KOURTIN Ismahane
IFAA SUP PARIS

Année Universitaire
2020-2021

Programmation
TP 6

Exercice 1 :

Écrire un programme qui lit une chaine de caractères et qui renvoie un dictionnaire de fréquence de chaque lettre dans la chaîne.

Exemple :

Donner votre chaine : brontosaurus

Renvoie :

{'a': 1, 'b': 1, 'n': 1, 'o': 2, 'r': 2, 's': 2, 't': 1, 'u': 2}

Exercice 2 :

Monsieur Germain est une personne très âgée. Il aimerait préparer une liste de courses à faire à l'avance. Ayant un budget assez serré, il voudrait que sa liste de courses soit dans ses capacités. Son seul petit souci est qu'il a une très mauvaise vue et n'arrive donc pas à voir le prix associé à chaque produit contenu dans le catalogue de courses.

Écrire une fonction « calcul_prix(produits, catalogue) » où :

- produits : est un dictionnaire contenant, comme clés, les produits souhaités par Monsieur Germain et comme valeurs associées, la quantité désirée de chacun d'entre eux,
- catalogue : est un dictionnaire contenant tous les produits du magasin avec leur prix associé.

La fonction retourne le montant total des achats de Monsieur Germain.

Exemple :

L'appel suivant de la fonction :

calcul_prix({"brocoli":2, "mouchoirs":5, "bouteilles d'eau":6}, {"brocoli":1.50, "bouteilles d'eau":1, "bière":2, "savon":2.50, "mouchoirs":0.80})

Renvoie :

13.0

Exercice 3 :

Lors de prises de notes, il nous arrive souvent de remplacer des mots par des abréviations (bonjour est remplacé par bjr par exemple). Nous allons utiliser un dictionnaire qui associe à chacune de ces abréviations sa signification.

Écrire une fonction « substitue(message, abreviation) » qui renvoie une copie de la chaîne de caractères « message » dans laquelle les mots qui figurent parmi les clés du dictionnaire « abreviation » sont remplacés par leur signification (valeur).

Exemple :

L'appel suivant de la fonction :

substitue("C. N. cpt 2 to inf", {"C." : "Chuck", "N." : "Norris", "cpt" : "counted", "2" : "two times", "inf" : "infinity"})

Renvoie :

Chuck Norris counted two times to infinity

Exercice 4 :

Écrire une fonction « valeurs(dico) » qui doit fournir, à partir du dictionnaire donné en paramètre, une liste des valeurs du dictionnaire triées selon leur clé.

Exemple :

L'appel de la fonction suivant :

valeurs({"three": "trois", "two": "deux", "one": "un"})

Renvoie :

['un', 'trois', 'deux']

Exercice 5 :

Un jury doit attribuer le prix du « Codeur de l'année ».

Afin de récompenser les trois candidats ayant obtenu la meilleure moyenne, nous vous demandons d'écrire une fonction « top_3_candidats(moyennes) » qui reçoit un dictionnaire contenant comme clés les noms des candidats et comme valeurs la moyenne que chacun a obtenue. Cette fonction doit retourner une liste contenant les noms des trois meilleurs candidats, par ordre décroissant de leurs moyennes.

Exemple :

L'appel suivant de la fonction :

top_3_candidats({"Candidat 7": 2, "Candidat 2": 38, "Candidat 6": 85, "Candidat 1": 8, "Candidat 3": 17, "Candidat 5": 83, "Candidat 4": 33})

Renvoie :

("Candidat 6", "Candidat 5", "Candidat 2")

Exercice 6 :

Écrire une fonction « compteur_lettres(texte) » qui renvoie un dictionnaire contenant toutes les lettres de l'alphabet associées à leur nombre d'apparition dans « texte ».

Les clés du dictionnaire seront les lettres de l'alphabet en minuscule. Si le texte contient des majuscules, celles-ci seront comptabilisées comme la lettre minuscule correspondante.

Le texte passé en paramètre ne comportera aucun caractère accentué.

Les espaces et autres caractères de ponctuation doivent être ignorés.

Exemple :

L'appel de la fonction suivant :

compteur_lettres("Dessine-moi un mouton !")

Renvoie :

{'a': 0, 'b': 0, 'c': 0, 'd': 1, 'e': 2,
'f': 0, 'g': 0, 'h': 0, 'i': 2, 'j': 0,
'k': 0, 'l': 0, 'm': 2, 'n': 3, 'o': 3,
'p': 0, 'q': 0, 'r': 0, 's': 2, 't': 1,
'u': 2, 'v': 0, 'w': 0, 'x': 0, 'y': 0,
'z': 0}

Exercice 7 :

Écrire une fonction « store_email(liste_mails) » qui reçoit en paramètre une liste d'adresses e-mail et qui renvoie un dictionnaire avec comme clés les domaines des adresses e-mail, et comme valeurs les listes d'utilisateurs correspondantes, triées par ordre croissant (UTF-8).

Exemple :

L'appel de la fonction suivante :

store_email(["ludo@prof.ur", "andre.colon@stud.ulb", "thierry@profs.ulb", "sébastien@prof.ur", "eric.ramzi@stud.ur", "bernard@profs.ulb", "jean@profs.ulb"])

Retourne le dictionnaire :

{ "prof.ur" : ["ludo", "sébastien"], "profs.ulb" : ["bernard", "jean", "thierry"], "stud.ulb" : ["andre.colon"], "stud.ur" : ["eric.ramzi"] }

Exercice 8 :

1) Écrire une fonction « const_dict_amis » qui reçoit une liste de couples « (prénom1, prénom2) » voulant dire que « prénom1 » déclare « prénom2 » comme étant son ami et qui construit un dictionnaire dont les clés sont les prénoms des personnes nommées (pour simplifier on suppose que deux personnes n'ont pas le même prénom) et la valeur de chaque entrée est la liste des amis de la personne trié par ordre croissant.

Si, dans la liste reçue, nous avons le couple « (prenom1, prenom2) », cela n'induit pas que « prenom1 » figure parmi les amis de « prenom2 ». Si le couple « (prenom2, prenom1) » n'est pas dans cette liste, nous aurons une amitié à sens unique ! Par contre le code doit s'assurer que les deux entrées sont créées dans le dictionnaire (même si l'ensemble des amis de 'prenom1' est vide).

Le dictionnaire retourné doit trié par ordre croissant selon les clés.

Exemple

L'appel suivant de la fonction :

const_dict_amis([("Quidam", "Pierre"), ("Thierry", "Michelle"), ("Thierry", "Pierre")])

Doit retourner :

{"Michelle" : [], "Pierre" : [], "Quidam" : ["Pierre"], "Thierry" : ["Michelle", "Pierre"]}

2) Écrire une fonction « symetrise_amis(d, englobe) » qui reçoit un dictionnaire « d » d'amis où les clés sont des prénoms, et les valeurs sont des ensembles de prénoms représentant les amis de chacun.

Cette fonction modifie le dictionnaire « d » de sorte que si une clé « prenom1 » contient « prenom2 » dans l'ensemble de ses amis, l'inverse soit vrai aussi.

La fonction accepte un second paramètre « englobe ». Si englobe est vrai, la fonction ajoutera les éléments nécessaires pour symétriser le dictionnaire « d ». Sinon, la fonction enlèvera les éléments nécessaires pour symétriser « d ».

Le dictionnaire retourné doit trié par ordre croissant selon les clés ainsi que les listes des amis.

Exemple 1

L'appel suivant de la fonction :

d = {"Bernadette": set(), "Michelle": {"Thierry"}, "Thierry": {"Bernadette", "Michelle"}}

symetrise_amis(d, True) ;

Doit retourner :

{"Bernadette": ["Thierry"], "Michelle": ["Thierry"], "Thierry": ["Bernadette", "Michelle"]}

Exemple 2

L'appel suivant de la fonction :

d = {"Bernadette": set(), "Michelle": {"Thierry"}, "Thierry": {"Bernadette", "Michelle"}}

symetrise_amis(d, False)

Doit retourner :

{"Bernadette": [], "Michelle": ["Thierry"], "Thierry": ["Michelle"]}