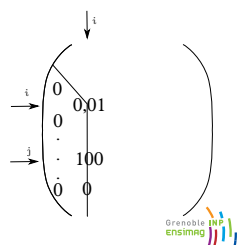


Le TP pivot de Gauss en bref

- Résolution de systèmes d'équations linéaires.

$$\left(\begin{array}{cccc|c} 1 & -1 & 2 & 1 & 1 \\ 2 & -4 & 1 & 1 & 6 \\ 1 & -1 & 1 & -1 & 2 \\ 3 & 1 & 1 & 2 & 1 \end{array} \right) \Leftrightarrow \begin{cases} x - y + 2z + t = 1 \\ 2x - 4y + z + t = 6 \\ x - y + z - t = 2 \\ 3x + y + z + 2t = 1 \end{cases}$$

- Itérations sur les lignes de la matrice
- À l'itération i , matrice de la forme \rightarrow
- Choix du pivot :
 - * coefficient $A[i, i]$ (si $\neq 0$, sinon permuter la ligne L_i avec une ligne $L_j, j > i$) \rightarrow pivot normal.
 - * systématiquement permuter la ligne L_i avec la ligne $L_j, j > i$ de $|A[i, j]|$ maximal \rightarrow pivot partiel.



matplotlib (suite)

- Discrétisation :

```
>>> x = numpy.arange(0, 1.1, 0.2)
array([ 0., 0.2, 0.4, 0.6, 0.8, 1. ])
```

- Remarque utile : on peut appliquer toutes les fonctions de numpy à un tableau.

```
>>> sin([ 0, pi/4, pi/3, pi/2 ])
array([ 0., 0.70710678, 0.8660254, 1. ])
```

```
>>> pl.plot(x, sin(x))
```

- Et sinon `fv = np.vectorize(f)` fait l'affaire !
- `y = list(map(f, x))` fonctionne aussi pourvu que `x` soit une liste.



Équations différentielles : odeint

- L'équation est donnée sous la forme $Y'(t) = F(Y(t), t)$ où F prend t comme deuxième argument :

```
>>> def f(y, t): y
```

```
ou encore
```

```
>>> f = lambda y, t: y
```

- On renvoie un tableau ndarray de valeurs approchées pour y .

```
>>> odeint(f, y0=1, t=np.linspace(0, 10, 1000))
array([[1.00000000e+00], [1.01006030e+00], ...,
       [2.20264698e+04]])
```

- Condition initiale : $y(t[0]) = y_0$
- En toute généralité y est un vecteur (pour pouvoir se ramener à l'ordre 1) donc `odeint` renvoie un tableau bidimensionnel.
- Pour ne récupérer ensuite qu'une composante on utilise une slice : `y[:, 0]`



Fonctions utiles

- Création et manipulation des tableaux

- ▶ `zeros((n, p))`
- ▶ `eye(n)`
- ▶ `random.random((n, p))`
- ▶ `fromfunction(f, (n, p))`
- ▶ `a.transpose()` ou `a.T`
- ▶ `dot(a, b)`

- Voir aussi : TP 5 CPP (pivot de Gauss). PDF protégé par mot de passe.

- Sous-bibliothèque `numpy.linalg`

- ▶ `inv(a)`
- ▶ `solve(a, b)`
- ▶ `qr(a)`
- ▶ `scipy.linalg.lu(a)`
- ▶ `eigvals(a)`



matplotlib

- Utilisation générale

```
import matplotlib.pyplot as pl
```

```
pl.plot([x1, x2, x3], [y1, y2, y3])
```

```
pl.title('Une jolie ligne brisée')
pl.grid()
pl.ylabel('Cette grandeur dépend')
pl.xlabel('de celle-ci')
...
pl.savefig('mongraphe.pdf')
pl.show()
```



scipy

- Résolution numérique d'équations `scipy.optimize`

- ▶ dichotomie : `scipy.optimize.bisect(f, a, b)`
- ▶ Newton : `scipy.optimize.newton(f, x0 [, f', f''])`
- ▶ plusieurs variables :


```
def f(x) : return (x[0] + x[1])**2,
              1 - x[1] + x[0]**2)
scipy.optimize.fsolve(f, (0, 0))
```

- Calcul d'intégrales `scipy.integrate` (trapèzes, Simpson)
- Equations différentielles `scipy.integrate.odeint`

