Programmation Orientée Objet en Python

#0 Programmation python

par David Albert

GEMA
GRANDES ÉCOLES DES MÉTIERS D'AVENIR

2023

Table des matières

01

Type de données

Les booléens

Valeurs : `True`, `False`

PEP 483 : `bool`

Représentation en machine

False	True
0	1

Opérateurs booléens

`and`, `or`, `not`

Les entiers

Valeurs : ..., -3, -2, -1, 0, 1, 2, 3, 4, ...

PEP 483 : `int`

Représentation en machine
La représentation des entiers en machine correspond sur la représentation des nombres entiers en base 2.

Opérateurs de comparaison

- égalité : `==`
- inégalité : `!=`
- infériorité : `<`, `<=`
- supériorité : `>`, `>=`

> Les opérateurs de comparaisons retournent un booléen (`True` ou `False`).

Les flottants

Valeurs : Permet de s'approcher d'une représentation des **nombres réels** (bien qu'incomplète).

PEP 483 : `float`

Représentation en machine
https://fr.wikipedia.org/wiki/Virgule_flottante

Opérateurs de comparaison

- égalité : `==`
- inégalité : `!=`
- infériorité : `<`, `<=`
- supériorité : `>`, `>=`

> Les opérateurs de comparaisons retournent un booléen (`True` ou `False`).

Les caractères

Valeurs : `'A'`, `'B'`, `'C'`, `';'`, `'!'`, `'0'`, `'1'`, ...

Représentation en machine

- Code ASCII
- unicode

> ! Le type caractère unique n'existe pas en tant que tel en python. Chaque caractère déclaré est interpréter par python comme une chaîne de caractère (cf ci-dessous).

Table ASCII

ASCII Code: Character to Binary

0	0011 0000	O	0100 1111	m	0110 1101
1	0011 0001	P	0101 0000	n	0110 1110
2	0011 0010	Q	0101 0001	o	0110 1111
3	0011 0011	R	0101 0010	p	0111 0000
4	0011 0100	S	0101 0011	q	0111 0001
5	0011 0101	T	0101 0100	r	0111 0010
6	0011 0110	U	0101 0101	s	0111 0011
7	0011 0111	V	0101 0110	t	0111 0100
8	0011 1000	W	0101 0111	u	0111 0101
9	0011 1001	X	0101 1000	v	0111 0110
A	0100 0001	Y	0101 1001	w	0111 0111
B	0100 0010	Z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	.	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	;	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	I	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(	0010 1000
N	0100 1110	l	0110 1100	)	0010 1001
				space	0010 0000

Les chaînes de caractères

Définition : Une liste de caractères.

Exemples :
`"Brice"`, `"la vie est belle !"`, `"#0$£ù%&-"`

PEP 483 : `string`

Opérateurs de bases

- assigner une variable : `my_str = "bidule"`
- accès au 3ème caractère : `my_str[2]`
- accès au dernier caractère : `my_str[-1]`
- sous-chaîne : `my_str[:2]`, `my_str[1:]`, `my_str[1:2]`, `my_str[::2]`
- longueur de la chaîne : `len(my_str)`
- changer la casse : `my_str.capitalize()`, `my_str.upper()`, `my_str.lower()`
- vérifier la casse : `my_str.islower()`, `my_str.isupper()`

02

Types composites

Les listes

Définition : Une liste ordonnée de données.

Exemples :
`[0, 21, 13, 7, 100]`, `[]`, `[True, True, False]`, ...

PEP 483 : `list`

Opérateurs de bases

- assigner une variable : `l = []`
- accès au 3ème caractère : `l[2]`
- accès au dernier caractère : `l[-1]`
- sous-listes : `l[:2]`, `l[1:]`, `l[1:2]`, `l[::2]`
- ajout d'un élément à la fin : `l.append(3)`
- supprimer du 3ème élément : `l.remove(2)`
- tri d'une liste : `sorted(l)`

Les dictionnaires

Définition : Un dictionnaire est une structure de données qui assimile des clés à des valeurs.

Exemples :
`{"nom" : "Fred", "age" : 20 }`, `{}`, `{True : "eat", False : [0, 1, 2]}`, ...

PEP 483 : `dict`

Opérateurs de bases

- assigner une variable : `d = {}`
- accès à une valeur (depuis sa clé) : `d[key]`
- ajout d'un élément à la fin : `d[key] = value`
- supprimer du 3ème élément : `del d[key]`
- accès aux clés : `d.keys()`
- accès aux valeurs : `d.values()`

03

Variables

Qu'est-ce qu'une variable ?

Définition

En informatique, les variables sont des symboles qui associent un nom (**l'identifiant**) à une **valeur**. Dans la plupart des langages, **les variables peuvent changer de valeur au cours du temps**.

De plus, **les variables ont un type** de valeur.

En python, la déclaration d'une variable se fait avec l'opérateur d'allocation `=`

Exemples :

```python
prenom = "Jonathan"          # variable de type chaîne de caractères (str)
age = 23                     # variable de type entier (int)
moyenne = 10.8               # variable de type flottant (float)
notes = [16, 12, 13, 9]      # variable de type liste d'entiers (list[int])
```

Portée des variables

Portée des variables (globale / locale)

```
var1 = 10 # var1 est globale

def foo(var2): # var2 est locale
    var3 = 30 # var3 est locale
    if (var3 > 0):
        var4 = 40
        print(var1, var2, var3, var4) # 10, 20, 30, 40

    print(var1, var2, var3, var4) # 10, 20, 30, Erreur

foo(20)
print(var1, var2, var3, var4) # 10, Erreur, Erreur, Erreur
```

Portée des variables

Visibilité des variables

```
def foo():

    level1 = 10

    def bar():
        level2 = 20

        def tutu():
            level3 = 30

            print("from tutu:", level1, level2, level3)

        print("from bar: ", level1, level2) # level3 NOT visible
        tutu()

    print("from foo: ", level1) # level2 or level3 NOT visible here
    bar()
```

04

Boucle et conditions

Conditions

1. if / else

```
if (cond):
    # code si vrai
else:
    # code si faux
```

2. if / else if / else

```
if (cond):
    # code si vrai
elif (cond2):
    # code si vrai
else:
    # code si faux
```

La notation **cond** dans les exemples ci-contre représente une expression quelconque qui renvoit un booléen.

Exemples :

- `if True:`
- `if is_winner:`
- `if (age < 30):`
- `if (age < 20 and name == "Mathéo"):`
- `if "John" in names:`

Boucles

1. for

```
for i in range(10):
    # ...
    # code ici
    # ...
```

2. while

```
while (cond):
    # ...
    # code ici
    # ...
```

05

Fonctions

Déclarer une fonction

Une fonction est définit grâce :

- au mot-clé `def` en python
- à un identifiant
- à des paramètres (optionnels)
- à un type de retour et une valeur de retour

Exemple 1: Calcul de l'aire d'un rectangle

```python
def aire_rectangle(longueur: int, largeur: int) -> int:
    return longueur * largeur
```

Exemple 2: Affichage des données d'une classe

```python
def display_user_data(user: User) -> None:
    print("--- User Data ---")
    print("name=", user.name)
    print("age=", user.age)
```

> **i** Une fonction qui ne retourne aucune valeur est généralement appelée **procédure**.

06

Packages & modules

Réutiliser du code

En plus d'être simple et intuitif, le langage de programmation Python possède l'avantage d'avoir une très large communauté.

Ainsi, pas besoin de tout réimplémenter. Si vous avez besoin d'une fonctionnalité, quelqu'un l'aura probablement déjà implémenté et partagé avant vous.

Un **module** est une collection de fonctions et méthodes qui peuvent être réutilisés dans une autre partie du code.

Un **package** est un ensemble de modules munis d'une documentation et conçus pour des besoins spécifiques.

Exemples de package python : numpy, pandas, plotly, Django, Flask, PyTorch, Scikit Learn, ...

Importer des fonctions & méthodes

Importer tout un module

```python
# Importer la library math
import math

# Retourne la racine carré de 9
print(math.sqrt(9))
```

Importer une fonction spécifique d'un module

```python
# Importer une fonction de la library math
from math import sqrt

# Retourne la racine carré de 9
print(sqrt(9))
```

Notes de fin

Le cours actuel n'est pas un cours de programmation en python mais un cours de programmation orientée objet (POO avec python).

Ce document est un bref récapitulatif de certaines notions qui doivent être maîtrisées pour la bonne compréhension du cours de POO.

Pour revoir les bases du langage python, vous pouvez suivre le cours OpenClassroom ci-dessous (gratuit) :

https://openclassrooms.com/fr/courses/7168871-apprenez-les-bases-du-langage-python

Pour approfondir le cours actuel de programmation orientée objet avec python, vous pouvez suivre le cours OpenClassroom ci-dessous (gratuit) :

https://openclassrooms.com/fr/courses/7150616-apprenez-la-programmation-orientee-objet-avec-python