~~REVEAL transition=convex&controls=1&show_progress_bar=1&build_all_lists=1&open_in_new_window=1~~

Programmer en Python

Généralités

- Qu'est-ce qu'un langage de programmation ?
- Compilation ou interprétation, ou… ?

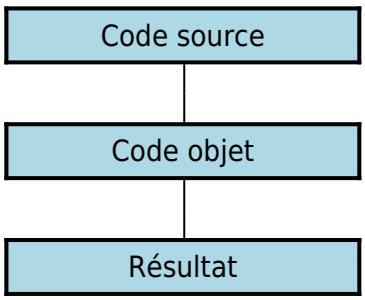
Rôle des langages de programmation

- Décrire des instructions dans un langage compréhensible par un être humain, mais transformable en d'autres instructions compréhensibles par l'ordinateur (langage machine)
- Automatiser le traitement de l'information;
- Effectuer des calculs, des simulations;
- Traiter l'information en temps réel;
- Fournir un interface à l'utilisateur;

Évolution des langages

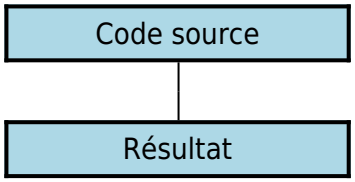
- L'assembleur (à partir des années 50's)
  - Mnémoniques équivalentes aux instructions machines, donc fonction du processeur utilisé
  - Instructions de bas niveaux (appel d'une variable en mémoire, opération arithmétique entre 2 opérandes,…)
- Fortran, Cobol, Pascal, C, Basic,… (années 60s et 70s)
  - Indépendants de l'ordinateur utilisé
  - Proche d'un langage courant, description procédurale
- Les langages à objets (années 80s et 90s)
  - Briques logicielles indépendantes et autonomes
  - Réutilisations aisées, sans devoir les approfondir
  - Java, C++, Python, perl, Ruby. . .sont les plus connus
- Des langages spécialisés (PHP, SQL,…)

Compilation et compilateur



- Etape de traduction du code source en langage machine
- Liaison éventuelle du code avec des bibliothèques existantes de code compilé
- Exécution ultérieure du code machine (sur un ordinateur ne disposant pas du compilateur par exemple)
- Le compilateur peut optimiser le code (passes multiples)

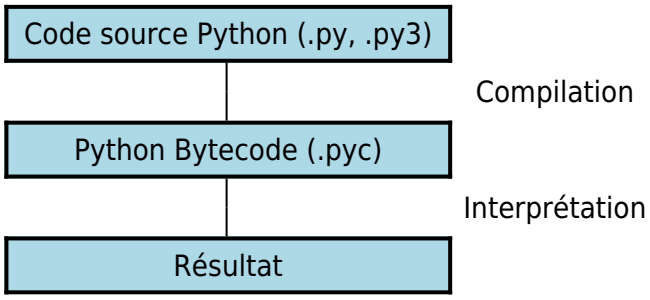
Interprétation et interpréteur



Interprétation

- Traduction dynamique du code source et exécution immédiate en répétant sans cesse :
  - lecture et analyse d'une instruction
  - exécution de l'instruction (si elle est valide)
- Le code est souvent moins optimisé, donc plus lent
- Il est nécessaire de disposer de l'interpréteur sur l'ordinateur
- On peut créer dynamiquement du code à interpréter pendant l'exécution
- On peut éviter la phase lente de compilation

Python / Langages à Bytecodes



- Pour Python (et d'autres langages), c'est un peu plus compliqué...
- Le programme est compilé vers un pseudo-code indépendant de l'ordinateur
- Le Bytecode est interprété par la suite
- Avantages :
  - Facilité de développement (cycle écriture-exécution rapide, "briques" logicielles)
  - Portabilité (même programme pour des ordinateurs et OS différents)

Premier aperçu de Python

- Avantages généraux
- Avantages techniques
- Avantages pour l'apprentissage
- Avantages pour le scientifique, le chimiste
- Les premiers pas avec Python

Avantages généraux

- langage de haut niveau (orienté objet)
- permet d'écrire des petits programmes ou suites d'instructions (scripts)
- licence libre (et gratuit)
- utilisable pour la programmation occasionnelle par des non-informaticiens
- nombreuses librairies existantes (modules)
- moderne et efficace pour les informaticiens
- excellente lisibilité intrinsèque du code

- bien documenté (aide et manuels en ligne, livres, forums, exemples…)

Avantages techniques

- mode interactif
- non déclaratif
- typage de haut niveau, dynamique et fort
- ramasse-miette intégré
- interfaçable avec d'autres langages (à partir de et vers)
- version de base "piles comprises"
    - module mathématique
    - accès aux fichiers et répertoires (+ formats de données standards)
    - compression, archivage, gestion de bases de données
    - fonctions génériques du système d'exploitation
    - réseau et communication, protocoles internet (+email, html)
    - multimedia (son, image)
    - interface graphique (Tkinter)
    - outils de documentation et gestion d'erreurs (débogage)
    - modules spécifiques Windows, Mac, Linux
    - …

Avantages pour l'apprentissage

- Installation aisée
    - de la version de base
    - de "distributions" étendues (avec des modules complémentaires)
- éditeur inclus (Idle) ou autre (SciTe, Pycharm, Eric,…)
- mode interactif pour les premiers essais
- principes de base identiques à de nombreux langages
- on n'est pas obligé d'utiliser toute la puissance du langage
- cycle d'écriture/essais très rapide

Avantages pour le scientifique, le chimiste

- possible de débuter en quelques jours
- alternative à des logiciels spécialisés (matlab, scilab,…)
- bon pour les calculs scientifiques, le graphisme, les simulations
- modules spécialisés
    - représentations graphiques 2D (Matplotlib)
    - représentations graphiques 3D (Mayavi, Vpython, VTK,)
    - calculs scientifiques (numpy, scipy,. . .)
    - traitement d'images (PIL)
    - chimie (pymol, mmtk, chimera,…)
    - biochimie (biopython)

Les premiers pas avec Python 3

- Sans installation : https://repl.it/languages/python3
- Python Setup and usage

Ídle3 : interface d'exécution et d'édition

Idle3 : interpréteur Python en console, avec exécution directe

```
>>> (8.314*300/24E-3)/101325
1.0256600049346163
>>>
```

- Commandes : copyrigth, credits, license(), quit, help, help(),...

Notion de variable

On peut attribuer des noms de variables, pas seulement pour des nombres...

```
>>> R=8.314
>>> L=0.001
>>> V=24*L
>>> n=1
>>> zero=273.16
>>> T=20+zero
>>> P=n*R*T/V
>>> atm=101325
>>> print(P,P/atm)
101555.51000000001 1.0022749568221072
>>>
```

Un peu de calcul

On peut effectuer quelques calculs sur des entiers :

```
>>> 1236*5698
7042728
>>> 12569+6233
18802
>>> 12+69+532+65-9
669
>>> 12356*458955
5670847980L
>>> 123*456
56088
>>> 123**456 ?? A ESSAYER ??
```

- On peut travailler avec des très très très grands nombres...

Division et division entière

```
>>> a =7/3
2.3333333333333335
>>> b = 7//3
2
```

```
>>>
```

- En python, chaque “objet” possède son type et un identifiant :
  - type(a)
  - id(a)
  - type(b)
  - id(b)

De nombreuses autres possibilités avec les nombres...

```
>>> Navogadro=6.02214199E23
>>> kboltzmann=1.3806505E-23
>>> print Navogadro*kboltzmann
8.31447334956
>>> 2**0.5
1.4142135623730951
>>> (5+2j)*(3-7j)
(29-29j)
>>> (1.+1./1E6)**1E6
2.7182804690957272
>>>
```

- Les expressions numériques s’évaluent en respectant les régles habituelles de priorités : parenthèses, exponentiation, multiplication, division, addition, soustraction (“PEMDAS”)
- On peut aussi travailler facilement avec des tableaux contenant des milliers de données !

Un peu de logique : le type booléen !

```
>>> 12 < 16
True
>>> 12 < 11
False
>>> 12 == 12, 12 == 13
(True, False)
>>> 12 < 16 or 12 < 11
True
>>> 12 < 16 and 12 < 11
False
>>> type(12 < 11)
<class 'bool'>
```

- Les tests, comparaisons et leurs combinaisons logiques sont utiles pour réaliser des opérations de manière conditionnelle. Pour la logique booléenne : cf. Algèbre de Boole

Les chaînes de caractères

- appelées aussi “string”
- mots, phrases, ou texte long
- délimitées par ’ (apostrophe) ou “ (guillemet)
- la casse est significative
- caractères accentués, spéciaux et chiffres permis (caractères Unicode)
- CONSEIL : éviter les accents dans les noms des variables

- peuvent comprendre des retours à la ligne (Enter) si délimitées par """

Les chaînes de caractères

```
>>> a=’bonjour’
>>> b="bonjour"
>>> c=’Bonjour’
>>> print a==b,a==c
True False
>>> d="pâté123#"
>>> print d
pâté123#
>>> é=d
>>> long="""un
deux
...
dix"""
>>> print(long)
un
deux
...
dix
>>>
```

Opérations sur les chaînes

```
>>> s=’Mons, le 15 septembre 2009’
>>> s[8:]
’ 15 septembre 2009’
>>> s.find(’le’)
6
>>> s.split()
[’Mons,’, ’le’, ’15’, ’septembre’, ’2009’]
>>> s.upper()
’MONS, LE 15 SEPTEMBRE 2009’
>>> s.replace(’ ’,’_’)
’Mons,_le_15_septembre_2009’
>>>
```

Créer son premier programme

- Utiliser Idle3 comme éditeur (ou tout autre éditeur)
- Sauvegarder
- Exécuter
- Fermer
- Rouvrir Idle3 et le programme
- Exécuter

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

```
""" Programme élémentaire en Python
pour afficher une chaîne de caractères
"""
chaine = 'Message : Hello World !'
print(chaine)
```

Un peu plus loin dans Python

Types de haut niveau