4.1 Rappels sur les dictionnaires

4.1.1 Définition

Définition

Un dictionnaire python est une structure de donnée qui permet d'associer une valeur à une clé. Cette clé peut être un entier, un str ou un tuple.
L'ensemble clé-valeur est appelé entrée.

Remarque

- Les dictionnaires ne comportent pas d'ordre. On ne peut donc pas trouver un élément via sa position, mais seulement via sa clé.
- Python possède nativement cette structure de données, ce qui n'est pas vrai pour tous les langages de programmation. Il faudrait alors l'implémenter si on souhaite l'utiliser.

Interface d'un dictionnaire

Voici l'interface d'un dictionnaire :

creer_dico() Permet de créer un dictionnaire vide

Inserer(dico,c,v) Permet d'ajouter l'entrée c-v

supprimer(dico,c Permet de supprimer l'entrée dont la clé est c

lire(dico,c) Permet de retourner la valeur de l'entrée dont la clé est c.

4.1.2 Utilisation des dictionnaires Python

L'implémentation des dictionnaires Python a été vu en première. Se référer au cours de l'an passé :-)

Compléxité de recherche d'un élément dans un dictionnaire

Un dictionnaire n'est pas une séquence ! Il n'y a pas d'ordre, et les éléments ne sont pas repérés par des indices.

Pour un tableau (list python) par exemple, la complexité de recherche d'un élément est linéaire, et dépend de la taille du tableau.

Ce n'est pas du tout le cas d'un dictionnaire : la complexité de recherche d'un élément est constante (ne dépend pas de la taille du dictionnaire ! !). Cela est dû à son implémentation avec des tables de hachage (ce n'est pas au programme de terminale, mais c'est très intéressant..Ne pas hésiter à faire des recherches sur le Web !)

4.1.3 Exercices

Exercice 4.1
Ecrire le script Python qui permet d'obtenir une structure de données de type dict qui doit être :

```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8, 'J': 9, 'K': 10,
'L': 11, 'M': 12, 'N': 13, 'O': 14, 'P': 15, 'Q': 16, 'R': 17, 'S': 18, 'T': 19, 'U': 20,
'V': 21, 'W': 22, 'X': 23, 'Y': 24, 'Z': 25}
```

Exercice 4.2
On considère le dico suivant :

```
dico={'nsi' :18,'maths':17,'svt':14,'français':14,'lv1':8,'physique':12,'HG':11}
```

1. Ecrire cet affichage :

```
En  nsi , Jean a 18 de moyenne
En  maths , Jean a 17 de moyenne
En  svt , Jean a 14 de moyenne
En  français , Jean a 14 de moyenne
En  lv1 , Jean a 8 de moyenne
En  physique , Jean a 12 de moyenne
En  HG , Jean a 11 de moyenne
```

2. Effacer les information relative au Français dans le dictionnaire

Exercice 4.3
Voici une citation célèbre de Gandhi :

La vie est un mystère qu'il faut vivre, et non un problème à résoudre.

Créer un dictionnaire qui associe à chaque lettre (clé) son occurrence (valeur). Par exemple la lettre 'a' apparait deux fois.

```
{'L': 1, 'a': 2, 'v': 3, 'i': 3, 'e': 7, 's': 3, 't': 4, 'u': 5, 'n': 4, 'm': 2, 'y': 1, '
è': 2, 'r': 5, 'q': 1, 'l': 2, 'f': 1, 'o': 3, 'p': 1, 'b': 1, 'à': 1, 'é': 1, 'd': 1}
```

Exercice 4.4
Le Titanic appareille de Southampton (Angleterre) le mercredi 10 avril à 12 h 15 .

Six heures plus tard, à 18 h 15, il fait escale dans la rade de Cherbourg. Il y débarque 24 passagers et en embarque 274, amenés par les transbordeurs Nomadic et Traffic. Il appareille à 20 h 10.

Le Titanic fait route vers l'Irlande. Il arrive à Queenstown (aujourdhui Cobh) le 11 avril à 11 h 30. Il débarque 7 passagers et en embarque 120. À 13 h 30, le paquebot appareille et entame sa traversée de l'Atlantique vers New York .

Le 14 avril, à 23 h 40 (heure locale, GMT-3), il percute un iceberg au large de Terre-Neuve. Il sombre le 15 avril à 2 h 20, causant la mort de 1 524 personnes.

Le fichier titanic.csv se trouve ici .

1. Importer le contenu du fichier dans une liste nommée titanic (une liste de dictionnaire) :

```
>>> titanic
[{'PassengerId': '1', 'Survived': '0', 'Pclass': '3', 'Name': 'Braund, Mr. Owen Harri
s', 'Sex': 'male', 'Age': '22', 'SibSp': '1', 'Parch': '0', 'Ticket': 'A/5 21171', 'F
are': '7.25', 'Cabin': '', 'Embarked': 'S'}, {'PassengerId': '2', 'Survived': '1', 'P
class': '1', 'Name': 'Cumings, Mrs. John Bradley (Florence Briggs Thayer)', 'Sex': 'f
```

2. Écrire un script qui détermine le nombre de survivants
3. Écrire un programme qui affiche le pourcentage de survivants par classe
4. Ecrire un programme qui donne le nombre de survivants embarqués à Cherbourg

Exercice 4.5

Lorsqu'on cherche un mot dans un dictionnaire (un vrai), on tourne peu de pages pour le trouver puisqu'ils sont triés par ordre alphabétique.

Maintenant, si on souhaite trouver tous les mots dans la seconde lettre est e, c'est impossible à moins de trier les mots par leur seconde lettre : il faudrait un dictionnaire différent pour chaque position de lettre.

L'objectif de cet exercice est de concevoir une sorte de dictionnaire qui permette de retrouver tous les mots ayant telle lettre à telle position.

Voici une liste de mots :

```
mots = ['eddard', 'catelyn', 'robb', 'sansa', 'arya', 'brandon',
       'rickon', 'theon', 'rorbert', 'cersei', 'tywin', 'jaime',
        'tyrion', 'shae', 'bronn', 'lancel', 'joffrey', 'sandor',
        'varys', 'renly', 'a' ]
```

1. construire un dictionnaire dico_bien_choisi avec :
   - La clé sous la forme d'un tupple (i,l), où i est la position de la lettre l(on commence à 0) et l est une lettre
   - La valeur associée à chaque clé est une liste contenant tous les mots contenant la lettre l à la i-ème position.
2. Créer une fonction liste_lettre_position de paramètres :
   - d : le dico bien choisi (dict)
   - l : la lettre choisi (str)
   - p : la position de la lettre (int)

```
>>> liste_lettre_position(d,'r', 2)
['rorbert', 'cersei', 'tyrion', 'varys']
```

Exercice 4.6

Voici un message codé avec un chiffrement de césar :

message =
"CFIJHLFETYVITYVLEDFKUREJLEUZTKZFEERZIVLEMIRZFEKFL
IEVGVLUVGRXVJGFLICVKIFLMVIGLZJHLZCJJFEKKIZVJGRIFIU
IVRCGYRSVKZHLVDRZEKVEREKJZFEJFLYRZKVKIFLMVIKFLJCVJ
DFKJUFEKCRJVTFEUVCVKKIVVJKVTVJKZDGFJJZSCVRDFZEJUVK
IZVICVJDFKJGRICVLIJVTFEUVCVKKIVZCWRLUIRZKLEUZTKZFE
ERZIVUZWWVIVEKGFLITYRHLVGFJZKZFEUVCVKKIV"

Décoder ce message en utilisant une approche fréquentielle : créer un dictionnaire avec comme clé les lettres de A à Z et comme valeur associé le pourcentage de cette lettre dans le message codé.

Ensuite, créer une fonction qui utilise ce dictionnaire afin de décoder le mot

☞ Le 'E' est la lettre la plus fréquente !

Exercice 4.7

Les réponses correctes d'un QCM de NSI sont stockées dans un dictionnaire nommé reponses_valides.
Les clés sont des chaînes de caractères de la forme "Q1". Les valeurs possibles sont des chaînes de caractères correspondant aux quatre réponses "a","b","c","d".
Les réponses données par Alice sont stockées dans le dictionnaire reponses_Alice.
Lorsqu'Alice n'a pas répondu à une question, la clé relatif à cette question n'est pas présente dans la dictionnaire de ses réponses.
La notation d'un QCM de NSI est la suivante : 3 points par réponse correcte, -1 point par réponse incorrecte et 0 si l'on n'a pas répondu.

```
reponses_valides = {"Q1":"c","Q2":"a","Q3":"d","Q4":"c","Q5":"b"}
reponses_Alice = {"Q1":"b","Q2":"a","Q3":"d","Q5":"a"}
```

Compléter la fonction **correction_qcm(reponses_Alice,reponses_valides)** qui, à partir des dictionnaires reponses_Alice et reponses_valides passées en argument renvoie le nombre de points obtenus au QCM par Alice.

Exercice 4.8

On pourra utiliser dans cet exercice la fonction 'len' qui renvoie la longueur d'une liste, et 'in' pour vérifier l'appartenance d'un élément dans une liste.
Dans cet exercice, on s'intéresse à la définition de fonctions permettant de manipuler un livre de recettes de cuisine. Comme tout bon livre de recettes qui se respecte, chaque recette décrit notamment l'ensemble des ingrédients qui la composent. À titre d'exemple, un livre de recettes de desserts pourrait contenir les informations suivantes :

Recette	Ingrédients
Gâteau au chocolat	chocolat, oeuf, farine, sucre, beurre
Gâteau au yaourt	yaourt, oeuf, farine, sucre
Crêpes	oeuf, farine, lait
Quatre-quarts	oeuf, farine, beurre, sucre
Kouign amann	farine, beurre, sucre

Un livre de recettes est donc représenté en Python par un dictionnaire de typedict[str :set[str]]

- les noms des recettes, de typestr, comme clés
- l'ensemble des ingrédients, de typeset[str], comme valeurs associées.

Ainsi, l'exemple précédent donnerait le livre de recettes suivant :

```
Livre_recettes={'gateau chocolat': ['chocolat','oeuf','farine','sucre','beurre'],
'gateau yaourt': ['yaourt','oeuf','farine','sucre'],
'crepes': ['oeuf','farine','lait'],
'quatre-quarts': ['oeuf','farine','beurre','sucre'],
'kouign amann': ['farine','beurre','sucre']}
```

1. Donner une définition de la fonction **nb_ingredients(D,r)** qui, étant donnés un livre de recettes D et le nom d'une recette r contenue dans D, renvoie le nombre d'ingrédients nécessaires à la recette r.

```
>>> nb_ingredients(Livre_recettes,'crepes')
3
>>> nb_ingredients(Livre_recettes,'gateau chocolat')
5
```

2. Donner une définition de la fonction **recette_avec(D,i)** qui, étant donnés un livre de recettes D et le nom d'un ingrédient i, renvoie la liste sous forme de tableau python des recettes qui utilisent cet ingrédient.

```
>>> recette_avec(Livre_recettes,'beurre')
['gateau chocolat', 'quatre-quarts', 'kouign amann']
>>> recette_avec(Livre_recettes,'lait')
['crepes']
```

3. Certaines personnes sont allergiques à certains ingrédients. On aimerait donc pouvoir ne conserver d'un livre de recettes que celles qui n'utilisent pas un ingrédient donné.Donner une définition de la fonction **recettes_sans(D,i)** qui, étant donnés un livre de recettes D et un ingrédient i, renvoie la liste (sous forme de tableau Python) des ingredients n'utilisant pas l'ingrédient i

```
>>> recette_sans(Livre_recettes,'farine')
[]
>>> recette_sans(Livre_recettes,'oeuf')
['kouign amann']
>>> recette_sans(Livre_recettes,'beurre')
['gateau yaourt', 'crepes']
```

4. Donner une définition de la fonction **tous_ingredients(D)** qui, étant donné un livre de recettes D,renvoie la liste sous forme de tableau python de tous les ingrédients apparaissant au moins une fois dans une recette de D.

```
>>> tous_ingredients(Livre_recettes)
['chocolat', 'oeuf', 'farine', 'sucre', 'beurre', 'yaourt',
'lait']
```

Exercice 4.9

Cet exercice est la suite de l'exercice précédent.

1. Tout livre de recettes contient une table des ingrédients permettant d'associer à chaque ingrédient l'ensemble des recettes qui l'utilisent.
   Une telle table est représentée en Python par le type dict[str : set[str]] dans lequel une clé est un ingrédient dont la valeur associée est l'ensemble des recettes qui l'utilisent.
   Donner une définition de la fonction **table_ingredients(D)** qui, étant donné un livre de recettes D, renvoie la table des ingrédients associée, comme dans l'exemple.

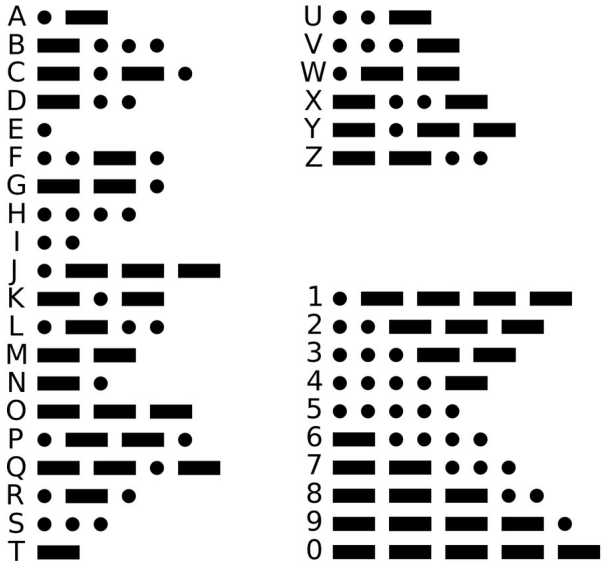
```
>>> table_ingredients(Livre_recettes)
{'chocolat': ['gateau chocolat'], 'oeuf': ['gateau chocolat', 'gateau yaourt', '
crepes', 'quatre-quarts'], 'farine': ['gateau chocolat', 'gateau yaourt', 'crepe
s', 'quatre-quarts', 'kouign amann'], 'sucre': ['gateau chocolat', 'gateau yaour
t', 'quatre-quarts', 'kouign amann'], 'beurre': ['gateau chocolat', 'quatre-quar
ts', 'kouign amann'], 'yaourt': ['gateau yaourt'], 'lait': ['crepes']}
```

2. Donner une définition de la fonction **ingredient_principal(D)** qui, étant donné un livre de recettes D, renvoie le nom de l'ingrédient utilisé par le plus grand nombre de recettes. On supposera ici que D contient au moins une recette et qu'il existe un et un seul ingrédient principal.

```
>>> ingredient_principal(Livre_recettes)
'farine'
```

Exercice 4.10

On considère l'alphabet braille :



Créer une fonction qui permet de décoder la citation suivante :

’— . ¯ . -. .- .-.. .-.. .- .. ... -.. .- -. ... .-.. . ... -... — .. ... .¯. .- .-. -.-. . ¯.- ..- . .— . ...- — ..- .-.. .- .. ..- .. ...- .-. . ... .- -. ... .- - . .-.-.- ...- .. ...- .-. . ¯..¯ .. -. - . -. ... . ¯ . -. - ¯..¯ . - ... ..- -.-. . .-. - — ..- - . .-.. .- ¯ — . .-.. .-.. . -.. . .-.. .- ...- .. . .-.-.- ¯ . - - .-. . . -. -.. . .-. — ..- - . - — ..- - -.-. . ¯.- ..- .. -. . - .- .. - .¯. .- ... .-.. .- ...- .. . .¯. — ..- .-. -. . .¯. .- ... -.. . -.-. — ..- ...- .-. .. .-. ¯..¯ .- .-.. ....- .-. . -.. . ¯ .- ¯ — .-. - ¯.- ..- . .— . -. .- ...- .- .. ... .¯. .- ... ...- . -.-. ..- .-.-.- ’

☛ Chaque caractère (lettre, ponctuation..) est séparée par un espace.

☛ Vous trouverez cette citation et le dictionnaire braille python dans le fichier annexe .txt