Introduction à Python

February 15, 2021

1 Introduction à Python

- Voir les matériels pédagogiques sur Moodle (cours/livre, memento, puis le sujet du TP SGF bientôt).
- Objectif : courte introduction du langage utilisé, ce n'est pas un cours Python.
- Python : impératif, interprété, typage dynamique, identation significatif.
- Python 2 vs Python 3.
- Notebook pour la présentation, mais éditeurs proposés, code exécutable, et interpréteur disponible.

1.1 Expressions

[1]:
```
2020  # int
```

[1]: 2020

[2]:
```
"Hello world!"  # str
```

[2]: 'Hello world!'

[3]:
```
True  # bool
```

[3]: True

[4]:
```
1 + 2 - 3  # int
```

[4]: 0

1.2 Variables

Documentation des types natifs

[5]:
```
hello_world_str = "Hello world!"

print(hello_world_str)
print(type(hello_world_str))
```

```
Hello world!
<class 'str'>
```

[6]:
```
twenty_twenty = 2020

print(twenty_twenty)
print(type(twenty_twenty))
```

```
2020
<class 'int'>
```

[7]:
```
true_value = True

print(true_value)
print(type(true_value))
```

```
True
<class 'bool'>
```

Typage dynamique fort

[8]:
```
# Exemple de typage dynamique
twenty_twenty = 2020

print(twenty_twenty)
print(type(twenty_twenty))
print()

twenty_twenty = False

print(twenty_twenty)
print(type(twenty_twenty))
```

```
2020
<class 'int'>

False
<class 'bool'>
```

1.3 Données structurées

Listes

[9]:
```
liste_entiers = [1, 5, 3, 7, 1, 2, 4]

print(liste_entiers)
print(type(liste_entiers))
```

```
[1, 5, 3, 7, 1, 2, 4]
<class 'list'>
```

[10]:
```
liste_entiers.append(8)
```

```
print(liste_entiers)
```

```
[1, 5, 3, 7, 1, 2, 4, 8]
```

[11]:
```
liste_entiers.insert(2, 9)

print(liste_entiers)
```

```
[1, 5, 9, 3, 7, 1, 2, 4, 8]
```

[12]:
```
second_value = liste_entiers[1]

print(second_value)
```

```
5
```

[13]:
```
liste_entiers[3] = 10

print(liste_entiers)
```

```
[1, 5, 9, 10, 7, 1, 2, 4, 8]
```

[49]:
```
# Fonction incluse : sorted
print(sorted(liste_entiers))
```

```
[1, 1, 2, 4, 5, 7, 8, 9, 10]
```

[51]:
```
# Possible, mais attention !
liste_bizarre = [1, 'Bonjour', False]

print(liste_bizarre)
print(type(liste_bizarre))
```

```
[1, 'Bonjour', False]
<class 'list'>
```

Tuples : immutables

[14]:
```
tuple_entiers = (5, 3, 7)

print(tuple_entiers)
print(type(tuple_entiers))
```

```
(5, 3, 7)
<class 'tuple'>
```

[15]:
```
first_value = tuple_entiers[0]

print(first_value)
print(type(first_value))
```

```
5
<class 'int'>
```

[16]:
```
tuple_entiers[0] = 10
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-16-271b40af42c1> in <module>
----> 1 tuple_entiers[0] = 10

TypeError: 'tuple' object does not support item assignment
```

Dictionnaires

[18]:
```
martin_dict = {'nom': 'Martin', 'age': 18, 'inscrit': True}

print(martin_dict)
print(type(martin_dict))
```

```
{'nom': 'Martin', 'age': 18, 'inscrit': True}
<class 'dict'>
```

[19]:
```
nom_martin = martin_dict['nom']

print(nom_martin)
print(type(nom_martin))
```

```
Martin
<class 'str'>
```

[20]:
```
# Ajout d'une valeur
martin_dict['ville'] = 'Rennes'

print(martin_dict)
```

```
{'nom': 'Martin', 'age': 18, 'inscrit': True, 'ville': 'Rennes'}
```

[21]:
```
# Suppression d'une valeur
del martin_dict['inscrit']

print(martin_dict)
```

```
{'nom': 'Martin', 'age': 18, 'ville': 'Rennes'}
```

Fonctions

Hello

[22]:
```
# Définition de la procédure hello_proc
def hello_proc(name):
```

```
    print('Hello %s!' % name)
```

[23]:
```
# Appel de la procédure hello_proc
hello_proc('world')
```

```
Hello world!
```

Factorielle

[24]:
```
# Définition de la fonction fact
def fact(n):
    if n < 1:
        return 1
    else:
        return n * fact(n-1)
```

[25]:
```
# Appel de la fonction fact
fact(5)
```

[25]: 120

1.4 Conditionnelle

[26]:
```
def max_10(x):
    if x > 10:
        return 10
    else:
        return x
```

[27]:
```
max_10(5)
```

[27]: 5

[28]:
```
max_10(15)
```

[28]: 10

Valeurs équivalentes à False

[29]:
```
# Les valeurs évaluées à false
def boolean_equivalent(value):
    if value:
        return True
    else:
        return False
```

[30]:
```
boolean_equivalent(0)
```

[30]: False

[31]:
```
boolean_equivalent('')
```

[31]: False

[32]:
```
boolean_equivalent([])
```

[32]: False

1.5 Boucles

Boucle for

[33]:
```
for x in liste_entiers:
    print(x + 5)
```

```
6
10
14
15
12
6
7
9
13
```

Boucle while

[35]:
```
i = 10
while i > 0:
    print('Décollage dans %d secondes.' % i)
    i -= 1  # Équivalent à i = i - 1
```

```
Décollage dans 10 secondes.
Décollage dans 9 secondes.
Décollage dans 8 secondes.
Décollage dans 7 secondes.
Décollage dans 6 secondes.
Décollage dans 5 secondes.
Décollage dans 4 secondes.
Décollage dans 3 secondes.
Décollage dans 2 secondes.
Décollage dans 1 secondes.
```

1.6 Entrées sorties

Formattage des sorties textuelles - `%s` : chaîne de caractère (str) - `%d` : entier (int) - `%f` : flottant (float) avec `%.2f` l'affichage de deux chiffres significatifs.

[36]:
```
# Exemple d'affichage avec formattage
print(martin_dict)

print('Bonjour %s de %s âgé de %d ans.'
      % (martin_dict['nom'], martin_dict['ville'], martin_dict['age']))

# Notez que ce qui se trouve après le '%' séparateur est un tuple.
```

```
{'nom': 'Martin', 'age': 18, 'ville': 'Rennes'}
Bonjour Martin de Rennes âgé de 18 ans.
```

[37]:
```
valeur_flottante = 5.168713567

print('Valeur saisie : %f' % valeur_flottante)
print('Équivalent entière : %d' % valeur_flottante)
print('Deux chiffres significatifs : %.2f' % valeur_flottante)
```

```
Valeur saisie : 5.168714
Équivalent entière : 5
Deux chiffres significatifs : 5.17
```

Saisie utilisateur

[38]:
```
input('Votre nom ? ')
```

```
Votre nom ? Test
```

[38]:
```
'Test'
```

[39]:
```
your_age = input('Votre age ? ')

print(type(your_age))

if your_age >= 18:
    print('Vous êtes majeur.')
else:
    print('Vous êtes mineur.')
```

```
Votre age ? 20
<class 'str'>
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-39-27178ac8af5c> in <module>
      3 print(type(your_age))
      4
----> 5 if your_age >= 18:
      6     print('Vous êtes majeur.')
      7 else:
```

```
TypeError: '>=' not supported between instances of 'str' and 'int'
```

[40]:
```
your_age = input('Votre age ? ')

your_age = int(your_age)
print(type(your_age))

if your_age >= 18:
    print('Vous êtes majeur.')
else:
    print('Vous êtes mineur.')
```

```
Votre age ? 18
<class 'int'>
Vous êtes majeur.
```

1.7 Importation

[41]:
```
import math
```

[42]:
```
math.sqrt(9)
```

[42]:
```
3.0
```

1.8 Bonus : typage

Uniquement une aide à la lecture, aucune vérification réelle ne sera exécutée !

[43]:
```
def sum_is_10(x: int, y: int) -> bool:
    return (x + y) == 10
```

[44]:
```
sum_is_10(5, 5)
```

[44]:
```
True
```

[45]:
```
sum_is_10('Test', 'Test')
```

[45]:
```
False
```

[46]:
```
def multiplication_et_addition(x: int, y: int, z: int) -> int:
    return (x * y) + z
```

[47]:
```
multiplication_et_addition(5, 5, 10)
```

[47]:
```
35
```

[48]:
```
multiplication_et_addition('Na ', 5, 'Batmaaan!')
```

[48]:
```
'Na Na Na Na Na Batmaaan!'
```

1.9 Bonus : commentaires

Plusieurs formats disponibles.

[52]:
```python
def multiplication_et_addition(x: int, y: int, z: int) -> int:
    """Multiplies x and y then adds z to the result.

    Args:
        x: The integer to multiply y with.
        y: The integer to multiply x with.
        z: The integer to adds to the multiplication result.

    Returns:
        The addition of z to the multiplication of x and y.
    """
    return (x * y) + z
```