Chapitre 1

Qu'est-ce que Python ?

Difficulté :

Vous avez décidé d'apprendre le Python et je ne peux que vous en féliciter. J'essayerai d'anticiper vos questions et de ne laisser personne en arrière.

Dans ce chapitre, je vais d'abord vous expliquer ce qu'est un langage de programmation. Nous verrons ensuite brièvement l'histoire de Python, afin que vous sachiez au moins d'où vient ce langage ! Ce chapitre est théorique mais je vous conseille vivement de le lire quand même.

La dernière section portera sur l'installation de Python, une étape essentielle pour continuer ce tutoriel. Que vous travailliez avec Windows, Linux ou Mac OS X, vous y trouverez des explications précises sur l'installation.

Allez, on attaque !

Un langage de programmation ? Qu'est-ce que c'est ?

La communication humaine

Non, ceci n'est pas une explication biologique ou philosophique, ne partez pas ! Très simplement, si vous arrivez à comprendre ces suites de symboles étranges et déconcertants que sont les lettres de l'alphabet, c'est parce que nous respectons certaines conventions, dans le langage et dans l'écriture. En français, il y a des règles de grammaire et d'orthographe, je ne vous apprends rien. Vous communiquez en connaissant plus ou moins consciemment ces règles et en les appliquant plus ou moins bien, selon les cas. Cependant, ces règles peuvent être aisément contournées : personne ne peut prétendre connaître l'ensemble des règles de la grammaire et de l'orthographe françaises, et peu de gens s'en soucient. Après tout, même si vous faites des fautes, les personnes avec qui vous communiquez pourront facilement vous comprendre. Quand on communique avec un ordinateur, cependant, c'est très différent.

Mon ordinateur communique aussi !

Eh oui, votre ordinateur communique sans cesse avec vous et vous communiquez sans cesse avec lui. D'accord, il vous dit très rarement qu'il a faim, que l'été s'annonce caniculaire et que le dernier disque de ce groupe très connu était à pleurer. Il n'y a rien de magique si, quand vous cliquez sur la petite croix en haut à droite de l'application en cours, celle-ci comprend qu'elle doit se fermer.

Le langage machine

En fait, votre ordinateur se fonde aussi sur un langage pour communiquer avec vous ou avec lui-même. Les opérations qu'un ordinateur peut effectuer à la base sont des plus classiques et constituées de l'addition de deux nombres, leur soustraction, leur multiplication, leur division, entière ou non. Et pourtant, ces cinq opérations suffisent amplement à faire fonctionner les logiciels de simulation les plus complexes ou les jeux super-réalistes. Tous ces logiciels fonctionnent en gros de la même façon :

- une suite d'instructions écrites en langage machine compose le programme ;
- lors de l'exécution du programme, ces instructions décrivent à l'ordinateur ce qu'il faut faire (l'ordinateur ne peut pas le deviner).

Une liste d'instructions ? Qu'est-ce que c'est encore que cela ?

En schématisant volontairement, une instruction pourrait demander au programme de se fermer si vous cliquez sur la croix en haut à droite de votre écran, ou de rester en tâche de fond si tel est son bon plaisir. Toutefois, en langage machine, une telle action demande à elle seule un nombre assez important d'instructions. Mais bon, vous pouvez

vous en douter, parler avec l'ordinateur en langage machine, qui ne comprend que le binaire, ce n'est ni très enrichissant, ni très pratique, et en tous cas pas très marrant. On a donc inventé des langages de programmation pour faciliter la communication avec l'ordinateur.

Le langage binaire est uniquement constitué de 0 et de 1. « 01000010011011110110111001101010011011110111010101110010 », par exemple, signifie « Bonjour ». Bref, autant vous dire que discuter en binaire avec un ordinateur peut être long (surtout pour vous).

Les langages de programmation

Les langages de programmation sont des langages bien plus faciles à comprendre pour nous, pauvres êtres humains que nous sommes. Le mécanisme reste le même, mais le langage est bien plus compréhensible. Au lieu d'écrire les instructions dans une suite assez peu intelligible de 0 et de 1, les ordres donnés à l'ordinateur sont écrits dans un « langage », souvent en anglais, avec une syntaxe particulière qu'il est nécessaire de respecter. Mais avant que l'ordinateur puisse comprendre ce langage, celui-ci doit être traduit en langage machine (figure 1.1).

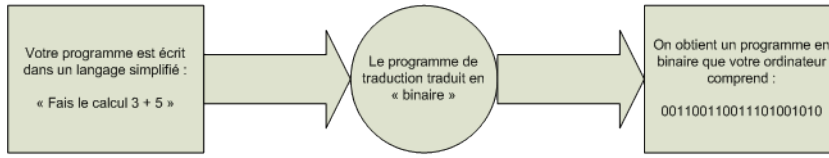

FIGURE 1.1 – Traduction d'un programme en langage binaire

En gros, le programmeur « n'a qu'à » écrire des **lignes de code** dans le langage qu'il a choisi, les étapes suivantes sont automatisées pour permettre à l'ordinateur de les décoder.

Il existe un grand nombre de langages de programmation et Python en fait partie. Il n'est pas nécessaire pour le moment de donner plus d'explications sur ces mécanismes très schématisés. Si vous n'avez pas réussi à comprendre les mots de vocabulaire et l'ensemble de ces explications, cela ne vous pénalisera pas pour la suite. Mais je trouvais intéressant de donner ces précisions quant aux façons de communiquer avec son ordinateur.

Pour la petite histoire

Python est un langage de programmation, dont la première version est sortie en 1991. Créé par **Guido van Rossum**, il a voyagé du Macintosh de son créateur, qui travaillait

à cette époque au *Centrum voor Wiskunde en Informatica* aux Pays-Bas, jusqu'à se voir associer une organisation à but non lucratif particulièrement dévouée, la **Python Software Foundation**, créée en 2001. Ce langage a été baptisé ainsi en hommage à la troupe de comiques les « Monty Python ».

À quoi peut servir Python ?

Python est un langage puissant, à la fois facile à apprendre et riche en possibilités. Dès l'instant où vous l'installez sur votre ordinateur, vous disposez de nombreuses fonctionnalités intégrées au langage que nous allons découvrir tout au long de ce livre.

Il est, en outre, très facile d'étendre les fonctionnalités existantes, comme nous allons le voir. Ainsi, il existe ce qu'on appelle des **bibliothèques** qui aident le développeur à travailler sur des projets particuliers. Plusieurs bibliothèques peuvent ainsi être installées pour, par exemple, développer des interfaces graphiques en Python.

Concrètement, voilà ce qu'on peut faire avec Python :

- de petits programmes très simples, appelés **scripts**, chargés d'une mission très précise sur votre ordinateur ;
- des programmes complets, comme des jeux, des suites bureautiques, des logiciels multimédias, des clients de messagerie...
- des projets très complexes, comme des progiciels (ensemble de plusieurs logiciels pouvant fonctionner ensemble, principalement utilisés dans le monde professionnel).

Voici quelques-unes des fonctionnalités offertes par Python et ses bibliothèques :

- créer des interfaces graphiques ;
- faire circuler des informations au travers d'un réseau ;
- dialoguer d'une façon avancée avec votre système d'exploitation ;
- ... et j'en passe...

Bien entendu, vous n'allez pas apprendre à faire tout cela en quelques minutes. Mais ce cours vous donnera des bases suffisamment larges pour développer des projets qui pourront devenir, par la suite, assez importants.

Un langage de programmation interprété

Eh oui, vous allez devoir patienter encore un peu car il me reste deux ou trois choses à vous expliquer, et je suis persuadé qu'il est important de connaître un minimum ces détails qui peuvent sembler peu pratiques de prime abord. Python est un langage de programmation **interprété**, c'est-à-dire que les instructions que vous lui envoyez sont « transcrites » en langage machine au fur et à mesure de leur lecture. D'autres langages (comme le C / C++) sont appelés « langages **compilés** » car, avant de pouvoir les exécuter, un logiciel spécialisé se charge de transformer le code du programme en langage machine. On appelle cette étape la « **compilation** ». À chaque modification du code, il faut rappeler une étape de compilation.

Les avantages d'un langage interprété sont la simplicité (on ne passe pas par une étape

de compilation avant d'exécuter son programme) et la portabilité (un langage tel que Python est censé fonctionner aussi bien sous Windows que sous Linux ou Mac OS, et on ne devrait avoir à effectuer aucun changement dans le code pour le passer d'un système à l'autre). Cela ne veut pas dire que les langages compilés ne sont pas portables, loin de là ! Mais on doit utiliser des compilateurs différents et, d'un système à l'autre, certaines instructions ne sont pas compatibles, voire se comportent différemment.

En contrepartie, un langage compilé se révélera bien plus rapide qu'un langage interprété (la traduction à la volée de votre programme ralentit l'exécution), bien que cette différence tende à se faire de moins en moins sentir au fil des améliorations. De plus, il faudra installer Python sur le système d'exploitation que vous utilisez pour que l'ordinateur puisse comprendre votre code.

Différentes versions de Python

Lors de la création de la Python Software Foundation, en 2001, et durant les années qui ont suivi, le langage Python est passé par une suite de versions que l'on a englobées dans l'appellation Python 2.x (2.3, 2.5, 2.6...). Depuis le 13 février 2009, la version 3.0.1 est disponible. Cette version casse la **compatibilité ascendante** qui prévalait lors des dernières versions.

Compatibilité quoi ?

Quand un langage de programmation est mis à jour, les développeurs se gardent bien de supprimer ou de trop modifier d'anciennes fonctionnalités. L'intérêt est qu'un programme qui fonctionne sous une certaine version marchera toujours avec la nouvelle version en date. Cependant, la Python Software Foundation, observant un bon nombre de fonctionnalités obsolètes, mises en œuvre plusieurs fois... a décidé de nettoyer tout le projet. Un programme qui tourne à la perfection sous Python 2.x devra donc être mis à jour un minimum pour fonctionner de nouveau sous Python 3. C'est pourquoi je vais vous conseiller ultérieurement de télécharger et d'installer la dernière version en date de Python. Je m'attarderai en effet sur les fonctionnalités de Python 3 et certaines d'entre elles ne seront pas accessibles (ou pas sous le même nom) dans les anciennes versions.

Ceci étant posé, tous à l'installation !

Installer Python

L'installation de Python est un jeu d'enfant, aussi bien sous Windows que sous les systèmes Unix. Quel que soit votre système d'exploitation, vous devez vous rendre sur le site officiel de Python. Pour cela, utilisez le code web suivant :

▷ Site officiel de Python
Code web : 199501

Sous Windows

1. Cliquez sur le lien `Download` dans le menu principal de la page.
2. Sélectionnez la version de Python que vous souhaitez utiliser (je vous conseille la dernière en date).
3. On vous propose un (ou plusieurs) lien(s) vers une version Windows : sélectionnez celle qui conviendra à votre processeur. Si vous avez un doute, téléchargez une version « x86 ». Si votre ordinateur vous signale qu'il ne peut exécuter le programme, essayez une autre version de Python.
4. Enregistrez puis exécutez le fichier d'installation et suivez les étapes. Ce n'est ni très long ni très difficile.
5. Une fois l'installation terminée, vous pouvez vous rendre dans le menu `Démarrer` > `Tous les programmes`. Python devrait apparaître dans cette liste (figure 1.2). Nous verrons bientôt comment le lancer, pas d'impatience...

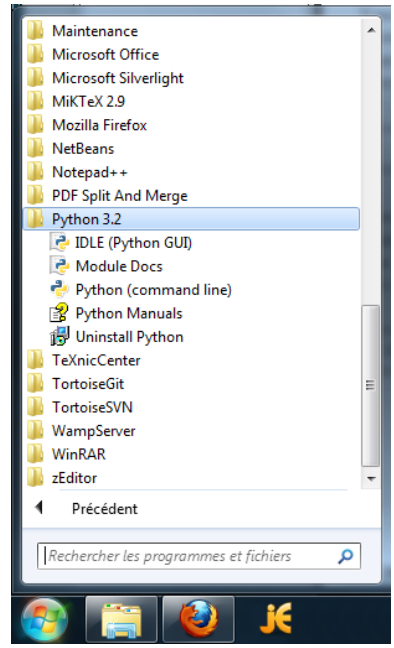

FIGURE 1.2 – Python est installé sous Windows

Sous Linux

Python est pré-installé sur la plupart des distributions Linux. Cependant, il est possible que vous n'ayez pas la dernière version en date. Pour le vérifier, tapez dans un terminal la commande `python -V`. Cette commande vous renvoie la version de Python actuellement installée sur votre système. Il est très probable que ce soit une version **2.x**, comme 2.6 ou 2.7, pour des raisons de compatibilité. Dans tous les cas, je vous conseille d'installer Python 3.x, la syntaxe est très proche de Python 2.x mais diffère quand même...

Cliquez sur `download` et téléchargez la dernière version de Python (actuellement « Python 3.2 compressed source tarball (for Linux, Unix or OS X) »). Ouvrez un terminal, puis rendez-vous dans le dossier où se trouve l'archive :

1. Décompressez l'archive en tapant : `tar -jxvf Python-3.2.tar.bz2` (cette commande est bien entendu à adapter suivant la version et le type de compression).
2. Attendez quelques instants que la décompression se termine, puis rendez-vous dans le dossier qui vient d'être créé dans le répertoire courant (`Python-3.2` dans mon cas).
3. Exécutez le script `configure` en tapant `./configure` dans la console.
4. Une fois que la configuration s'est déroulée, il n'y a plus qu'à compiler en tapant `make` puis `make altinstall`. Ces commandes ont pour but de compiler Python. La commande `make altinstall`, en particulier, crée automatiquement les liens vers la version installée. Grâce à `altinstall`, vous pouvez être sûrs que la version que vous installez n'entrera pas en conflit avec une autre déjà présente sur votre système.

Sous Mac OS X

Téléchargez la dernière version de Python. Ouvrez le fichier `.dmg` et faites un double-clic sur le paquet d'installation `Python.mpkg`

Un assistant d'installation s'ouvre, laissez-vous guider : Python est maintenant installé !

Lancer Python

Ouf ! Voilà qui est fait !

Bon, en théorie, on commence à utiliser Python dès le prochain chapitre mais, pour que vous soyez un peu récompensés de votre installation exemplaire, voici les différents moyens d'accéder à la ligne de commande Python que nous allons tout particulièrement étudier dans les prochains chapitres.

Sous Windows

Vous avez plusieurs façons d'accéder à la ligne de commande Python, la plus évidente consistant à passer par les menus `Démarrer` > `Tous les programmes` > `Python 3.2`

> `Python (Command Line)`. Si tout se passe bien, vous devriez obtenir une magnifique console (figure 1.3). Il se peut que les informations affichées dans la vôtre ne soient pas les mêmes, mais ne vous en inquiétez pas.

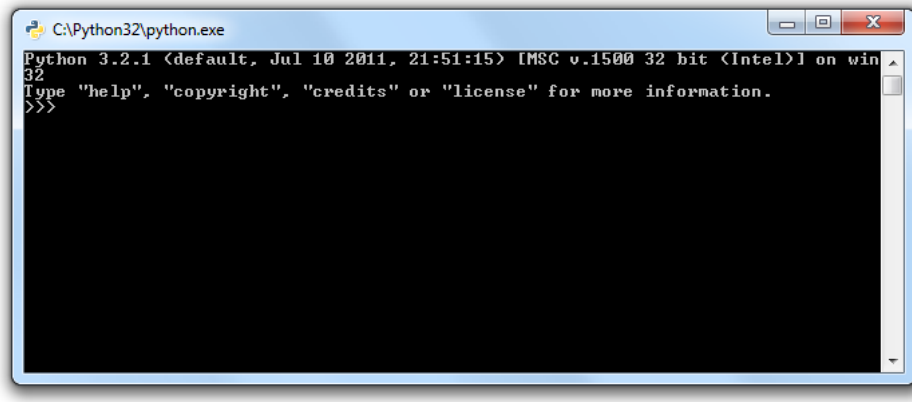

FIGURE 1.3 – La console Python sous Windows

Qu'est-ce que c'est que cela ?

On verra plus tard. L'important, c'est que vous ayez réussi à ouvrir la console d'interprétation de Python, le reste attendra le prochain chapitre.

Vous pouvez également passer par la ligne de commande Windows ; à cause des raccourcis, je privilégie en général cette méthode, mais c'est une question de goût. Allez dans le menu `Démarrer`, puis cliquez sur `Exécuter`. Dans la fenêtre qui s'affiche, tapez simplement « python » et la ligne de commande Python devrait s'afficher à nouveau. Sachez que vous pouvez directement vous rendre dans `Exécuter` en tapant le raccourci `Windows` + `R`.

Pour fermer l'interpréteur de commandes Python, vous pouvez tapez « exit() » puis appuyer sur la touche `Entrée`.

Sous Linux

Lorsque vous l'avez installé sur votre système, Python a créé un lien vers l'interpréteur sous la forme **python3.X** (le X étant le numéro de la version installée).

Si, par exemple, vous avez installé Python 3.2, vous pouvez y accéder grâce à la commande :

```
1 $ python3.2
2 Python 3.2 (r32:88445, Mar  4 2011, 22:07:21)
```

```
3 [GCC 4.3.3] on linux2
4 Type "help", "copyright", "credits" or "license" for more
      information.
5 >>>
```

Pour fermer la ligne de commande Python, n'utilisez pas CTRL + C mais CTRL + D (nous verrons plus tard pourquoi).

Sous Mac OS X

Cherchez un dossier `Python` dans le dossier `Applications`. Pour lancer Python, ouvrez l'application `IDLE` de ce dossier. Vous êtes prêts à passer au concret !

En résumé

- Python est un langage de programmation interprété, à ne pas confondre avec un langage compilé.
- Il permet de créer toutes sortes de programmes, comme des jeux, des logiciels, des progiciels, etc.
- Il est possible d'associer des **bibliothèques** à Python afin d'étendre ses possibilités.
- Il est portable, c'est à dire qu'il peut fonctionner sous différents systèmes d'exploitation (Windows, Linux, Mac OS X,...).

Chapitre 2

Premiers pas avec l'interpréteur de commandes Python

Difficulté :

Après les premières notions théoriques et l'installation de Python, il est temps de découvrir un peu l'interpréteur de commandes de ce langage. Même si ces petits tests vous semblent anodins, vous découvrirez dans ce chapitre les premiers rudiments de la syntaxe du langage et je vous conseille fortement de me suivre pas à pas, surtout si vous êtes face à votre premier langage de programmation.

Comme tout langage de programmation, Python a une syntaxe claire : on ne peut pas lui envoyer n'importe quelle information dans n'importe quel ordre. Nous allons voir ici ce que Python mange... et ce qu'il ne mange pas.

Où est-ce qu'on est, là ?

Pour commencer, je vais vous demander de retourner dans l'interpréteur de commandes Python (je vous ai montré, à la fin du chapitre précédent, comment y accéder en fonction de votre système d'exploitation).

Je vous rappelle les informations qui figurent dans cette fenêtre, même si elles peuvent être différentes chez vous en fonction de votre version et de votre système d'exploitation.

```
1 Python 3.2.1 (default, Jul 10 2011, 21:51:15) [MSC v.1500 32
      bit (Intel)] on win 32
2 Type "help", "copyright", "credits" or "license" for more
      information.
3 >>>
```

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

Attends, attends. C'est quoi cet interpréteur ?

Souvenez-vous, au chapitre précédent, je vous ai donné une brève explication sur la différence entre langages compilés et langages interprétés. Eh bien, cet interpréteur de commandes va nous permettre de tester directement du code. Je saisis une ligne d'instructions, j'appuie sur la touche Entrée de mon clavier, je regarde ce que me répond Python (s'il me dit quelque chose), puis j'en saisis une deuxième, une troisième... Cet interpréteur est particulièrement utile pour comprendre les bases de Python et réaliser nos premiers petits programmes. Le principal inconvénient, c'est que le code que vous saisissez n'est pas sauvegardé (sauf si vous l'enregistrez manuellement, mais chaque chose en son temps).

Dans la fenêtre que vous avez sous les yeux, l'information qui ne change pas d'un système d'exploitation à l'autre est la série de trois chevrons qui se trouve en bas à gauche des informations : >>>. Ces trois signes signifient : « je suis prêt à recevoir tes instructions ».

Comme je l'ai dit, les langages de programmation respectent une syntaxe claire. Vous ne pouvez pas espérer que l'ordinateur comprenne si, dans cette fenêtre, vous commencez par lui demander : « 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 cela pour dire que, si vous saisissez n'importe quoi dans cette fenêtre, la probabilité est grande que Python vous indique, clairement et fermement, qu'il n'a rien compris.

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

```
1 >>> premier test avec Python
2 File "<stdin>", line 1
```

```
premier test avec Python
^
SyntaxError: invalid syntax
>>>
```

Eh oui, l'interpréteur parle en anglais et les instructions que vous saisirez, comme pour l'écrasante majorité des langages de programmation, seront également en anglais. Mais pour l'instant, rien de bien compliqué : l'interpréteur vous indique qu'il a trouvé un problème dans votre ligne d'instruction. Il vous indique le numéro de la ligne (en l'occurence la première), qu'il vous répète obligeamment (ceci est très utile quand on travaille sur un programme de plusieurs centaines de lignes). Puis il vous dit ce qui l'arrête, ici : `SyntaxError: invalid syntax`. Limpide n'est-ce pas ? Ce que vous avez saisi est incompréhensible pour Python. Enfin, la preuve qu'il n'est pas rancunier, c'est qu'il vous affiche à nouveau une série de trois chevrons, montrant bien qu'il est prêt à retenter l'aventure.

Bon, c'est bien joli de recevoir un message d'erreur au premier test mais je me doute que vous aimeriez bien voir des trucs qui fonctionnent, maintenant. C'est parti donc.

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

C'est assez trivial, quand on y pense, mais je trouve qu'il s'agit d'une excellente manière d'aborder pas à pas la syntaxe de Python. Nous allons donc essayer d'obtenir les résultats de calculs plus ou moins compliqués. Je vous rappelle encore une fois qu'exécuter les tests en même temps que moi sur votre machine est une très bonne façon de vous rendre compte de la syntaxe et surtout, de la retenir.

Saisir un nombre

Vous avez pu voir sur notre premier (et à ce jour notre dernier) test que Python n'aimait pas particulièrement les suites de lettres qu'il ne comprend pas. Par contre, 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 saisit un nombre et l'interpréteur le renvoie. Mais dans bien des cas, ce simple retour indique que l'interpréteur a bien compris et que votre saisie est en accord avec sa syntaxe. De même, vous pouvez saisir des nombres à virgule.

```
>>> 9.5
```

```
2 9.5
3 >>>
```

Attention : on utilise ici la notation anglo-saxonne, c'est-à-dire que le point remplace la virgule. La virgule a un tout autre sens pour Python, prenez donc cette habitude dès maintenant.

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

Opérations courantes

Bon, il est temps d'apprendre à utiliser les principaux opérateurs de Python, qui vont vous servir pour la grande majorité de vos programmes.

Addition, soustraction, multiplication, division

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

```
1 >>> 3 + 4
2 7
3 >>> -2 + 93
4 91
5 >>> 9.5 + 2
6 11.5
7 >>> 3.11 + 2.08
8 5.1899999999999995
9 >>>
```

Pourquoi ce dernier 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. C'est pourquoi, en programmation, on préfère travailler autant que possible avec des nombres entiers. Cependant, vous remarquerez que l'erreur est infime et qu'elle n'aura pas de réel impact sur les calculs. Les applications qui ont besoin d'une précision mathématique à toute épreuve essayent de pallier ces défauts par d'autres moyens mais ici, ce ne sera pas nécessaire.

Faites également des tests pour la soustraction, la multiplication et la division : il n'y a rien de difficile.

Division entière et modulo

Si vous avez pris le temps de tester la division, vous vous êtes rendu compte que le résultat est donné avec une virgule flottante.

```
>>> 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 utilise le symbole « // ». Il permet d'obtenir la partie entière d'une division.

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

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

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

Ces notions de *partie entière* et de *reste de division* ne sont pas bien difficiles à comprendre et vous serviront très probablement 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'occurence, 3).
- Pour obtenir le modulo d'une division, on « récupère » son reste. Dans notre exemple, $10/3 = 3$ et il reste 1. Une fois que l'on a compris cela, ce n'est pas bien compliqué.

Souvenez-vous bien de ces deux opérateurs, et surtout du modulo « % », dont vous aurez besoin dans vos programmes futurs.

En résumé

- L'interpréteur de commandes Python permet de tester du code au fur et à mesure qu'on l'écrit.
- L'interpréteur Python accepte des nombres et est capable d'effectuer des calculs.
- Un nombre décimal s'écrit avec un point et non une virgule.
- Les calculs impliquant des nombres décimaux donnent parfois des résultats approximatifs, c'est pourquoi on préfèrera, dans la mesure du possible, travailler avec des nombres entiers.