

Première partie

Présentation du cours et du langage support

Les transparents sont accessibles à l'adresse <http://etudiant.istic.univ-rennes1.fr/current/L1L2/ps/> mais sous une forme plus compacte. Ils y sont déposés en général après le cours.

1 Introduction

Ce que l'informatique n'est pas :

- jouer à des jeux vidéos ;
- saisir un document et l'imprimer ;
- surfer sur le web
- lire son courrier électronique ;
- ...

Ceci correspond à l'utilisation d'outils mis à disposition sur un ordinateur.

L'informatique, qu'est-ce que c'est ?

- c'est la science des procédés de calcul et du traitement de l'information. Un procédé de calcul est appelé un **algorithme** ou **programme**.
- c'est faire des programmes justes, efficaces, faciles à faire évoluer, lisibles, ...

Pourquoi de l'informatique ?

- parce que c'est obligatoire ;
- parce ce que, en tant que scientifique, vous allez devoir l'utiliser ;
- parce que cela est utile ;
 - pour savoir ce qu'est un programme ;
 - pour pouvoir en écrire ;
 - pour avoir une connaissance des principes de fonctionnement d'un ordinateur.

Objectif du cours : apprendre à concevoir et mettre au point des (petits) programmes. Contenu du cours :

- écrire des expressions ;
- écrire des fonctions ;
- écrire des expressions conditionnelles ;
- utiliser des données structurées ;
- écrire des itérations ;
- manipuler des structures de données « complexes »
- ...

Difficultés souvent rencontrées :

- la rigueur d'un ordinateur qui est moins tolérant qu'une personne ;
- la simplicité apparente alors que les difficultés apparaissent à l'usage ;
- comprendre les réactions de l'ordinateur et en particulier les réactions cachées.

Le langage support utilisé pour ce cours est Python (**version 3**).

Pourquoi Python ?

- pour mettre en œuvre les notions vues (celles ci sont indépendantes du logiciel utilisé)
- Python est un langage de haut niveau portable, gratuit et utilisé par de nombreux scientifiques.
- sa syntaxe est simple, il offre des structures de données évoluées
- il est dynamiquement typé

2 Qu'est-ce qu'un ordinateur ?

Un ordinateur est composé, physiquement, des éléments matériels suivants :

1. un *processeur* qui exécute des programmes ,
2. une *mémoire* dans laquelle sont stockés les programmes et les données permettant le fonctionnement de l'ordinateur au cours d'un calcul ,
3. des dispositifs d'*entrées/sorties* : écran, clavier, souris, imprimante, qui permettent de communiquer avec l'ordinateur,
4. une (ou plusieurs) *mémoires de masse* (disques par exemple), dans lesquelles sont mémorisées de façon permanente des programmes et les données (dans des *fichiers*).

Un ordinateur est livré avec un ensemble de *logiciels* qui en permettent l'utilisation.
Le principal logiciel s'appelle un *système d'exploitation*.
Ce logiciel permet aux programmes conçus par des utilisateurs, ou réalisés par des professionnels, d'être exécutés sur l'ordinateur.
On appelle ces programmes des *programmes d'application*.

3 Comment fonctionne Python ?

3.1 Interprétation ou compilation ?

comme Java, Python combine Les deux!!

1. le code Python est compilé ;
2. cela produit un code intermédiaire (Bytecode) ;
3. ce code est interprété par l'interpréteur Python ;
4. le résultat est affiché (sauf erreur!).

```
>>> 2**3 + 2**2
```

```
12
```

3.2 Documentation en ligne

Dans l'interpréteur, la saisie de *help()* vous permettra d'obtenir une description en anglais de certains éléments.

Sinon, le site <http://docs.python.org/3.3/> vous donnera accès à la documentation (toujours en anglais) de la version de Python utilisée dans ce cours.

4 Langage, syntaxe et sémantique

Pour communiquer (dialoguer) avec une application, on utilise un langage dont les phrases sont des expressions. Ce langage possède une **syntaxe** et une **sémantique**.

La **syntaxe** est la forme des expressions. La syntaxe doit respecter une grammaire pour être reconnu par une application.

La **sémantique** est la signification (ou l'effet) d'une expression.

Une erreur de syntaxe est une faute de grammaire

```
>>>2 +
```

```
SyntaxError: invalid syntax
```

L'interpréteur vous le signale immédiatement

Une erreur sémantique correspond à l'obtention d'un résultat qui n'est pas celui attendu :

```
>>>5 // 2
```

```
2
```

Le problème ne vient pas de l'interpréteur mais de ce que vous lui avez demandé de faire !

La recherche de telles erreurs et leur correction est une tâche importante de tout programmeur!!!

5 Premiers pas

5.1 Calculer avec Python

En lançant l'interpréteur, vous pourrez utiliser Python comme une « simple » calculatrice.

En tapant *python3* depuis une « ligne de commande » vous obtiendrez :

```
Python 3.3.3 (default, Jan 15 2014, 18:35:36)
[GCC 4.7.2 20120921 (Red Hat 4.7.2-2)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

```
2+5
2 - 3 # espaces optionnels
7+3*5
(7+3)*5
20/3
```

Dans le cadre de ce cours, nous utiliserons l'environnement de travail « Idle » (commandes « idle3 »)
Là aussi, les trois caractères `>>>` correspondent à l'invite de commande pour la saisie d'une expression Python

N.B. : Idle vient de « Integrated DeveLopment Environment » (généralement noté IDE), mais est aussi une référence à « Eric Idle », membre fondateur de la troupe des « Monthy Python », le nom du langage venant de cette troupe.

5.2 Types numériques de base

Il y a trois types numériques de base dans Python

int : suite de chiffres, éventuellement précédée d'un signe. Correspond aux entiers relatifs.

float : les nombre « réels ». Pour être considéré comme tel, un nombre doit comporter un « . » ou exposant de 10.

complex : les nombres complexes. Ils sont noté « x + yj ».

En cas de combinaison dans une même expression, c'est le type le plus général qui l'emporte

Attention : bien insisté sur le fait que les flottants ne sont pas les réels, mais une représentation d'une partie de ceux-ci. Ce sont des nombres à virgule flottante, représentés en mémoire à l'aide d'une mantisse et d'un exposant.

Évoquer, si le temps, le problème de la précision en faisant les deux opérations suivantes :

`10*0.1` et `0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1`

Parmi les autres types, en voici deux que nous allons rencontrer très rapidement :

bool : une des deux valeurs booléennes « vrai » ou « faux » ; **Attention** : les valeurs booléennes sont considérées comme numériques par Python. Faire des exemples d'opérations.

str : chaîne de caractères (suite de caractères entre « " » ou « ' » .

On peut obtenir le type d'une expression grâce à la fonction `type()`

Exemple 5.1. Portable : faire des exemples avec la fonction « `type()` »

```
type(2)
type(2.) type(2e1) type(2E1)
type(4+6j) type(1j*1j)
type(True) type(False)
type("chaîne") type('autre chaîne') type("c")
True + False True + 2
```

5.3 Conversion de valeurs

Les opérandes des opérateurs algébriques (+, -, *, etc.) doivent être de même type.

Si ce n'est pas le cas, il y a conversion vers le type le plus général : entier vers réel vers complexe.

— `2 + 3` donne 5

— `2 + 3.1` donne 5.1 et `2 + 3.` donne 5.

— `2 + (3+7j)` donne (5+7j) et `1j*1j` donne (-1+0j)

Donner au tableau (et sur le portable) les principaux opérateurs, en particulier « `**` », « `%` » et « `//` », ainsi que les fonctions **abs**, **min**, **max** et **pow**.

5.4 Données et variables

Pour manipuler les données, Python a besoin de variable : il s'agit d'un emplacement mémoire contenant une valeur et désigné par un identificateur.

Symboles : ou identificateurs, suite de lettres et de chiffres (plus le `_`) commençant par une lettre Par ex. `x1`, `Z2B6`, `Ident`, mais pas `2x`, ou `Z/8`.

Certains sont prédéfinis (une trentaine) Python les connaît et leur a donné une signification particulière. Par ex. `and`, `or`, `not`, `def`, `if`, `else`, `elif`, `while`, `for`, `in`, ...

Les afficher sur le portable en tapant **`help()`** puis **keywords**

5.5 Affectation

Cela permet d'établir le lien entre le nom d'une variable et sa valeur (son contenu) : on va affecter une valeur à une variable.

```
>>>m= 7
```

```
>>>pi=3.14159
```

```
>>>message="Quoi d'autre ?"
```

Pour afficher le contenu d'une variable, vous avez deux possibilités :

- dans l'interpréteur, vous tapez le nom de la variable
- dans un programme (vu par la suite), vous utilisez **toujours** la fonction *print*

```
>>> m
7
>>>print (m)
7
```

Bien insisté sur la différence entre « rendre un résultat » et « afficher un résultat ».

Typage d'une variable

Python utilise un typage **dynamique**.

Le type d'une variable est fonction du type de la valeur qui y est affectée.

Une même variable peut donc changer de type au cours du temps.

```
variable = 7    -> variable contient un entier
```

```
variable = 12.  - > elle contient maintenant un flottant
```

Pour la lisibilité et la maintenance du code, il est fortement déconseillé de changer le type d'une variable. Tolérance pour `int` et `float` !

L'effet de chaque affectation peut être décrit comme suit :

- créer et mémoriser le nom de la variable;
- lui attribuer un type bien déterminé;
- créer et mémoriser une valeur particulière;
- établir le lien entre le nom de la variable et l'emplacement mémorisant la valeur.

L'affectation peut être multiple

```
x = y = 7
```

```
x,a, msg = 5, 5.2,"Et ensuite ?"
```

Attention : c'est à utiliser avec parcimonie car cela peut nuire à la lisibilité du code !

6 Priorité des opérations

En l'absence de parenthèses, les opérations ont un *ordre de priorité*, qui décide dans quel ordre les calculs sont effectués :

- la puissance d'abord,
- le moins (et le plus) unaire
- la division / ou //, la multiplication *, le modulo %
- et enfin le + et le - binaire (comme dans $a+b$ ou $a-b$).

Si deux opérations ont la même priorité, elles sont évaluées de gauche à droite sauf ** qui est de droite à gauche.

```
>>> 45 + 2 - 512/(2**2**3 * 4) * 8
```

l'évaluation est faite dans l'ordre suivant :

- $2**3 = 8$
- $2**8 = 256$
- $256*4 = 1024$
- $512/1024 = 0.5$
- $0.5*8 = 4.$
- $45+2 = 47$
- enfin $47-4. = 43.$

Que donne le calcul avec « // » à la place de « / » ?