L2 MIASHS – Informatique S3
TD5 : Erreurs et tests

L'objectif de ce TD est d'acquérir des réflexes pour évaluer la correction de son code. Les question qu'on va se poser sont donc :

— est-ce que mon code s'exécute ?
— est-ce qu'il a le comportement attendu ?
— comment localiser les problèmes ?

Exercice 1

Pour commencer on va travailler sur le fichier `tri.py` qui contient une fonction censée trier une liste de nombres par ordre croissant.

1. Commençons par tester le fichier : l'exécution produit une erreur. Quel est le message d'erreur ? Quelle information cela nous donne-t-il ? Corrigez le code de manière à ce qu'il s'exécute.

   Indication :

   ```
   File "/home/shplin/Documents/Enseignement2020/2020S5/TD3/Correction/sanstitre1.py",
   line 5, in tri
       if l[i]>l[i+1] :
   IndexError: list index out of range
   ```

   — Le message d'erreur ci-dessus se lit de la manière suivante : Dans la fonction `tri`, ligne 5, il y a une erreur de type `IndexError`. Plus précisément cette erreur est `list index out of range`.
   — Cela signifie qu'on essaye d'accéder à un élément de la liste, qui n'existe pas (l'indice n'est pas bon).
   — Pour comprendre, je propose d'ajouter dans la boucle, juste avant la ligne concernée (celle du if), un affichage `print(i,i+1)` pour voir quelle vlaeur prennent les indices et où est le problème.
   — Rappel : les indices d'une listes vont de `0` à `len(l)-1`.

2. Le résultat affiché correspond-il à ce qu'on attendait d'une fonction qui trie une liste de nombre. Cela suffit-il à affirmer que le programme est correct ? Pourquoi ?

3. Tester la fonction avec la liste `[12,3,54,3,4,43,1]`. Quel est le problème ? Corrigez le code de manière à ce qu'il corresponde à ce qu'on attendait : "trier une liste de nombres par ordre croissant".

   Indication : Il peut être utile d'ajouter cette fois-ci un affichage de la liste à chaque passage dans la boucle `for n in...`, pour voir ce qui coince.

Quelques remarques :

Une étape essentielle pour s'assurer de la correction d'un programme est de spécifier précisément ce qu'on attend de lui.

Exemple : pour "une fonction qui trie une liste". On peut se demander

— de quel type sont les éléments de la liste : des entiers ? des chaînes de caractères ?

— dans quel ordre doit-on trier ? croissant ? décroissant ? lexicographique ?

— que se passe-t-il si ces conditions ne sont pas remplies ?

— que se passe-t-il si la liste est vide ?

Ici, la fonction du fichier `tri.py` triait donc une liste de nombre, par ordre croissant, et en ne traitant que le cas de listes ne contenant que des nombres. Une liste vide était supposée triée (on la renvoie telle quelle).

En fait, souvent dans nos exercices, ils sont assez simples et précis qu'il n'y ait pas besoin de préciser. Par contre il reste la deuxième étape : s'assurer que les objectifs sont atteints. Et à notre niveau, le plus simples est de faire cela avec des tests.

Pour finir, listez les tests qui ont été nécessaires pour trouver les erreurs dans cet exercice. Quels tests supplémentaires aurait-on du ajouter pour s'assurer un peu plus de la correction du programme ?

Exercice 2

L'objectif de cet exercice est d'écrire une fonction qui prend une chaîne de caractère comme paramètres et retourne l'ensemble des caractères les plus présents dans la chaîne. *Exemples :* Pour la chaîne, `"Hello a tous"`, la fonction retourne : `{'l', 'o', ' '}`. Pour la chaîne, `"Bonjour a toi mon ami"`, la fonction retourne : `{'o',' '}`

On peut aborder se problème en se demandant comment le décomposer en sous-fonctions.

On peut décomposer les opérations à faire pour répondre au problème en trois parties :

1. la construction d'un dictionnaire des occurences des lettres dans la chaîne,
2. la recherche de la plus forte valeur du dictionnaire,
3. la construction d'une liste des clés du dictionnaire dont la valeur associée est cette plus forte occurence.

En découpant ainsi, on peut écrire trois fonctions qui répondent à chacune de ces opérations.

Si on les écrit, puis on teste que chacune fait bien ce qu'elle est censée faire, on obtient ainsi une garantie sur le total. Pour cela il faut écrire précisément ce qu'on attend de chaque fonction :

1. Une fonction `diccocc(s)` qui prend une chaîne `s` en paramètre et retourne le dictionnaire des occurences des aractères de cette chaînes
2. Une fonction `maxival(d)` qui prend un dictionnaire comme paramètre et retourne le max des valeurs du dictionnaire.
3. Une fonction `cles(d,v)` qui prend comme paramètre un dictionnaire `d` et une valeu `v` et retourne l'ensemble des clés de `d` dont la valeur associée est `v`

4. Une fonction `plusfrequent(s)` qui prend une chaîne de caractère en paramètre et retourne l'ensemble des caractères les plus fréquents dans la chaîne. (en utilisant les autres fonctions).

   Ecrivez ces 4 fonctions, vérifiez leur fonctionnement en testant chacune d'elle.