

## Programmation scientifique Présentation du cours

francois.poulet@univ-rennes1.fr

<http://etudiant.istic.univ-rennes1.fr/current/I1I2/ps>

CM : 2 x 2 heures présentiel

3 x 2 heures FOAD (supports en ligne)

TD : 5 x 2 heures (bat.32A)

TP : 5 x 2 heures (dernier TP noté) bat.2

**2 absences injustifiées en TP = 0 note TP**

Evaluation : Note = (3xExam + TP) / 4



## Plan – Partie I

- introduction
- qu'est ce qu'un ordinateur?
- qu'est ce que Python?
- langage, syntaxe et sémantique
- premiers pas
  - calcul avec Python
  - types numériques
  - conversions de valeurs
  - données et variables
  - affectation
- priorités des opérateurs

Ifsic – Université de Rennes 1

2

## Introduction

- informatique = traitement automatique de l'information
- traitement automatique :
  - algorithme (comment faire pour résoudre le problème) : TD
  - programme (comment mettre en œuvre en pratique) : TP
- programmes justes, efficaces, réutilisables...

Ifsic – Université de Rennes 1

3

## Introduction

- objectif : concevoir et mettre au point des programmes
- contenu :
  - écrire des expressions
  - écrire des fonctions
  - écrire des conditionnelles
  - données structurées
  - itérations
  - structures de données « complexes »
  - ...

Ifsic – Université de Rennes 1

4

## Introduction

- difficultés rencontrées:
  - intolérance totale aux erreurs !!!
  - simplicité apparente, difficultés réelles
- comprendre les erreurs

Ifsic – Université de Rennes 1

5

## Introduction

- langage support : Python (v.3)
- pourquoi?
  - mise en œuvre des notions
  - haut niveau
  - portable
  - gratuit
  - récent, très utilisé dans beaucoup de domaines
  - syntaxe simple, structures de données évoluées
  - dynamiquement typé

Ifsic – Université de Rennes 1

6

## Introduction

- assembleur : 1948
- Fortran : 1956
- Cobol, Lisp : 1959
- Algol : 1960
- Basic : 1965
- Pascal : 1966
- C, Prolog, Smalltalk : 1972
- C++ : 1983
- Perl : 1986
- html : 1990
- **Python : 1991**
- Java, PHP : 1995

## Qu'est-ce qu'un ordinateur?

- composition :
  - un processeur (le centre de calcul)
  - une mémoire vive (les données)
  - E/S : écran, clavier, souris... (interactions utilisateur)
  - une mémoire de masse (HD)

## Python

- comment ça marche?
- 2 types de langages :
  - compilé
  - interprété
- quelle différence ?
  - compilé : langage source -> langage processeur (ensemble du programme)
  - interprété : langage source -> traduit au fur et à mesure

## Python

- et alors Python?
- les 2 !!!
- code source compilé en un code intermédiaire
- code intermédiaire interprété par l'interpréteur Python
- affichage du résultat
- `>>> 2**3 + 2**2`
- 12

## Python

- `help()` => description de certains éléments du langage
- <http://docs.python.org/3.x/> => doc en anglais de Python
- beaucoup de livres / sites sur Python

## Python

- utilisation d'un langage
  - syntaxe
  - sémantique
- erreur de syntaxe : faute d'orthographe / grammaire
- `>>>2 +`
- `SyntaxError: invalid syntax`
- signalé immédiatement par l'interpréteur

## Python

- sémantique :
  - facilité d'écriture de programmes maintenables
  - différents styles de programmation (impérative, fonctionnelle...)
- erreur sémantique résultat inattendu
- `>>>5 // 2`
- `2`
- syntaxiquement correct
- pas forcément le résultat attendu...

## Python

- lancer l'interpréteur python (utilisation comme une calculatrice)
- tapez `python3` & (ligne de commande)
- vous obtenez :  
Python 3.5.3 (default, May 10 2017, 15:05:55)  
[GCC 6.3.1 20161221 (Red Hat 6.3.1-1)] on linux  
Type "help", "copyright", "credits" or "license" for more  
`>>>`
- utilisation de l'environnement de travail idle (commande `idle3&`)

## Python

- 3 types numériques de base :
  - `int` : suite de chiffres (entiers relatifs)
  - `float` : nombres réels (doit comporter un `.` ou exposant de 10)
  - `complex` : les nombres complexes (`x + yj`)
- en cas de combinaison, résultat du type le plus général

## Python

- autres types :
  - `bool` : valeur booléenne (vrai / faux)
  - `str` : chaîne de caractères (entre « ' » ou « " »)
  - `type()` : retourne le type d'une expression

## Python

- opérandes des opérateurs algébriques (+, -, \*, etc) doivent être de même type
- si types différents conversion vers type le plus général : entier vers réel vers complexe
  - `2 + 3` donne 5
  - `2 + 3.` donne 5.
  - `2 + 3.1` donne 5.1
  - `2 + (3+7j)` donne (5+7j)
  - `1j * 1j` donne (-1 + 0j)

## Python

- données et variables
  - pour manipuler des données il faut une variable (désignée par un identificateur) pour la stocker en mémoire
  - identificateur est suite de lettres ou chiffres (plus le `_`) commençant par une lettre

## Python

- affectation permet établir le lien entre le nom d'une variable et sa valeur
- on affecte une valeur à la variable :
- `>>>m = 7`
- `>>>pi = 3.1415926`
- `>>>message = "Quoi d'autre?"`

## Python

- afficher le contenu d'une variable :
- dans l'interpréteur, taper le nom de la variable
- dans un programme, utiliser toujours la fonction print
- `>>>m`
- `7`
- `>>>print (m)`
- `7`

## Python

- typage d'une variable
- Python utilise le typage dynamique :
- la variable prend le type de la valeur affectée
- une variable peut donc changer de type
- `variable = 7` -> variable de type int
- `variable = 12.` -> variable de type float
- **fortement déconseillé de changer le type**

## Python

- effet d'une affectation :
- créer et mémoriser le nom de la variable
- lui attribuer un type
- créer et mémoriser sa valeur
- établir le lien entre le nom de la variable et son emplacement mémoire

## Python

- priorité des opérations :
- sans parenthèse, ordre de priorité :
- la puissance d'abord
- le plus et le moins unaire
- la division / ou //, la multiplication \*, le modulo %
- le + et le - binaire ( $a + b$  ou  $a - b$ )
- si deux opérations de même priorité, évaluation de gauche à droite sauf \*\* de droite à gauche

## Python

- `>>> 45 + 2 - 512 / (2**2**3 * 4) * 8`
- l'évaluation s'effectue dans l'ordre suivant :
- `2**3 = 8` (\*\* plus prioritaire, éval de gauche à droite)
- `2**8 = 256`
- `256 * 4 = 1024`
- `512/1024 = 0.5` (même priorité / et \*, droite à gauche)
- `0.5*8 = 4.`
- `45 + 2 = 47`
- `47-4. = 43.`
- que donne le calcul avec // à la place de / ?