Laboratoire # 1
Ramper avec Python

Avant d'essayer de marcher, il faut bien commencer quelque part !

Semaine 1

Ce premier laboratoire a pour objectif de s'assurer que vous un environnement de développement Python fonctionnel sur votre machine, et de vous familiariser un peu avec les différents outils.

1 Installation des logiciels utiles

Pour commencer, vous devez installer les logiciels nécessaires pour le cours, sur votre ordinateur. Si vous utilisez les ordinateurs du département d'informatique et de génie logiciel, vous aurez accès déjà accès à Python 3 ainsi qu'à PyCharm, un environnement de développement pour Python. Nous vous recommandons par contre de procéder à l'installation sur votre propre ordinateur, pour vous permettre de travailler à la maison.

Procédez donc à l'installation et à la configuration, en lisant le tutoriel disponible dans la partie « contenu et activités » du site Web du cours, puis sur la page « Python, logiciels et tutoriels ».

2 Premiers pas

Maintenant que nous avons installé PyCharm, prenons le temps de ne PAS l'utiliser... Nous allons donc commencer juste avec l'interpréteur de Python basique et s'amuser un peu. Si anodins que semblent ces petits tests, vous découvrirez les premiers rudiments de la syntaxe du langage que nous allons apprendre en détails dès la deuxième semaine de la session. Ouvrez l'interpréteur interactif Python (si vous ne savez pas comment vous y prendre, consultez le tutoriel sur le site Web du cours).

Vous devriez obtenir une fenêtre avec plus ou moins le contenu suivant :

```
Python 3.4.3 |Anaconda 2.3.0 [...]
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Python vous souhaite à sa façon, la bienvenue dans son interpréteur de commandes.

C'est quoi cet interpréteur ?

Cet interpréteur de commandes va nous permettre de tester directement du code. J'entre une ligne d'instruction, je fais entrée (enter), je regarde ce que me dit Python (s'il me dit quelque chose), puis j'en entre une deuxième, une troisième... Cet interpréteur est particulièrement utile pour comprendre la base de Python et réaliser nos premiers petits programmes. Le principal inconvénient, en contre-partie, étant que le code que vous entrez n'est pas sauvegardé si vous ne le faites pas manuellement, mais on verra un peu plus loin comment le conserver.

Dans la fenêtre que vous avez sous les yeux, l'information qui ne change pas d'un système d'exploitation à l'autre et qui est, somme toute, la plus importante, est la série des trois chevrons qui se trouvent en bas à gauche des informations : >>>. Ces trois signes signifient : je suis prêt à recevoir tes instructions. C'est ce qu'on appelle l'invite de commande de l'environnement Python.

Les langages de programmation respectent une syntaxe claire. On ne peut espérer que l'ordinateur « comprenne » si, dans cette fenêtre, vous commenciez à lui dire : j'aimerais que tu me codes un jeu vidéo génial. Et autant que vous le sachiez tout de suite (bien qu'à mon avis vous vous en doutiez), on est très loin d'obtenir des résultats aussi spectaculaires à notre niveau.

Tout ça pour dire que si vous entrez n'importe quoi dans cette fenêtre, il est plus que probable que Python vous indique, clairement et fermement, qu'il n'a rien compris.

Si par exemple vous entrez « premier test avec Python », vous obtenez le résultat suivant :

```
>>> premier test avec Python
  File "<stdin>", line 1
    premier test avec Python
                ^
SyntaxError: invalid syntax
```

L'interpréteur vous indique qu'il a trouvé un problème dans votre ligne d'instruction. Il vous donne la ligne (en l'occurrence la première) qu'il vous répète obligeamment (ceci est très utile quand on travaille sur un programme de plusieurs centaines de lignes) puis vous dit ce qui l'arrête, ici : `SyntaxError: invalid syntax`. Limpide n'est-ce pas ? Ce que vous avez entré est incompréhensible pour Python. Enfin, la preuve qu'il n'est pas rancunier, c'est qu'il vous réaffiche une série de trois chevrons, montrant bien qu'il est prêt à retenter l'aventure !

Essayons un autre exemple qui fonctionne.

```
>>> print('Hello world!')
Hello world!
```

L'environnement Python vous permet donc d'explorer la syntaxe de Python, d'accéder à l'aide interactive et de déboguer des programmes courts. Un environnement graphique de programmation (qui n'est rien d'autre qu'un éditeur de texte plus convivial) comme IDLE, Eclipse ou PyCharm vous permettront d'écrire des séquences d'instructions et de les sauvegarder dans un fichier, ce qui s'avèrera très pratique lorsque nous aurons de « vrais » problèmes à résoudre.

Avant tout, l'environnement Python est un véritable terrain de jeu interactif. Tout au long du cours, vous verrez des exemples comme celui-ci :

```
>>> 1 + 1
2
```

La partie que vous devez entrer est `1 + 1`. Vous pouvez entrer n'importe quelle expression ou commande Python valide. N'ayez pas peur, le pire qui puisse arriver est qu'un message d'erreur s'affiche. Les commandes sont exécutées immédiatement (une fois appuyé sur Entrée), les expressions sont évaluées immédiatement et l'environnement affiche le résultat. Ici, le résultat affiché est 2, 1 + 1 est en fait une expression Python valide, dont le résultat est bien sûr 2.

Vos premières instructions : un peu de calcul mental pour l'ordinateur

Nous allons essayer d'obtenir des résultats à des calculs pas compliqués du tout. Je vous rappelle encore une fois qu'exécuter les tests sur votre machine est une très bonne façon de vous rendre compte de la syntaxe et surtout, de la retenir.

Entrer un nombre

Vous avez pu voir, sur notre premier test que Python n'aimait pas particulièrement les suites de lettres qu'il ne comprend pas. Cependant, l'interpréteur adore les nombres. D'ailleurs, il les accepte sans sourciller, sans une seule erreur :

```
>>> 7
7
>>>
```

D'accord, ce n'est pas extraordinaire. On entre un nombre, et l'interpréteur retourne ce nombre.

Mais dans bien des cas, ce simple retour indique qu'il a bien compris et que ce que vous avez entré est en accord avec sa syntaxe. Vous pouvez de même entrer des nombres à virgule :

```
>>> 9.5
9.5
>>>
```

Attention : on utilise ici la notation anglo-saxonne, le point remplace la virgule. La virgule a un tout autre sens pour Python, prenez donc cette habitude dès maintenant. Les nombres à virgule se comportent parfois étrangement, c'est pourquoi il faut les manier avec prudence.

```
>>> 8.13 + 3.4
11.530000000000001
>>>
```

... pourquoi ce résultat approximatif ?

Python n'y est pas pour grand chose. En fait, le problème vient en grande partie de la façon dont les nombres à virgule sont écrits dans la mémoire de votre ordinateur. Cependant, vous remarquerez que l'erreur est infime et qu'elle n'a pas d'impact formidable sur les calculs. Les applications qui ont besoin d'une précision mathématique à toute épreuve essayent de corriger ces défauts par d'autres moyens, mais ici ce ne sera pas nécessaire. Il se peut que sur votre machine, l'évaluation de `8.13 + 3.4` n'ait pas retourné une approximation : ça dépend de votre système d'exploitation, de l'architecture de votre microprocesseur, etc.

Il va de soi que l'on peut tout aussi bien rentrer des nombres négatifs (vous pouvez faire l'essai d'ailleurs).

Addition, soustraction, multiplication, division

Pour effectuer ces opérations, on utilise respectivement les symboles « + », « - », « * » et « / ».

```
7
>>> 3 + 4
7
>>> -2 + 93
91
>>> 9.5 + 2
11.5
>>>
```

Faites également des tests pour la soustraction, la multiplication et la division, il n'y a rien de nouveau pour vous j'en suis sûr !

Division entière et modulo

Si vous avez pris le temps de tester la division, vous vous êtes rendu compte que le résultat est donné en virgule flottante.[1]

[1] En Python 2, le résultat aurait été un nombre entier, car diviser ensemble deux nombres entiers donnait nécessairement un nombre entier. Ce comportement a été modifié dans Python 3.

```
>>> 10/5
2.0
>>> 10/3
3.3333333333333335
>>>
```

Il existe deux autres opérateurs qui permettent de connaître le résultat d'une division entière et le reste de cette division. Le premier opérateur est « // ». Il permet d'obtenir la partie entière d'une division.

```
>>> 10 // 3
3
```

L'opérateur « % » que l'on appelle « modulo » permet de connaître le reste de la division :

```
>>> 10 % 3
1
```

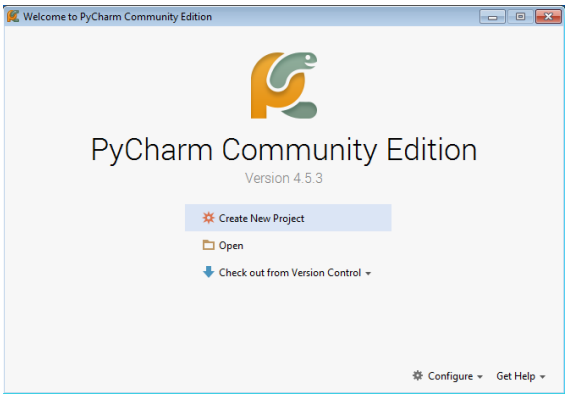
Cette notion de partie entière et de reste de division n'est pas bien difficile à comprendre et vous servira par la suite. Si vous avez du mal à en saisir le sens, sachez donc que : la partie entière de la division de 10 par 3 est le résultat de cette division, sans tenir compte des chiffres au-delà de la virgule (en l'occurrence, 3). Pour obtenir le reste de 10/3, on regarde quelle est la partie entière. Ici elle est de 3 (10//3). On multiplie cette partie entière par 3 (celui de 10/3) et on obtient... 9. Entre 9 et 10, on a 1... et c'est le reste de notre division. Vous devriez retrouver ces chiffres facilement en posant la division sur papier.

3 Programmer son premier script avec PyCharm

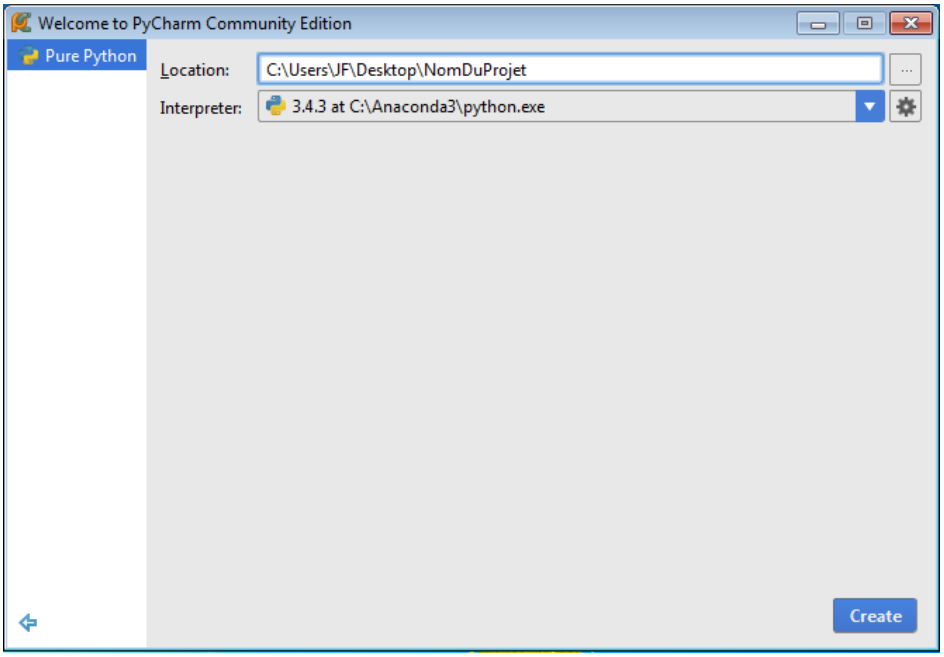
Utiliser l'interpréteur de commande est très utile, mais les commandes que vous tapez ne sont pas sauvegardées et ne peuvent donc réutilisées. Nous devons être capable de sauvegarder nos commandes dans un fichier et pouvoir les exécuter autant de fois que c'est nécessaire.

Il existe plusieurs environnements de programmation, certains plus simples et certains plus complexes, qui contiennent des fonctionnalités permettant de se simplifier la vie ou t'automatiser certaines tâches. Pour notre cours, nous recommandons PyCharm, dont la version Community est un logiciel libre.

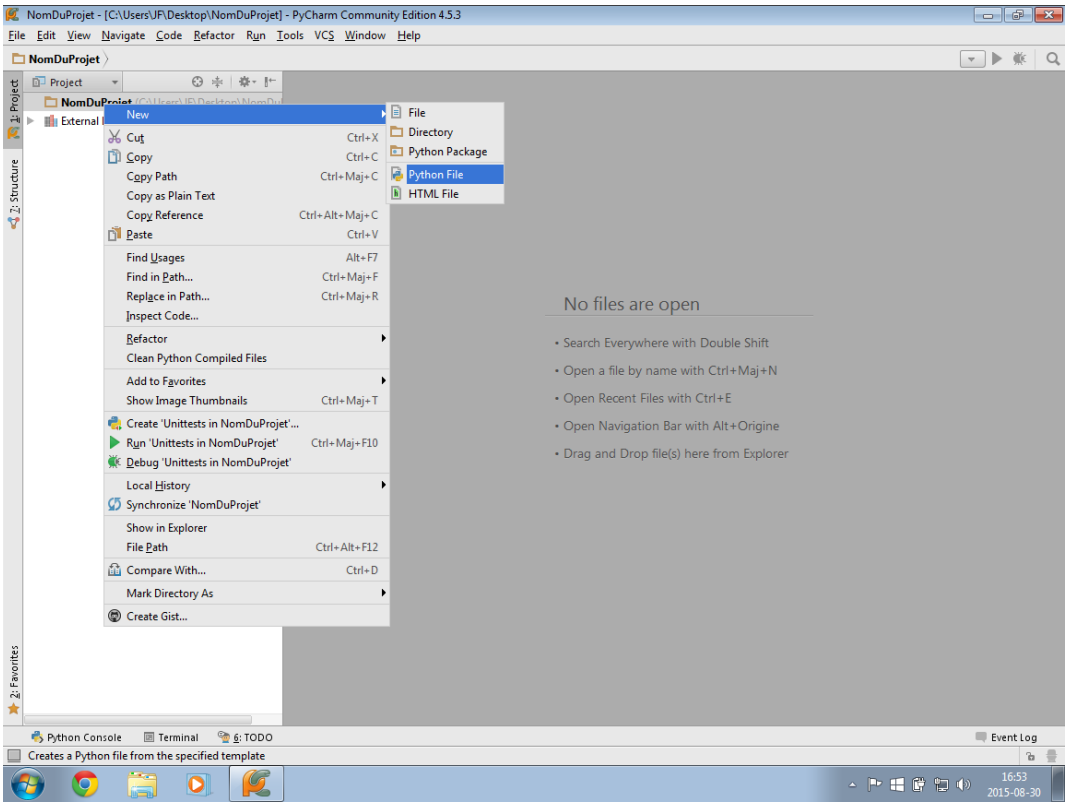
Démarrez Pycharm, puis choisissez de créer un nouveau projet.



Choisissez un emplacement pour votre projet, puis donnez lui un nom. Prenez note de l'endroit où vous avez sauvegardé votre projet : c'est à cet endroit que seront créés les fichiers de code Python `.py`. Vérifiez que l'interpréteur choisi est bel et bien Python 3, et non Python 2.



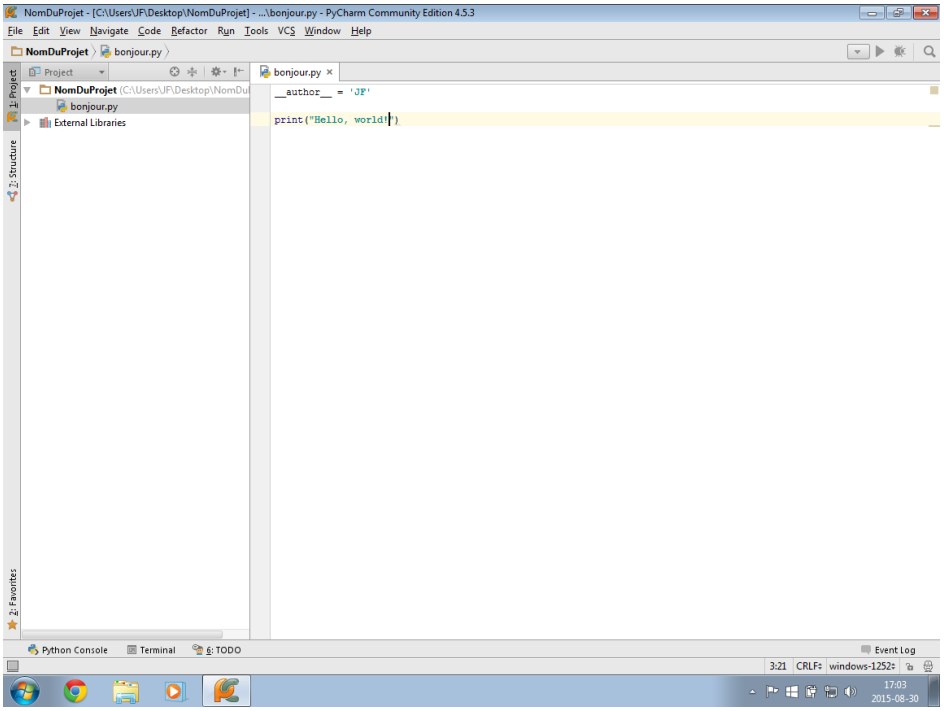
PyCharm ouvrira maintenant votre nouveau projet, qui ne contient encore aucun fichier. Pour créer un nouveau fichier de code Python, faites un clic droit (avec le bouton de droite de la souris) sur le répertoire racine de votre projet, puis choisissez New, puis Python file. Choisissez un nom pour votre fichier, par exemple `bonjour`. Remarquez que le nouveau fichier apparaît à gauche dans la liste des fichiers contenus dans votre projet, et que celui-ci possède l'extension de fichier `.py`. L'extension `.py` indique qu'il s'agit d'un fichier contenant un programme Python.



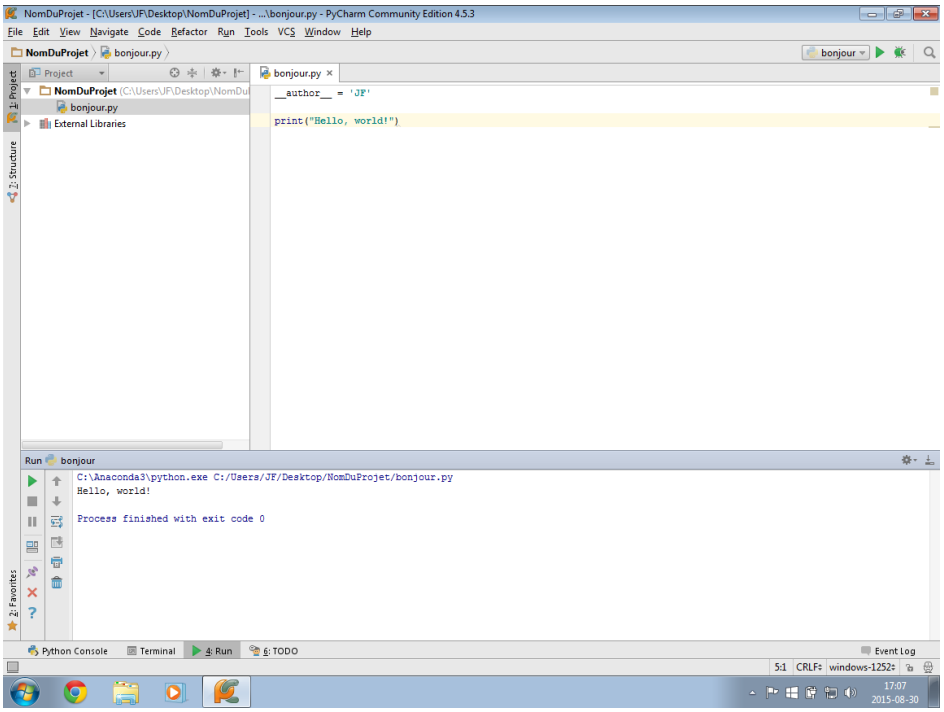
Il y a une tradition en informatique. Le premier programme que vous exécuteriez dans un nouveau langage est le programme « Hello World ». Ce programme ne fait rien d'autre qu'afficher les mots « Hello World ». C'est une tradition du fait que nous nous concentrons uniquement dans la mécanique d'écriture d'un premier programme. Vous trouverez ici : http://en.wikibooks.org/wiki/List_of_hello_world_programs, plus de 200 exemples de tels programmes.

Tapez dans votre nouveau fichier la commande suivante :

```
print("Hello, world!")
```



Vous pouvez exécuter votre programme, sélectionnez dans le menu Run, puis Run..., et choisissez votre programme dans la liste[2]. Le résultat de l'exécution apparaîtra dans un terminal dans le bas de la fenêtre, tel que montré ci-après.

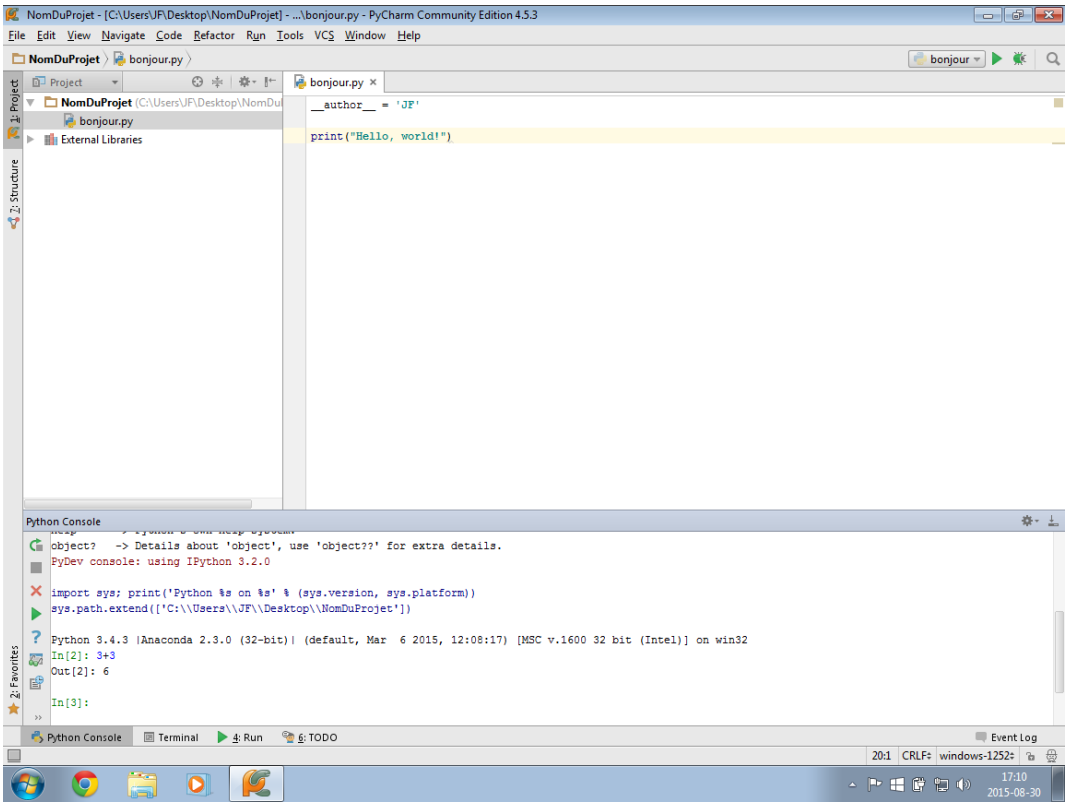


[2]Vous pouvez également utiliser le raccourci clavier indiqué dans le menu, c'est à dire les touches `alt+maj+F10`. Il est très pratique lorsqu'on utilise souvent une option d'un logiciel d'apprendre le raccourci-clavier associé.

Félicitations, vous venez d'écrire et d'exécuter votre premier programme en Python !

Si vous voulez éditer ou exécuter un programme Python déjà existant, démarrez PyCharm et ouvrez le projet (dossier) dans lequel ce programme a été sauvegardé. Vous pourrez ensuite choisir le fichier à exécuter dans la liste à gauche, puis utiliser le menu « Run » ou les raccourcis-clavier associés.

Notez finalement que PyCharm vous permet d'exécuter des commandes Python dans son interpréteur de commandes intégré. Pour l'utiliser, cliquez sur le bouton « Python Console » dans le bas à gauche de la fenêtre, ou via le menu « Tools » puis « Python Console ».



Jusqu'à présent, c'est assez simple, non ? Mais il y a beaucoup d'autres choses que vous pouvez faire dans l'environnement Python. Si vous êtes bloqués, que vous ne vous rappelez pas une commande ou les paramètres d'une fonction, vous pouvez obtenir de l'aide de manière interactive. Tapez simplement `help` puis Entrée.

4 Obtenir de l'aide

Si vous avez besoin rapidement d'information sur n'importe quelle fonction ou instruction dans Python, alors vous pouvez utiliser la fonctionnalité help. Cela est très utile particulièrement quand on utilise le prompt de l'interpréteur. Par exemple, tapez `help(print)` - cela affiche l'aide pour la fonction `print()` qui est utilisée pour afficher des choses à l'écran.

Note : Tapez q pour sortir de l'aide.

De la même manière, vous pouvez obtenir de l'information sur pratiquement n'importe quoi dans Python. Utilisez `help()` pour appendre plus de choses sur help lui-même !

Pour avoir de l'aide sur les opérateurs comme ==, !=, etc., il faut mettre des guillemets à l'intérieur (par exemple `help('==')`) pour que Python comprenne ce que l'on veut faire.

Le help() de l'interpréteur Python

```
>>> help
Type help() for interactive help, or help(object) for help about object.
```

L'aide a deux modes de fonctionnement. On peut obtenir de l'aide sur un objet, ce qui affiche sa documentation et renvoi au prompt de l'environnement Python. On peut également entrer dans le mode interactif de l'aide, qui vous permet d'interroger continuellement l'aide sur des mots-clés. Pour entrer dans le mode interactif de l'aide, tapez `help()` et appuyez sur Entrée.

```
>>> help()

Welcome to Python 3.4!  This is the interactive help utility.

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/3.4/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help>
```

Notez que l'invite n'est plus >>> mais `help>`, ce qui vous rappelle que vous êtes dans l'aide interactive. Vous pouvez maintenant entrer n'importe quel mot-clé, commande, nom de module, nom de fonction... à peu près n'importe quel terme que Python puisse identifier, pour obtenir de la documentation le concernant.

```
help> print
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
```

```
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.

help> quit

You are now leaving help and returning to the Python interpreter.
If you want to ask for help on a particular object directly from the
interpreter, you can type "help(object)".  Executing "help('string')"
has the same effect as typing a particular string at the help> prompt.
>>>
```

Pour obtenir de la documentation sur la fonction `print()`, tapez simplement `print` puis Entrée. L'aide interactive affichera le nom de la fonction, une brève description, les paramètres de la fonction et leurs valeurs par défaut, etc. Si la documentation vous semble obscure, ne vous inquiétez pas, nous étudierons ces concepts dans les chapitres suivants.

Bien entendu, l'aide interactive ne sait pas tout. Si vous entrez autre chose qu'une commande, un module, une fonction ou un autre mot réservé de Python, l'aide interactive haussera simplement ses épaules virtuelles.

Pour sortir de l'aide interactive, `quit` puis Entrée. L'invite redevient >>> pour indiquer que vous avez quitté l'aide interactive et que vous êtes revenu dans l'environnement Python.

Accès à la documentation en ligne de Python

La documentation en ligne de Python est très complète. Vous pouvez lui accéder en vous rendant sur https://docs.python.org/3/ dans votre navigateur Web.

Comme vous pouvez le constater, il y a beaucoup d'aide de disponible ! Notez que c'est un tutoriel qui peut vous aider à démarrer avec Python. C'est également une référence sur le langage ainsi que sur les modules que nous pouvons importer (nous verrons cela dans le module 3).

Le site officiel de Python, www.python.org, fournit également beaucoup d'aide et inclus des forums incontournables.

La solution finale consistant, bien évidemment, à vous référer à Google en associant le nom de la fonction que vous recherchez aux mots clés « python » et « example » pour avoir des exemples d'utilisation de ladite fonction en python (en anglais car c'est dans cette langue que sont la majorité des bons exemples).

Vous pouvez également utiliser le forum disponible sur le site Web du cours, mais n'oubliez pas : ne donnez jamais de code source relatif à un travail pratique, car ce serait du plagiat. Tentez d'isoler votre question du contexte du travail pratique. Cet exercice vous permettra peut-être même de trouver votre réponse par vous-même, car vous aurez dû réfléchir davantage afin de bien formuler la question !