Python: conteneurs

1 Exercices d'échauffement

Exercice 1 - Manipulation de listes

Soit une liste `L = [23,34,23,56,45,12]`

- Ajouter la valeur 11 à la fin de la liste.
- Ajouter les valeurs 12 et 13 à la fin de la liste.
- Afficher le premier élément, les deux premiers éléments, le dernier élément, les deux derniers éléments.
- Ajouter la valeur 1 à chacun de ses éléments (trois manières différentes : +, for et intention)
- Construire la liste "paires" qui contient les nombres paires de `L` et la liste "impaire" qui contient les nombres "impaires" de `L` (2 manières différentes : boucle for et liste compréhension)
- Ajouter la valeur 3.5 entre 3 et 4.
- Supprimer la valeur 3.5.
- Inverser l'ordre des éléments de `L`.
- Demander à l'utilisateur de fournir un nombre au hasard et dire si ce nombre est présent dans `L`.

Exercice 2 - Écrire un programme qui demande cinq nombres à l'utilisateur, puis les affiche dans l'ordre croissant :

```
Entrez un nombre: 1
Entrez un nombre: 45
Entrez un nombre: 65
Entrez un nombre: 3
Entrez un nombre: 1
[1, 1, 3, 45, 65]
```

Exercice 3 - Manipulation de dictionaires

On donne le dictionnaire ci-dessous :

```
d = {'nom': 'Dupuis', 'prenom': 'Jacque', 'age': 30}
```

- Corriger l'erreur dans le prénom, la bonne valeur est '`Jacques`'.
- Afficher la liste des clés du dictionnaire.
- Afficher la liste des valeurs du dictionnaire.
- Afficher la liste des paires clé/valeur du dictionnaire.
- Ecrire la phrase "Jacques Dupuis a 30 ans".

2 Algorithmes sur …

A Les listes

Pour se simplifier, on ne considère que des listes qui contiennent des nombres … mais elles pourraient contenir toute sorte d'objets !

On considère les listes suivantes :

```
L = [23,34,23,56,45,12]
L2 = [45,56,67,78]
```

Exercice 4 - Calcul de la somme

Q.1) Écrire un programme qui calcule la somme des éléments de la liste L

Exercice 5 - Calcul du maximum

Q.1) Écrire un programme qui calcule la valeur maximum de la liste L

Q.2) Modifier le programme pour que celui-ci calcule l'argmax, c'est-à-dire l'indice auquel est stockée la valeur maximale de la liste.

Exercice 6 - Écrivez un programme qui calcule l'intersection des deux listes (c'est-à-dire une liste contenant tous les éléments présents dans les deux listes) [sans utiliser l'opération + entre listes]

B Dictionnaires

Exercice 7 -

On dispose d'un dictionnaire associant à des noms de commerciaux d'une société le nombre de ventes qu'ils ont réalisées. Par exemple :

```
ventes={"Dupont":14, "Hervy":19, "Geoffroy":15, "Layec":21}
```

Q.1) Écrire un programme qui calcule le nombre total de ventes dans la société.

Q.2) Écrire un programme qui donne le nom du vendeur ayant réalisé le plus de ventes. Si plusieurs vendeurs sont ex-aequo sur ce critère, donner le nom de l'un d'entre eux.

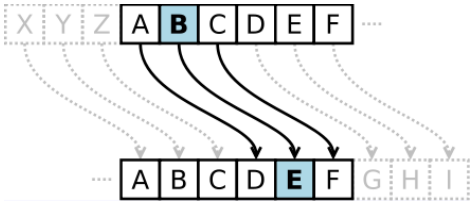
Exercice 8 - Histogramme

Écrire un programme qui affiche le nombre de tirages de la somme de deux dés, sur 10000 tirages successifs, pour chaque somme possible. Les nombres seront stockés dans un dictionnaire, indexé par la somme des tirages. Le résultat devrait ressembler à ça:

```
somme = 2 : 290 tirages
somme = 3 : 593 tirages
somme = 4 : 827 tirages
somme = 5 : 1137 tirages
somme = 6 : 1411 tirages
somme = 7 : 1648 tirages
somme = 8 : 1343 tirages
somme = 9 : 1115 tirages
somme = 10 : 794 tirages
somme = 11 : 579 tirages
somme = 12 : 263 tirages
```

Exercice 9 - Codage par décalage

Le codage par décalage est une technique (simple) d'encryptage d'un texte. Elle consiste à associer un caractère de l'alphabet à une une autre lettre en les décalant d'une valeur donnée.



Les deux lignes suivantes permettent de construire une liste de caractères (à tester dans un terminal) :

```
import string
chars = list(string.ascii_lowercase)
```

Q.1) Définir une variable `decal`*, puis construire un dictionnaire qui associe chacun des caractères de l'alphabet avec le caractère décalé (permutation circulaire) de* `decal`.

Q.2) Le dictionnaire est une structure de données qui assure la correspondance entre les caractères en clair et le caractéres codés !

Q.3) Définir un texte dans une variable `text_clair` *puis écrire le programme de sorte à construire une chaîne codée (réfléchir à l'algorithme en amont)*

Pour manipuler les chaînes de caractères comme des listes, il est possible de transformer les textes en une liste de caractères et réciproquement en vous aidant de l'exemple ci-dessous :

```
text_clair='ceci est totalement secret'
l = list( text_clair )
print(l)
new_texte=''.join(l)
print(new_texte)
```

Pour les caractères qui ne sont pas dans l'alphabet (les espaces, les ponctuations, etc) vous les supprimerez du texte codé.

Q.4) Modifier la construction du dictionnaire en construisant un alphabet aléatoire. La difficulté est ici d'assurer qu'un caractère n'est associé qu'à un seul et un seul caractère codé. La solution consiste alors à utiliser la fonction `random.shuffle(l)` *qui réordonne aléatoirement une liste. Proposer un méthode pour ce codage. Quel est son inconvénient majeur par rapport au codage précédent ?*

Exercice 10 - Calcul des points

Q.1) Écrire un programme qui commence par définir un dictionnaire associant à un nom une liste de points obtenus à un jeu. À chaque fois qu'un utilisateur joue, il a un nouveau score et tous les scores doivent être collectés : quelle structure de données allez vous utiliser ?

Q.2) Ecrire à la suite, un programme qui permet de saisir un nouveau score : on donne le nom du joueur et son score

Pour faciliter la suite de l'exercice, il est conseillé de rentrer manuellement des données dans le programme pour éviter d'avoir à saisir tout depuis le début.

Q.3) Une fois la saisie effectuée, donner le meilleur score de chaque joueur

Q.4) Déterminer le gagnant du jeu (le joueur qui a le meilleur score) ; si il y a des ex aequo, vous vous contenterez de donner le nom d'un des joueurs

Q.5) Déterminer le joueur le plus régulier (celui qui a la meilleure moyenne de scores) ; si il y a des ex aequo, vous vous contenterez de donner le nom d'un des joueurs

3 Exemple d'application : retour sur les fichiers

Exercice 11 - Traitement d'un collection de fichiers

L'objectif est de reprendre l'exemple du TP précédent pour lequel vous aviez réalisé un programme permettant de générer une commande de transformations géométriques d'une image à l'aide de `gdalwarp`. La solution permettait de faire le traitement d'un unique fichier donné par l'utilisateur.

Dans cet exercice, on vous fait utiliser le module `glob` qui permet de lister facilement des fichiers.

Q.1) En utilisant le module `glob`*, lister l'ensemble des fichiers* txt *dans un répertoire saisie par l'utilisateur. Vous utiliserez pour cela la fonction* `glob.glob('*.txt')`.

Q.2) Modifier le code du TP précédent de sorte à traiter avec `gdalwarp` *tous les fichiers* tif *d'un répertoire donné par l'utilisateur.*

Q.3) Compléter votre code en collectant pour chaque image traité le temps de calcul de la

transformation (voir ci-dessous pour le calcul des temps).

q.3.1) Quelle structure de données allez vous utiliser ?

```
import time
tmps1=time.clock()
# code
# code
tmps2=time.clock()
print("Temps d'execution = {}\n".format(tmps2-tmps1) )
```