Python3 en condensé

1 Conventions de style d'écriture

Toutes les conventions décrites dans cette partie respectent celles décrites dans PEP 8 -- Style Guide for Python Code.

1.1 Indentation

En Python, utilisez quatre espaces d'indentation comme dans l'exemple ci-dessous.

```
a = 15
while  a != 1:
   if a % 2 == 0:
      a = a // 2
   else:
      a = 3 * a + 1
```

1.2 Longueur des lignes

Évitez les lignes de code trop longues. Une convention est de limiter les lignes à 79 caractères.

1.3 Continuation des lignes

Il arrive qu'une expression conduise à une ligne trop longue.

- Une expression contenant une parenthèse ou une accolade ou un crochet ouvrant peut être poursuivie à la ligne suivante :

```
In [1]: print(1+2+3+
   ...: 4+5+6+
   ...: 7+8+9)
45
```

- Pour de longues expressions ne commençant pas par une parenthèse, accolade ou crochet ouvrant, on peut utiliser le caractère de continuation de ligne (\) :

```
a = \
 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9
```

1.4 Espaces

Votre code doit être « aéré » avec des espaces.

- affectations : i = i + 1
  et non : i=i+1
- expressions : 2 * (x + 1) / (x - 1)
  et non : 2*(x+1)/(x-1)
- énumérations séparées par des virgules : [a, b, c], print(x, y)
  et non : [a,b,c], print(x,y)

2 Conventions de nommage

2.1 Constantes

Les constantes doivent être nommées par des identificateurs en lettres capitales, avec éventuellement des blancs soulignés ou chiffres. RACINE_DEUX = 1.414

2.2 Variables

Les variables doivent être nommées par des identificateurs ne comprenant que des lettres bas de casse, avec éventuellement des blancs soulignés ou chiffres.

Ne jamais commencer un nom de variable par un chiffre.

Pour un nom de variable à une seule lettre, ne pas utiliser les lettres :

- **l** (L minuscule qui risque d'être confondu avec le chiffre 1). Si nécessaire, utiliser L.
- **O** (o majuscule qui risque d'être confondu avec le chiffre 0).
- **I** (i majuscule qui risque d'être confondu avec le chiffre 1).

```
joueur1 = 'calbuth'
joueur2 = 'timoleon'
numero_partie = 5
```

2.3 Fonctions

Les fonctions doivent être nommées en lettres bas de casse, éventuellement avec des blancs soulignés ou chiffres.

```
def celsius_en_fahrenheit(celsius):
    fahrenheit = 9 / 5 * celsius + 32
    return fahrenheit
```

2.4 Mots-clés en Python3

Un mot-clé est un identificateur ayant un sens prédéfini dans le langage de programmation. Les mots-clés ne peuvent pas être utilisés comme identificateur de constantes, variables, fonctions, etc ..., sous peine de provoquer une erreur de syntaxe (SyntaxError).

Liste des mots clés du langage Python (dans la version 3).

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

3 Commentaires

Tout ce qui suit le caractère # est un commentaire, et est ignoré par Python.

On distingue :

- les commentaires en fin d'une ligne de code :

```
TAUX_INTERET = 0.035  # taux d'intérêt annuel à 3.5%.
```

a, b, c = 1., -1., 1. # les trois coefficients d'un trinôme

- les lignes de commentaires :

```
delta = b * b - 4 * a * c
if delta >= 0.:
    # calcul des deux racines reelles éventuellement confondues
    x1 = (-b + sqrt(delta)) / (2 * a)
    x2 = (-b + sqrt(delta)) / (2 * a)
```

- les blocs de lignes de commentaires :

```
# ------------------------------------------
# authors: Calbuth, Timoleon
# date: 10/07/2017
# object: example of a multiline comment
# ------------------------------------------
```

4 Valeurs littérales

En programmation, on désigne par littéral toute suite de caractères désignant sa propre valeur.

4.1 Le littéral None

En Python, None est une valeur littérale qui désigne tout ce qui n'a pas de valeur (comme la valeur d'un appel à la fonction print par exemple).

- Par exemple, si on saisit et on exécute la ligne suivante :

```
print('Timoleon') == None
```

alors on obtient :

```
Timoleon
Out[1]: True
```

None est une valeur d'un type spécial qui est NoneType. C'est la seule valeur de ce type.

```
type(None)
NoneType
```

4.2 Les entiers

Les littéraux entiers sont obtenus tout naturellement par leur écriture habituelle en base 10.

- Par exemple, si on saisit et on exécute la ligne suivante :

```
123
```

alors on obtient :

```
Out[2]: 123
```

On peut aussi donner des valeurs littérales, commençant par le nombre 0, dans d'autres bases que la base décimale. On préfixe alors l'écriture littérale par 0b pour le binaire (base 2), 0o pour l'octal (base 8) ou 0x pour l'hexadécimal (base 16).

- Par exemple, si on saisit et on exécute la ligne suivante :

0b1111011

alors on obtient :
Out[3]: 123

- Si on saisit et on exécute la ligne suivante :

0o173

alors on obtient :
Out[4]: 123

- Si on saisit et on exécute la ligne suivante :

0x7B

alors on obtient :
Out[5]: 123

En Python3, il n'y a aucune limitation sur la taille des entiers (hormis la taille de la mémoire de la machine sur laquelle le programme s'exécute).

Les littéraux entiers ne contiennent jamais de point. Il ne faut pas confondre 123 qui est un entier (type int) et 123. qui est un flottant (type float).

type(123)
int

type(123.)
float

4.3 Les flottants

Les littéraux flottants s'écrivent uniquement en base 10. Ils comprennent tous un point dans leur écriture.

Par exemple :
123.5

Ils peuvent être exprimés en notation scientifique :

- Si on saisit et on exécute la ligne suivante :

123e100

alors on obtient :
Out[7]: 1.23e+102

Les flottants sont limités à environ 17 chiffres significatifs, et les flottants non nuls sont, en valeur absolue, compris entre 10^{-308} et 10^{308} environ.

4.4 Les chaînes de caractères

Les littéraux chaînes de caractères sont délimités par deux apostrophes (') ou deux guillemets anglais (").

"Timoleon est un homme politique grec."
a la même valeur que
'Timoleon est un homme politique grec.'

Les deux façons d'écrire une chaîne de caractères permettent d'inclure si besoin une apostrophe en délimitant avec des guillemets.

- Par exemple, si on saisit et on exécute la ligne suivante :

"l'après-midi"

alors on obtient :
Out[8]: "l'après-midi"

Les deux façons d'écrire une chaîne de caractères permettent d'inclure si besoin des guillemets en délimitant avec des apostrophes.

- Par exemple, si on saisit et on exécute la ligne suivante :

'Il dit : "Bonjour !"'

alors on obtient :
Out[9]: 'Il dit : "Bonjour !"'

Avec ces délimiteurs, les chaînes de caractères doivent être écrites sur une seule ligne.

Si nécessaire, il est possible de les écrire sur plusieurs lignes avec le caractère de continuation de ligne \.

- Si on saisit et on exécute les lignes suivantes :

'Timoleon \
est un homme \
politique grec.'

alors on obtient :
Out[10]: 'Timoleon est un homme politique grec.'

Avec un délimiteur triplé, triple apostrophe (''') ou triple guillemet anglais (""") les littéraux chaînes peuvent être écrits sur plusieurs lignes.

On les utilise souvent dans la documentation des fonctions (docstring).

"""
Timoleon
est un homme
politique grec.
"""
Out[11]: '\nTimoleon\nest un homme\npolitique grec.\n'

Explication : Les \n écrits dans une chaîne de caractères signifient "new line".

En résumé :

- \ est une continuation de ligne.
- \n est un passage à une nouvelle ligne.

4.5 Les booléens

Deux valeurs littérales seulement pour les booléens : True et False. Attention : ce n'est ni TRUE, ni true.

5 Opérateurs

5.1 Opérateurs arithmétiques

Opérateur	Signification
+	Addition
-	Soustraction
*	Multiplication
**	Exponentiation
/	Division flottante
//	Division euclidienne (on dit aussi division entière) : quotient
%	Division euclidienne : reste (on dit aussi le modulo)

5.2 Opérateurs relationnels

Opérateur	Signification
‹	Inférieur
›	Supérieur
‹=	Inférieur ou égal
›=	Supérieur ou égal
==	Égal
!=	Différent

5.3 Opérateurs booléens

and	Et logique
or	Ou logique
not	Négation

5.4 Priorité des opérateurs

Classement des opérateurs par priorité décroissante :

**	Mise à la puissance
*, /, //, %	13 // 4 donne 3 (quotient dans la division euclidienne) ; 13 % 4 donne 1 (reste dans la division euclidienne)
+, -	
not	
and	
or	

- Par exemple, si on saisit et on exécute :

3 + 4 * 5 alors on obtient : Out[12]: 23
et non 35 car * est prioritaire sur +.

En résumé :
3 + 4 * 5 = 3 + (4 * 5)

- Par exemple, si on saisit et on exécute :

True or False and False alors on obtient Out[13]: True
et non False car and est prioritaire sur or.

En résumé :
True or False and False = True or (False and False)