TP d'informatique n°4 : dictionnaires « for »

PCSI 2023 – 2024

♡ Les exercices indiqués par un cœur dans la marge sont ceux qu'il faut absolument savoir faire très rapidement et adapter en cas de besoin.

I Avant le tp

Lire l'énoncé des deux exercices ci-dessous (il n'est pas demandé de les faire, mais de lire l'énoncé pour gagner du temps).

II Introduction

Créer dans le corps de votre programme au début une liste
$Lfruits = [$"$orange$","$pomme$","$orange$","$poire$","$orange$","$orange$"$]$ qui nous servira pour les tests.

Exercice n°1 Écrire une fonction `cherche_ind(x,L:list)->int` prenant en argument un objet x et une liste L et qui renvoie un entier :
- qui est l'indice de x dans L si x appartient à L ;
- -1 par convention si x n'appartient pas à L.

Par exemples :
♡
- `cherche_ind("orange",Lfruits)` doit renvoyer 0 ;
- `cherche_ind("poire",Lfruits)` doit renvoyer 3 ;
- `cherche_ind("tomate",Lfruits)` doit renvoyer -1.

Exercice n°2 Dans cet exercices, qui est la suite du précédent, on doit créer à partir d'une liste, un « inventaire », c'est-à-dire compter le nombre d'occurence de chaque élément de la liste de départ. On va donc écrire une fonction `inventaire(L:list)->(list,list)` qui prend en argument une liste L, éventuellement avec doublon, et qui renverra une liste $objets$ qui contiendra les mêmes éléments que L mais sans doublons, et une liste $nombres$ qui contiendra des entiers non nuls tels que $nombres[i]$ est le nombre de fois où $objets[i]$ apparait dans la liste L.
Par exemple, si $L = ['a','b','c','b']$ alors notre fonction inventaire doit calculer et renvoyer les deux listes suivantes : $objets = ['a','b','c']$ et $nombres = [1,2,1]$.
Le résultat signifie que $'a'$ est présent une fois dans la liste L, $'b'$ deux fois etc...
Aide si besoin en note de bas de page [1].

III Notion de dictionnaire

1. Introduction

Un dictionnaire est une collection non ordonnée d'objets. À chaque « clé », on associe une « valeur ». On peut dans un tout premier temps voir cela comme une "liste" où les "indices" peuvent être des chaines

1. supposons que l'exemple ci-dessus représente un état intermédiaire de l'algorithme, c'est-à-dire qu'en fait L contient d'autres termes, $L = ['a','b','c','b',...]$ et qu'avec une boucle for on construit les listes objets et nombres au fur et à mesure. Si l'élément suivant de L est $'a'$, comment faut-il modifier les listes $objets$ et $nombres$? et si l'élément suivant de L est $'b'$? comment savoir où modifier $nombres$? et si l'élément suivant est $'d'$? Avant d'avoir regarder le moindre élément de L comment initialiser les deux listes ? Une fois toutes ces questions maitrisées, il suffit de faire une boucle for (et peut être un test ? !). Ce type de raisonnement où l'on construit une solution de proche en proche est très fréquent en informatique.

de charactères, des entiers, des flottants[2] ou des tuples[3]. Par contre, la manière dont un dictionnaire est implémentée est très différente d'un simple tableau, cela a des conséquences importante, notamment sur le fait que la recherche de la présence ou non d'une clé dans un dictionnaire est une opération très rapide. Vous verrez cela l'an prochain en détail, pour cette année, nous nous contentons d'un point de vue utilisateur.

2. Opérations élémentaires sur un dictionnaire

Vous devez maitriser les syntaxes suivantes :

- création d'un dictionnaire vide : $nom\_de\_variable = \{\}$;
- création d'un dictionnaire non vide : $nom\_de\_variable = \{cle1 : valeur1, cle2 : valeur2\}$;
- ajout d'une paire $clé : valeur$ dans un dictionnaire nommé d : $d[clé] = valeur$;
- modification de la valeur associée à une clé : $d[clé] = valeur2$
- accès à la valeur associée à une clé : $d[clé]$ (attention, renvoie une erreur si la clé n'est pas présente dans le dictionnaire) ;
- nombre d'association clé :valeur : $len(d)$;
- copie d'un dictionaire : $d2 = d.copy()$ (c'est une opération couteuse)
- recherche de la présence d'une clé dans un dictionnaire (renvoie un booléen, $True$ ou $False$) : $cle\ in\ d$. On peut donc utiliser cela dans un if, un $while$ ...
  Remarque important : cette recherche de la présence d'une clé est une opération rapide dans un dictionnaire contrairement à la recherche d'un élément dans une liste. Plus précisément, c'est une opération dont le temps d'éxécution ne dépend presque pas de la taille du dictionnaire alors que dans le cas d'une liste, le temps d'exécution de "in" est proportionnel à la taille de la liste.

3. Mise en pratique

Exercice n°3 créer un dictionnaire edt ayant pour clé les matières : maths, phys, chimie, si et ayant pour valeurs correspondantes le nombre d'heure dans l'Emploi Du Temps.

Exercice n°4 modifier le dictionnaire pour ajouter les matières LV1, français et info.

Exercice n°5 comment un utilisateur peut il interroger/utiliser le dictionnaire pour savoir le nombre d'heure d'info ? comment un utilisateur peut il savoir si vous étudiez ou non la matière "hist-geo" ?

Exercice n°6 comment modifier (dans le dictionnaire) le nombre d'heure d'info ?

4. Parcours d'un dictionnaire

Si on doit parcourir toutes les paires clé :valeur d'un dictionnaire on peut utiliser les syntaxes suivantes :

- `for k in d:` ; à chaque passage dans la boucle, k correspondra à une clé de d, la boucle se termine lorsque l'on a parcouru toutes les clés de d ;
- `for k in d.keys():` ; équivalent au précédent, à éviter.
- `for (k,v) in d.items():` ; à chaque passage dans la boucle, k correspondra à une clé de d et v à la valeur correspondante, c'est-à-dire que $v = d[k]$, la boucle se termine lorsque l'on a parcouru toutes les paires clé :valeur de d.

2. nombres "réels" en gros
3. Plus précisément, le type utilisé pour la clé doit être "hashable", et en particulier non mutable. Le type des valeurs est non restreint, cela peut être des objets simples, des listes, des fonctions etc...

Exercice n°7 Utiliser une des syntaxe pour afficher l'ensemble des matières présentes dans le dictionnaire edt.

Exercice n°8 Refaire l'exercice "inventaire" du début (Exercice n°2) en utilisant un dictionnaire plutôt que deux listes. Les clés seront les objets et les valeurs seront le nombre d'occurence correspondant.

Exercice n°9 Écrire une fonction `inverse` qui prend en argument un dictionnaire et qui inverse les clés et les valeurs. Ainsi si $d = \{1 : 2, 3 : 4\}$ alors `inverse(d)` doit renvoyer $\{2 : 1, 4 : 3\}$. Cela pourrait servir pour fabriquer un "dictionnaire" français-anglais à partir d'un dictionnaire anglais-français (on fera l'hypothèse que les valeurs ne présentent pas de doublons).

Exercice n°10 On considère un déplacement sur une grille, chaque case est repéré par un tuple (ligne, colonne). On suppose que chaque case ne permet d'aller qu'à une seule autre case au plus. On enregistre cela dans un dictionnaire "parent" pour lequel les clés sont des tuples représentant une case et les valeurs sont des tuples qui représentent la case permettant d'aller à la clé. Par exemple, si $d[(2,3)]$ vaut $(1,3)$, cela signifie qu'à partir de la case (1,3), on peut aller sur la case (2,3).

Écrire une fonction $chemin(dico,dep,arr)$ qui prend en argument un tel dictionnaire, une position de départ, une position d'arrivée et qui renvoie le chemin (s'il existe) permettant d'aller de dep à arr

IV À faire pour la prochaine séance

Exercice n°11 Rappeler sans regarder les pages précédentes quelles sont les opérations élémentaires sur un dictionnaire et la syntaxe correspondante en python. Vérifier ensuite que vous n'avez rien oublié et que la syntaxe est correcte.

Exercice n°12 Créer un dictionnaire fr_en qui contient en clé les mots français $physique$, je, $aime$ et en valeur les mots anglais $physic$, I, $love$

Exercice n°13 Écrire une fonction[4] `traduction(phrase:list, dico:dict)->list` qui prend en argument une "phrase" sous forme d'une liste de mot et un dictionnaire dico qui contient les traduction mot à mot ; votre fonction renverra la traduction mot à mot de la phrase (sous forme d'une liste de mot). Par exemple `traduction(["je","aime",'physique'], fr_en)` doit renvoyer ["I",'love','physic'].

Exercice n°14 Écrire une fonction `inventaire(L:list)->dict` qui prend en argument une liste et qui renvoie un dictionnaire ou les clés correspondent aux éléments de L et les valeurs au nombre d'occurence dans L.

4. à ne pas montrer à votre prof d'anglais ...