Les mots-clefs (/forum/sujet/les-mots-clefs-25832)

Liste de tous les mots-clefs en Python 3.

realmagma (/fr/members/k277cxxw6p43)

12 septembre 2010 à 10:15:43

Priez de ne **PAS** poster ici, sinon il n'y a aucune chance qu'il soit mis en Post-it. Veuillez me MP pour toutes suggestions, remarques, fautes ou autres.

Bonjour,

Je vous propose dans ce topic de retrouver tous les mots clefs liés au langage de programmation **Python** dans sa version numéro trois. Ce topic ne remplace et n'est en aucun cas un tutoriel, officiel ou non; et sera édité fréquemment.

Définition: Un mot-clé est un nom ayant une signification spéciale, et qui ne peut pas être utilisé comme identificateur de classe, de méthode ou de variable.
Source: wikibooks.org (https://fr.wikibooks.org/wiki/Accueil)

En somme, un mot clef ne peut en aucun cas être affecté comme une simple variable, car son nom est réservé.
Chaque mot-clef a sa **propre signification**.

Il en existe en tout trente. (Nous y reviendrons)

Liste des mots-clefs en Python(V3)

and	del	from	not	while
as	elif	global	or	with
assert	else	if	pass	yield
break	except	import	print	-
class	exec	in	raise	-
continue	finally	is	return	-
def	for	lambda	try	-

#1 Le mot-clef: and

Description: Le mot-clef '**and**' est un booléen logique, signifiant '**ET**'. S'utilise le plus souvent dans une condition.
Exemple:
En effet, son nom est bien '**bob**' ET son **argent (1000)** est supérieur à 500.

```
nom, argent = "bob", 1000

if nom == "bob" and argent > 500:
    print("Vous pouvez achetez cette télévision
else:
    print("Vous n'avez pas assez de sous.")
```

```
Vous pouvez achetez cette télévision !
```

#2 Le mot-clef: as

Description: Le mot-clef '**as**' signifie '**en tant que**'; parfois utile dans un gros projet. Contrôleur d'espaces noms. Il permet entre autre, de renommer un module ou le nom d'un fichier pour éviter les conflits (entre des modules qui pourraient avoir le même nom)
Exemple:
Ici, ctime() nous donne la date actuelle.

```
#On import le module time en tant que t
import time as t

t.ctime()
time.ctime()
```

```
'Sat Sep 11 19:49:40 2010'
```

```
'Sat Sep 11 19:49:40 2010'
```

#3 Le mot-clef assert

Description: Le mot-clef **'assert'**, permet de rajouter des contrôles pour le débogage d'un programme.
Exemple:

```
def assert(test, données)
```

On peut omettre l'argument'données'. '**assert**' lève une exception **AssertionError**.

```
if __debug__ :
    if not test :
        raise AssertionError, données
```

#4 Le mot-clef break

Description: Le mot-clef **'break'**, permet de sortir d'une boucle.
Exemple:
Lorsque le compteur atteint '10', on sort de la boucle grâce à celui-ci

```
compteur = 0

#On commence par une boucle infinie
while True:
    compteur += 1
    print(compteur, end = '')

    #Quand le compteur vaut 10, on sort de la b
    if compteur >= 10:
        break
```

```
12345678910
```

#5 Le mot-clef class

Description: Le mot-clef **'class'** est indispensable pour créer un classe en programmant Objet. OOP
Exemple:
On créer une classe personnage avec quelques attributs.

```
class Personnage(object):
    """Création d'une classe <Personnage>"""
    def __init__(self, nom):
        """Les attributs
        """
        self.nom = nom
        self.prenom = "bob"

        #Pour le fun
        self.force = 0
        self.magie = 0
        self.pa    = 10
        self.pm    = 5
```

#6 Le mot-clef continue

Description: Le mot-clef **'continue'** permet de repartir au début de la boucle courante.
Exemple:
On exécute les instructions une à une, dès que l'on tombe sur **'continue'** on repart au début de la boucle sans exécuter les instructions qui suivent

```
while compteur <= 10:
    avancer(x, y)
    lancerPouvoirMagique(enemieProche)

    #Quel acharnement !
    if enemieProche is not mort:
        continue

    compteur += 1
```

#7 Le mot-clef def

Description: Le mot-clef **'def'** permet de construire une fonction.
Exemple:
On définit la fonction 'addition' grâce à '**def**'

```
def addition(nombre1, nombre2):
    return nombre1 + nombre2

print(addition(1, 1))
```

```
2
```

#8 Le mot-clef del

Description: Le mot-clef **'del'** prend un argument et le supprime.
Exemple:
Il peut effacer des variables, des fonctions et même des classes !

```
maListe = ["Bonbon", "Chocolat", "Fruits", "Stea
del(maListe[0])
print(maListe)

del(maListe)
print(maListe)
```

```
["Chocolat", "Fruits", "Steak"]

Traceback (most recent call last):
  File "<pyshell#91>", line 1, in <module>
    print(maListe)
NameError: name 'maListe' is not defined
```

#9 Le mot-clef elif

Description: Le mot-clef **'elif'** '**sinon si**' s'utilise pour une condition, notamment après un '**if**' ou un autre '**elif**'; c'est une alternative secondaire.
Exemple:

- si ...
--- ...
- sinon-si (elif) ...
--- ...
C'est soit l'un, soit l'autre

```
couleur = "rouge"

if couleur == "verte":
    #Instruction(s)

elif couleur == "rouge":
    #Elle est d'occasion ;p
    acheterVoiture(1900)
    print("Voiture acheté")
```

```
Voiture acheté
```

#10 Le mot-clef else

Description: Le mot-clef **'else'** '**sinon**' s'utilise pour une condition, l' alternative dernière.
Exemple:

- si ...
--- ...
- sinon-si (elif) ...
--- ...
- sinon (else)
--- ...
Si le bloc '**si**' et '**sinon-si**' n'est pas exécuté, alors le bloc '**sinon**' (else) le sera

```
couleur = "grise"

if couleur == "verte":
    #Instruction(s)

elif couleur == "rouge":
    #Elle est d'occasion ;p
    acheterVoiture(1900)

#Si elle n'est ni verte, ni rouge; elle est pro
else:
    #Ce bloque sera exécuté
    vendreVoiture(1900)
    print("Voiture vendue")
```

```
Voiture vendue
```

#11 Le mot-clef except

Description: Le mot-clef **'except'** permet de gérer une ou plusieurs exceptions.
Exemple:
Il est préférable de préciser de quelle(s) exception(s) il s'agit

```
a = 18
b = 0

try:
    a/b
except ZeroDivisionError:
    print("La division par 0 est impossible")
```

```
La division par 0 est impossible
```

#12 Le mot-clef exec

Description:
Exemple:

#13 Le mot-clef finally

Description: Le mot-clef **'finally'** s'utilise avec les bloc '**try**' et '**except**'. Le code dans le bloc 'finally' est exécuté quoi qu'il arrive, exception ou non.
Exemple:

- try:
--- On essaye ce code

- except:
--- S'il y a une exception, ce code est exécuté
- finally:
--- Quoi qu'il arrive, ce code sera automatiquement exécuté après ceux-là

```
a = 18
b = 0

try:
    a/b
except ZeroDivisionError:
    print("La division par 0 est impossible")
finally:
    print("Le code après finally est exécuté")
```

```
La division par 0 est impossible
Le code après finally est exécuté
```

#14 Le mot-clef for

Description: Le mot-clef **'for'** signifiant '**pour**' s'utilise pour créer des boucles.
Exemple:
Dans une boucle du type **'for'** il n'y a pas besoin de créer des variables au préalable pour la parcourir.

```
for i in range(5):
    print(i)
```

```
01234
```

#15 Le mot-clef from

Description: Le mot-clef **'from'** signifiant '**de**', est utilisé lors d'un import de module.
Exemple:
S'utilise sous la forme de: from nomDuModule import *

```
from Tkinter import *
```

#16 Le mot-clef global

Description: Le mot-clef **'global'** s'utilise pour modifier une valeur dans une fonction depuis le programme principal, où les variables sont dites '**locales**'
Exemple:
La valeur de la variable '**var**' située dans la fonction **maFonction** change car elle est '**globale**'

```
def maFonction():
    global var
    print("var =", var)

#Programme principal
var = 5
maFonction()
```

```
var = 5
```

#17 Le mot-clef if

Description: Le mot-clef **'if'** signifiant '**si**', est une condition.
Exemple:

- si la condition est vérifiée
--- Ce bloc est exécuté

```
pomme = "oui"

if pomme == "oui":
    print("J'achète cette pomme")
else:
    print("Je ne prends pas cette pomme")
```

```
J'achète cette pomme
```

#18 Le mot-clef import

Description: Le mot-clef **'import'** permet d'importer un module.
Exemple:
Importation du type: import math. Importation de toutes les méthodes du module **math**

```
import math
math.sqrt(10)
```

```
3.1622776601683795
```

#19 Le mot-clef in

Description: Le mot-clef **'in'** signifiant '**dans**' ou même '**contient**'.
Exemple:
Pour chaque lettre dans maChaine
affichier lettre

```
maChaine = "abcde"

for lettre in maChaine:
    print(lettre, end = '')
```

```
abcde
```

#20 Le mot-clef is

Description: Le mot-clef **'is'** signifiant '**est**'. Contrairement à ce que l'on pourrait penser, ce n'est pas une façon de représenter en lettre le signe '**==**', et encore moins avec des variables du (type: string).
Exemple:
On voit clairement qu'il ne faut pas confondre '**==**' et '**is**' dans l'exemple ci-dessous

```
class MaClasse:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def __eq__(self, other):
        return self.a == other.a and self.b ==

a = MaClasse(2, 3)
b = MaClasse(2, 3)

a is b #False
a == b #True

b = a
a is b #True
```

```
False

True
True
```

#21 Le mot-clef lambda

Description: Le mot-clef **'lambda'** ou plus communément dit '**à la volée**' permet de créer rapidement une fonction d'une ligne.
Exemple:
Dans cette exemple, on ré-écrit la fonction 'ajouterUn' à la volée.

```python
def ajouterUn(nombre):
    return nombre + 1

resultat = print(ajouterUn(10))


#Maintenant avec le mot-clef 'lambda'
resultat = lambda nombre: nombre + 1
print(resultat(25))
```

```
11
26
```

#22 Le mot-clef not

Description: Le mot-clef **'not'** exprime une négation.
Exemple:
S'utilise le plus souvent pour dire: Si tu n'est pas/n'a pas...

```python
motSecret = "Love_Python"

if not "a" in motSecret:
    print("le mot {0} ne contient pas cette lett
```

```
Le mot Love_Python ne contient pas cette lettre.
```

#23 Le mot-clef or

Description: Le mot-clef **'or'** est un booléen logique signifiant '**ou**'. Il s'utilise le plus souvent dans une condition.
Exemple:
- Si bob est fort en Programmation ou en Français
--- Affichier: Je l'embauche !

Ça tombe bien parce que bob n'est fort quand Français

```
Mr_bob = "Fort en Français"

if (Mr_bob is "Fort en Français") or (Mr_bob is
    print("Je l'embauche !")
```

```
Je l'embauche !
```

#24 Le mot-clef pass

Description: Le mot-clef **'pass'** permet de rendre un bloc passif.
Exemple:
On ne peut pas par exemple laisser le bloc suivant sans rien à l'intérieur; sinon il plante.

```
nbrDePommes = 60

if nbrDePommes == 60:
    pass

#Le reste du code
```

"Le soit-disant mot-clef" print

Description: **'print'** n'est plus un mot-clef sous la version 3. de Python. C'est à présent une fonction.
Exemple:

```
fonctionPrint = print

fonctionPrint("Hello")
```

```
Hello
```

#25 Le mot-clef raise

Description: Le mot-clef **'raise'** permet de lever une exception.
Exemple:
L'exception ZeroDivisionError est levée.

```
a, b = 10, 0

try:
    a/b
    raise ZeroDivisionError("Impossible de divis
except:
    print("Vous ne pouvez pas diviser", a, "par"
```

#26 Le mot-clef return

Description: Le mot-clef **'return'** permet de retourner 0 ou plusieurs argument(s) dans une fonction (principalement).
Exemple:
On retourne var²

```
def fonction(var):
    return var * var

print(resultat = fonction(5))
```

```

```

#27 Le mot-clef try

Description: Le mot-clef **'try'** signifiant '**essayer**' permet de tester un bloc d'instruction.
Exemple:
On essaye de convertir la valeur numérique entrée par l'utilisateur

```
valeur = input("Entrez un valeur\n>")

try:
    valeur = int(valeur)

#except TypeError:
    #Du code

finally:
    print("La valeur entrée est", valeur)
```

```
Entrez une valeur
>6
La valeur entrée est 6
```

#28 Le mot-clef while

Description: Le mot-clef **'while'** signifiant '**tant que**' permet de construire une boucle.
Exemple:
On créer une boucle grâce à lui. On n'oublie pas d'implémenter une condition pour éviter les boucles infinies.

```python
compteur = 0

while compteur <= 5:
    print(compteur, end = '')
    compteur += 1
```

```
12345
```

#29 Le mot-clef with

Description: Le mot-clef **'with'** signifiant '**avec**' est utilisé pour les manipulations de fichiers et d'objets.
Exemple:
Dans ce cas, le mot-clef '**with**' permet d'utiliser le fichier déjà ouvert.

```python
with open(monFichier, r+) as f:
    #Le reste du code
```

#30 Le mot-clef yield

Description: Le mot-clef **'yield'** retourne successivement plusieurs valeurs. (Connaître ce que sont les Itérateurs et les Générateurs)
Exemple:
Dans cette exemple, la fonction **pommeItérateur** renverra successivement '**prendrePomme**', puis '**laverPomme**' au prochain appel, puis '**mangerPomme**'; et cela ainsi de suite

```
def pommeIterateur():

    while True:
        for i in ("prendrePomme", "laverPomme",
            yield i

#Programme principal
action = pommeIterateur()

for i in range(6):
    print(action.next())
```

```
prendrePomme
laverPomme
mangerPomme
prendrePomme
laverPomme
mangerPomme
```

 PARTIE EN CHANTIER ...

Priez de ne **PAS** poster ici, sinon il n'y a aucune chance qu'il soit mis en Post-it. Veuillez me MP pour toutes suggestions, remarques, fautes ou autres. Merci.

---

nohar (/fr/members/c2v9qc734dm8)

(/fr/members/c2v

12 septembre 2010 à 13:32:59

Tu n'as pas compris l'utilisation du mot-clé is .
Il n'est pas DU TOUT équivalent à l'opérateur ==.

Il ne teste pas l'égalité, mais l'identité.

```
>>> class MaClasse:
...     def __init__(self, a, b):
...             self.a = a
...             self.b = b
...     def __eq__(self, other):
...             return self.a == other.a and se
...
>>> a = MaClasse(2, 3)
>>> b = MaClasse(2, 3)
>>> a is b
False
>>> a == b
True
>>> b = a
>>> a is b
True
```

C'est pour cette raison que la comparaison entre chaines de

caractères avec is est très fortement déconseillée...

Edit:
Et ce n'est pas parce que j'ai posté ici que ce sujet ne deviendrait pas un post-it.
Il le deviendra si les modos jugent que le premier post est de qualité et constitue une source d'aide pour les débutants.
BTW il vaudrait mieux penser au contenu du topic et de voir ensuite s'il est vraiment post-itable plutôt que de commencer un topic avec la ferme intention qu'il le devienne.

Zeste de Savoir (http://zestedesavoir.com), le site qui en a dans le citron !

Les mots-clefs

Après avoir cliqué sur "Répondre" vous serez invité à vous connecter pour que votre message soit publié. ✖

Attention, ce sujet est très ancien. Le déterrer n'est pas forcément approprié. Nous te conseillons de créer un nouveau sujet pour poser ta question. ✖

Style | Taille | Editeur • Markdown

HTML