IPT : Cours 3

Présentation de Python et de l'IDE Spyder

MPSI : Prytanée National Militaire

Pascal Delahaye

28 septembre 2015

1 La programmation

Programmer consiste à demander à un ordinateur d'effectuer des tâches, appelée aussi *instructions* ou *commandes*. Or, l'être humain et l'ordinateur ne parlent pas le même langage : l'ordinateur ne sait reconnaître qu'une succession de 0 et de 1 (le langage machine), et il est bien entendu illusoire de penser qu'un homme puisse apprendre un tel langage. Pour permettre à l'un et à l'autre de communiquer, on fait appel à Un langage de programmation : basic, pascal, fortran, C++, caml... ou encore python.

1. La rédaction des programmes dans le langage choisi se fait alors avec un logiciel appelé *interface graphique*
2. Lors de l'exécution des commandes ou du programme, un autre logiciel appelé *compilateur, interpréteur ou noyau*, traduit le langage informatique en langage machine et effectue les tâches demandées.

Python est un langage parmi beaucoup d'autres, permettant le développement de programmes informatiques.

1.1 Les particularités d'un langage

Les différents langages informatiques se différencient par :

1. la syntaxe des instructions : mots clés utilisés, espaces, points virgules, deux points, opérateurs...
2. les *fonctions de bases* disponibles : `print()`, `input()`, `eval()`, fonctions mathématiques...
3. les *bibliothèques* disponibles contenant des fonctions supplémentaires
4. les structures de données (ou *classes*) disponibles : listes, chaines de caractères, piles, dictionnaires...
5. les *méthodes* et les *fonctions* associées à ces structures disponibles

Remarque 1. Même si beaucoup d'utilisateurs confondent souvent les deux notions :

1. Une *fonction* est un programme qui traitent les données en argument et renvoie un résultat à l'utilisateur. Par exemple `len(L)`,...

2. Une *méthode* est un programme qui effectue un traitement sur une structure de donnée, sans pour autant renvoyer un résultat à l'utilisateur. Par exemple : `L.reverse()`, `L.append(a)`, ...

Dans ce cours, le langage choisi est Python 3.4

1.2 Interface graphique

Pour rédiger et exécuter un programme en python, il existe de nombreux logiciels (appelés *interfaces graphiques* ou *IDE : Environnement de Développement Intégré*) possibles. Parmi les interfaces possibles (Notebook, tKinder, IDLE, Eclipse, bloc-note, word...), nous avons choisi d'utiliser le logiciel "Spyder". Celui-ci présente de nombreux avantages d'aide à la programmation.

1.3 Installation

Vous pouvez télécharger librement la distibution *Anaconda3-2.3.0-Windows-x86.exe* en vous rendant à l'adresse suivante :

http ://continuum.io/downloads#py34

Cette distribution comprend à la fois le module de langage *Python version 3.4*, l'interface graphique *Spyder* (IDE) et inclut les bibliothèques que nous serons amenés à utiliser (*matplotlib, numpy, scipy, sympy*). Elle présente enfin l'avantage d'être *portable*, c'est à dire qu'elle peut être utilisée depuis n'importe quel support externe (clé USB, DD externe, carte mémoire SD) sans avoir besoin d'être installée sur l'ordinateur utilisé.

2 Présentation de l'environnement SPYDER

Spyder est constitué de 3 fenêtres :

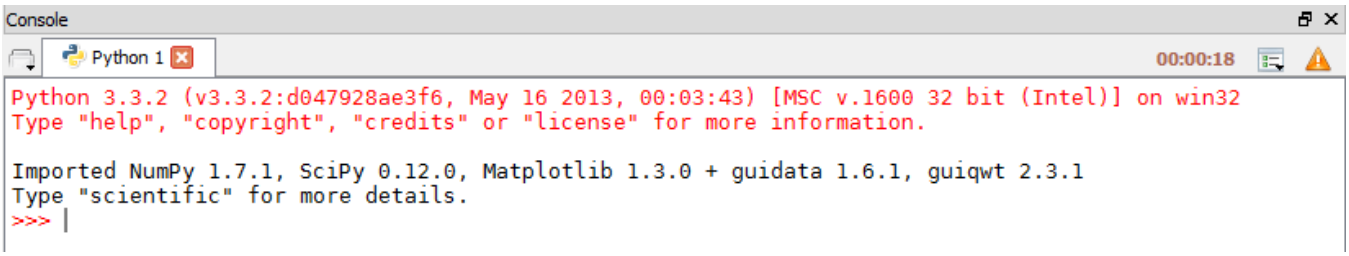
1. L'*éditeur* qui permet de rédiger des programmes.

2. La *console* qui permet de tester des commandes (onglet : Console IPython) et qui renvoie les résultats des programmes rédigés dans l'éditeur (autres onglets)

3. L'*explorateur* avec pour onglets :
   (a) L'*inspecteur d'objets* qui donne des informations sur l'utilisation des fonctions activées
   (b) L'*explorateur de variables* qui donne la liste et les valeurs de toutes les variables qui ont été crées
   (c) L'*explorateur de fichiers* qui donne accès au disque dur

2.1 La console

Au démarrage de Python, la console affiche la version Python qui est utilisée ainsi que les différentes bibliothèques automatiquement importées. Le curseur qui clignote après le $''>>>''$ indique l'endroit où sont entrées les commandes.



Dans la console taper une à une les instructions suivantes, en validant à la fin de chaque ligne :

```
>>> A = 2
>>> A

>>> print("toto", 6, "titi")
>>> 4 + 2

>>> 5/3
>>> 5//3

>>> a = input("entrer une valeur : ")
>>> a
```

Les flèches ↑ et ↓ permettent de retrouver les commandes tapées précédemment dans la cession.

Remarque 2. Dans la console, il faut avoir exécuté la première commande avant de pouvoir taper la seconde ; on dit que la console est le *mode interactif* de Python. Elle sert donc à l'exécution de commandes simples ou à tester des commandes pour lesquelles on a des doutes. La console n'est pas adaptée à la construction d'un programme puisque ceux-ci, en général, sont constitués de plusieurs lignes de commandes (*les instructions*). Les programmes sont donc rédigés dans la fenêtre *éditeur*.

2.2 L'éditeur

Nous utiliserons la fenêtre "éditeur" lors de la création de programmes...

Cet environnement facilite la rédaction d'un programme en :

1. imposant des indentations lorsque c'est nécessaire ;
2. mettant en couleur les fonctions, les mots clés et les chaînes de caractères (entre guillemets) ;
3. proposant une aide pour la gestion des parenthèses
4. indiquant par un panneau ⚠ les erreurs éventuelles de syntaxe ;
5. affichant une aide contextuelle pour l'utilisation des fonctions Python.

Une fois le programme rédigé, on l'enregistre dans un fichier *.py puis on peut l'exécuter à l'aide de la touche F5. Les résultats s'affichent alors dans la console intéractive ou une console spécifique dédiée au fichier contenant le programme.

Remarque 3. Dans l'éditeur, il est possible d'ouvrir plusieurs fichiers correspondant chacun à des programmes différents.

2.3 La 3ème fenêtre : l'explorateur

1. L'inspecteur d'objets :

   Dans la zone *objet* de l'inspecteur d'objets, taper successivement : `print`, `eval` puis `sqrt`.
   Vous constatez que l'inspecteur d'objet reconnaît ces différentes fonctions et vous donne des informations sur leur utilisation.
   Recommencer cette opération en tapant `print(`, `eval(` puis `sqrt(` dans la console.
   Ces fonctions sont, là encore, reconnues par l'inspecteur d'objets qui renvoie automatiquement leur description.

   ⚠ L'inspecteur d'objets reconnaît automatiquement les fonctions actives et donne une information sur leur utilisation. Si aucune information sur la fonction apparaît dans l'interpréteur, c'est soit que cette fonction n'existe pas soit qu'elle n'a pas encore été importée depuis la bibliothèque qui la contient !
   Vérifiez cela en tapant dans la console `ln(2)` puis `log(2)`.

2. L'explorateur de variables :
   Il contient toutes les variables qui ont été définies lors de la cession de travail, en précisant : leur type et leur valeur. Vérifiez ainsi qu'à l'ouverture de spyder, l'explorateur de variables ne contient que les variables `e` et `pi`.

Créer les variables `a`, `b` et `c` à l'aide des instructions suivantes :

```
a = "toto va à la plage"
b = 3.14
c = 3,14,15
```

Vérifiez alors que l'explorateur de variables donne bien la liste de toutes les variables globales accessibles ou définies par l'utilisateur.

3. L'explorateur de fichiers :
   Vérifiez que l'explorateur de fichiers donne accès à l'arborescence du disque dur. A priori, nous n'aurons pas besoin de l'utiliser...

2.4 Le débogueur

Il arrive très fréquemment que les programmes construits comportent des erreurs. Spyder comprend un module d'aide à la correction, le *débogueur*, que l'on utilise lorsque les erreurs sont difficiles à détecter. Il permet l'exécution instruction après instruction d'un programme et permet ainsi de détecter les instructions posant problème.

Copier le programme suivant dans l'éditeur :

```
x = 10
y = 7
x = x+ y
y = x
x = 5 / (x-y)
```

Pour lancer le débogueur, on sélectionne la fonction "Déboguer" dans le menu "exécution" (ou plus rapidement on tape : "ctrl + F5"). La première instruction s'affiche dans la console.

1. **Pour l'exécuter pas à pas :**
   On tape "n" (comme "next") puis "entrer". On vérifie alors l'état des variables dans l'explorateur et on recommence pour lancer la deuxième instruction... On constate que le programme s'arrête sur la 5ème instruction en affichant un message d'erreur.

   On peut ainsi suivre pas à pas l'exécution d'un programme et repérer les instructions problématiques.

2. **Pour fixer des points d'arrêt :**
   Au lieu d'exécuter un programme pas à pas, on peut décider d'arrêter son exécution au niveau de certaines instructions. On commence alors par fixer des *points d'arrêt* devant les instructions avant lesquelles on souhaite s'arrêter. Pour cela, il suffit de double-cliquer dans la marge de gauche devant l'instruction. On tape alors "c" puis "entrer" pour indiquer au programme de "continuer".

Exercice : 1

1. Taper le programme suivant dans l'éditeur puis l'exécuter.

```
i = 10
while i != 0:
      i = 1-i
      print(i)
```

   Qu'observez-vous ?...

2. Exécuter ce programme pas à pas et observer les valeurs prises par la variable "i" après chaque instruction. Expliquer le comportement observé à la première question.

3. Placer un point d'arrêt à un endroit approprié pour montrer son comportement sans avoir besoin de détailler les étapes inutiles.