Formation au langage de programmation Python

Initiation à Python

Partie III
good habits – générateurs – structures – fonctions

Formateur : IBRAHIM M. S.
ibrahim.ms@gmail.com

du 17/10 au 20/10 2016

Plan

Bonnes pratiques de programmation : code source

En ce qui concerne le code

- code lisible (simple) et commenté, convention de nommage, modulaire
- indentation, smart editor, coloration syntaxique, linter, doxygen

Commentaires – Ecrire du code commenté intégralement

- fichiers, scripts, classes, modules, fonctions, méthodes, constructeurs
- profit : aide contextuelle intégrée à l'éditeur de texte, gain de temps

Documentation – Générer les documentations pdf html pdf

- reStruredText, Docutils, Sphinx, Autodoc, Napoleon, Doxygen...

Pythonic philosophie

- Zen of Python : import this — PEP8 — encodage — shebang

Bonnes pratiques de programmation : conventions

Outils

- pylint, pychecker, recommendations PEP8 , vérifier la version Python

Conventions :

- de nommage

- fonctions — lowercase_underscore_format
- exception/classes — CapitalizedWordFormat
- constantes de module — ALL_CAPS_UNDERSCORE_FORMAT
- self [cls] — premier paramètre de méthode[de classe]
- opérateurs — surcharger les opérateurs pour les classes

- d'import de modules

- importer le strict nécessaire — ordre : stdlib $\longrightarrow$ tiers $\longrightarrow$ personnels

Fonctions à la volée : les fonctions anonymes

Fonctions anonymes

- fonction sans le mot-clef def
- la fonction tient en une instruction

Syntaxe

- lambda arguments : expression

Utilisation

- (lambda arguments_fictifs : expr) (argument_reels)
- f = lambda arguments_fictifs : expr ; f (argument_reels)

Exemple

- g = lambda L x : [k for k in range(len(L)) if L[k]==x]

Conteneurs en compréhension (ou intention)

Syntaxe

- new_list = [function(item) for item in old_list if condition(item)]

Cas d'utilisation

- filtrer des éléments d'une liste sur une condition
- appliquer une fonction aux éléments d'une liste

Exemples

- a = [1,4,2,7,1,9,0,3,4,6,6,6,8,3]
- c = [x for x in a if x > 5]
- L = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
- M = [2**x for x in L]
- L = ["5", "10", "15"]
- b = [str(int(x)*10) for x in L]

Structures de contrôle classiques

Conditions logiques

- quelques ajout : A in B — X is Y — X is not Y
- conditions ensemblistes : inclusion d'ensemble

Branchements

- if (cond) : (bloc indenté)
- elif : (bloc indenté)
- else : (bloc indenté)

tant que

- while (cond) : (bloc indenté)

itération

- for var in itérateur : (bloc indenté)

Fonctions

Utilité

- meilleure lisibilité du code, test posssible des fonctionnalités
- maintenabilité accrue et réutilisatilité (non redondance)
- un pas vers la modularité : les modules

Déclaration

- def nom (argument) : (corps intenté)
- docstring facultative mais recommendée
- présence d'un return obligatoire : sinon None imposé

Appel de la fonction

- le passage des arguments est positionnel
- on peut utiliser des argument nommés : ordre arbitraire

Exercice 1

```
def f(a,b,c) :
    """ f doc """
    a**b+c
```

- f(2, 31, -1)
- f(a=2, 31, -1)
- f(2, c=31, a=-1)
- f(2, c=31, b=-1)
- f(c=3, a=31, b=-1)
- f(b=3, c=31, a=-1)

Exercice 2

- écrire une fonction moyenne : moyenne d'une liste de nombres
- écrire une fonction moyenne : moyenne de ses arguments

Exercice 2 : nombre variable d'arguments

- def moyenne(L) : return (sum(L)/len(L))
-
  - def moyenne(*arg) : return (sum(arg)/len(arg))
  - def moyenne(*arg) : L = [i for i in arg] ; return sum(L)/len(L)