ITI 1520 - Devoir 4

Disponible: le 19 octobre, 2020
Date de remise: lundi, le 2 novembre, 2020, 8:00
SVP notez que le devoir n'est pas accepter après cette date.

Vous devez faire ce travail **individuellement**. Vous devez soumettre un fichier d4_VOTRE_NUMERO_ETUDIANT.zip contant un fichier pour chaque question :
d4q1_VOTRE_NUMERO_ETUDIANT.py
d4q2_VOTRE_NUMERO_ETUDIANT.py
d4q3_VOTRE_NUMERO_ETUDIANT.py
d4q4_VOTRE_NUMERO_ETUDIANT.py

Les noms des fichiers et les noms des fonctions doivent être les noms requises, parce qu'on va utiliser des tests automatiques pour la correction. Si le nom d'une fonction n'est pas le nom requis, la note sera zéro pour la fonction.
Ajouter des commentaires pour chaque fonction (avec une description courte, contrat de type et preconditions).
Si un fichier .py donne erreur de syntaxe, la note sera zéro pour la question.
Si vous envoyez le devoir plusieurs fois, la dernière version sera note.
SVP noter qu'on va utiliser un outil logiciel pour détecter du plagiat. En cas deux devoirs sont identique ou très similaires, la note sera zéro pour les deux.
Pour ce devoir ce n'est pas permis d'utiliser des *sets* est des *dictionnaires* Python.

Barème : total de 20 points. Devoir 4 est 7% de la note finale.

Question 1 (2 points) Créez une fonction Python, appelée nombreDivisibles, qui prend une liste de nombres et un entier positive n, et qui retourne le nombre d'éléments divisible par n trouvés dans la liste. Dans la partie principale du programme, demandez à l'usager d'introduire la liste et l'entier n, invoquez la fonction et affichez le résultat.

Exemple avec le programme principal :

Veuillez entrer une liste des entiers par des espaces: 1 4 2 5 8
Veuillez entrer un entier positif: 2
Le nombre des éléments divisibles par 2 est 3

Exemples d'appels de la fonction dans l'interpréteur Python:
>>> nombreDivisibles([10, 2, 3, 4, 5, 6], 3)
2

Suggestion : Pour lire une liste de clavier, vous pouvez utilise, par exemple:

```
s = input('Veuillez entrer une liste des entiers par des espaces:').strip().split()
a=[]
for item in s:
     a.append(int(item))
```

Question 2 (3 points) Créez une fonction Python, appelée sequenceDesDeux, qui prend une liste de nombres et qui donne True s'il y a au moins une séquence de deux éléments consécutifs égaux, et False dans le cas contraire. Dans la partie principale du programme, demandez à l'usager d'introduire la liste, appelez la fonction et affichez le résultat. Assurez-vous que la fonction est efficace et qu'elle s'arrête dès que le résultat est connu.

Exemples (plusieurs exécutions du programme principal):

Veuillez entrer une liste des valeurs séparées par des espaces: 3 1.0 1.0 7 8.5
True

Veuillez entrer une liste des valeurs séparées par des espaces: 1 6 3 3 3
True

Veuillez entrer une liste des valeurs séparées par des espaces: 2 1 5 3.6 1 2 1
False

Veuillez entrer une liste des valeurs séparées par des espaces: 5
False

Exemple d'appel de la fonction dans l'interpréteur Python:

```
>>> sequenceDesDeux([1,5,5.0,4])
True
>>> sequenceDesDeux([])
False
>>> sequenceDesDeux([1,5,3])
False
```

Question 3 (3 points) Créez une fonction Python, appelée sequenceMax, qui prend une liste de nombres et qui donne la longueur de la plus longue séquence d'éléments consécutifs égaux. Entrez 1 s'il n'y a aucune séquence. Dans la partie principale du programme, demandez à l'usager de saisir la liste, appelez la fonction et affichez le résultat.

Exemples (plusieurs exécutions du programme principal):

Veuillez entrer une liste des valeurs séparées par des espaces: 1 2 1 3 3 3 6 1 1
3
Veuillez entrer une liste des valeurs séparées par des espaces: 1 5.5 2 2 2 7 3 3 3 3
4
Veuillez entrer une liste des valeurs séparées par des espaces: 3 1.0 7 1.0
1

Exemples des tests dans l'interpréteur Python :

```
>>> sequenceMax([])
1
>>> sequenceMax([1,2.4,1,1,1,1,6,7])
4
>>> sequenceMax([2,3,4])
1
```

Question 4 (12 points) Jeu des Scrabble et Anagrammes

Pour cette question, implémentez un programme qui trouve les anagrammes de mots, pour aider quand vous jouez un jeu de scrabble.

Pour deux mot (séquences des lettres) w1 et w2, ont dit que w2 est une *anagramme* de w1 si :
1. w2 et w1 ont exactement les mêmes lettres dans n'importe quel ordre (w2 est une permutation de w1) et
2. w2 est un mot dans la langue française (pour ce devoir on utilise la liste des mots en français donner dans le fichier french_noaccents.txt).

EXEMPLES:
lapin est une anagramme de *alpin*
mais est une anagramme de *amis*
pinla n'est pas une anagramme de *lapin* (ils ont les mêmes lettres, mais ils ne sont pas des anagrammes parce que *pinla* n'est pas un mot en français)
amis n'est pas une anagramme de *amies* parce qu'ils n'ont pas exactement les mêmes lettres.

Téléchargez le fichier etudinats.zip et étudiez le fichier test_q4.txt pour voir des exemples exécution. Il y a aussi des exemples des tests dans les commentaires des fonctions dans le fichier que vous devrez compléter d4q4_xxxxxx.py.
Remplacez xxxxxx dans le nom du fichier avec votre numéro d'étudiant. Exécutez le fichier pour voir que-ce que le code déjà donne fait (starter code). Après ça, ajutez votre code pour compléter ce ficher, dans les endroits indiquées. Il y a quelques fonctions qui sont déjà complètes. Vous devez compléter les autres. Ce n'est pas permis de modifier le code donné, sauf les parties avec l'instruction *pass*, qui doit être remplacé par votre code.

Vous devez aussi compléter le programme principal dans les endroits indiquées. C'est permis d'ajouter des fonctions au besoin. Dans le programme principal l'usager peut choisir option 1 pour analyser les mots d'un fichier (par exemple le fichier small_fr.txt ou autre fichier que vous créez aves de phrases on français, sans les accents). L'analyse inclue quels mots ont le nombre maximal des anagrammes et quels mots n'ont pas des anagrammes; aussi quels mots ont exactement k anagrammes, pour un k donné par l'usager (k >=0). Comme option 2, l'usager peut demander d'aide pour le jeu de scrabble, avec toutes les lettres données ou avec ces lettres sauf une d'entre eux. Si l'usage entre autre chose que 1 ou 2, le programme s'arrête.

Quelques suggestions:
C'est recommandé de coder toute les fonctions avant de compléter le programme principal. Ça aide de compléter les fonctions en ordre parce qu'elles utilisent les fonctions définies avant. Sauf pour les fonctions word_anagrams et create_clean_sorted_nodupicates_list l'ordre n'est pas importante.

La fonction test_letters(w1, w2) détermine si deux mots w1 et w2 ont exactement les mêmes lettres dans n'importe quel ordre (ils sont des permutations). Il faut trouver un algorithme pour ça.
Au besoin, vous pouvez convertir une chaine des caractères dans une liste ou une liste dans une chaine des caractères. Voici des exemples :

```
s='abcd'
q=list(s)   # donne la liste ['a', 'b', 'c', 'd']
q.append('z')
```

nstr=''.join(q) # donne la chaine 'abcdz'

La méthode join de la classe String prend chaque élément de la liste (q dans ce cas) et les ajoute à la chaine (dans ce cas la chaine vide ''). Si la chaine est ':' elle ajoute le caractère : entre les éléments
':'.join(myList) # donne 'a:b:c:d:z'

Notes pour le fichier declaration_VOTRE_NOM.txt si vous avez besoin:
Ce fichier doit contenir la référence pour code qui n'est pas écrit par vous même **si c'est le cas**. Ça inclue code donné par un collègue ou autre personne, ou trouvé sur l'internet, réseaux sociale (comme Stack Overflow, discord, ou autre). Ça n'inclue pas le code donne en BrightSpace dans les notes de classe, labo, etc.
Pour chaque question ou vous avez utilisé du code donné ou trouvé, il faut:
1. le numéro de question
2. copier-coller le code emprunter. Ça inclue code donné/trouvé et modifier très peu.
3. le nom de la source: personne, site Internet ou autre
Vous allez perdre des points la question, mais au moins vous évitez une accusation de plagiat. En cas de plagiat, la note est zéro et le cas doit être rapporté au doyen. Le même est valable si quelqu'un montre son code a un collègue.
Si vous n'utilisez pas cette déclaration, c'est équivalent à déclarer que tous le code a été écrit par vous-même.