

Python

Les bases

Introduction
Variables
Pro tips
Fonctions
Conditions
Boucles

1. Python

Introduction

Introduction

- Python : presque 30 ans !
 - Robuste, grosse communauté
- + de 85 000 modules via PyPi
 - RESTful, BDD, analyse de data...
- Multiplateforme & open source

Introduction

- On peut faire tout type d'application :
 - WebApp
 - Client léger
 - Client lourd
 - Jeu vidéo
- Langage moins verbeux
 - Pas d'accolades mais de l'indentation
 - Syntaxe plus légère

2. Python

Variables & protips

Variables

- Comme dans d'autres langages, il existe de nombreux types de données :
 - String
 - Int
 - Float
 - ...
- Pour comparer, python n'a besoin que de "=". Il n'existe pas de "==="

Variables

- Petit rappel sur différents langages :
 - **PHP** : typage faible dynamique
 - PHP en roue libre sur les variables
 - **Java, C** : typage fort statique
 - Ultra strict : un int reste un int !
 - **Python** : typage fort dynamique
 - Mitigé : Ok, mon int devient float, mais par contre on fait pas n'importe quoi lorsqu'on travaille avec différents types !

Variables

- Petit rappel sur différents langages :
 - **PHP** : typage faible dynamique
 - PHP en roue libre sur les variables
 - **Java, C** : typage fort statique
 - Ultra strict : un int reste un int !
 - **Python** : typage fort dynamique
 - Mitigé : Ok, mon int devient float, mais par contre on fait pas n'importe quoi lorsqu'on travaille avec différents types !

Variables

```
10  
12.5  
J'suis une chaîne de caractère  
Je suis une chaîne  
de caractères  
multiligne !
```

```
# coding: utf-8  
  
maVar = 10  
print maVar  
  
maVar = 12.5  
print maVar  
  
maVar = "J\'suis une chaîne de caractère"  
print maVar  
  
maVar = """Je suis une chaîne  
de caractères  
multiligne !"""  
print maVar
```

Variables

```
11
a = a
b = b
a = b
b = a
10
10
15
```

```
# PRO TIPS #
|
# coding: utf-8

# Je suis commentaire monoligne

"""
Ceci est un commentaire
multiligne. En vrai, c'est
une chaîne de caractère. Mais
sans instruction, elle est
totalement ignorée lors de
l'exécution du programme
"""

maVar = 10
maVar += 1 # Incrémentation
print maVar
```

```
maVar = 10
maVar += 1 # Incrémentation
print maVar

# Permutation
a = "a"
b = "b"
print "a = " + a
print "b = " + b
a,b = b,a
print "a = " + a
print "b = " + b

# Multi-affectation
a = b = 10
print a
print b

# Le backslash pour couper une instruction
print 10 + 10 \
- 5 # Ici, l'opération correspond à 10 + 10 - 5 donc 15.
```

Variables - Scope

- Variables **locales** :
 - Si définies avant l'appel d'une fonction, elles seront accessibles en lecture seule
 - Si définies dans une fonction -> portée de fonction
- Variables **globales** :
 - Mot-clé : **global** maVar
 - **Le moins de variable globale possible !**

3. Python

Fonctions

Fonctions

- Une fonction en Python ressemble à ceci :

```
def hello():  
    print "Hello World !"  
  
hello();|
```

- “def” est le mot-clé pour définir une fonction
- “:” est le “point d’entrée” de la fonction
- On appelle ensuite simplement la fonction comme dans la plupart des autres langages, via son nom

Fonctions

- Quatre fonctions très utiles :
 - **print**("str") -> Permet d'afficher du contenu + saut de ligne
 - **type**(obj) -> Permet de connaître le type d'une variable
 - **isinstance**(obj, class) -> **TRUE** si votre variable(obj) est du type(class) passé en paramètre
 - **input**("str") -> Permet de proposer un prompt à l'utilisateur

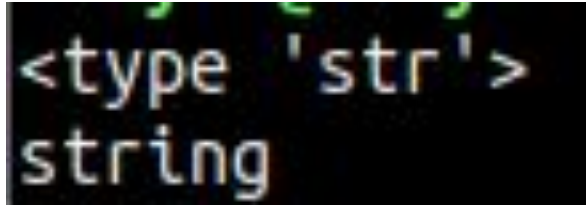
Fonctions

```
maVar = "a"

print type(maVar)

# isinstance(obj, class)
# obj => variable
# class => type (int, float, str...)

if (isinstance(maVar, str)):
    print "string"
```



<type 'str'>
string

A terminal window showing the output of the Python code. The first line displays the type object <type 'str'>, and the second line displays the string value 'string'.

4. Python

Conditions

Conditions

- Prédicat : c'est la condition de votre "if"
 - Exemple : `if a == 5: ...` <- Le prédicat est "a == 5"
- Comme dans les autres langages, retourne true ou false
- Liste des opérateurs :
 - `<`, `>`, `<=`, `>=`, `==`, `!=`

Conditions

- Prédicat : c'est la condition de votre "if"
 - Exemple : `if a == 5: ...` <- Le prédicat est "a == 5"
- Comme dans les autres langages, retourne true ou false
- Liste des opérateurs :
 - <, >, <=, >=, ==, !=

Conditions

- Trois mots-clés de condition :
 - and (si, a ET b)
 - or (si, a OU b)
 - is not (si a N'EST PAS b)
- Il n'y a pas de SWITCH ... CASE en Python, même si on peut l'implémenter en trichant

Conditions

```
nbr = 4

if nbr > 0:
    print("higher than zero")
elif nbr < 0:
    print("lower than zero")
else:
    print("equals to zero")
```

5. Python

Boucles

Boucles

For, while et enumerate. C'est tout !

- Comme Python se veut abordable et le plus simple possible, beaucoup de mots-clés verbeux ont été enlevés. Parmi eux : do...while, foreach, switch...case

Boucles

```
# WHILE
nbr = 0
while (nbr < 10) :
    print nbr
    nbr += 1

# FOR
string = "Hello World !"
for char in string:
    print(char)
```

Boucles

- **Break** -> Stop la boucle
- **Continue** -> Reprend

au début de la boucle
sans exécuter la suite

```
# CONTINUE keyword
nbr = 0
while (nbr < 10) :
    if(nbr % 2 != 0):
        print nbr
        nbr += 1
        continue
    print "Un nombre pair !"
    nbr += 1
```

```
# BREAK keyword
nbr = 0
while (nbr < 10) :
    if(nbr % 2 != 0):
        print nbr
        nbr += 1
        print "Impair !"
        break
    print "Un nombre pair !"
    nbr += 1
```


Boucles

- **Enumerate** permet d'afficher une liste sous forme clé => valeur

```
# ENUMERATE  
my_list = ['apple', 'pear', 'orange', 'banana']  
for c, value in enumerate(my_list):  
    print(c, value)
```

```
(0, 'apple')  
(1, 'pear')  
(2, 'orange')  
(3, 'banana')
```