Langage Python

Aziz DAROUICHI

Chapitre 1: Introduction à Python

Plan

- Bref historique
- Principales caractéristiques de Python
- Choix de Python
- Interpréteur de commandes Python
- Variables
- Les types de données en Python
- Python et le typage dynamique
- Opérateurs et expressions
- Fonctions mathématiques : le module math
- Commentaires
- Fonctions standards d'entrée/ sortie
- Structures conditionnelles
- Structures répétitives (boucles)
- Branchement inconditionnel
- Références

Bref historique

- **1989:** Le langage Python a été développé aux Pays-Bas par ***Guido Von Rossum***.
- **1991:** La première version est sortie
- **2001: Python Software Foundation** est créée
- **19/12/2017:** Sortie de la version: **Python 3.6.4**
- **04/02/2018:** Sortie des versions: **Python 3.5.5 et Python 3.4.8**
- **02/08/2018:** Sortie des versions: **Python 3.5.6 et Python 3.4.9**
- **20/10/2018:** Sortie des versions: **Python 3.7.1 et Python 3.6.7**
- **24/12/2018:** Dernières versions: **Python 3.7.2 et Python 3.6.8**
- Site officiel : **www.python.org**

Principales caractéristiques de Python

- **Langage Open Source:** libre et gratuit.
- **Portable** : le même code source peut être exécuté sous tous les systèmes d'exploitation (Windows, linux, Mac OS, . . .).
- **Simplicité du langage :** Syntaxe claire et cohérente.
- **Langage Script:** Python fait partie des langages script interprétés contrairement à Java, au C/C++ qui sont des langages compilés. Ce qui lui permet d'être plus rapide au développement, de comporter moins de ligne (50% de moins). Par contre, Il est plus lent à l'exécution.
- **Extensible:** Au-delà de la multitude de librairies et de modules déjà existent, il est possible d'en développer pour ses propres besoins.

Principales caractéristiques de Python

- **Modulable:** Python permet de séparer les programmes en modules qui peuvent être réutilisés dans d'autres programmes en Python.
- **Syntaxe aisée:** La syntaxe de Python est très simple et, combinée à de nombreux types de données évolués (comme les listes, dictionnaires, tuples,...), ce qui conduit à des programmes à la fois très compacts et très lisibles. De plus, Python ne nécessite aucune déclaration de variable. Les variables sont créées lors de leur première assignation.
- **Peu d'erreurs…**

  Tout comme Java, Python possède un système d'exception qui facilite la gestion des erreurs.

Principales caractéristiques de Python

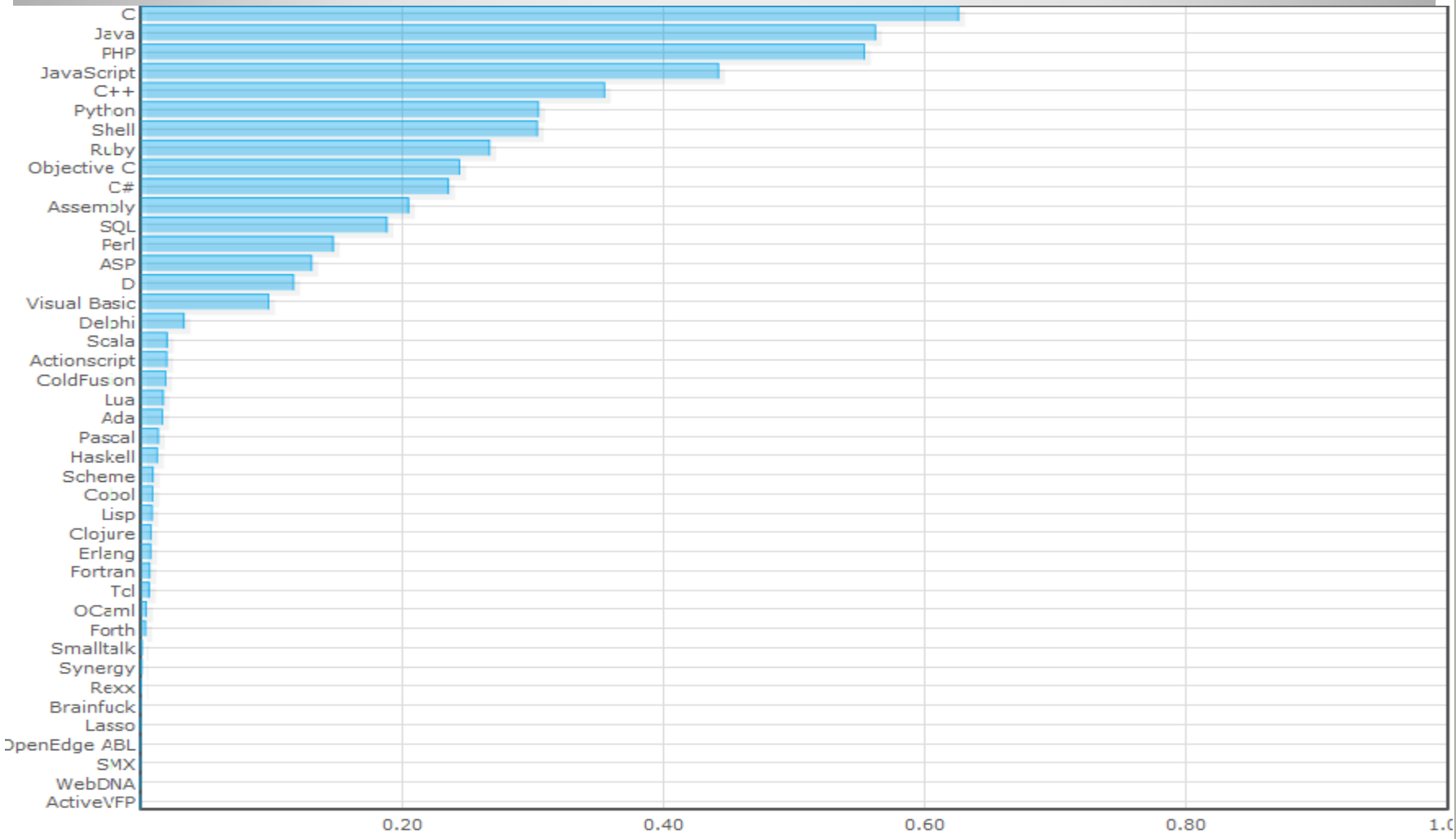
- Python est de plus en plus utilisé dans de nombreux domaines :
  - Wikipedia, Google, Yahoo,...
  - Pour le calcul scientifique (NASA,…)
  - Facebook, Instagram, Amazon, New York times,...
  - Data analysis (analyse de données)
  - Machine learning
  - Deep learning
  - Data science
  - Big data
  - Robotique
  - …

Principales caractéristiques de Python

- **Python est orienté objet .**
- **Langage interprété.**
- **Python est dynamique:**
  - *pas de déclaration de type des variables.*
  - *Le type d'une variable peut changer.*
- Deux modes d'exécution:
  - *À l'aide de l'interpréteur*
  - *Par appel au compilateur Python*
- ...

Choix de Python

Normalized Comparison 2014



Choix de Python

Normalized Comparison 2015

Language Rank	Types
1. Java	
2. C	
3. C++	
4. Python	
5. C#	
6. R	
7. PHP	
8. JavaScript	
9. Ruby	
10. Matlab	

Normalized Comparison 2016

Language Rank	Types
1. C	
2. Java	
3. Python	
4. C++	
5. R	
6. C#	
7. PHP	
8. JavaScript	
9. Ruby	
10. Go	

Choix de Python

Normalized Comparison 2017

Language Rank	Types
1. Python	
2. C	
3. Java	
4. C++	
5. C#	
6. R	
7. JavaScript	
8. PHP	
9. Go	
10. Swift	

Normalized Comparison 2018

Language Rank	Types
1. Python	
2. C++	
3. Java	
4. C	
5. C#	
6. PHP	
7. R	
8. JavaScript	
9. Go	
10. Assembly	

Choix de Python

TIOBE Index for August 2016

Aug 2016	Aug 2015	Change	Programming Language
1	1		Java
2	2		C
3	3		C++
4	4		C#
5	5		Python
6	7	^	PHP
7	9	^	JavaScript
8	8		Visual Basic .NET
9	10	^	Perl
10	12	^	Assembly language

Choix de Python

TIOBE Index for September 2017

Sep 2017	Sep 2016	Change	Programming Language
1	1		Java
2	2		C
3	3		C++
4	4		C#
5	5		Python
6	7	^	PHP
7	6	v	JavaScript
8	9	^	Visual Basic .NET
9	10	^	Perl
10	12	^	Ruby
11	18	^^	R
12	11	v	Delphi/Object Pascal
13	13		Swift

Choix de Python

TIOBE Index for August 2018

Aug 2018	Aug 2017	Change	Programming Language
1	1		Java
2	2		C
3	3		C++
4	5	^	Python
5	6	^	Visual Basic .NET
6	4	v	C#
7	7		PHP
8	8		JavaScript
9	-	^^	SQL
10	14	^^	Assembly language
11	11		Swift
12	12		Delphi/Object Pascal
13	17	^^	MATLAB
14	18	^^	Objective-C
15	10	vv	Ruby

Interpréteur de commandes Python

Lancer Python

- Démarrer>Tous les programmes>Python 3.7>Python (Command Line)
- Démarrer, puis cliquer sur Exécuter. Dans la fenêtre qui s'affiche, tapez simplement « py »
- Windows + R puis saisir py

Fermer Python

- Taper « exit() » puis appuyer sur la touche Entrée.

Variables

- Une variable est un nom donné à une zone mémoire pour stocker une valeur.
- Le nom d'une variable (Identificateur) doit respecter les règles suivantes :
  1. Il est formé d'une suite de lettres, de chiffres et de '_' ("underscore"). Le premier caractère de cette suite ne peut pas être un chiffre;
  2. Les lettres accentuées sont interdites ;
  3. Le langage Python est sensible à la casse. L'identifiant «abc» est différent de « ABC » ou « aBc ».
  4. Le nom d'une variable doit être différent des mots réservés du langage Python.

Les mots réservés de Python

- Les mots clés réservés du langage ne peuvent pas être utilisés comme nom de variables. La version 3 de Python compte 33 mots :

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

NB: Les mots réservés du langage Python doivent être écrits en minuscules.

Variables

- **Affectation:**
  - nomDeLaVariable = valeur.
- **Convention de nommage:**
  - La variable en minuscules et à remplacer les espaces éventuels par un espace souligné « _ ».

    Exemple: mon_age
  - Une autre convention utilisée consiste à passer en majuscule le premier caractère de chaque mot, à l'exception du premier mot constituant la variable.

    Exemple: monAge

Les types de données en Python

Il existe différents types de variable en Python:

- **int :** les nombres entiers : « entiers relatifs »
- **float:** les nombres flottants: « nombre à virgule »
- **bool :** les booléens : une variable de ce type ne peut prendre que 2 valeurs **True** ou **False**,

Les types de données en Python

- **str:** Les chaînes de caractères (string)
- On peut écrire une chaîne de caractères de différentes façons :
  - entre guillemets ("une chaîne de caractères") ;
  - entre apostrophes ('une chaîne de caractères') ;
  - entre triples guillemets ("""une chaîne de caractères""").

Les types de données en Python

- **complex:** les nombres complexes
- Les complexes sont définis classiquement en notation cartésienne. La partie réelle et la partie imaginaire (j) sont de type flottant.
- **Exemple:**

- 1+2j→1+2j
- (1.5+2.5j).real→1.5
- (1.5+2.5j).imag→2.5
- (1+2j).conjugate( )→1-2j

Python et le typage dynamique

- En Python, le type de la variable est déterminé au moment de l'affectation.
- Le type d'une variable peut changer au cours du temps. Ceci constitue une particularité intéressante de Python : le typage des variables sous Python est un typage dynamique.
- On peut connaitre le type courant d'une variable avec la fonction type: **type(nomDeVariable)**

```
>>>a = - 5
>>> type(a)
<class 'int'>
```

Opérateurs et expressions

Opérateurs arithmétiques

Opérateur	Description	Exemple	Résultat
+	addition	5 + 2	7
-	soustraction	5 - 2	3
*	multiplication	5 * 2	10
**	exposant	5**2	25
/	division réelle	5/2	2. 5
//	division entière	5//2	2
%	modulo	5%2	1

Opérateurs et expressions

Opérateurs de comparaison

Opérateur	Description	Exemple	Résultat
==	Égalité	3==3	True
!=	Différence	3!=3	False
>	Strictement supérieur à	3>3	False
<	Strictement inférieur à	3<5	True
>=	Supérieur ou égal	3>=3	True
<=	Inférieur ou égal	3<=2	False

Opérateurs et expressions

Opérateurs logiques

Opérateur	Description
and	Vrai si les deux arguments sont vrais, faux sinon
or	Vrai si au moins l'un des deux arguments est vrai, faux sinon
not	Négation logique

Opérateurs et expressions

Opérateur	Description
is, is not	Test d'identité
in , not in	Appartenance à une séquence

Fonctions mathématiques : le module math

- Il est possible d'utiliser les fonctions mathématiques usuelles (**sin**, **cos**, **exp**, **ln**,...) et les constantes (**π**, **e**, ...) en Python.

- Cependant ces fonctions et constantes ne sont pas directement accessibles au démarrage de Python : elles sont définies dans un « module » dédié, nommé « **math** », que l'on charge en mémoire pour les utiliser.

Fonctions mathématiques : le module math

❑ Il faut les importer dans la session en cours, via l'une des syntaxes suivantes :

- importation des seuls éléments nécessaires :

```
>>> from math import  cos, pi
>>> cos(pi)
-1.0
```

- importation de tout le module:

```
>>> from math import  *
>>> cos(pi)
-1.0
```

Commentaires

- Il est très important de bien commenter les programmes.
- Insérer des commentaires dans un programme permet d'expliquer (en langue naturelle) le comportement du programme.
- En Python, un commentaire est un texte précédé du symbole # (tout ce qui suit le # sur la même ligne est un commentaire).
- Ce texte n'est pas interprété.

```
x = 15          # on affecte la valeur 15 à la variable x
```

Fonctions standards d'entrée/ sortie

- Saisie au clavier: **input**
- Affichage à l'écran: **print**

Fonction input()

- ***Exemple*:**

```
>>> prenom=input("Entrer votre prénom :")
Entrer votre prénom :Mohamed
>>> prenom
'Mohamed'
```

- La fonction **input()** renvoie toujours une chaîne de caractères. Afin de récupérer un nombre, il est nécessaire de procéder à une conversion .
- La fonction **input()** accepte un paramètre facultatif : le message à afficher à l'utilisateur.

Fonction input()

```
>>> a=input("Entrer un nombre entier : ")
Entrer un nombre entier : 7
>>> print(3*a)
777
>>> a=int(a) #conversion d'un type str en type int
>>> print(3*a)
21
>>>
```

- La ligne a=int(a) permet de convertir le caractère '7' en un nombre entier égal à 7.
- On peut aussi écrire plus rapidement :

```
a=int(input("Entrer un nombre entier : "))
```

- La fonction **float( )** convertit la chaîne de caractères en float.

Fonction **print()**

```
print (‘Bonjour')
print ("Cela fonctionne aussi avec des guillemets")
```

Bonjour

Cela fonctionne aussi avec des guillemets

- L'utilisation de la fonction **print()** entraîne un retour chariot (passage à la ligne) obligatoire. On peut, si on ne veut pas passer à la ligne après l’écriture, ajouter l’option **end=" "**

Fonction print()

Exemple 1 :

```
print ("Bonjour. ", end=" ")
print ("Je ne suis pas à la ligne  ! ")

    Bonjour. Je ne suis pas à la ligne !
```

Exemple 2 :

```
x=5
print ("x = ",end=" ")
print (x)
```

Output : x = 5

Structures de contrôle

- **Structures conditionnelles**
- **Structures répétitives (boucles)**
- **Branchement inconditionnel**

Test simple: *l'instruction* if

***Syntaxe*:**

if *condition* :

bloc d'instructions

Si la condition est vraie le bloc d'instructions est exécuté, sinon on fait rien.

l'instruction **if**

Exemple*: *calcul de la valeur absolue d'un nombre réel.

```
X=float(input("saisir la valeur de X :"))
Val = X
if X < 0:
    Val = -X
print("valeur absolue de ", X ,"=", Val)
```

Test avec alternative: if ... else

***Syntaxe*:**

```
if condition:
        bloc d'instructions 1

else :
        bloc d'instructions 2
```

Le « bloc d'instructions 1 » *est exécuté si la condition est vraie. Sinon, c'est le* ***« bloc d'instructions 2 »*** *qui est exécuté.*

If … else

Exemple : ***maximum de deux nombres réels***

```
x = float(input("Saisir la valeur de x?"))
y = float(input("Saisir la valeur de y?"))
if x<y:
        M=y
else:
        M=x
print("le maximum de x et y est :",M)
```

Tests imbriqués

- Il existe une instruction plus efficace, qui combine le else et le if : « **elif** »

- L'instruction « **elif** » permet d'avoir le même résultat avec un code plus compact.

if … elif … else

***Syntaxe*:**

```
if  condition1:
        Instruction(s) si la condition1 est vérifiée
elif condition2:
        Instruction(s) si la condition2 est vérifiée
elif condition3:
        Instruction(s) si la condition3 est vérifiée
else:
        Instruction(s) si aucune des 3 conditions n'est vérifiée
```

if ... elif ... else

Exemple 1: ***signe d'un nombre entier***

```
a = int(input(" Saisir un entier:"))

if a > 0 :
      print(a," est positif")
elif a < 0 :
      print(a," est negatif")
else:
      print(a," est nul" )
```

Conditions composées

- La condition peut s'écrire avec des opérateurs logiques, par exemple: (m >=0 and m <= 20).
- Les lois de *De Morgan* sont fréquemment utilisées pour simplifier les tests:

1) not (A or B) == (not A) and (not B)
2) not (A and B) == (not A) or (not B)

- **Exemple:**

not(x >= 0 and x <10) == (x < 0 or x >= 10)

Boucle while

***Syntaxe*:**

```
while condition:
        bloc d'instructions
```

- Les instructions **doivent être indentées** par rapport au mot **while**.

Boucle for

***Syntaxe*:**

```
for variable in range(debut, fin, pas):
        instructions
```

- Les instructions **doivent être indentées** par rapport au mot **for**.

Boucle for

- Voici la syntaxe de la boucle **for** pour parcourir une séquence:

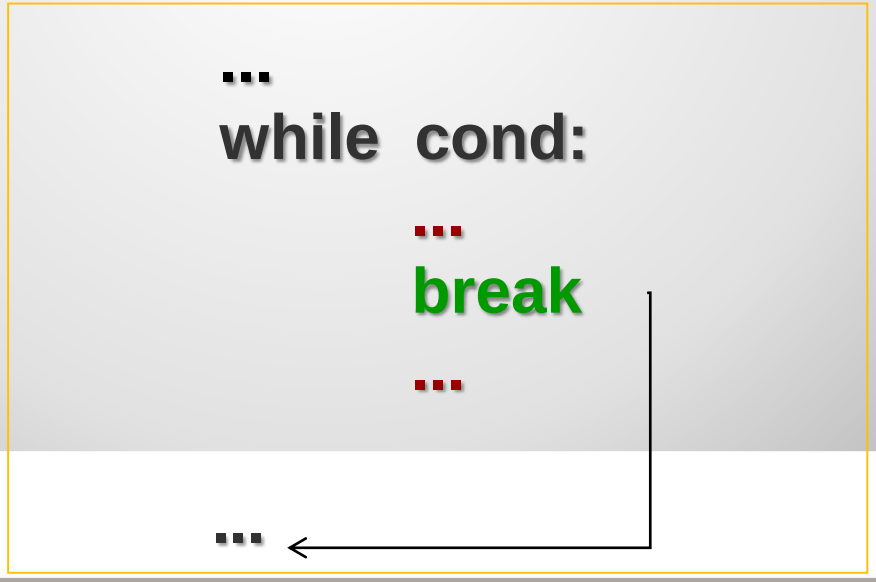
```
for element in sequence:
    instructions
```

- *element* est une variable créée par la boucle **for**, ce n'est pas à vous de l'instancier.
- Elle prend successivement chacune des valeurs figurant dans la séquence parcourue.

Branchement inconditionnel

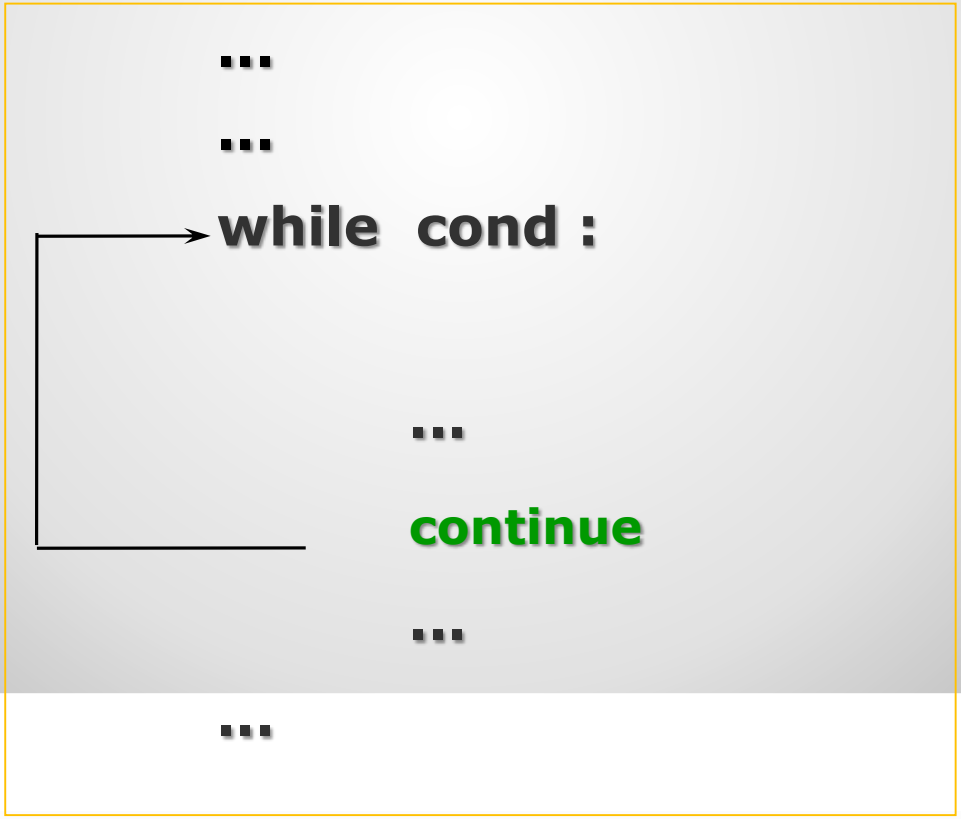
Instruction **break**

- L'instruction **break** peut être employée à l'intérieur des boucles.
- Elle permet tout simplement d'interrompre le déroulement d'une boucle et de passer à la première instruction qui suit la boucle.
- En cas de boucles imbriquées, l'instruction **break** fait sortir de la boucle la plus interne.
- **Fonctionnement :**



Instruction continue

- Retourne au début de la boucle en sautant les instructions qui la suivent.
- ***Fonctionnement***



Instruction pass

- « pass » ne fait rien du tout mais, comme on ne peut avoir une expression qui n'est suivie de rien, « pass » peut être utilisé pour combler ce vide.

Références

1. https://openclassrooms.com/fr/courses/235344-apprenez-a-programmer-en-python
2. http://apprendre-python.com/
3. https://www.tutorialspoint.com/python/