

Introduction au langage Python

Massimo Stefani, Gymnase du Bugnon, le 31.05.19

Sources:

- [Pensez en Python](#) par Allen B. Downey,
- [Site officiel de Python](#)
- [w3schools.com](#)

Exemples:

- Quelques exemples ont été directement copiés de *Pensez Python*.

Images:

- PrintScreen

Menu

- [Introduction](#)
- [Console](#)
- [Exécution d'un script](#)
- [Le langage](#)
 - [La fonction input](#)
- [Affection](#)
 - [Opérations arithmétiques](#)

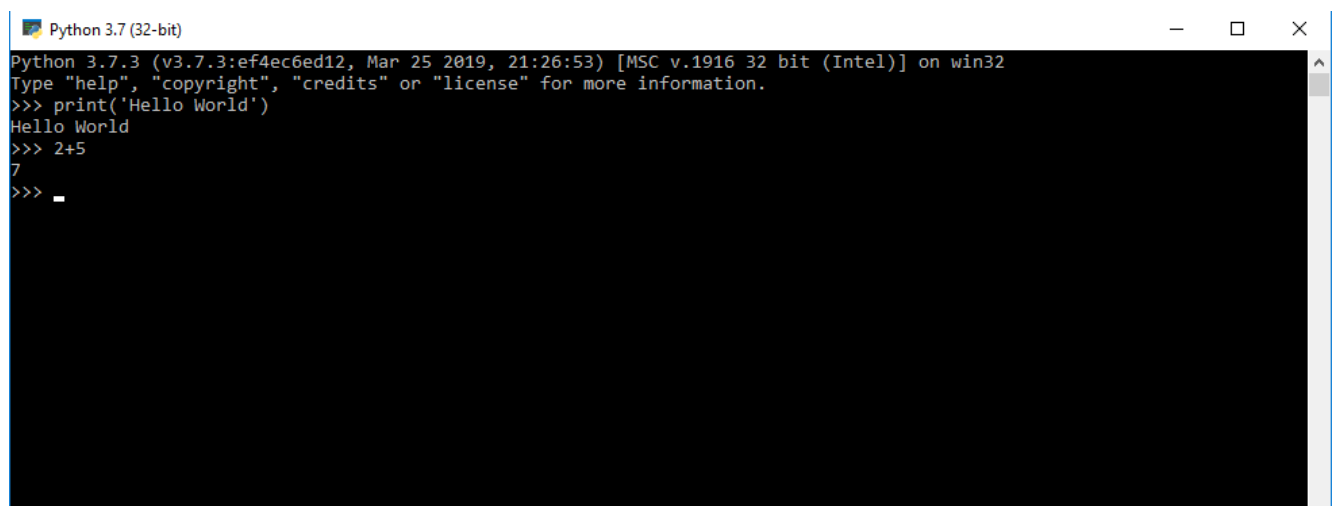
Introduction

Python a été créé en 1990 par le programmeur néerlandais Guido Van Rossum. Ce langage de programmation orienté objet est un langage qui a été conçu pour être facile à apprendre. Sa logique et facilité de lecture est un grand avantage. Grâce à cela, il est très accessible aux personnes débutantes dans le monde de la programmation.

Console

Python est un langage qui peut être exécuté de différentes façons. Une d'entre elles est l'exécution dans la console (Linux ou Windows). Une invite `>>>` (prompt) est affichée afin d'entrer le code souhaité et recevoir une réponse instantanée. Cette méthode est utilisée pour exécuter des commandes basiques ou faire des tests. Voici quelques captures d'écrans avec des exemples:

- Exemple 1: `print('Hello World')`
- Exemple 2: `2+5`



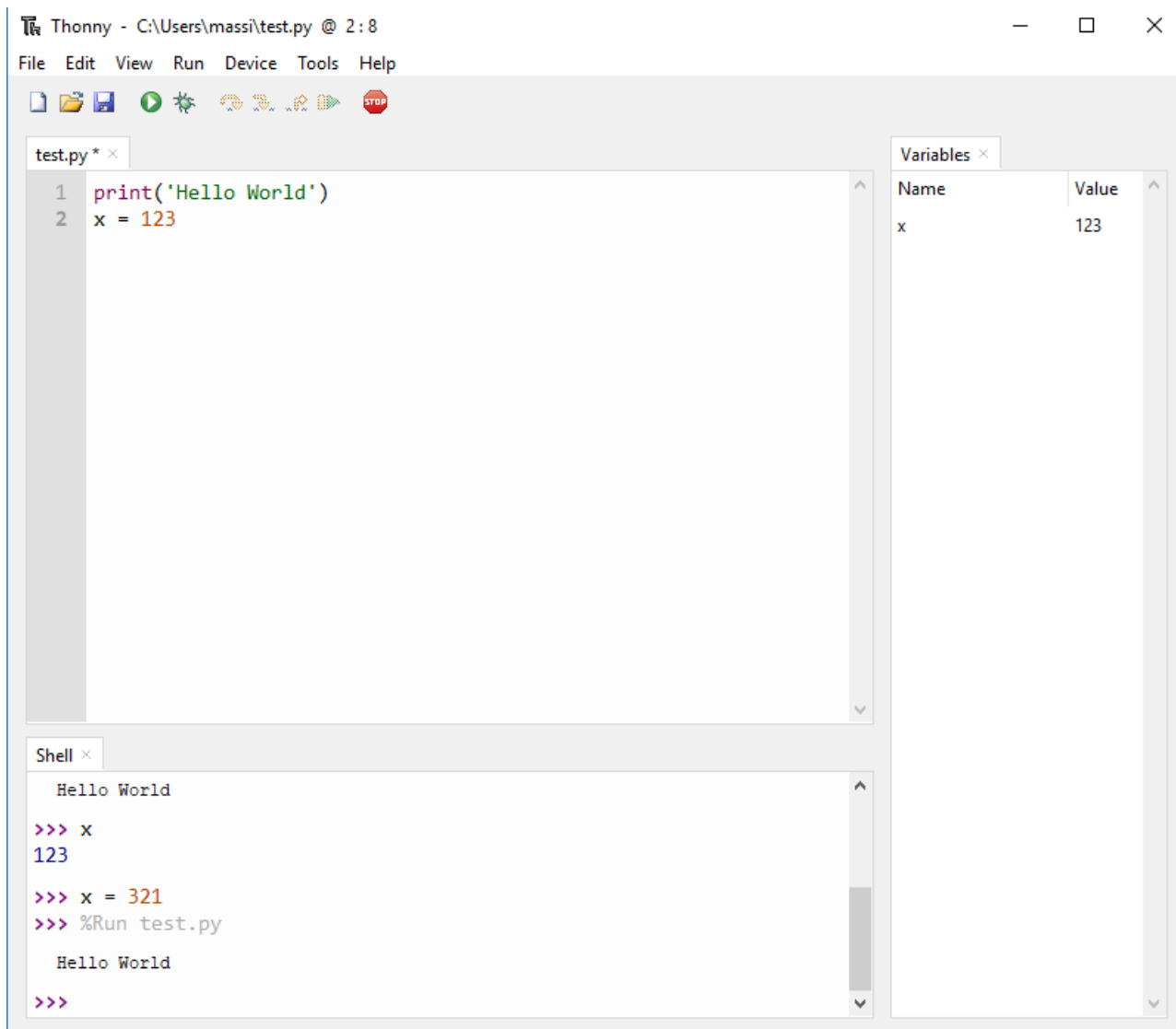
```
Python 3.7 (32-bit)
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello World')
Hello World
>>> 2+5
7
>>> _
```

Dans Jupyter Notebook, `In[i]` serait notre prompt `(>>>)` et `Out[i]` la console.

Execution d'un script

Python peut être utilisé comme un langage de script. Il peut être enregistré dans un fichier `.py`. Et exécuté dans un programme dans un ordre précis. Le programme l'interprète et donne le résultat.

Thonny est un des ces programmes compatibles avec Python.



Nous pouvons voir la console, l'état des variables et le contenu du fichier `test.py`

Le langage

Python est un langage qui fonctionne avec des valeurs. Chacune de ces valeurs ont un `type`. Que ce soit des lettres, des nombres entiers ou des nombres à virgules, Python les définit d'une manière spécifique:

- `str` : Ce type est attribué aux valeurs se formant grâce à une chaîne des caractères.
- `int` : Ce type est attribué aux nombres entiers.

- **float** : Ce type est accordé aux nombres à virgules.

Les chaînes des caractères (string) sont de type **str**

In []:

```
type('Hello, World!')
```

Les nombres entiers (integers) appartiennent au type **int**

In [5]:

```
type(2)
```

Out[5]:

int

Les nombres à virgule (floating-point numbers) leur est accordé le type **float**

In [2]:

```
import math  
type(math.pi)
```

Out[2]:

float

Néanmoins, d'autres types sont aussi présents dans Python. Le type **bool** est un type qui se caractérise pour avoir deux valeurs définies: `True` ou `False`. La plupart du temps ces valeurs sont utilisées dans les expressions booléennes; pour définir un état ou des comparaisons.

In [12]:

```
type(True), type(False)
```

Out[12]:

(bool, bool)

In [13]:

```
5 == 5, 6 == 5
```

Out[13]:

(True, False)

In [15]:

```
5 != 6, 5 > 6, 5 < 6, 5 >= 6, 5 <= 6
```

Out[15]:

(True, False, True, False, True)

Les types peuvent être également utilisés comme fonctions.

`int()` Transforme une chaîne de caractères composée de chiffres comme un **int**

In [17]:

```
int('32')
```

```
Out[17]:
```

```
32
```

`float()` Transforme des entiers et des chaînes de caractères en **float**

```
In [ ]:
```

```
float(32)
```

```
In [ ]:
```

```
float('35')
```

`str()` Convertit son argument en un chaîne.

```
In [1]:
```

```
str(32)
```

```
Out[1]:
```

```
'32'
```

```
In [2]:
```

```
str(35.5415)
```

```
Out[2]:
```

```
'35.5415'
```

La fonction `input()`

Cette fonction a comme but de laisser l'opportunité à l'utilisateur d'utiliser le clavier et la souris pour compléter des informations. L'utilisateur peut alors lui-même assigner des variables.

```
In [ ]:
```

```
question = "Age? "  
reponse = raw_input(question)
```

```
In [7]:
```

```
reponse
```

```
Out[7]:
```

```
'21'
```

Il est important de souligner que dans la langue française il existe l'apostrophe (`'`) ceci est un problème, car Python va l'interpréter comme la fermeture ou ouverture d'une chaîne des caractères. C'est pour cela qu'on utilise `"""` .

Affection

Dans Python, il est possible de créer nos propres variables et leur donner une valeur précise.

Les programmeurs doivent donner des noms à leurs variables. Ces noms doivent pouvoir expliquer à quoi ces variables servent.

Ces variables doivent suivre certains critères:

- Il ne faut jamais commencer une variable par un chiffre. Les caractères spéciaux ne sont pas non plus admis.
- Dans le but d'avoir des variables lisibles. Les programmeurs utilisent les `_` à la place des espaces.
- On ne peut pas utiliser les mots clés réservés par Python

Exemple d'une bonne affectation:

In [3]:

```
message = 'Ceci est un message ayant comme type: str'
x = 5
y = 2.5
```

Exemple d'affectations erronées.

In [31]:

```
76trombones = 'grande parade'
```

```
File "<ipython-input-31-flae388c90a0>", line 1
76trombones = 'grande parade'
      ^
```

SyntaxError: invalid syntax

In [1]:

```
plus@ = 1000000 # Caractère '@' non admis.
```

```
File "<ipython-input-1-bd97e817a9f8>", line 1
plus@ = 1000000 # Caractère '@' non admis.
      ^
```

SyntaxError: invalid syntax

In [2]:

```
class = 'Théorie avancée de la fermentation alcoolique'
```

```
File "<ipython-input-2-112b05e2b14c>", line 1
class = 'Théorie avancée de la fermentation alcoolique'
      ^
```

SyntaxError: invalid syntax

Voici une liste des mots réservés de Python:

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
True	del	global	not	with
and	elif	if	or	yield
as	else	import	pass	
assert	break	in	raise	

Ces valeurs peuvent être utilisées et nous pouvons faire des opérations avec celles-ci. Afin d'afficher les variables nous pouvons les imprimer avec `print()` ou directement sur l'entrée.

In [6]:

```
print(message)
print(x)
print(y)
```

```
Ceci est un message ayant comme type: str
5
```

2.5

In [5]:

```
x
```

Out[5]:

```
5
```

Opérations arithmétiques

Grâce à Python il est également possible d'effectuer des opérations arithmétiques. Or vous ne pouvez pas les appliquer pour les chaînes des caractères.

Il y a deux exceptions: `+` et `*`. Vous pouvez enchaîner des chaînes des caractères grâce à l'addition et vous pouvez les répéter grâce à la multiplication.

In [9]:

```
x+y, x*y, x/y, x**y
```

Out[9]:

```
(7.5, 12.5, 2.0, 55.90169943749474)
```

Python respecte l'ordre des opérations. `40 * 2 + 5 != 40 * (2 + 5)`

In [1]:

```
40 * 2 + 5
```

Out[1]:

```
85
```

In [2]:

```
40 * (2 + 5)
```

Out[2]:

```
280
```

Avec les chaînes des caractères.

In [12]:

```
premier = 'plate'  
second = 'forme'  
premier + second
```

Out[12]:

```
'plateforme'
```

In [13]:

```
premier*5
```

Out[13]:

```
'plateplateplateplateplate'
```

Comme vous avez pu constater précédemment. J'ai écrit la chose suivante: `import math`. Grâce à cela, il m'a été possible d'importer un module. Dans ce cas, le module `math` rassemble toutes les opérations possibles pour effectuer des opérations plus compliquées.

In [3]:

```
math.sqrt(2) / 2 #Racine carrée de 2 divisée par 2
```

Out[3]:

```
0.7071067811865476
```

Vous pouvez apprendre plus sur le module appliquant: `help(math)`

In []:

```
help(math)
```