Rappel sur les bonnes règles d'écriture du code

NSI terminale - JB Duthoit

Coder de façon lisible et claire (spécifications)

Votre code doit être facile à lire par une autre personne. Et cette autre personne peut très bien être-vous deux ans plus tard....on est parfois surpris de ne pas comprendre ce que l'on a bien pu faire - :) !

Pour bien comprendre l'intérêt, il faut garder à l'esprit que vous passerez plus de temps à relire votre code qu'à l'écrire !

Généralités

Votre code doit :

- Beau (c'est mieux que laid :-))
- Explicite
- Simple (c'est mieux que complexe)
- Complexe mieux que compliqué
- Aéré
- Documenté. En Python les commentaires commencent par un # .

En python

Vous trouverez tous les détails sur le site officiel de Python.
En résumé :

- En python ne mélangez jamais tabulations et espaces
- Une ligne de code ne devrait pas dépasser 79 caractères.
- Encoder en Utf-8
- Importer les bibliothèques en début de programme sur des lignes séparées.
- Il y a de nombreuses conventions sur les espaces :
  - On met un (et un seul) espace avant et après les affectation, comparaisons, booléens et opérations
  - On ne met pas d'espace pour le reste (, { ,[...
- Pour les fonctions, variables et méthodes tout en minuscule avec _ pour séparer les mots.
- Pour le nom des classes : majuscule pour chaque nouveau mot et mots collés. (Exemple : MaClasse)
- En ce qui concerne la documentation à l'intérieur des fonctions :

- Lorsque vous créez une fonction, vous devez la documenter. C'est le rôle du docstring. Le docstring se place juste après la création de la fonction par def. Il commence et termine par trois guillemets "
- Votre docstring doit décrire le rôle de la fonction, puis les paramètres passés en arguments (type et rôle), ainsi que le type de ce qui est retourné
- On accède au docstring en tapant :

```
nom_de_ma_fonction.__doc__
```

0.1 Affectation

NSI terminale - JB Duthoit

Exercice 0.1

L'indice de Masse Corporel est un nombre réel utilisé en médecine. Sa formule est pour une masse en kilos et une taille en mètres :

$$IMC = \frac{masse}{taille^2}$$

1. Écrire en langage Python une fonction `masse_corporelle()` qui :
   - Demande la masse de l'utilisateur en kg (nombre réel)
   - Demande la taille de l'utilisateur en cm (nombre entier).
   - affiche l'IMC de l'utilisateur
2. Écrire en langage Python une fonction `masse_corporelle_2(masse,taille)` qui a comme paramètre `masse` en kg et `taille` en mètres. La fonction renvoie l'indice de masse corporelle.

Testez votre programme avec différentes valeurs.

Exercice 0.2

Écrire une fonction nommée `dire_ bonjour(nom)` ayant comme paramètre une chaine de caractère `nom` et qui affiche bonjour suivi du mot.
Exemple : `dire_ bonjour("Paul")` affiche `"Bonjour Paul"`.

Remarque

⚠ Il ne faut surtout pas confondre :

- Une fonction qui **affiche** quelque chose, et dans ce cas on utilisera la commande `print`. Dans ce cas, la fonction est d'ailleurs plutôt appelée **procédure** (car elle ne renvoie rien - en réalité, elle renvoie `None`)
- Une fonction qui **renvoie** quelque chose, et dans ce cas in utilise la commande `return`

Exercice 0.3

L'énergie cinétique d'un objet de masse m (en kg) qui se déplace à la vitesse v (en m/s) est donnée par la formule :

$$E_c = \frac{1}{2} \times m \times v^2$$

Écrire une fonction nommée `get_ energie_ cinetique(m,v)` ayant comme paramètres la masse `m` et la vitesse `v` qui renvoie l'énergie cinétique.
Quelle est l'énergie (en Joule) d'un objet de 3 kg se déplaçant à 1.56 m/s ?

Exercice 0.4

Écrire une fonction `div_ euclidienne(a,b)` qui prend en paramètre deux nombres entiers `a` et `b`, qui effectue la division euclidienne de a par b et qui renvoie le couple (r,q), respectivement le reste et le quotient de cette division euclidienne.

0.2 Instructions conditionnelles

NSI terminale - JB Duthoit

Exercice 0.5

Écrire une fonction python `renverse(a,b)` qui prend en paramètre deux entiers `a` et `b` et qui retourne b et a (en inversant l'ordre).
Par exemple `renverse(a,b)` va renvoyer `(b,a)`

Exercice 0.6

On considère la fonction suivante.

```
def examen(x,y,z,t):
        m = (2 * x + 2 * y + z + t) / 6
        if m >= 10:
                print("Le candidat est reçu")
        else :
                print("le candidat est refusé")
        return None
```

1. Documenter la fonction en ajoutant un docstring.
2. Proposer des pré-conditions écrites sur les variables x, y, z et t en utilisant l'instruction assert qui assurent le bon usage de cette fonction examen.

Exercice 0.7

Écrire une fonction `signe_prod(a,b)` qui reçoit deux entiers a et b relatifs et qui renvoie `True` si le produit est positif et `False` sinon.
⚠ On ne doit pas calculer le produit des deux nombres.

Exercice 0.8

Écrire une fonction `categorie(age)` qui a pour paramètre l'age de l'enfant(qui sera un entier) et qui renvoie sa catégorie :

- "Poussin" de 6 à 7 ans
- "Pupille" de 8 à 9 ans
- "Minime" de 10 à 11 ans
- "Cadet" après 12 ans

Exercice 0.9

Écrire une fonction Python `est_voyelle(lettre)` pour tester si `lettre` est une voyelle ou non. La fonction renvoie un booléen.

Exercice 0.10

Écrivez une fonction `triangle(a,b,c)` pour vérifier qu'un triangle est équilatéral, isocèle ou scalène.
Remarque :
Un triangle équilatéral est un triangle dans lequel les trois côtés sont égaux.
Un triangle scalène est un triangle qui a trois côtés inégaux.
Un triangle isocèle est un triangle avec (au moins) deux côtés égaux.
Production attendue :

- `triangle(6,8,12)` renvoie `"Triangle scalène"`

0.3 Les boucles

NSI terminale - JB Duthoit

Exercice 0.11

Écrire une fonction `carre(n)` qui lit en entrée un entier `n` et écrit tous les carrés des nombres inférieurs à n, dans l'ordre croissant. Par exemple, si l'entrée est 16, la sortie sera :
1
4
9

Exercice 0.12

Créer une fonction `somme(n)` de paramètre `n` et qui retourne la somme des entiers naturels jusque n.

- avec une boucle pour
- avec une boucle tant que

Exercice 0.13

Écrire une fonction `tablemulti(n)` qui reçoit en entrée un nombre entier `n` compris entre 1 et 10 et affiche en sortie la table de multiplication de ce nombre. Par exemple, si la fonction prend comme argument le nombre 7, il affichera la table de multiplication :
1 fois 7 = 7
2 fois 7 = 14
3 fois 7 = 21
etc...

Exercice 0.14

Écrire une fonction qui permet d'écrire un tableau d'entiers aléatoires qui s'arrête dès que la somme des nombres dépasse une certaine valeur. la fonction `make_list(deb,fin,maxi)` prend en entrée 3 valeurs entières `deb`, `fin` et `maxi` correspondant respectivement à la plus petite valeur des nombres aléatoires, à la plus grande et à la valeur à ne pas dépasser en faisant la somme de ces nombres.

Exercice 0.15

Écrire une fonction qui demande à l'utilisateur d'entrer des notes entières, sans que l'on sache à l'avance combien de notes il va saisir : l'arrêt de la saisie se produit dès que l'utilisateur saisit un nombre négatif. Le programme doit calculer et afficher la moyenne (Attention : le nombre négatif saisi ne doit pas compter dans la moyenne !). On nommera cette fonction moy elle ne prendra aucune entrée et donnera un élément de type float en sortie.

Exercice 0.16

Pour commencer l'ordinateur va choisir au hasard un nombre entier compris entre 1 et 100.
L'utilisateur doit alors deviner ce nombre comme ceci : L'utilisateur propose un nombre. L'ordinateur lui dit s'il est trop petit ou trop grand, et ainsi de suite jusqu'à ce que l'utilisateur ait trouvé le bon nombre. Possibilité d'ajouter un compteur qui donnera le nombre d'essais utilisés.

0.4 Les chaînes

Exercice 0.17
Définissez une fonction `minuchaine(chaine)` qui renvoie le résultat de la conversion de chaine en minuscules.

Exercice 0.18
Définissez une fonction `minuchaine2(chaine)` qui retourne le mot avec la première lettre en majuscule et le reste en minuscule.

Exercice 0.19
Écrire une fonction `bien_trie(l)` qui reçoit un tableau `l` de trois chaines de caractères et qui renvoie `True` si les trois mots sont rangés par ordre alphabétique et `False` Sinon.

Exercice 0.20
Écrire une fonction qui prend une chaine de caractère en entrée et qui retourne la même chaîne mais en ayant au préalable effacé le premier et le dernier caractère. (Vous vous assurerez que la chaîne a une longueur d'au moins 2.) Par exemple, pour l'entrée `Fairy`, une fonction correcte écrira `air`.

Exercice 0.21
Écrire une fonction qui prend en entrée une chaîne et qui retourne en sortie la même chaîne mais avec le premier et le dernier caractères échangés. (Vous vous assurerez que la chaîne a comme longueur au moins 2.)
Par exemple, pour l'entrée `Fairy`, une fonction correcte écrira `yairF`.
☛ Indice : utilisez votre solution de l'exercice précédent comme point de départ.

Exercice 0.22
Écrire une fonction qui prend en entrée une chaine de caractères, et qui retourne la chaine dans l'ordre inverse.

Exercice 0.23
Écrire une fonction qui prend en entrée une chaine de caractères et qui renvoie `True` si la chaine de caractères est un palindrome, `False` sinon.
☛ Il est possible d'utiliser la fonction précédente !

Exercice 0.24
Une sous-chaîne est une séquence consécutive de caractères à l'intérieur d'une autre chaîne.
La même sous-chaîne peut se trouver plusieurs fois à l'intérieur de la même chaîne : par exemple :

- 'expertiser' a la sous-chaîne 'er' 2 fois
- 'banane trans-panaméenne' a la sous-chaîne 'an' 4 fois

Écrire une fonction `sous_chaine(aiguille,bottedefoin)` qui prend deux lignes d'entrées, dont la première s'appelle *aiguille* et la seconde *bottedefoin* et qui renvoie nombre de fois que *aiguille* se produit comme une sous-chaîne de *bottedefoin*.

0.5 Les tableaux

0.5.1 Copie superficielle et copie profonde

Exercice 0.25
Analyser les lignes suivantes, et prévoir le résultat à l'appel de l1,l2,l3,l4,l5 et l6.

```
# copie superficielle
l1 = [5,6]
l2 = l1
l1.append(7)
l3 = [5,8]
l4 = l3
l4.append(9)
# copie profonde
l5 = [7,8]
l6 = copy.deepcopy(l5)
l5.append(9)
```

0.5.2 Parcours de tableau

Exercice 0.26
Écrire une fonction `get_max(liste)` qui prend en paramètre un tableau d'entiers et renvoie le max des éléments de celle-ci, sans ordonner la liste.

Exercice 0.27
Écrire une fonction `get_max_position(l)` qui prend en paramètre un tableau d'entiers `l` et qui renvoie le couple (le tuple) `(PG,IG)`, respectivement la plus grande valeur de la liste et la position de cette valeur maximale dans la liste.

Exercice 0.28
Cette fonction nommée `heure(t,d)` prend deux paramètres :

- un "temps de démarrage" exprimé dans une horloge de 24 heures avec des zéros, comme 08 :30 ou 14 :07.
- La deuxième ligne est une durée d de quelques minutes.

Le programme devra afficher la nouvelle heure, c'est-à-dire d minutes après l'heure de départ.
Par exemple, pour l'entrée `heure(12:30,47)`, la sortie correcte serait `13:17`. Toutes les heures doivent être formatées sous forme de nombres entre 00 :00 et 23 :59, mais le délai peut aller au-delà de minuit. Par exemple, sur l'entrée (23 :59,13), la sortie correcte est 00 :12.

Exercice 0.29
On s'intéresse à la création de tableaux en Python.

1. Créer un tableau de taille 10 contenant uniquement la valeur `False`. (On pourra utiliser la syntaxe de la multiplication d'une liste.)
2. Créer un tableau de tableaux (tableau 10×10) contenant uniquement la valeur `False`. (On pourra utiliser une boucle ou bien un produit de listes en compréhension.)
3. On définit d'abord

```
l = [False] * 10
tab = []
for i in range(10):
    tab.append(l)
```

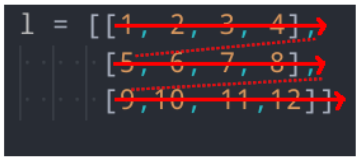
Expliquer quel est le problème posé par cette méthode et proposer une correction.

Exercice 0.30

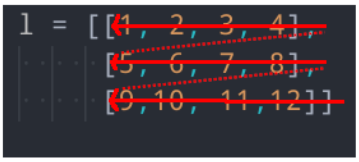
Soit

$$l = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]$$

1°) Parcours Largeur 1 Écrire un algorithme qui parcourt « en largeur » d'abord de gauche à droite puis de haut en bas, chacun des éléments de chacune des sous-listes (de la liste l), en l'affichant dans le Terminal. On pourra utiliser deux boucles `for` imbriquées.



2°) Parcours Largeur 2 Modifier l'algorithme précédent pour parcourir « en largeur » mais de droite à gauche et de haut en bas, donc tout à l'envers par rapport à la question précédente.



3°) Parcours Profondeur 1 Parcourir « en profondeur » d'abord de haut en bas et de gauche à droite, chacun des éléments de chacune des sous-listes (de la liste l).

4°) Parcours Profondeur 2 Parcourir « en profondeur » d'abord de bas en haut et de gauche à droite, donc tout à l'envers par rapport à la question précédente.

0.6 Dictionnaire

NSI terminale - JB Duthoit

Exercice 0.31

Antoine a obtenu les notes suivantes :

```
notes = {'nsi':18,'Maths':16,'SP':13,'Français':14, 'svt':10}
```

1. Créer une fonction `moyenne(d)`, de paramètre un dictionnaire `d`, et qui retourne la moyenne obtenue.
2. Créer une fonction qui produit un affichage du type "relevé de notes" :

```
Antoine a 18 en NSI
Antoine a 16 en Maths
...
Antoine a 10 en SVT
Antoine à ... de moyenne
```

#

Exercice 0.32

Écrire un programme Python pour convertir le nom du mois en un nombre de jours. Accédez à l'éditeur
Sortie attendue :

Liste des mois : janvier, février, mars, avril, mai, juin, juillet, août,, Septembre Octobre Novembre Décembre
Entrez le nom du mois : février
Nombre de jours : 28/29 jours

Exercice 0.33

Écrire un programme Python `compteur(d1,d2)` pour combiner deux dictionnaires en ajoutant des valeurs pour les clés communes.

```
d1 = {'a': 100, 'b': 200, 'c': 300}
d2 = {'a': 300, 'b': 200, 'd': 400}
assert compteur(d1,d2) == {'a': 400, 'b': 400, 'd': 400, 'c': 300}
```

#

Exercice 0.34

Écrire une fonction `transformation(chaine)` Python pour créer un dictionnaire à partir d'une chaîne.
Remarque : faire correspondre chaque lettre de la chaine par son nombre de lettres de la chaîne.

```
transformation("mathématiques") ==
{'m':2,'a':2,'t':2,'h':1,'é':1,'i':1,'q':1,'u':1,'e':1,'s':1 }
```

#

0.7 Tri

NSI terminale - JB Duthoit

Exercice 0.35

Écrire une fonction *tri_concatenation*(l1,l2) qui prend en argument deux listes triées A et B, et qui renvoie la liste triée C, C résultant de la concaténation de A et de B.

```
def tri_concatenation(l1,l2):
    '''
    l1 et l2 sont des listes triées
    la fonction retourne une nouvelle liste, qui est la concaténation triée de l1 et l2
    '''

    #
```

0.8 Autres

Exercice 0.36

Écrire un programme Python pour trouver les nombres qui sont divisibles par 7 et multiple de 5, entre 1500 et 2700 (tous deux inclus)

Exercice 0.37

Écrire une fonction Python `tableau(m,n)` qui prend en entrée deux chiffres m (ligne) et n (colonne) et génère un tableau à deux dimensions. La valeur de l'élément dans la i-ème ligne et la j-ème colonne du tableau doit être i * j.

Remarque :
i = 0,1 .., m-1
j = 0,1, n-1.

Données de test : Lignes = 3, Colonnes = 4 `tableau(3,4)` renvoie `[[0, 0, 0, 0], [0, 1, 2, 3], [0, 2, 4, 6]]`

Exercice 0.38

Réécrivez les nombres de l'entrée à la sortie. Arrêtez le traitement de l'entrée après avoir lu le nombre 42. Tous les nombres en entrée sont des entiers à un ou deux chiffres.
Entrée :
2
88
17
42
17
9
Sortie :
2
88
17