Stage LIESSE Python/BD

L'écosystème Python

Benjamin Wack (Benjamin.Wack@imag.fr)

Contributions de J. Courant et S. Gonnord

Université Joseph Fourier Grenoble 1

Juin 2014

Sommaire

Sommaire

Quelques IDE

- Spyder
  - inclut toutes les bibliothèques intéressantes
  - confortable
  - Python 2.x uniquement (mais Python 3 accessible en beta)
  - pas d'explorateur de pile
- IDLE
  - inclus dans la distrib standard sous windows
  - Python portable sur clé USB
  - très basique, n'inclut pas `numpy`/`scipy`.
- Wing IDE 101
  - pensé pour l'enseignement
  - explorateur de pile
  - version gratuite limitée, notamment la gestion des points de contrôle
- ...

Un point fort des IDE : débogueur

- Pas à pas et explorateur de variables

Un point fort des IDE : débogueur

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

Un point fort des IDE : débogueur

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

Sommaire

Listing des fonctions et constantes

- Pour accéder au contenu d'un module :

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

- Mais aussi :

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

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

Aide embarquée

- Un doute sur une fonction ?

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

ceil(...)
    ceil(x)

    Return the ceiling of x as an int.
    This is the smallest integral value >= x.
```

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

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

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.
```

Mais aussi...

- IPython
  - autocomplétion TAB
  - historique amélioré (persistent, `Out[4]`, Ctrl+R)
  - accès au code source `??`
  - `%time`

Mais aussi...

- IPython
  - autocomplétion TAB
  - historique amélioré (persistent, `Out[4]`, Ctrl+R)
  - accès au code source `??`
  - `%time`
- Sage
  - Alternative à Maple, Matlab... : calcul formel et numérique, traceur
  - **Très** pénible à installer, en particulier sous Windows...
  - ... mais accessible en ligne (`http://www.sagenb.org/`)

Mais aussi...

- IPython
  - autocomplétion TAB
  - historique amélioré (persistent, `Out[4]`, Ctrl+R)
  - accès au code source `??`
  - `%time`
- Sage
  - Alternative à Maple, Matlab... : calcul formel et numérique, traceur
  - **Très** pénible à installer, en particulier sous Windows...
  - ... mais accessible en ligne (`http://www.sagenb.org/`)
- py2exe
- pyhtmlizer, pygmentize...
- ...

Sommaire

numpy

`from numpy import *` ou bien `import numpy as np`

Objet central : `ndarray` ou `numpy.array` (à ne pas confondre avec `array.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]
```

numpy

`from numpy import *` ou bien `import numpy as np`

Objet central : `ndarray` ou `numpy.array` (à ne pas confondre avec `array.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`

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)`
- 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()
```

`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.

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

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

Équations différentielles : `odeint`

- L'équation est données 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, 1, x)
array([[1.],[1.22],[1.49],[1.82],[2.22],[2.72]])
```

- 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 slice.

Sommaire

Fichier texte

- En lecture

```
monfichier = open('lenom.txt', 'r')
l = monfichier.readline()
for l in monfichier.readlines():
...
```

- En écriture

```
monfichier = open('lenom.txt', 'w')
monfichier.write('plof')
```

- Dans tous les cas

```
monfichier.close()
```

- Alternative :

```
with open('lenom.txt', 'w') as monfichier:
    l = monfichier.readline()
```

Fichier texte

- Supprimer des caractères en début et/ou fin de ligne

```
ligne.strip()
ligne.lstrip('...')
```

- Découper une ligne selon un séparateur

```
ligne.split(',')
```

- Compter les occurrences d'une sous-chaîne

```
ligne.count('0')
```

- cf librairie `string`

Image

- Récupérer une image

```
from PIL import Image
mb = Image.open('mario.png')
mb.show()
mb.size
```

- `mbtab = numpy.array(mb)` donne le tableau bidimensionnel des pixels.
- 1 pixel = 1 entier dans $[0; 255]$ (niveaux de gris)
  ou un tableau de 3 entiers (couleurs)

`Image` (suite)

- Créer une image à partir d'un tableau

```
newmb = Image.fromarray(mbtab)
newmb.save('fich.jpg')
```

- En Python 2.x : simplement `import Image`
- Écrire et dessiner dans un pdf :

```
from reportlab.pdfgen import canvas
```

Mais aussi...

- random
- decimal, fractions
- zipfile, urllib
- antigravity
- ...