Python 2.3 Syntax Reference

Types

Type	Catégorie	Forme littérale
None	Nulle	None
Bool	Booléen	True, False
Int	Nombres	25, -25
Long		25L, -25L
Float		25.0, 25.0E3
Complex		3+4j
String	Séquences	"abc", """...""""
Unicode str.		u"abc"
Tuple		(1, 2)
List*		[1, 2, 3] [f(x) for x in seq]
Dict*	Mapping	{"a":1; "b":2}
File	Fichiers	File("nom.txt")
Function	Exécutable	def func(): ...
Class		class objDef(): ...

* = mutable[1], (rien) = non mutable[2]

Caractères littéraux spéciaux

\	Continuation de lignes
\\	Backslash
\'	Guillemet simple
\"	Double guillemet
\n	Retour de ligne
\xhh	Valeur héxa (\x00 à \xff)
\uhhh	Valeur héxa Unicode

Convention de nommage des variables

Préfixe	Exemple	Type
s	sTexte	String
b	bValid	Booléen
i	iCount	Int
ii	iiCount	Long
l	lCollect	List
t	tCoord	Tuple
d	dTel	Dict

[1] les modif. ont lieu sur l'objet
[2] les modif. ont lieu sur une copie de l'objet

Expressions

Affectation et affectation augmentée

Opération	Description
x = y	Affectation
del x	Suppression
x += y	x = x + y
x -= y	x = x - y

Opérations arithmétiques

Opération	Description
x + y	Addition
x - y	Soustraction
x * y	Multiplication
x / y	Division
x ** y	Puissance
x % y	Modulo (reste de x divisé par y)
-x	Moins unaire
+x	Plus unaire

Opérations de comparaison

Opération	Description
x < y	Plus petit que
x > y	Plus grand que
x == y	Egal à
x != y	Différent de (identique à <>)
x >= y	Plus grand ou égal à
x <= y	Plus petit ou égal à

Opérations sur les dictionnaires

Opération	Description
x = d[k]	Indexation selon la clé k
d[k] = x	Affectation selon la clé k
del d[k]	Efface un élément selon la clé k
len(d)	Nombre d'éléments du dict.

Opérations sur les séquences

Opération	Description
s + r	Concaténation
s * n	Fait n copies de s
s % d	Mise en forme (String slt)
s[i]	Indexation
s[i:j]	Slicing
x in s	Appartenance
for x in s: ...	Itération sur les élts de s
len(s)	Nombre d'éléments
min(s)	Elément le plus petit
max(s)	Elément le plus grand
s[i] = x	Affectation à un élément
s[i:j] = r	Affectation à un *slice*
del s[i]	Supprime un element
del s[i:j]	Supprime un *slice*

Caractères de mise en forme des String

Caractère	Format de représentation
%d	Nombre entier
%5d	Idem sur 5 positions
%f	Nombre décimal
%.2f	Idem avec 2 décimales slt
%s	Chaîne de caractères
%25s	Idem sur 25 positions
%-25s	Idem, aligné à gauche
%c	Caractère simple
%%	Caractère %

Fonctions internes de Python

Opération	Description
abs(x)	Valeur absolue de x
divmod(x,y)	Retourne (int(x / y), x % y)
round(x,[n])	Arrondit x au multiple le plus proche de 10^{-n}

Mode d'exécution

Contexte	Mode	Exemple
Expression	Immédiat	a == 2
Instructions		print "hello"
Invoc. fonc.		func(10)
Invoc. méthode		string.upper()
Déf. fonction	Différé	def func(n): ...
Déf. classe		class obj(): ...
Importation		from string ...
		import string

Fonctions

```
# Définition de fonction (différé)
def funcName( param1, param2):
   """description de la fonction"""
   instructions qui utilisent param1 et param2
   return expr

# Invocation (immédiat)
result = funcName( val1, val2)
```

Modules

```
import string
from string import replace
dir( string)
dir( string.replace)
```

Aide

```
print func.__doc__
help( string)
help( string.replace)
```

Python 2.3 Syntax Reference

Modèle de script

```
# -*- coding: cp1252 -*-
"""Description du programme"""

import sys

def main():
   """Programme principal"""
   …
if __name__ == "__main__": main()

# eof
```

Structure des lignes de code

Les instructions d'un programme sont terminées par un retour de ligne (\n)
ou par un point-virgule ;

Une longue instruction peut être répartie sur plusieurs lignes en utilisant le caractère de continuation de ligne \ :

```
a =   math.cos( 3 * ( x - n) + \
      math.sin( 3 * ( y - n)
```

Le caractère de continuation de ligne n'est toutefois pas nécessaire à l'intérieur des parenthèses (…), crochets […], accolades { … } et triples guillemets """ … """.

Le caractère dièse # dénote un commentaire, qui s'étend jusqu'à la fin de la ligne.

L'indentation est utilisée pour dénoter les blocs de code, tels que le corps des fonctions, conditions, boucles et classes.

L'indentation doit être consistante au sein des instructions d'un bloc:

```
if expr :
   instr1     # consistant
   instr2
else :
   instr3
      instr4  # pas consistant
```