Initiation à Python - leçon 3.1.2

s1
Dans cette séquence, nous allons découvrir les types de base.

---
s2

Dans Python 2.7, il y a plusieurs types de base : un type pour les booléens, deux types pour les entiers, un type pour les nombres flottants, un type pour les complexes, un dernier type pour les chaînes de caractères que nous avons déjà découvertes précédemment.

Le type d'un objet peut être obtenu grâce à la fonction type.

Avant d'entrer dans le détail, examinons ces exemples que je vous propose de reproduire.

Une variable booléenne prend les valeurs False ou True. Dans notre exemple, on déclare une variable a qui vaut l'expression 1==2. Dans cette expression l'opérateur == signifie que l'on teste si l'égalité entre 1 et 2 est vraie ou fausse. Ici, bien entendu, la valeur de cette expression est False. Le type de la variable a est booléen.

L'expression 1+2+3 appartient à la classe des entiers.
L'expression 1+2+3. appartient à la classe des flottants à cause du point signifiant que l'on a affaire à un nombre décimal.
L'expression 2**100 - qui signifie que l'on élève 2 à la puissance 100 - est un entier long

Voici comment définir un nombre complexe.

Revoici enfin un type chaîne de caractères.

Nous découvrirons plus loin les types composés permettant de définir des objets capables de contenir des ensembles de variables de même type ou de types différents.
---
s3

Commençons par les booléens.

Voici la liste des opérations possibles sur les booléens et leur résultat.

Dans Python comme dans beaucoup de langages, on peut être amené à comparer deux valeurs. Le résultat de cette comparaison est un booléen.
Voici les opérateurs qui permettent de comparer des booléens en Python. On notera la différence entre le test d'égalité (deux signes =) et l'instruction d'affectation d'un objet à une variable (un simple signe =).
On notera que l'on peut enchaîner les comparaisons. Dans l'exemple présenté, on compare d'abord 1 par rapport à 4 et 4 par rapport à 2.
On notera également que l'on peut comparer des chaînes de caractères. La comparaison se fait selon l'ordre lexicographique.

---
s4

Un mot sur les entiers. Dans Python 2.7, il y a donc deux types d'entiers les int et les long. Pour l'utilisation quotidienne, on ne se préoccupe pas de savoir si l'on travaille avec un type ou l'autre. Les int sont des entiers codés sur un nombre fixé de bits. Les entiers longs permettent de dépasser cette limite et de coder les entiers avec un nombre de bits arbitraire.

Voici la liste des opérations possibles sur les entiers et le type du résultat.

La somme de deux entiers, leur différence, leur produit.

Attention, avec Python 2.7 la division entre deux entiers produit un entier, comme dans l'exemple présenté.

L'opération modulo renvoie le reste de la division entière ; l'opération double division qui renvoie la division entière.

Un mot sur la fonction cmp (fonction qui n'est pas spécifique aux entiers) qui permet de comparer deux objets et qui ici dans le cas de deux entiers renvoie -1

Je vous propose de reproduire les exemples présentés.

---
s5

Un mot sur les flottants. Voici la liste des opérations possibles sur les flottants et le type du résultat.
Un mot sur la fonction round qui renvoie l'entier le plus proche.
Pour pouvoir utiliser les fonctions floor et ceil qui renvoie respectivement l'entier le plus grand directement inférieur et l'entier le plus petit directement supérieur, il faut importer le module

math grâce à l'instruction import. Math est l'une de ces bibliothèques spécialisées dont nous avons parlé au tout début de ce module et qui étendent Python avec de nouvelles fonctions. Nous préciserons plus loin ce concept, mais nous pouvons voir tout de suite que math est un objet Python et que floor et ceil en sont deux des méthodes.

Je vous propose de reproduire les exemples présentés.

---
s6

Un mot sur les nombres complexes. Les nombres complexes sont construits à partir d'entiers ou de flottants. On notera que le nombre complexe traditionnellement noté i est noté j ou J en Python, et que cette lettre j (ou J) doit être obligatoirement être utilisée comme suffixe d'une valeur (de type 'int' ou 'float') afin d'être reconnue sans ambiguïté.
Ainsi, dans l'exemple présenté, a est bien un nombre complexe.

Un nombre complexe est un objet Python : on affiche ici ses attributs partie entière et partie imaginaire. On peut également lui appliquer un certain nombre de méthodes, par exemple la méthode conjugate. Dans cet exemple, on déclare une nouvelle variable b qui référence le complexe conjugué de a.

Dans ce nouvel exemple 4+3j est toujours un nombre complexe, même si à la ligne précédente on a défini préalablement une variable j entière. Mais, attention à la syntaxe : la variable a = 4+3*j n'est pas un nombre complexe.

Je vous propose de reproduire les exemples présentés.

---
s7

Enfin, un mot sur les chaînes de caractères. Les chaînes de caractères sont définies entre apostrophes ou entre guillemets.
Voici la liste des opérations possibles sur les chaînes de caractères et le type du résultat.
L'opérateur + signifie que l'on concatène deux chaînes, comme dans l'exemple présenté.

Attention, dans une chaîne de caractères, Python code chaque caractère sur un seul octet ce qui n'offre que 256 possibilités, ce qui est insuffisant pour décrire les dizaines de milliers de glyphes qui existent de par le monde. Pour coder les caractères et symboles des différents alphabets existants, il a fallu mettre au point une norme mondiale, dite "Unicode" et définir différents jeux de caractères ("charset" en anglais) qui sont des tables associant à un caractère donné un symbole. Par exemple, dans le charset "latin1" le caractère à (a accent grave) a la valeur 224.

Malheureusement le jeu de caractères par défaut dans Python est souvent l'ASCII qui est une norme américaine qui a "oublié" les caractères accentués.

Ainsi, pour utiliser des caractères accentués dans Python, il faut utiliser la représentation unicode de ce caractère et le préciser à Python, en préfixant avec le caractère u une chaîne contenant un caractère accentué, comme dans l'exemple suivant.

Je vous propose de reproduire tout de suite les exemples présentés.

---
s8

Une chaîne de caractères est un objet. On dispose donc de méthodes ou fonctions spécifiques qui existent pour cet objet. Elles sont listées ci-dessous grâce à la commande dir. Nous n'allons pas les examiner toutes. Pour obtenir de l'aide sur une méthode, il faut utiliser la commande help() de la façon suivante.

Je vous propose de reproduire l'exemple présenté ici pour la méthode upper et de tester par vous-même les méthodes replace, find et capitalize.

---
s9

Un mot sur les opérateurs de comparaison
Il ne faut pas confondre les opérateurs == et is.

== teste l'égalité des valeurs de deux objets
l'opérateur is compare l'identité de deux objets, c'est-à-dire l'adresse mémoire, l'endroit où ils sont rangés en mémoire.

Dans l'exemple présenté a est un entier, b est une variable référençant le même objet que a (ainsi a et b ont la même identité).
Dans notre exemple, c référence un flottant. Les deux objets référencés par les variables a et c sont égaux au sens de leur valeur, mais pas de leur identité. Ils ont la même valeur, mais ce ne sont pas les mêmes objets.

---
s10

Dans cette leçon, nous avons découvert les types de base de Python : booléens, entiers, flottants, complexes et chaînes de caractères. Dans la prochaine leçon, nous allons découvrir des objets nouveaux qui permettent de ranger des ensembles d’objets ayant pour types ces types de base.