R1.01 Initiation au développement
Feuille de TP n°3
Prise en main de VSCode – Boucle `for` calcul cumulatif

Qu'est-ce qu'on fait aujourd'hui ?

Cette feuille a pour objectifs principaux :

- Comprendre le fonctionnement de la boucle for
- Utiliser la boucle for pour faire des calculs cumulatifs
- Utiliser le débugger pour comprendre le fonctionnement d'une fonction

Exercice 1 *Fonction mystère*

1.1 En utilisant le débugger sur le fichier `tp3_sources.py`, exécutez la fonction `mystere_exo2` avec les entrées `[1,4,6,-2,-5,3,10]` et `[-4,5,-11,-56,5,-11]`

1)

1.2 Indiquez en commentaire ce que contiennent les variables `xxx` et `yyy` au début de chaque tour de boucle.

xxx= 4 yyy=3

1.3 Renommez la fonction et les variables afin que les noms soient significatifs et complétez la docstring

1.4 Ecrivez la fonction de test qui effectue au moins 4 assert

Exercice 2 *Débuggage d'une boucle*

La fonction `min_sup` du fichier `tp3_sources.py` doit calculer dans une liste le plus petit élément supérieur à une certaine valeur. Or lorsqu'on tape dans le terminal

```
pytest-3 -v tp3_sources.py
```

on s'aperçoit que la fonction n'est pas correcte.

2.1 Exécutez la fonction avec le débugger et essayez de déterminer pourquoi la fonction ne retourne pas toujours le bon résultat.

2.2 Corrigez la fonction pour qu'elle passe les tests.

Exercice 3 *Débuggage d'une boucle (bis)*

La fonction `nb_mots` du fichier `tp3_sources.py` doit calculer le nombre de mots contenus dans une phrase en se basant sur les espaces. Il s'avère que la fonction ne passe pas les tests fournis.

3.1 Exécutez la fonction avec le débugger et essayez de déterminer pourquoi la fonction ne retourne pas toujours le bon résultat.

3.2 Complétez les commentaires pour indiquer ce que valent les variables au début de chaque tour de boucle

3.3 Corrigez la fonction pour qu'elle passe les tests.

Exercice 4 *Ecrire des boucles*

Pour chacune des fonctions demandées, vous devrez la documenter avec une docstring, écrire une fonction de tests comportant au moins 4 `assert` différents et indiquer dans un commentaire ce que valent vos variables de calcul au début de chaque tour de boucle.

4.1 Ecrire une fonction qui retourne la somme des nombres pairs d'une liste d'entiers.

Par exemple sur la liste `[12,13,6,5,7]` le résultat doit être `18` car seuls `12` et `6` sont pairs.

4.2 Ecrire une fonction qui retourne la dernière voyelle d'une chaîne de caractères.

Par exemple pour la chaîne `"buongiorno"` le résultat doit être `"o"` et pour la chaîne `"bonjour"` le résultat doit être `"u"`. Si la chaîne ne contient aucune voyelle le résultat doit être `None`.

4.3 Ecrire une fonction qui donne la proportion de nombres strictement négatifs dans une liste.

Par exemple pour la liste `[4,-2,8,2,-2,-7]` le résultat doit être `0.5` car la liste contient 3 nombres négatifs sur 6. Si la liste est vide, le résultat doit être `None`.

Exercice 5 *La fonction Range*

Comme pour l'exercice précédent, vous devrez documenter avec une docstring chaque fonction demandée et écrire une fonction de tests comportant au moins 4 `assert` différents. Vous devrez aussi indiquer dans un commentaire ce que valent vos variables de calculs au début de chaque tour de boucle.

5.1 Ecrire une fonction qui fait la somme des `n` premiers entiers

Par exemple si `n` vaut `4` le résultat doit être `10` qui est le résultat de `1+2+3+4`.

5.2 Ecrire une fonction qui permet de calculer le terme U_n d'une suite de Syracuse

Une suite de Syracuse se définit à partir d'un nombre val_init strictement positif de la manière suivante :

- $U_0 = val\_init$
- $U_n = \begin{cases} U_{n-1}/2 & \text{si } U_{n-1} \text{ est pair} \\ 3 \times U_{n-1} + 1 & \text{si } U_{n-1} \text{ est impair} \end{cases}$

Par exemple pour val_init égale à 6, U_0 vaut 6, U_1 vaut $6/2 = 3$, U_2 vaut $3 \times 3 + 1 = 10$ etc. Remarquez que pour calculer U_i, il faut calculer tous les termes de U_0 à U_{i-1}

Exercice 6 *Des boucles suite*

Chaque fonction devra être commentée par une docstring et un commentaire indiquant ce que doivent contenir les variables de calcul au début de chaque tour de boucle. Vous devrez fournir une fonction de test comprenant au moins 4 `assert` différents.

Implémentez les fonctions de la feuille TD2.