Institut National Universitaire Champollion

L2 INF

Projet Module Algo avancée

Anagrammes

Vous placerez dans un répertoire nommé Votrenom_Votreprenom *vos fichiers python 3.4 ainsi qu'un fichier d'explications au format pdf*
Comme d'habitude, vous zipperez le répertoire et déposerez le fichier zip obtenu sur moodle dans la zone réservée.

Deux mots sont anagrammes s'ils sont constitués des mêmes lettres sans tenir compte des accents ou des majuscules. Par exemple "châtain" et "chantai" ou bien "Pancho" et "chapon" ou encore "élancé", "enlacé" et "lancée" sont anagrammes.
L'ensemble des mots utilisés dans ce projet est le dictionnaire du scrabble fourni : ods.txt
Dans ce projet, tous les mots sont supposés écrits en lettres majuscules sans accents.

1 Travail à réaliser

1.1 Anagrammes stricts

Ecrire les fonctions :

1. est_anagramme qui prend deux mots en paramètres et renvoie un booléen vrai si ces deux mots sont anagrammes.
2. anagrammes qui prend en paramètre un mot et renvoie la liste de tous ses anagrammes present dans le dictionnaire fourni (fichier ods.txt) .
3. best qui prend une longueur de mot $\ell \geq 2$ et renvoie la liste des mots de longueur ℓ qui possèdent le plus d'anagrammes. On obtiet par exemple :
   · 15 mots de longueur 2 possédent 2 anagrammes
   · 1 mots de longueur 3 possédent 4 anagrammes : AIR
   · 3 mots de longueur 4 possédent 6 anagrammes : RUSE TSAR SALI

Avec l'aide éventuelle des fonctions précédentes, répondre aux questions :
Combien de mots possèdent un nombre maximal d'anagrammes (sans compter leurs anagrammes !) ? Les mots qui ont le plus grand nombre d'anagrammes ont-ils tous la même longueur ?

1.2 Anagrammes non stricts (Optionnel)

On appelle n-presque-anagramme d'un mot tout mot contennant toutes les lettres du mot de départ plus n lettres **supplémentaires**.
Ecrire les fonctions :

1. est_presque_anagramme qui prend deux mots et un entier $n \geq 0$ en paramètre et renvoie un booléen vrai si les deux mots sont presque-anagrammes pour la valeur n. Par exemple :
   · est_presque_anagramme("PAR","DRAP",1)=True.
   · est_presque_anagramme("PAR","DRAPA",2)=True.
2. presque_anagrammes qui prend en paramètre un mot et un entier $n \geq 0$ et renvoie la liste de tous ses "n-presque-anagrammes" .
   Par exemple : presque_anagramme("PAR",1) est constituée des 20 mots : PARA, RAPA, PARC, DRAP, APRE, EPAR , PARE, RAPE, PAIR, PARI, PRIA, RIPA, PRAO, PARS, RAPS, PART, PARI, RAPT, PARU, RUPA

Dans la suite, on suppose $n = 1$.
Combien de mots possèdent un nombre maximal de presque-anagrammes !) ? Les mots qui ont le plus grand nombre de presque-anagrammes ont-ils tous la même longueur ?

1.3 Anagrammes stricts composés de plusieurs mots

Dans cette partie on fournit une chaîne de caractères et on cherche des mots dont la concaténation est une anagramme strict de la chaîne de départ. Par exemple :
– ETFROMAGE donne 192 solutions en particulier MEGA FORTE, ET FERMA GO, RE FAT ME GO,...
– SOURIS donne 32 solutions en particulier : ROUSSI, OURS SI, SOIS RU, OR SI US,...
Ecrire les fonctions :

1. A1 qui prend un mot et renvoie la liste des listes de mots anagrammes du mot de départ. Par exemple :
   · A1("ROSE")=[[EROS],[ES, OR],[OR,SE],[ORES],[OS, RE],[OSER],[SORE]]. 8 solutions
   · A1("PROSE")=[[ES,PRO],[OR,SEP],[OS,PRE],[PERSO],[PORES],[POSER], [PRO, SE],[PROSE],[PSORE],[REPOS],[SPORE]]. 11 solutions
   · A1("CHAMPOLION") donne 599 solutions dont CHAPON MOLLI, LOCH LAMPION, MOLLAH PICON ou OH CLIM LAPON...
2. A2 qui fait comme A1 et ne renvoie que les solutions constituées au plus de n mots. Par exemple :
   · A2("CHAMPOLLION",1)=[]. pas de solutions
   · A2("CHAMPOLLION",2) contient 11 solutions
   · A2("CHAMPOLION",3) donne 261 solutions
   · A2("CHAMPOLION",4) donne 599 solutions
   · A2("CAROLINEETFLORIAN",2) donne 445 solutions dont CORRELATION FINALE et AIOLI CONFRATERNEL

2 Barème indicatif

- Codage : 14 pts
  – Anagrammes stricts : 6
  – Anagrammes stricts de plusieurs mots : 6

  - Option : anagrammes non stricts ou idées personnelles : 2
- Tests : vous rendrez au moins 2x3=6 fichiers. Pour chaque partie, un fichier de fonctions et un fichier de tests :p11.py, Tests_p11.py, option.py, Test_soption.py, A.py, Tests_A.py
  Au maximum la note de codage peut être diminuée de 5 pts selon la qualité des tests unitaires.
- Forme : 2pts
  Explications dans le code quand cela est nécessaire, cartouches pour les fonctions (décrivant les Entrées, les Sorties et l'action accomplie)sous forme de triplets de Hoare éventuellement en français, choix des identificateurs...
- Document texte explicatif : 4pts
  Il doit être au format pdf, contenir 4 pages maximum. Dans ce document, vous préciserez :
  1. Qu'est ce qui a été traité ? testé ? qu'est ce qui est fonctionnel ?
  2. Quels choix d'algorithmes avez-vous fait ? sur quels critères ?
  3. Quelles sont les limitations du programme : temps d'exécution, taille des chaînes...
  4. Quels sont les problèmes rencontrés.