

Institut Préparatoire aux Études d'Ingénieurs de Tunis



Leila Ben Othman

Première année de classes préparatoires
MP-PC

Année universitaire: 2017 - 2018

PYTHON: Introduction et prise en main

Plan

① Présentation du langage

- Site officiel

- Contributeurs - Licence - Versions

- Caractéristiques du langage

- Domaine d'applications

- Installation - IDLE

② Mode shell et mode script

③ Premiers pas en PYTHON

- Typeage dynamique, objet, variable

- Types élémentaires en PYTHON

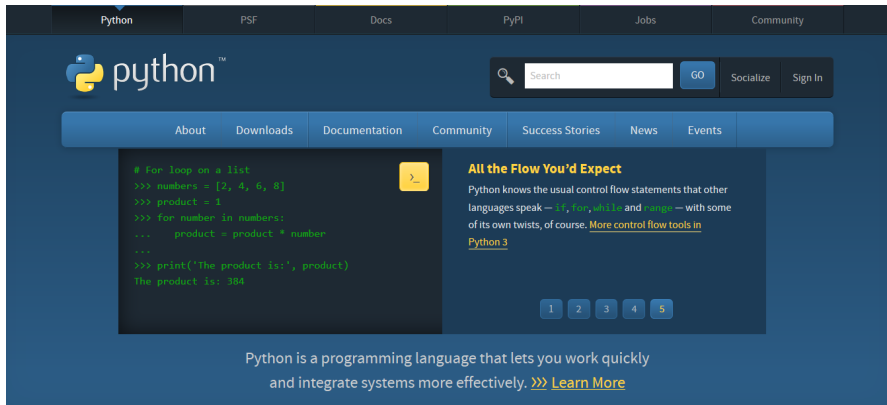
- Affectation

- Conversion de types

- Les bibliothèques en PYTHON

Site officiel

- <https://www.python.org/>



The screenshot shows the Python.org homepage with a dark blue background. At the top, there is a navigation bar with links: Python, PSF, Docs, PyPI, Jobs, and Community. Below this is the Python logo and a search bar. A secondary navigation bar contains links: About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area features a code snippet on the left and an article titled "All the Flow You'd Expect" on the right. The code snippet demonstrates a for loop and a product calculation. The article text discusses Python's control flow statements. At the bottom, a banner states: "Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)".

```
# For loop on a list
>>> numbers = [2, 4, 6, 8]
>>> product = 1
>>> for number in numbers:
...     product = product * number
...
>>> print('The product is:', product)
The product is: 384
```

All the Flow You'd Expect

Python knows the usual control flow statements that other languages speak — `if`, `for`, `while` and `range` — with some of its own twists, of course. [More control flow tools in Python 3](#)

1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)

Contributeurs - Licence - Versions

- En 1989, Guido van Rossum (Institut National de Recherche en Mathématique et Informatique - Pays-Bas) a commencé ses travaux sur Python.
- PYTHON est diffusé en 1991
- Contributeurs: PYTHON Software Foundation
- Licence libre
- Versions
 - Deux versions de base: version 2 et version 3
 - Actuellement PYTHON est à sa version 3.6 (Septembre 2017)
 - PYTHON 2 et PYTHON 3 sont légèrement incompatibles

Caractéristiques du langage

- Haut niveau
- Orienté objet (Héritage multiple, Surcharge des opérateurs, etc)
- Libre
- Portable
- Interprété
- Dynamiquement typé
- Efficace (gestion d'exceptions)
- Dôté d'un gestionnaire de mémoire
- Facile à apprendre/à lire/à écrire/à maintenir (syntaxe très simple et combinée à des types de données évolués (listes, dictionnaires, etc)
- Evolutif: Communauté très active
- Puissant: doté de bibliothèques très variées (numpy, scipy, etc)

Domaine d'applications

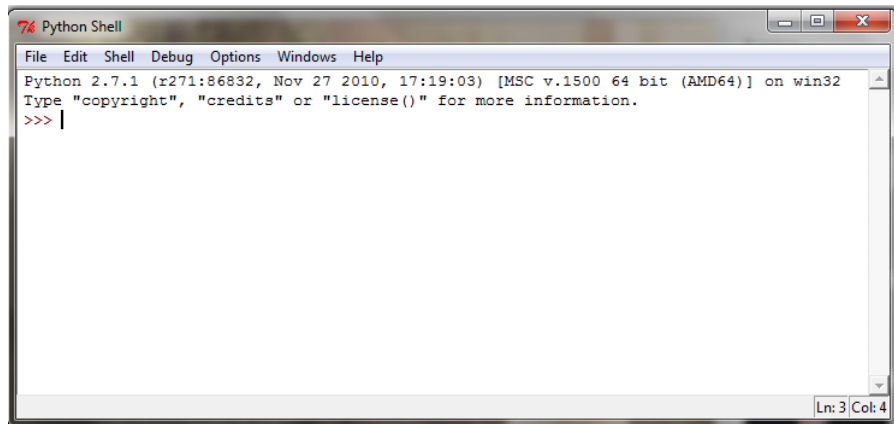
Que peut-on faire avec PYTHON?

- du calcul scientifique/statistique (numpy, scipy)
- des graphiques (matplotlib)
- de la bioinformatique (Biopython)
- Data science (pandas, scikit-learn)
- des jeux (Pygame)
- des applications web
- interfacier des systèmes de gestion de bases de données
- etc

Installation - IDLE

- Sous Windows, pour installer PYTHON, il suffit de le télécharger à partir du site officiel et de l'installer.
- Une fois Python installé, vous avez deux modes pour travailler: **le mode shell** et **le mode script**.
- PYTHON offre un environnement de développement intégré (IDE: Integrated Development Environment): **IDLE**
- IDLE propose un certain nombre d'outils:
 - un éditeur de texte (pour le mode script, avec coloration syntaxique, indentation automatique et auto-complétion)
 - un débogueur pour tester le programme
 - un interpréteur pour exécuter le programme
- Il existe d'autres IDE pour PYTHON: Spyder, Eclipse, Pydev, NetBeans, etc

Mode shell



The image shows a screenshot of a 'Python Shell' window. The title bar is 'Python Shell' with standard Windows window controls. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Windows', and 'Help'. The main text area displays the Python 2.7.1 startup message: 'Python 2.7.1 (r271:86832, Nov 27 2010, 17:19:03) [MSC v.1500 64 bit (AMD64)] on win32'. Below this, it says 'Type "copyright", "credits" or "license()" for more information.' and the prompt '>>>' is shown with a cursor. The status bar at the bottom right indicates 'Ln: 3 Col: 4'.

```
Python 2.7.1 (r271:86832, Nov 27 2010, 17:19:03) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

Mode shell

```
> > > 4 + 4  
8
```

> > > prompt ou invite de commande

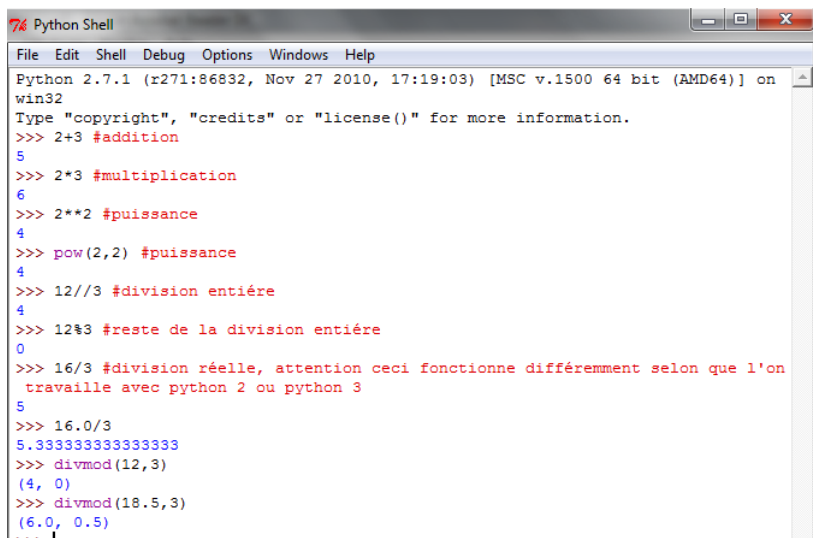
4 + 4 instruction PYTHON, entrée par l'utilisateur

8 réponse de l'interprète PYTHON

- Quelques opérations simples

- $4 + 4$
- $4 - 4$
- $4 + 2 * 2$
- $(4 + 2) * 2$
- $10 / 2$
- $10 \% 2$

Opérations élémentaires



The screenshot shows a 'Python Shell' window with a menu bar (File, Edit, Shell, Debug, Options, Windows, Help) and a command prompt interface. The prompt is 'Python 2.7.1 (x271:86832, Nov 27 2010, 17:19:03) [MSC v.1500 64 bit (AMD64)] on win32'. The user has entered several commands and their outputs are shown. Comments in red text explain the operations. The operations include addition, multiplication, exponentiation, and division (both integer and floating-point), as well as the divmod function.

```
Python 2.7.1 (x271:86832, Nov 27 2010, 17:19:03) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 2+3 #addition
5
>>> 2*3 #multiplication
6
>>> 2**2 #puissance
4
>>> pow(2,2) #puissance
4
>>> 12//3 #division entière
4
>>> 12%3 #reste de la division entière
0
>>> 16/3 #division réelle, attention ceci fonctionne différemment selon que l'on
travaille avec python 2 ou python 3
5
>>> 16.0/3
5.333333333333333
>>> divmod(12,3)
(4, 0)
>>> divmod(18.5,3)
(6.0, 0.5)
```

Opérations élémentaires

In [1]: $4+4$

Out [1]: 8

In [2]: $6-4$

Out [2]: 2

In [3]: $2^{**}4$

Out [3]: 16

In [4]: `pow`(2,4)

Out [4]: 16

In [5]: $12//3$

Out [5]: 4

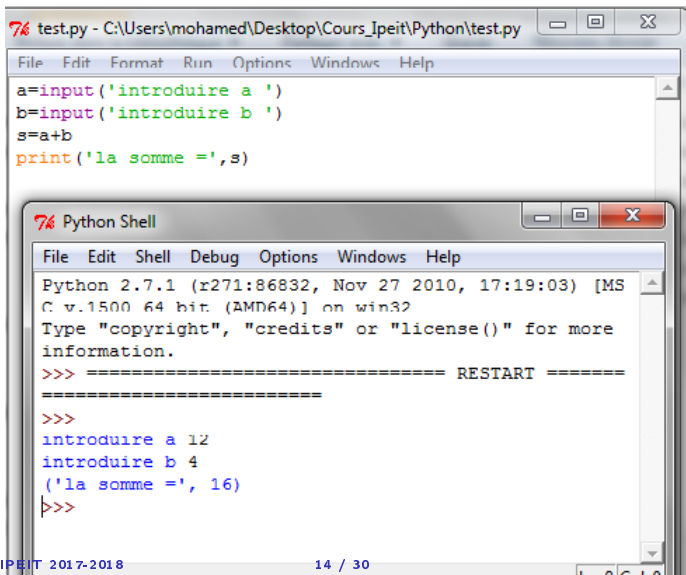
Opérations élémentaires

```
In [6]: 16/3  
Out [6]: 5.3333333333
```

```
In [7]: 20%2  
Out [7]: 0
```

```
In [8]: divmod(20,2)  
Out [8]: (10,0)
```

Mode script



The image shows two overlapping windows on a Windows desktop. The top window is a text editor titled 'test.py - C:\Users\mohamed\Desktop\Cours_Ipeit\Python\test.py'. It contains the following Python code:

```
a=input('introduire a ')
b=input('introduire b ')
s=a+b
print('la somme =',s)
```

The bottom window is titled 'Python Shell' and shows the execution of the script. It displays the Python version (2.7.1), the operating system (win32), and the input/output of the script:

```
Python 2.7.1 (r271:86832, Nov 27 2010, 17:19:03) [MS
C v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more
information.
>>> ===== RESTART =====
>>>
introduire a 12
introduire b 4
('la somme =', 16)
>>>
```

Typage dynamique, objet, variable

- PYTHON est un langage orienté Objet
⇒ Tout est objet : données, fonctions, modules, etc
- Un objet PYTHON possède:
 - une valeur (des données matérialisées par des valeurs d'attributs)
 - des opérations applicables (méthodes)
 - un emplacement mémoire (adresse): la fonction `id` permet de retourner l'adresse (identifiant d'un objet).
 - un type (nature de l'objet): la fonction `type` permet de retourner le type d'un objet (classe).
- On n'a pas besoin de spécifier le type d'un objet qu'on manipule ⇒ PYTHON gère ça automatiquement. On parle d'un typage dynamique

Typage dynamique, objet, variable

- Pour manipuler des objets et les distinguer l'un de l'autre, on utilise des variables. On dit qu'une variable fait référence à un objet.
- Une affectation `x=5`, implique:
 - la création d'un objet de type entier (int) avec une valeur égale à 5.
 - Le référencement de cet objet par la variable `x`.
- Un objet ne peut changer ni d'adresse mémoire ni de type.
- Quand un objet n'est plus référencé, il est détruit automatiquement (le mécanisme de ramasse miette ou de garbage collector).

Types élémentaires en PYTHON

- NoneType: seule valeur possible est **None**. C'est la valeur retournée par une fonction qui ne retourne rien;
- int: les nombres entiers;
- float: les nombres réels;
- complex: les nombres complexes;
- bool: **True**, **False**
- str: les chaînes de caractères

Types élémentaires en PYTHON

```
In [1]: x=4;type(x)  
Out [1]: <class 'int'>
```

```
In [2]: x=4.5;type(x)  
Out [2]: <class 'float'>
```

```
In [3]: x=4+5j;type(x)  
Out [3]: <class 'complex'>
```

```
In [4]: x=True;type(x)  
Out [4]: <class 'bool'>
```

```
In [5]: x='Bonjour';type(x)  
Out [5]: <class 'str'>
```

```
In [6]: x=None;type(x)  
Out [6]: <class 'NoneType'>
```

Création d'un objet de type complex

```
> x=1+5j ou bien x=complex(1,5)
> type(x)
<class 'complex'>
```

Manipulation d'un nombre complexe

```
> x.real # partie réelle
1.0
> x.imag # partie imaginaire
5.0
> x.conjugate() # le conjugué
(1-5j)
```

Égalité de valeur, égalité physique

- L'opérateur `==` test l'égalité de valeur entre objets.
- L'opérateur `is` test l'égalité physique entre objets (égalité des adresses mémoires)

Remarques:

- On parle d'égalité de valeur lorsque deux variables référencent la même valeur à deux emplacements mémoires pouvant être différents. Dans ce cas, nous avons deux objets différents qui ont une même valeur.
- On parle d'égalité physique lorsque deux variables référencent le même emplacement mémoire, ce qui implique l'égalité de valeur. Dans ce cas, nous avons un objet unique et par conséquent une valeur unique.

Égalité de valeur, égalité physique

In [1]: x=4;y=4; id(x); id(y)

Out [1]: (54564007,54564007)

In [2]: x==y,x is y

Out [2]: (True,True)

In [3]: a=4+5j;b=4+5j;id(a),id(b)

Out [3]: (12034569,12034465)

In [4]: a==b, a is b

Out [4]: (True,False)

In [5]: a=4+5j;b=a;id(a),id(b)

Out [5]: (12055465,12055465)

In [6]: x==b,a is b

Out [6]: (True,True)

Affectation simple, multiple, parallèle

```
In [1]: x=4;id(x) # affectation simple
```

```
Out [1]: 12034569
```

```
In [2]: x=y=z=4;id(x),id(y) # affectation multiple
```

```
Out [2]: (12034569,12034569,12034569)
```

```
In [3]: x,y,z=4,5,6;id(x),id(y),id(z) # affectation parallèle
```

```
Out [3]: (12034569,12034570,12034571)
```

```
In [4]: x,y=y,x;id(x),id(y) # permutation
```

```
Out [4]: (12034570,12034569)
```

Autres syntaxes des opérateurs mathématiques

```
In [1]: x=4;x+=1;x # x=x+1
```

```
Out [1]: 5
```

```
In [2]: x-=x;x # x=x-x
```

```
Out [2]: 0
```

```
In [3]: x=4;x*=2 # x=x*2
```

```
Out [3]: 8
```

```
In [4]: x/=x;x # x=x/x
```

```
Out [4]: 1
```

Conversion de types

- On peut passer d'un type à un autre en utilisant les fonctions suivantes : `int()`, `float()`, `str()`, `complex()`
- La conversion n'est pas toujours possible. Par exemple, il n'est pas possible de convertir un type `complex` en `float` ou `int`.
- Attention, ces fonctions créent de nouveaux objets avec le type spécifié et ne modifient pas le type de l'objet d'origine.

Conversion de types

La fonction `int()`

```
In [1]: int(4.5);
```

```
Out [1]: 4
```

```
In [2]: int(True)
```

```
Out [2]: 1
```

```
In [3]: int(False)
```

```
Out [3]: 0
```

```
In [4]: int('45')
```

```
Out [4]: 45
```

```
In [5]: int(1+5j)
```

```
Out [5]: Erreur
```

Conversion de types

La fonction float()

```
In [1]: float(4);
```

```
Out [1]: 4.0
```

```
In [2]: float(True)
```

```
Out [2]: 1.0
```

```
In [3]: float(False)
```

```
Out [3]: 0.0
```

```
In [4]: float('45')
```

```
Out [4]: 45.0
```

```
In [5]: float(1+5j)
```

```
Out [5]: Erreur
```

Conversion de types

La fonction `complex()`

```
In [1]: complex(4);
```

```
Out [1]: 4+0j
```

```
In [2]: complex(6.5)
```

```
Out [2]: 6.5+0j
```

```
In [3]: complex(True)
```

```
Out [3]: 1+0j
```

```
In [4]: complex(False)
```

```
Out [4]: 0j
```

```
In [5]: complex('4+5j')
```

```
Out [5]: 4+5j
```

Conversion de types

La fonction `str()`

```
In [1]: str(4);
```

```
Out [1]: '4'
```

```
In [2]: str(6.5)
```

```
Out [2]: '6.5'
```

```
In [3]: str(True)
```

```
Out [3]: 'True'
```

```
In [4]: str(False)
```

```
Out [4]: 'False'
```

```
In [5]: str(4+5j)
```

```
Out [5]: '4+5j'
```

L'import de bibliothèque

Fonctions mathématiques supplémentaires

```
In [1]: sqrt(4)
```

```
Out[1]: NameError: name 'sqrt' not defined
```

```
In [2]: import math # méthode 1
```

```
In [3]: math.sqrt(4)
```

```
Out [3]: 2.0
```

```
In[4]: import math as m # méthode 2 - avec alias
```

```
In[5]: m.sqrt(4)
```

```
out[5]: 2.0
```

```
In[6]: from math import * # méthode 3
```

```
In[7]: sqrt(4)
```

```
out[7]: 2.0
```

L'import de bibliothèque

Fonctions mathématiques supplémentaires

In [1]: `dir(math)` # affiche toutes les fonctions de la bibliothèque math

Out[1]: ['__doc__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'hypot', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']

In [2]: `help(m.sin)` # affiche une description de la fonction

Out[2]:

Help on built-in function sin in module math:

`sin(...)`

`sin(x)`

Return the sine of x (measured in radians).