UNIVERSITE PARIS 8
Vincennes - Saint-Denis

Rapport du TP 02

Programmation concurrente

Sarah KADI

Université Paris 8 | Vincennes - Saint-Denis

Introduction :

Le rapport comportera la compréhension acquise après la réalisation du tutoriel sur le langage python puis entamera la partie « problèmes rencontrés » qui évoquera toutes les difficultés rencontrées.

Compréhension :

Notions acquises sur le langage python :

Les commentaires : pour introduire une ligne comportant un commentaire, cette dernière doit commencer par le symbole #.

En introduisant ce dernier, tout ce qui va le suivre sur la même ligne sera considéré comme un commentaire et ne sera donc pas pris en compte par l'interpréteur. Si le symbole # est entré dans une chaine de caractères, celui-ci sera considéré comme un simple symbole (un caractère) et non comme un symbole indiquant un commentaire.

La Saisie :

- En python, la saisie d'un calcul peut être effectuée directement dans l'interpréteur ou le résultat sera retourné.

  Le fonctionnement des calculs reste le même au niveau de la priorité des opérateurs et des parenthèses.

- Le mélange des types « *int* » pour entier et « *float* » pour réel donnera toujours un résultat de type « *float* », par exemple on aura : 1 – 0.5 = 0.5 ou alors 2 + 3.5 = 5.5.
- La division de deux nombres retourne toujours un nombre de type « *float* » (même si les deux nombres entrés sont de type « *int* »). Par contre, pour utiliser une division entière on remplacera le symbole habituel de la division « / » par deux barres obliques « // ».
- Le reste d'une division entière s'obtient à l'aide du symbole « % », on aura par exemple 6//4, une opération représentant donc la division entière de 6 par 4 ce qui donne le résultat : 1. 1 * 4 = 4 donc le reste de cette division est 2 (car 6 – (1*4) = 2). C'est donc pour cela que 6%4 = 2.
- La puissance s'écrit de la manière suivante : 5^2 s'écrit 5 ** 2.

- L'affectation d'une valeur à une variable se fait à l'aide du symbole « = » comme sur le langage C, mais à la différence de ce dernier, la déclaration d'une variable sans valeur engendre une erreur lors de la compilation, on ne peut donc pas écrire par exemple : « int a ».
- L'utilisation de la variable « _ » directement sur l'interpréteur permet de récupérer la dernière expression affichée.
- L'utilisation des nombres complexes se fait en ajoutant un « j » ou un « J » pour indiquer la partie imaginaire.

Les chaines de caractères :

Une chaine de caractère en python peut s'écrire sous la forme : 'chaine de caractères' entre des guillemets simples, ou alors entre des guillemets doubles tel que "chaine de caractères ".

Pour différencier un guillemet simple « ' » d'une fin de chaine de caractères, le symbole « \ » doit être placé avant ce guillemet, prenons un exemple :

Pour afficher la chaine de caractères « l'école », cette dernière doit être déclarée ainsi : 'l\'école', dans le cas ou notre chaine de caractères est placée entre des guillemets doubles, il est inutile de « protéger » le guillemet simple, ainsi donc la chaine de caractères « l'école » s'écrira "l'école".

Pour avoir un affichage de la chaine de caractères ne contenant pas de guillemets, l'utilisation de la fonction print() est recommandée. En effet, dans la fonction print(), pour éviter que l'interpréteur interprète le symbole « \ » comme un symbole spécial, il faudrait précéder la chaine de caractères à l'intérieur du print() de la lettre « r », exemple : pour afficher la chaine de caractères suivante : « to\to » il faudrait la déclarer ainsi : print(r'to\to').

Pour concaténer deux chaines de caractères, il suffit d'écrire le symbole « + » entre les deux chaines, exemple : 'tata' + 'toto' ce qui nous donne « tatatoto ».

La concaténation de cette manière qui est également possible pour les variables, exemple : var1 = "toto" et var2 = "tata".

print(var1 + var2) donne « tototata ».

Nous pouvons également répéter une chaine de caractères n fois en l'écrivant sous la forme n * 'chaine de caractères'. Cependant, cela ne fonctionne pas avec les variables.

Si deux chaines de caractères sont écrites chacune entre guillemets et côte à côte, celles-ci seront concaténées automatiquement mais cela ne fonctionne bien évidemment pas avec les variables.

La chaine de caractère est indexée, c'est à dire que si on ne veut afficher que la seconde lettre d'un mot par exemple, cela se fera de la manière suivante :

var1 = 'toto'

print(var1[1]) donne le résultat « o ».

Nous pouvons également effectuer un décompte, c'est à dire commencer par la fin, ce qui veut dire que pour afficher la dernière lettre on effectue un print(var1[-1]).

Nous pouvons également obtenir une sous chaine en indiquant l'indice auquel on aimerait s'arrêter, par exemple :

s = 'exemple'

print(s[0:2]) donne le résultat : « ex ».

La fonction « len() » renvoie la longueur d'une chaine de caractères.

Différentes écritures provoquant des erreurs au niveau de l'interpréteur :

- Donner un indice trop grand à l'intérieur de la fonction print().
- Essayer d'affecter une valeur à un caractère dans la chaine de caractères.

Les listes :

Les listes sous python sont écrites entre crochets et les éléments sont séparés par des virgules. Ces derniers ne sont pas forcément du même type même s'ils appartiennent à la même liste.

Les listes peuvent être indicées et découpées, pour afficher un élément d'une certaine position dans une liste il suffit de l'indiquer comme ceci :

Liste = [1,2,4,5]

Liste[2] affichera « 4 ».

Nous pouvons également découper la liste de manière vu précédemment sur le chapitre « les chaines de caractères » et le résultat renvoie une copie

superficielle de la liste contenant le découpement demandé. Aussi, les listes possèdent la même manière de concaténation que les chaines de caractères, mais à la différence de ces dernières, remplacer un élément de la liste est possible. Il est également possible de rajouter un élément à la lise à l'aide de la fonction « append() ». Prenons un exemple :

Liste.append(23) remplacera la liste originale par Liste = [1,2,4,5,23]

On peut également remplacer plusieurs valeurs en même temps comme par exemple :

Liste[0:2] = ['a','b'] remplacera la liste par Liste = ['a', 'b', 4, 5, 23].

Pour connaitre la taille de la liste il suffit de faire appel à la fonction « len() ».

Il est également possible d'imbriquer des listes comme ceci :

Liste1 = [1,2,3]

Liste2 = ['a','b','c']

Liste3 = [Liste1,Liste2]

Liste3 donnera donc [[1,2,3], ['a','b','c']]

Pour afficher un élément d'une « sous liste » : par exemple pour afficher le caractère 'b' dans la liste Liste3 il suffira d'écrire Liste3[1][1].

Autres notions :

Il est possible d'effectuer une affectation multiple en python par exemple :

a veut 1 et b vaut 2, on peut simplement écrire a,b = 1,2.

IMPORTANT : l'indentation est la méthode utilisée par python pour gérer les instructions, dans le cas du non-respect de l'indentation, l'interpréteur engendrera des erreurs.

Problèmes rencontrés :

Aucun problème n'a été rencontré lors de la réalisation du tutoriel.