Python 3 : Types et Instructions de Bases

Licence 2 Génie Logiciel

Ibrahima SY

Institut Supérieur Informatique

11/18/2021

Introduction au typage dynamique

Notion d' objet

- En python tout est un objet
- Un objet est un morceau de code qui va contenir des données.
- En plus il contient un ensemble de mécanisme qui permettent de manipuler ces données qu'on appelle méthodes
- Les objets ont tous un types
- Le type est le comportement par défaut qui va être défini pour les objets

Introduction Typage Dynamiques

Notion d' objet

Pour référencer un objet, il faut affecter un objet à un nom de variable avec une notation que l'on appelle la notation d'affectation

```
moyenne = 12
```

Le nom des variables en Python peut être constitué de lettres minuscules (a à z), de lettres majuscules(A à Z), de nombres (0 à 9) du caractères souligné (_)

```
age = 18
Age = 30
_age = 40
age_1 = 30
age_1 = 32
1age = 43 # incorrect
```

Introduction au Typage Dynamique

Le typage dynamique

L' orsque vous taper $a = 3$ dans la console , Python va réaliser 3 opérations :

1. Python crée l'entier 3 dans l'espace des objets
2. Python crée une variable a dans l'espace des variables(espace de nommage)
3. Python crée une référence entre la variable a et l'entier 3

Introduction au Typage Dynamique

Le typage dynamique

- Si on effectue cette opération a = "Licence GL" python déréférence l'objet 3 pour que la variable a référence désormais l'objet a = "Licence GL"
- Le typage dynamique veut dire qu'en Python, le type n'est pas lié à la variable qui référence l'objet mais à l'objet .
- del a permet de supprimer la variable a de l'espace des variables.
- Si l'objet n'a plus de référence, un mécanisme qui s'appelle mécanisme de garbage collector libère la mémoire de l'ordinateur.

Introduction au Typage Dynamique

Le typage dynamique

Il est impossible d'utiliser les mots clés du langage (ci-dessous) pour nommer des variables.

```
False    def      if       raise
None     del      import   return
True     elif     in       try
and      else     is       while
as       except   lambda   with
assert   finally  nonlocal yield
break    for      not
class    from     or
continue global   pass
```

Type Numérique

En python il existe 4 types numériques :

int $\Longrightarrow$ entier

float $\Longrightarrow$ nombre décimaux / réels

complex $\Longrightarrow$ nombres complexes

bool $\Longrightarrow$ les booléens (qui sont un sous ensemble des int)

Type Numérique

Le type int

- On peut vérifier le type avec la fonction built-in type()

```
i = 1
i * 2 # multiplication
i + 2 # addition
i - 3 # soustraction
i = i + 5
```

- Les entiers en Python sont des objets de précision illimitée. Python n'a aucun problème à faire des opérations sur des entiers extrêmement grands.

Type Numérique

Type Float

Ce type est utilisé pour stocker des nombres à virgule flottante, désignés en anglais par l'expression floating point numbers. Pour cette raison, on appelle ce type : float. En français, on parle de flottant.

```
f = 3.6
```

Le type Booléen

Les booléens sont le résultat d'opérations logiques et ont deux valeurs possibles : True ou False Il est d'ailleurs codé comme un entier. False étant l'entier 0 et True étant l'entier 1.

```
f = True
```

Type Numérique

Le type complex

- La constante complexe que nous notons i en français, se note j en Python.
- Un nombre complexe, c'est 2 nombres float mis l'un à côté de l'autre

```
z = 3 + 4j
```

On peut additionner n'importe quel type numérique. Python se charge de faire la conversion pour nous. Lors de la conversion, on peut avoir une perte de précision.

```
3 + 3.56789388338 #(int + float renvoie float)
3 + 3.27726 + 1+4j #( int + float + complex ) renvoie un nombre complex
```

Type Numérique

On peut convertir des nombres à l'aide de fonctions built-in :

1. int() convertit en entier (l'opération s'appelle la troncation)
2. float() convertit en float
3. complex() convertit en complex
4. str() convertir en chaîne de caractère

Les Chaînes de caractères

- Pour créer une chaîne de caractère , il faut l'entourer soit par des apostrophes ‘’ , soit par des guillemets “” Les chaînes de caractères sont des objets **immuables**
- Les chaînes de caractères contiennent également de nombreuses méthodes les fonctions qui manipulent les chaînes de caractères retournent un nouvel objet chaîne de caractères.
- Pour accéder à l'intégralité des méthodes qui existent sur un objet particulier, on utilise la fonction built-in dir().

La méthode str.title()

Cette méthode met la première lettre de chaque mot en majuscule.

Les Chaînes de caractères

La méthode str.repalce()

Cette méthode permet de remplacer un mot de la chaîne

La méthode str.isdecimal()

Un certain nombre de méthodes de chaînes de caractères permettent également de faire des comparaisons ou de faire des tests. Ainsi, si on prend la chaîne '123' et que l'on souhaite la convertir en entier, on pourrait vouloir s'assurer que cette chaîne de caractères représente bien un nombre décimal.

Les Chaînes de caractères

La méthode str.repalce()

Formater une chaîne de caractères : la méthode str.format()

Cette méthode permet d'insérer des objets à l'intérieur d'emplacements spécifiés dans la chaîne de caractères par des accolades :

```
age = 12
nom = "Ansou"

"{} a {} ans cette annee ".format(nom, age)
```

```
## 'Ansou a 12 ans cette annee '
```

Les Séquences

Les séquences sont un ensemble de types regroupant les list, les tuple, les str et les bytes.

Définition de séquence

Une séquence en Python est un ensemble fini et ordonné d'éléments indicés de 0 à $n-1$ si on a n éléments.

Les Les Séquences

Opérations de base sur une séquence “chaîne de caractère“

Commençons avec une chaîne de caractère : **s = ‘egg, bacon’**

- L’opération s[0] permet d’accéder au premier élément de la séquence, c’est-à-dire **‘e’**
- Pour connaître le nombre d’éléments d’une séquence, il faut utiliser la fonction built-in len
- Toutes les séquences supportent également le test d’appartenance, qui est une opération très puissante en Python. ‘egg’ in s veut dire est-ce que **‘egg’** est dans **s**. L’opération renvoie True. De même on peut faire ‘egg’ not in s qui est le test de non-appartenance et qui retourne ici False.

Les Séquences

Opérations de base sur une séquence “chaîne de caractère“

- Ensuite on peut faire de la concaténation de séquence. s + ‘, and beans’ rajoute **‘, and beans’** à notre séquence et renvoie donc **‘egg, bacon, and, beans’** . Cette concaténation produit une nouvelle chaîne de caractères puisqu’une chaîne de caractère n’est pas **mutable**.
- L’opération index() permet de trouver la première occurrence par exemple de la lettre **g** : s.index(‘g’) renvoie **1** puisque la première occurrence de la lettre **g** c’est la deuxième lettre à la place **1**.

Les Les Séquences

Opérations de base sur une séquence “chaîne de caractère“

- L'opération count() permet de compter le nombre d'éléments, ici le nombre de **‘g’** que l'on a dans **s**
- On peut également appliquer les fonctions built-in min() et max() qui retournent le minimum et le maximum d'une séquence. min(s) renvoie **‘ ’** et max(s) renvoie **‘o’** .Comme notre séquence est une chaîne de caractère, le **min** et le **max** utilisent l'ordre lexicographique : **l'espace** pour le minimum et la lettre **O** pour le maximum.
- L'opération de **Shalow copy** permet de copier un certain nombre de fois un élément : ‘x’*30 produit 30 fois **‘x’**. Nous verrons par la suite que cette opération produit en fait une shalow copy qui peut avoir des effets de bord lorsque la séquence multipliée n'est pas un immuable mais un objet de type mutable.

Les Séquences

Opérations de base sur une séquence “chaîne de caractère“

L'opération de slicing est valable pour toutes les séquences (chaînes de caractères aussi bien que liste). Pour commencer nous créons une chaîne de caractère s = ‘egg, bacon’

```
s = "egg, bacon"
s[0:3]
```

```
## 'egg'
```

Liste

- La liste représente un type extrêmement souple et puissant. Une liste est une séquence d'objets hétérogènes.
- Elle ne stocke que des références vers les objets. Par conséquent, la taille de l'objet liste est indépendante du type d'objets qui sont référencés.
- La liste c'est un objet mutable . Cela veut dire qu'on peut le modifier là où il est stocké.
- L'avantage de cette mutabilité c'est qu'on n'a pas besoin de faire une copie de l'objet pour le modifier.

Liste

Liste et séquence, mêmes opérations

- Toutes les opérations applicables aux séquences sont applicables aux listes .
- Prenons une liste mêlant une variable **i=4** , une chaîne de caractère **'spam'**, un flottant **3.2** et un booléen **True**.

```
liste = ["spam", 1,2,3, True] # creation d' un objet liste
liste[0] # on accede au prmier elementb de la liste
liste[0] = 6   # on change la premiere valuer de cette liste
liste[0] = liste[0] + 1 # on peut directement faire des operation sur la liste
len(liste)  # le nombre d'elts dans la liste
"spam" in liste # test appartenance
```

Liste

OpérationS de slicing

- On peut également réaliser des opérations de slicing sur une liste. **liste[1:3]** prend tous les éléments de 1 inclus à 3 exclu .
- On peut aussi faire des opérations d'affectation sur des slice **liste[1:3] = [1,2,3]**
- Cette opération d'affectation sur un slice est un moyen très simple d'effacer des éléments dans une liste. Pour **liste=[16,1,2,3,True]**, si on entre **a[1:3] = []**, les éléments entre 1 inclus et 3 exclu sont effacés.
- On peut également utiliser l'instruction del pour enlever des éléments dans un slice. Avec **del liste[1:2]**, l'élément à l'indice 1 est effacé.

Liste

OpérationS de slicing

```
i = 4 # crèation d' une variable
liste = [i, "spam", 3.2, True]
liste[1:3] # prend tous les elts 1 inclus à 3 exclu
liste[1:3]=[1,2,3] # les éléments entre 1 inclus et 3 exclu sont remplacés par [1,2,3]
del liste[1:2] # intruction del pour enlever les elts dans un slice
```

\textcolor Les méthodes {blue}{list.append()} et list.extend()

- L'opération list.append() ajoute un objet à la fin d'une liste.
- L'opération list.extend() prend une séquence et ajoute chaque élément de cette séquence à la fin de la liste. C'est l'équivalent de list.append() sur a chaque élément de la séquence.

La méthodes list.sort()

- Pour une liste **a = [1,5,3,1,7,9,2]**, on peut appeler la méthode list.sort() qui va trier les éléments de la liste.
- *ATTENTION!* list.sort() fonctionne en place

Liste

La méthodes list.sort()

Il existe des opérations permettant de passer d'une chaîne de caractère à une liste et d'une liste à une chaîne de caractère. C'est une opération fréquemment utilisée pour accéder à des fichiers que l'on veut traiter avec python.

Pour commencer, on crée une chaîne de caractères s = 'spam egg beans'.

- L'objectif est de séparer cette chaîne de caractère en colonnes. Pour cela, on utilise la fonction built-in **split()** qui est une fonction des chaînes de caractères
- On peut retransformer la liste en chaîne de caractères avec la syntaxe : **" "** **".join(list)** . L'espace entre guillemets avant **.join()** désigne le séparateur que **.join()** va mettre entre chaque élément.

Liste

La méthodes list.sort()

```
"spam, egg bacon".split()  # separe une chaine de caratere en liste
"spam, egg bacon".split(",")  # on peut aussi specifier un separteur (" ", " /", ",/")
"".join(['sapm', ' egg bacon ']) # on transforme une liste en chaine de caratere
" ".join(['spam', ' egg bacon']) #on peut specifier le séparateur
```

Introduction aux fonctions

- La déclaration d'une fonction en python est très simple on utilise la syntaxe suivante :
- **def** : pour spécifeir qu'on définit une fonction
- **Nom_de_la _fonction** : porte le nom de la fonction
- **Corps de la fonction** : contient les blocs instructions
- **return** : permet de retourner une valeur de la fonction
- Exemple de fonction qui calcul le carré d'un nombre passé en argument

Introduction aux modules

- Un module est un fichier python qui finit en .py. Lorsqu'on importe ce fichier, cela crée un objet module.
- L'idée des modules, c'est de mettre des opérations similaires dans le même fichier.
- Un module est une sorte de boîte à outil que l'on importe quand on a besoin de l'ouvrir.

- Pour importer un module, on utilise l'instruction import suivie du nom du module. Exemple avec le module random :
- On peut accéder à tous les objets d'un module avec la fonction dir( ).

```
import random
```

- Avec help(random) ,Python affiche toute l'aide liée à random. help() est très pratique pour regarder le fonctionnement d'une méthode particulière issue d'un module : help(random.randint) nous indique

Introduction aux modules

Avec la librairie standard on peut :

- Faire de la programmation parallèle ou de la programmation asynchrone
- Faire de la persistance de données, une opération de sérialisation pour garder une copie des objets sur le disque dur
- Faire de la communication sur Internet. Python supporte quasiment tous les protocoles classiques sur Internet
- Formater les données et lire des formats spécifiques de données sur Internet
- Manipuler des fractions ou des nombres décimaux
- Écrire des expressions régulières
- Gérer des dates et des calendriers
- Interagir avec le système de fichiers, créer des fichiers, des répertoires, parcourir des répertoires
- Faire de la compression de fichiers
- Écrire des interfaces graphiques

Introduction aux modules

En plus de sa librairie, Python a également des centaines de milliers de modules écrits par des groupes de développement ou par des individus que l'on peut charger et importer notre programme comme **Django** pour faire du dévellopement Web