IPT : Cours 4

Les Bibliothèques de Python

MPSI-Schwarz : Prytanée National Militaire

Pascal Delahaye

27 mars 2015

Python est un langage de programmation et permet donc de créer ses propres fonctions (ou programmes). Cependant, ce langage ne serait pas très utile s'il fallait soi-même reconstruire toutes les fonctions usuellement utilisées en programmation. Le succès de Python parmi les informaticiens a eu pour conséquence la mutualisation du travail de chacun et la mise à disposition de tous d'un grand nombre de fonctions nouvelles regroupées dans des bibliothèques. On trouve ainsi des bibliothèques destinées à effectuer des représentations graphiques `matplotlib` avec le module `matplotlib.pylab`, des bibliothèques adaptées à l'algèbre `numpy` et l'algèbre linéaire `numpy.linal`, des bibliothèques consacrées au calcul formel `sympy`, des bibliothèques de calcul scientifique (ou calcul numérique) `scipy` ...etc...

1 La bibliothèque de base

Dans le travail précédent, nous avons vu que certaines fonctions faisaient partie intégrante du langage Python : on dit qu'elles font partie de la *bibliothèque de base*. Par exemple `print()`, `input()`, `sin()`, `log()`, `sqrt()`, `arccos()`, `tanh()`, `abs()`, `floor()` ... Ces fonctions ont été intégrées au langage car elles sont d'utilisation courante.

Parmi les *built-in functions* on trouve :

1. Les fonctions mathématiques usuelles
2. `input()`, `print()`
3. `max()` et `min()` : renvoie le maximun et le minimum d'une séquence de nombres entiers ou flottants
4. `divmod()` : renvoie le quotient et le reste d'une division euclidienne
5. `floor()` : renvoie la partie entière d'un nombre décimal
6. `bin()` : renvoie l'expression binaire d'un entier relatif (sous la forme d'une chaine de caractères)
7. `len()`, `range()`, `sorted()`, `sum()` : pour les listes
8. `random()` : renvoie un floattant compris entre 0 et 1
9. `help()` : renvoie des informations sur l'objet en argument (fonction, module, bibliothèque...)
10. `dir()` : renvoie le nom des objets (fonctions, symboles et modules) définis dans une bibliothèque

D'autres fonctions, plus spécifiques à une utilisation donnée, sont stockées dans des bibliothèques : *sympy*, *numpy*, *matplotlib*, *scipy*... Toutes ces bibliothèques sont gratuites et disponibles sur le net sous la forme de répertoires ou de fichier exécutable d'installation.

2 Installation et importation d'une nouvelle bibliothèque

Certaines de ces bibliothèques sont d'ores et déjà installées sur les ordinateurs. C'est le cas des 4 bibliothèques précédentes lorsque Python a été installé via la distribution winPython32bits. Pour s'en assurer, on peut aller vérifier que les répertoires *sympy*, *numpy*, *matplotlib*, *scipy* sont bien présents sur le disque dur à l'adresse suivante :

`\ WinPython-32bit-3.3.1.0\ python-3.3.1\ Lib\ site-packages\ .`

Remarque 1. **Installation d'une nouvelle bibliothèque :**
Il existe beaucoup d'autres bibliothèques disponibles gratuitement sur internet (plus de 200). Pour les installer sur votre ordinateur, il suffit de télécharger le répertoire portant le nom de la bibliothèque (et contenant les fichiers de fonctions) et de le copier à l'adresse précédente, avec les autres bibliothèques.

Remarque 2. **Importation d'une nouvelle bibliothèque ou fonction :**
Les fichiers `*.py` contenus dans les bibliothèques sont des fichiers TEXT contenant la programmation en python des nouvelles fonctions disponibles. Lorsqu'un fichier contient la définition de plusieurs fonctions, il porte le nom de *module*.

Pour pouvoir utiliser ces nouvelles fonctions dans la console ou dans l'éditeur, il faut au préalable les avoir importées (oo dit aussi *activées*).
Pour, par exemple, importer les fonctions contenues dans une bibliothèque (par exemple `turtle`), on dipose de plusieurs options :

1. `>>> import turtle` Importe toutes les fonctions en ajoutant le préfixe "turtle" devant le nom des fonctions
2. `>>> import turtle as tl` Importe toutes les fonctions en ajoutant le préfixe "tl" devant le nom des fonctions
3. `>>> from turtle import *` Importe toutes les fonctions sous leur nom d'origine

Si, seules les fonctions d'un module "toto" nous intéressent, on fait la même chose en remplaçant `turtle` par `turtle.toto`

⚠ Même si la troisième option semble attrayante, elle présente un inconvénient majeur lorsque des fonctions de différents modules portent le même nom. En effet, l'importation de fonctions sous la forme `>>> from turtle import *` fait disparaître toutes les fonctions antérieurement importées qui portent le même nom que les nouvelles fonctions.

Remarque 3.

1. On remarquera que les bibliothèques `numpy`, `matplotlib` et `scipy` sont automatiquement importées dans la console de Spyder lors de l'ouverture du logiciel. Quant à `sympy`, l'importation devra se faire manuellement.
2. Cependant, lors d'un travail dans l'éditeur, il faudra impérativement en début de programme, procéder à l'importation des toutes les fonctions utilisées par le programme.

3 Comment connaître les fonctions contenues dans une bibliothèque ?

Il n'existe pas de moyen simple pour obtenir la liste des fonctions définies dans une bibliothèque.
Les instructions suivantes donne des informations sur les objets définis dans la bibliothèque, mais ne donne pas de détail des fonctions définies dans les modules contenus par la bibliothèque.

```
>>> dir('numpy')      # Donne la liste des objets (fonctions, symboles, modules)
                        définis dans la bibliothèque
>>> help('numpy')     # Donne également tout le détail relatif à chacune des fonctions
```

`Exemple 1.`
Consultez l'aide pour obtenir des informations sur la bibliothèque *matplotlib*.
Vous constaterez qu'il n'est pas très facile de s'y retrouver...
En pratique, il sera beaucoup plus efficace :

1. soit de connaître par coeur le nom des fonctions que l'on souhaite utiliser,
2. soit de consulter le "tableau de fonctions" que je vous ai distribués.

On pourra alors consulter l'aide dans l'inspecteur d'objets pour retrouver comment utiliser ces fonctions.

4 Importation de fonctions dans un programme de l'éditeur.

Dans la console, la plupart des fonctions des différentes bibliothèques sont d'ores et déjà activées. Mais attention... ce n'est pas le cas des fonctions contenues dans la bibliothèque *sympy* !

⚠ En revanche lorsqu'on utilise l'éditeur, il est indispensable dans le texte du programme, de préciser la provenance (ie les bibliothèques et les modules) des différentes fonctions utilisées.

Pour importer seulement certaines fonctions de la bibliothèque `bibli` (bibliothèque fictive) :

1.	`from bibli import f`	importe uniquement la fonction `f` de la bibliothèque sous le nom `f`
2.	`from bibli import f,g`	importe uniquement les fonctions `f` et `g` de la bibliothèque

Remarque 4. Si la fonction est contenu dans le module *module* de la bibliothèque *bibli*, il faut taper :

```
from bibli.module import f
```

5 Comment retrouver la façon d'utiliser une fonction ?

Pour savoir :

1. à quoi sert une fonction donnée
2. dans quelle bibliothèque elle se trouve
3. comment l'utiliser

il suffit de taper la fonction dans la console sous la forme `fonction(` et de lire les informations renvoyées par l'inspecteur d'objets. Cette technique fonctionne uniquement lorsque la fonction recherchée a d'ores et déjà été importée.

```
>>> plot(
```

Remarque 5.

1. Eh oui ! !... L'aide est en anglais...
2. L'aide proposée donne souvent des exemples d'utilisation dont il est possible de s'inspirer.
3. L'inspecteur d'objet nous indique en particulier la bibliothèque qui contient cette fonction. Cette information est fondammentale pour procéder à l'importation (ou l'activation) d'une fonction dans l'éditeur.

6 Les 5 bibliothèques principales

1. **math** :
   Cette bibliothèque contient les fonctions mathématiques usuelles.
   On l'utilisera dans l'éditeur lors de la conception de programmes utilisant des fonctions mathématiques.

2. **sympy** :
   Pour le calcul formel (factorisation, développement, simplification, dérivée, intégrales, primitives, equations différentielles, développements limités...)

   ⚠ Les fonctions de cette bibliothèque ne sont pas compatibles avec fonctions mathématiques de la bibliothèque de base. Il faut donc impérativement importer les propres fonctions mathématiques de cette bibliothèque.
   On utilisera aussi le module **sympy.abc** pour l'importation de variables utilisées dans les expressions algébriques.

3. **mathplotlib.pylab** :
   Pour les représentations graphiques (nuages de points, fonctions, courbes paramétrées)

4. **numpy** :
   Pour la manipulation des complexes et des tableaux

5. **numpy.linal** :
   Pour effectuer des opérations d'algèbre linéaire (matrices...)

6. **scipy.integrate** :
   Pour les fonctions d'analyse numérique (résolution approchée d'équations différentielles, calcul approché d'intégrales...)

`Exemple 2.` Vérifiez que les fonctions de la bibliothèque `sympy` ne reconnaissent que ses propres fonctions mathématiques.

La fonction `plot()`

La fonction `plot()` est disponible dans les deux bibliothèques `matplotlib.pylab` et `sympy`.
Cependant, ces deux fonctions s'utilisent de façons différentes :

1. La fonction `plot()` de `matplotlib.pylab` :
   La syntaxe de base est la suivante :

   `plot([...],[...])` avec les deux listes contenant les abscisses et les ordonnées des points

2. La fonction `plot()` de `sympy` :
   La syntaxe de base est la suivante :

   `plot(f(x),(x,a,b))` avec $\begin{cases} f(x) \text{ l'expression de la fonction} \\ x \text{ la variable} \\ a,\ b \text{ les bornes de l'intervalle} \end{cases}$

   ⚠ Pour utiliser cette syntaxe, il faut auparavant :
   (a) avoir déclaré une variable "x" avec la fonction `var()` de `sympy`
   (b) avoir importé depuis `sympy` les fonctions mathématiques utilisées

`Exemple 3.` Tracer la courbe représentative de $x \mapsto \cos(x)$ sur l'intervalle $[-5,\ 5]$ à l'aide de chacune des deux fonctions `plot()` disponibles.

Pour utiliser la fonction plot() de ***`matplotlib.pylab`****, on pourra exécuter les instructions*

```
X = arange(-5,5,0.1)
Y = cos(X)
```