Initiation à la programmation en Python

Damien Vergnaud

École Normale Supérieure

10 mars 2010

Table des matières

Modules

- On peut ranger les définitions de fonctions se rapportant à une même application au sein d'un script commun baptisé module.
- Un module est sauvegardé sous forme d'un fichier dont le nom a la forme `<nom du module>.py`.
- Pour utiliser un module, il faut se servir de l'instruction `import <nom du module>`.
- L'exécution de cette instruction consiste à exécuter le script définissant le module (ce script peut contenir des instructions autres que des définitions de fonctions).
- Pour importer un module, Python a besoin de connaître le chemin qui permet d'accéder au fichier correspondant. Ce chemin doit apparaître dans la liste des chemins possibles stockés dans la variable `path` du module `sys`.

Modules - Première méthode d'importation

```
>>> import random
>>> random.randint(0,10)
9
```

Regardons de plus près cet exemple :

- L'instruction `import` vous permet d'importer toutes les fonctions du module `random`
- Ensuite, nous utilisons la fonction (ou méthode) `randint(a,b)` du module random; attention cette fonction renvoie un nombre entier aléatoirement entre a inclus et b inclus.

Modules - Deuxième méthode d'importation

- Pour disposer d'une fonction du module:

Syntaxe

```
from [module] import [fonction]
```

- Pour disposer de toutes les fonctions d'un module:

Syntaxe

```
from [module] import *
```

```
from math import *
racine = sqrt(49)
angle = pi/6
print sin(angle)
```

Modules courants

- `math` : fonctions et constantes mathématiques de base (sin, cos, exp, pi...).
- `sys` : passage d'arguments, gestion de l'entrée/sortie standard etc...
- `os` : dialogue avec le système d'exploitation.
- `random` : génération de nombres aléatoires.
- `time` : permet d'accéder aux fonctions gérant le temps.
- `calendar` : fonctions de calendrier.
- `urllib` : permet de récupérer des données sur internet depuis python.
- `Tkinter` : interface python avec Tk (permet de créer des objets graphiques; nécessite d'installer Tk).
- `re` : gestion des expressions régulières.
- `pickle` : écriture et lecture de structures python (comme les dictionnaires par exemple).
- ...

Obtenir de l'aide sur les modules importés

- Pour obtenir de l'aide sur un module rien de plus simple, il suffit d'invoquer la commande help :

```
>>> import random
>>> help(random)
```

- Il est aussi possible d'invoquer de l'aide sur une fonction particulière d'un module en la précisant de la manière suivante :

```
>>> help(random.randint)
```

Utilisation de fichiers

- Il est important de dissocier les données des programmes qui les utilisent en rangeant ces données dans des fichiers séparés.

- Le module os contient des fonctions qui permettent de localiser les fichiers :
  - `getcwd()`: Retourne le chemin du répertoire courant

    `chdir(<ch>)`: Change le répertoire courant qui prend la valeur donnée par la chaîne de caractères `<ch>`

    `path.isfile(<ch>)`: Retourne un booléen qui indique s'il existe un fichier dont le chemin est la chaîne de caractères `<ch>`

    `path.isdir(<ch>)`: Retourne un booléen qui indique s'il existe un répertoire dont le chemin est la chaîne de caractères `<ch>`

```
>>> from os import chdir
>>> chdir("/home/exercices")
```

Les deux formes d'importation

```
>>>>>> import os
>>> rep_cour = os.getcwd()
>>> print rep_cour
```

```
>>> from os import getcwd
>>> rep_cour = getcwd()
>>> print rep_cour
```

Utilisation de fichiers

- Pour utiliser un fichier identifié par le chemin `ch` dans un programme Python, il faut commencer par l'ouvrir par l'appel de fonction `open(<ch>, [<mode>])` qui retourne un objet de type `file`.

- Le paramètre facultatif `<mode>` indique le mode d'ouverture du fichier :
  - `r` : mode lecture (le fichier doit exister préalablement)
  - `w` : mode écriture (si le fichier existe, les données sont écrasées, sinon le fichier est créé)
  - `a` : mode ajout (si le fichier existe, les données écrites vont l'être après celles existantes, sinon le fichier est créé)

- Si le mode est omis, le mode par défaut est `r`.

Utilisation de fichiers

- un objet de type `file` est associé à des attributs et des méthodes. En voici quelques-unes :
  - `read([<n>])` : Retourne la chaîne des `<n>` caractères restants.
  - `write(<ch>)` : Ecrit la chaîne de caractères `<ch>`.
  - `close()` : Ferme le fichier quand il est fini d'être utilisé.
  - `seek(<n>)` : Choisit le caractère `<n>` comme position courante du fichier.
  - `tell()`: Retourne le caractère en position courante.

Exemple

- Créez un fichier dans un éditeur de texte que vous sauverez dans votre répertoire avec le nom 'exemple.txt', par exemple :

```
Ceci est la premiere ligne
Ceci est la deuxieme ligne
Ceci est la troisieme ligne
Ceci est la quatrieme et derniere ligne
```

```
>>> filin = open('exemple.txt','r')
>>> filin
<open file 'exemple.txt', mode 'r' at 0x40155b20>
>>> filin.readlines()
['Ceci est la premiere ligne\n', 'Ceci est la deuxieme ligne\n',
'Ceci est la troisieme ligne\n',
'Ceci est la quatrieme et derniere ligne\n']
>>> filin.close()
>>> filin
<closed file 'exemple.txt', mode 'r' at 0x40155b20>
```

Exemple

```
Ceci est la premiere ligne
Ceci est la deuxieme ligne
Ceci est la troisieme ligne
Ceci est la quatrieme et derniere ligne
```

```
>>> filin = open('exemple.txt','r')
>>> lignes  = filin.readlines()
>>> for i in lignes:
...   print i
...
Ceci est la premiere ligne

Ceci est la deuxieme ligne

Ceci est la troisieme ligne

Ceci est la quatrieme et derniere ligne
>>> filin.close()
```

Exemple

```
Ceci est la premiere ligne
Ceci est la deuxieme ligne
Ceci est la troisieme ligne
Ceci est la quatrieme et derniere ligne
```

```
>>> filin = open("exemple.txt","r")
>>> filin.read()
'Ceci est la premiere ligne\nCeci est la deuxieme ligne\nCeci est la troisi
>>> filin.close()
```

Exemple

```
Ceci est la premiere ligne
Ceci est la deuxieme ligne
Ceci est la troisieme ligne
Ceci est la quatrieme et derniere ligne
```

```
>>> filin = open("exemple.txt","r")
>>> filin.tell()
0L
>>> filin.readline()
'Ceci est la premiere ligne\n'
>>> filin.tell()
27L
>>> filin.seek(0)
>>> filin.tell()
0L
>>> filin.readline()
'Ceci est la premiere ligne\n'
>>> filin.close()
```

Exemple

```
>>> animaux = ['girafe', 'hippopotame', 'singe', 'dahu' , 'ornithorynque']
>>> filout = open('exemple2.txt','w')
>>> for i in animaux:
...   filout.write(i)
...
>>> filout.close()
>>>

$ more exemple2.txt
girafehippopotamesingedahuornithorynque
$
```