Quelques conventions de présentation

Structuration d'un programme
et Guide Style – PEP 8

Sommaire

- Conventions de présentation (Style guide)
- Structuration générale d'un programme

- Contrôle dans spyder :

Outils / Preferences > Editor > Code Introspection/Analysis
select the tickbox next to Style analysis (PEP8)

- Reformat dans pycharm

PEP 8

- Python Enhancement Proposal
  - Un code dont on est l'auteur peut être difficile à relire si on l'abandonne quelque temps
  - Lire le code d'un autre développeur est toujours plus délicat
  - Si votre code doit être utilisé par d'autres, il doit être facile à reprendre (à lire et à comprendre)

Objectifs

- Homogénéité de la bibliothèque standard
- Lisibilité
- Bonnes pratiques qui se propagent

Lignes

- Taille maximum d'une ligne : 79 caractères
- Longues lignes et continuité
  - continuation de ligne implicite à l'intérieur de ( ) [ ] { }

```
def __init__(self, first, second, third,
                   fourth, fifth, sixth):
        output = (first + second + third
                    + fourth + fifth + sixth)
```

  - Sinon, utiliser le "\"

```
if  width == 0 and height == 0 and \
    color == 'red' and emphasis == 'str' :

    etat = "perdu"
```

Saut de lignes

- Les sauts de ligne → facteur de lisibilité du code

```
def plus_petit_diviseur(n):
    """
    @param n:   nombre entier
    @return:    le plus petit diviseur de n autre que 1
    """

    # on teste les diviseurs potentiels de n à partir de 2
    for d in range(2, n + 1):
        if n % d == 0:        # on s'arrête forcément à n
            break

    return d
```

- Deux lignes vides entre éléments 'top-niveau' (classes)
- Une ligne entre les éléments 'internes' (méthodes)

Espaces

- Espaces dans les expressions et définitions
  - Un espace après "," ";" ":"
  - Pas d'espace avant
  - Un espace de chaque côté d'un opérateur
  - Pas d'espace après "{" "[" "("
  - Pas d'espace entre le nom d'une fonction et sa liste d'arguments
  - Pas d'espace entre le nom d'un dictionnaire et un index

Espaces - Exceptions

- Groupement des opérateurs mathématiques ayant la priorité la plus haute

  a = x*2 - 1 b = x*x + y*y c = (a+b) * (a-b)

- Pas d'espace autour du signe = dans le passage de paramètres :

  def fonction(arg=valeur)

Conventions de nommage

- Modules : un ou plusieurs mots attachés et en minuscules

  jeudevie.py

- Classes : un ou plusieurs mots attachés dont chaque 1ère lettre est en majuscules

  PersonnageArmé

- Fonctions et variables, méthodes et attributs : un ou plusieurs mots séparés par "_" et en minuscules

  creation_annuaire

Guide Style

- Commentaires :
  - Valeur ajoutée : s = 0 # initialisation de s à 0
  - Description du fonctionnement d'une partie du code
  - Entêtes des fonctions, des modules
  - etc.
  - Mettre systématiquement VOS NOMS en début de chaque script

- Docstring

  """ documentation """
  - tous les modules publics
  - toutes les fonctions
  - toutes les classes
  - toutes les méthodes de ces classes
  - création automatique de documentation : help(fct)

- Il n'y a aucun saut de ligne avant ou après la docstring.

Exemple de docstring en format Epytext (pour doxygen)

```
def racine(a, eps):
    """

    @param a:   valeur dont on veut calculer la racine
    @param eps: erreur autorisée
    @return:    valeur approchée de la racine carrée de a
    """
    r = 2.
    while abs(r * r - a) > eps:
        r = (r + a / r) / 2

    return r
```

Exemple de docstring format numpy
(sous spyder)

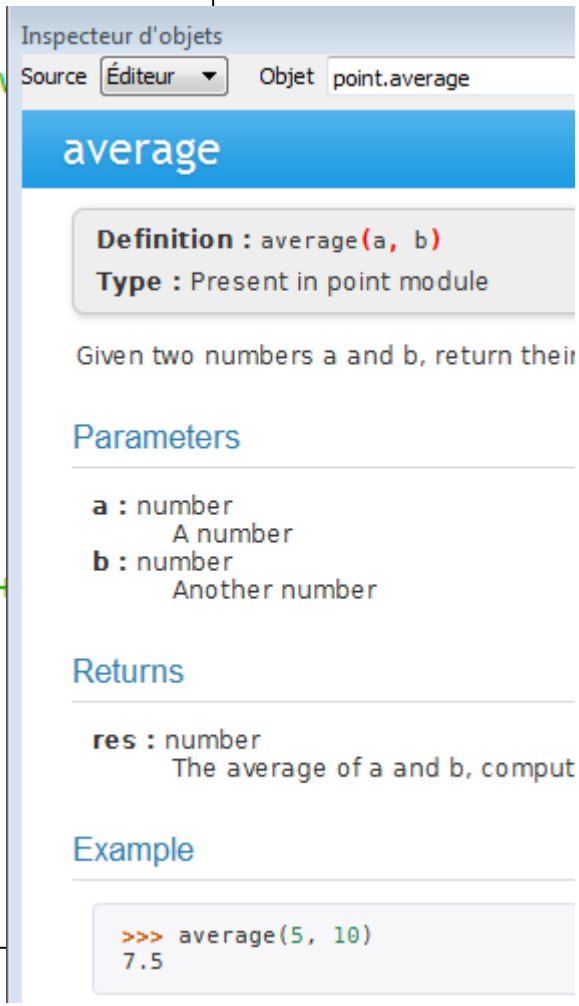
```
def average(a, b):
    """
    Given two numbers a and b, return their average v

    Parameters
    ----------
    a : number
      A number
    b : number
      Another number

    Returns
    -------
    res : number
      The average of a and b, computed using 0.5*(a +

    Example
    -------
    >>> average(5, 10)
    7.5

    """
```



Recommandations de codage

None

```
x = None
if x:
    print("on ne passe pas ici")
else:
    print(« on passera")
```

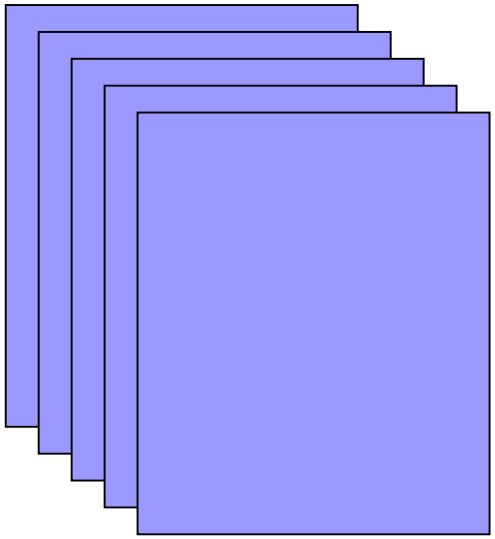
```
x = ""
if x:
    print("on ne passe pas ici")
else:
    print("on passera")
```

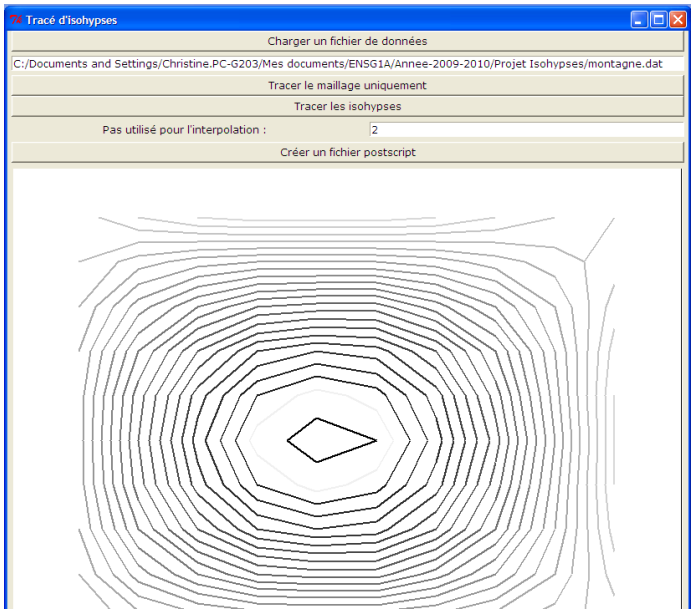
```
ma_liste = []
If not ma_liste:
    print("on ne passe pas ici")
else:
    print("on passera")
```

```
bool = True
If bool:      # ou if bool == True
    print("on ne passe pas ici")
else:
    print("on passera")
```

Organisation des Programmes

Construction d'un programme





Fichiers sources

Application qui communique directement a l'ordinateur

Organisation d'un programme

- Code réparti dans plusieurs fichiers : *Les modules*
- Un module regroupe :
  - Classes
  - Fonctions
  - Variables
- Un module par classe
- Extension *.py*
- 2ème niveau d'organisation : les paquets (packages : organisation des modules dans plusieurs répertoires)

Organisation d’un module

- Entête
- Clauses d'importation
- Classes et fonctions
- Instructions ("main")

Importation de modules

- La directive *import*
- Permet d'utiliser le code contenu dans des modules python

```
from module import fonction1
from module import fonction2

from module import *

import module
# dans ce cas : utilisation des fonctions
module.fonction
```

Modules

- Modules de la bibliothèque standard
  - Modules système : sys, os …
  - Modules pour le calcul numérique : math
  - Modules structures de données : array, time, datetime
- Modules personnels
- Bibliothèques tierces : base de données, numerical python, jeux et 3D…

Modules standards

- Primitives (fonctions ou méthodes) :
  - Fonctions directement accessibles dans l'interpréteur
  - Également appelé built-ins
  - help(x) permet d'afficher de l'aide sur l'utilisation de x
- De nombreuses fonctionnalités existent dans les modules standards
- Eviter de réinventer la roue

Mise au point d'un programme

Mise au point d'un programme

- Recherche des erreurs
  - Erreurs de syntaxe
    - Concerne la structure du programme
    - Détectée par l'éditeur
      - Pb de parenthésage, d'indentation, etc.
  - Erreurs à l'exécution
    - Erreurs décelées par l'interpréteur au cours de l'exécution du programme
    - Erreurs appelées également des exceptions
      - Exemple : division par 0, boucle infinie, etc.
  - Erreurs sémantiques
    - Par d'erreurs détectées par l'interpréteur
    - Le programme fournit des résultats, mais pas les bons !!