Les dictionnaires Python

1 Définition

1.1 Définition

Comme avec les tableaux dynamiques (type list en Python), les dictionnaires sont une structure linéaire qui permet de "stocker" des données.
Chaque élément d'un dictionnaire est composé de deux parties :

- La clé
- La valeur

⚠ A l'inverse des tableaux, les dictionnaires ne sont pas indexés, c'est à dire que les valeurs ne sont pas repérées par un indice.

Définition 3.1

Un dictionnaire est une structure de données où les différentes valeurs sont associées à des clés.
Le couple (clé-valeur) est appelé une entrée.
Les différentes valeurs, à la différence d'un tableau, ne sont pas indexés, mais associées à des clefs.

☞ L'accès aux valeurs se fait en utilisant des fonctions de hachage, très utilisé en informatique, mais hors programme dans le cadre de la NSI.

1.2 Mon premier dictionnaire

La structure des dictionnaires est implémenté nativement dans python. C'est un objet qui possède des méthodes.

```
mon_dico_vide = {} # ou bien mon_dico_vide= dict()
mon_dico = {"Pierre": "0689787475", "Romane": "0656585978"}
```

Remarque

On utilise des accolades pour définir un dictionnaire

Dans l'exemple précédent, "Pierre" et "Romane" sont des clés.
Tandis que "0689787475" et "0656585978" sont des valeurs.

Les clés peuvent être des nombres, des chaînes de caractères, des tuples.
Les valeurs peuvent être des nombres, des chaînes de caractères, des tuples, des listes..etc..

1.3 Ajout d'un élément

```
mon_dico["Aline"] = "0789868583"
```

1.4 Accès à la valeur associée à une clé

```
mon_dico["Pierre"]
```

Remarque

Il est aussi possible d'accéder à une valeur du dictionnaire en utilisant la méthode get(), mais nous étudierons cela davantage en terminale.

1.5 Effacer une entrée

```
del mon_dico["Pierre"]
```

⚠ Cela efface la clé et la valeur associée.

Exercice 3.1
Soit le dictionnaire :

```
>>> d = {'nom': 'Dupuis', 'prenom': 'Jacque', 'age': 30}
```

1. (a) Afficher le nom 'Dupuis'
   (b) puis le prénom 'Jacque',
   (c) et enfin l'âge '30'
2. Corriger l'erreur dans le prénom, la bonne valeur est 'Jacques'.
3. Ecrire la phrase "Jacques Dupuis a 30 ans".

2 Les méthodes associées au dictionnaires

Comme tout objet, les dictionnaires possèdent différentes méthodes. En première, nous étudierons uniquement 3 méthodes, qui permettent de parcourir un dictionnaires (avec un for par exemple).

2.1 La méthode dico.keys()

```
>>> ani2 = {'nom':'singe', 'poids':70, 'taille':1.75}
>>> for key in ani2.keys():
        print(key)
```

2.2 La méthode dico.values()

```
>>> ani2 = {'nom':'singe', 'poids':70, 'taille':1.75}
>>> for val in ani2.values():
        print(val)
```

2.3 La méthode dico.items()

```
>>> ani2 = {'nom':'singe', 'poids':70, 'taille':1.75}
>>> for key,val in ani2.items():
        print (key,val)
```

Exercice 3.2

Soit le dictionnaire :

```
>>> d = {'nom': 'Dupuis', 'prenom': 'Jacques', 'age': 30}
```

1. Afficher la liste des clés du dictionnaire.
2. Afficher la liste des valeurs du dictionnaire.
3. Afficher la liste des entrées clé/valeur du dictionnaire.

Exercice 3.3

Dans cet exercice, nous nous familiarisons avec les manipulations de dictionnaires sur une thématique de magasin en ligne.
«Chez Geek and sons, tout ce qui est inutile peut s'acheter, et tout ce qui peuts'acheter est un peu trop cher.»
La base de prix des produits de Geek and sons est représentée en Python par un dictionnaire de type dict avec :— les noms de produits, de type str, comme clés— les prix des produits, de type float, comme valeurs associées.

1. Donner une expression Python pour construire la base des prix des produits correspondant à la table suivante :

Nom du produit	Prix TTC
Sabre laser	229
Mitendo DX127	30
Coussin Linux	74.5
Slip Goldorak	30
Station Nextpresso1	84.60

2. Créer une fonction **prix_moyen(dico)** qui prend en argument un dictionnaire (où les clefs sont des str, et les valeurs des float) et qui retourne le prix moyen des produits disponibles.
3. Créer une fonction **fourchette_prix(dic,mini,max)** qui, étant donné un prix minimum mini, un prix maximum maxi et une base de Prix, retourne l'ensemble des noms de produits disponibles dans cette fourchette de prix.
4. La notion de panier est un concept omniprésent dans les sites marchands, Geeks and sons n'échappe pas à la règle. En Python, le panier du client sera représenté par un dictionnaire de type dict avec :
   - les noms de produits comme clés
   - une quantité d'achat comme valeurs associées

   Donner une expression Python correspondant à l'achat de 3 sabres lasers, de 2 coussins Linuxet de 1 slip Goldorak.

5. Créer la fonction **prix_achats(panier,base_prix)** qui, étant donné un panier d'achat Panier et une base de Prix, retourne le prix total correspondant.

Exercice 3.4

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 valeur de la clef correspondant au nom de l'exercice est None.
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","Q4":None,"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 3.5

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 3.6

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'
```