Thème : Exercices simples sur les ensembles et dictionnaires

Exercice 1 : Magasin en ligne

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 peut s'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[str:float]` avec :

— les noms de produits, de type `str`, comme clés
— les prix des produits, de type `float`, comme valeurs associées.

Question 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	229.0
Mitendo DX	127.30
Coussin Linux	74.50
Slip Goldorak	29.90
Station Nextpresso	184.60

Question 2

Donner une définition de la fonction `disponibilite` qui étant donné un nom de produit `prod` et une base de prix `Prix`, retourne `True` si le produit est présent dans la base, ou `False` sinon.

Question 3

Donner une définition de la fonction `prix_moyen` qui, étant donné une base de prix (contenant au moins un produit), retourne le prix moyen des produits disponibles.

Par exemple :

```
>>> prix_moyen({'Sabre Laser': 229.0,
                'Mitendo DX': 127.30,
                'Coussin Linux': 74.50,
```

```
                'Slip Goldorak': 29.90,
                'Station Nextpresso': 184.60})
129.06
```

Question 4

Donner une définition de la fonction `fourchette_prix` 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.

Par exemple :

```
>>> fourchette_prix(50.0, 200.0, {'Sabre Laser': 229.0,
                                  'Mitendo DX': 127.30,
                                  'Coussin Linux': 74.50,
                                  'Slip Goldorak': 29.90,
                                  'Station Nextpresso': 184.60})
{'Coussin Linux', 'Mitendo DX', 'Station Nextpresso'}
```

Question 5

Le *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[str:int]` 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 *Linux* et de 1 slip *Goldorak*.

Question 6

Donner une définition de la fonction `tous_disponibles` qui, étant donné un panier d'achat `Panier` et une base de `Prix`, retourne `True` si tous les produits demandés sont disponibles, ou `False` sinon.

Question 7

Donner une définition de la fonction `prix_achats` qui, étant donné un panier d'achat `Panier` et une base de `Prix`, retourne le prix total correspondant.

Par exemple :

```
>>> prix_achats({'Sabre Laser': 3, 'Coussin Linux': 2, 'Slip Goldorak': 1},
                {'Sabre Laser': 229.0,
                 'Mitendo DX': 127.30,
                 'Coussin Linux': 74.50,
```

```
                  'Slip Goldorak': 29.90,
                  'Station Nextpresso': 184.60})
865.9
```

Remarque : on supposera que tous les articles du paniers sont disponibles dans la base de produits.

Exercice 2 : Recettes de cuisine

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 type `dict[str : set[str]]`

- les noms des recettes, de type `str`, comme clés
- l'ensemble des ingrédients, de type `set[str]`, comme valeurs associées.

Ainsi, l'exemple précédent donnerait le dictionnaire `Dessert` suivant :

```
Dessert = {
       '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'}
       }
```

Dans la suite de cet exercice, on pourra utiliser l'alias de type `Recette` pour `dict[str : set[str]]`

Question 1

Donner une définition de la fonction `nb_ingredients` 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`.

Par exemple:

```
>>> nb_ingredients(Dessert, 'crepes')
3
```

```
>>> nb_ingredients(Dessert, 'gateau chocolat')
5
```

Question 2

Donner une définition de la fonction `recette_avec` qui, étant donnés un livre de recettes `D` et le nom d'un ingrédient `i`, renvoie l'ensemble des recettes qui utilisent cet ingrédient.

Par exemple :

```
>>> recette_avec(Dessert, 'beurre')
{'gateau chocolat', 'kouign amann', 'quatre-quarts'}
```

```
>>> recette_avec(Dessert, 'lait')
{'crepes'}
```

```
>>> recette_avec(Dessert, 'fraise')
set()
```

Question 3

Donner une définition de la fonction `tous_ingredients` qui, étant donné un livre de recettes `D`, renvoie l'ensemble de tous les ingrédients apparaissant au moins une fois dans une recette de `D`.

Par exemple :

```
>>> tous_ingredients(Dessert)
{'beurre', 'chocolat', 'farine', 'lait', 'oeuf', 'sucre', 'yaourt'}
```

Question 4

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` qui, étant donné un livre de recettes `D`, renvoie la table des ingrédients associée.

Par exemple :

```
>>> table_ingredients(Dessert)
{'chocolat': {'gateau chocolat'},
 'oeuf': {'crepes', 'gateau chocolat', 'gateau yaourt', 'quatre-quarts'},
 'lait': {'crepes'},
 'yaourt': {'gateau yaourt'},
 'farine': {'crepes',
```

```
  'gateau chocolat',
  'gateau yaourt',
  'kouign amann',
  'quatre-quarts'},
 'beurre': {'gateau chocolat', 'kouign amann', 'quatre-quarts'},
 'sucre': {'gateau chocolat',
  'gateau yaourt',
  'kouign amann',
  'quatre-quarts'}}
```

Question 5

Donner une définition de la fonction `ingredient_principal` 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.

Par exemple :

```
>>> ingredient_principal(Dessert)
'farine'
```

Question 6

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` qui, étant donnés un livre de recettes `D` et un ingrédient `i`, renvoie un nouveau livre de recettes ne contenant que des recettes de `D` n'utilisant pas l'ingrédient `i`.

Par exemple :

```
>>> recettes_sans(Dessert, 'farine')
{}
```

```
>>> recettes_sans(Dessert, 'oeuf')
{'kouign amann': {'beurre', 'farine', 'sucre'}}
```

```
>>> recettes_sans(Dessert, 'beurre')
{'gateau yaourt': {'farine', 'oeuf', 'sucre', 'yaourt'},
 'crepes': {'farine', 'lait', 'oeuf'}}
```

Exercice 3 : Statistiques sur les lettres

Dans cet exercice, on effectue quelques calculs statistiques sur les fréquences de lettres dans des textes (chaînes de caractères).

Les fréquences (ou nombre d'occurrences) des lettres sont représentées sous la forme d'un dictionnaire de type `dict[str:int]` avec :

— des lettres (caractères) comme clés
— des entiers naturels (fréquence du caractère) pour les valeurs associées

Pour séparer les lettres de la langue française des autres caractères possibles dans les chaînes, on utilise la fonction suivante :

```
def est_lettre(c):
    """ str -> bool
    Hypothèse : len(c) == 1    (caractère)
    Retourne True si le caractère c est une lettre, ou False sinon."""

    return ((c >= 'a') and (c <= 'z')) \
        or ((c >= 'A') and (c <= 'Z')) \
        or (c in {'é', 'è', 'à', 'ù', 'œ'})
```

Question 1

Définir la fonction `frequences_lettres` qui étant donnée un chaîne de caractère `s` retourne les fréquences des lettres de `s` sous la forme d'un dictionnaire de type `dict[str:int]`.

Par exemple :

```
>>> frequences_lettres('alea jacta est')
{'j': 1, 'e': 2, 't': 2, 'c': 1, 'a': 4, 's': 1, 'l': 1}
```

```
>>> frequences_lettres("l'élève")
{'é': 1, 'e': 1, 'v': 1, 'l': 2, 'è': 1}
```

Question 2

Définir une fonction `lettre_freq_max` qui retourne la lettre de fréquence maximale dans un dictionnaire `Freqs` de fréquences.

Par exemple :

```
>>> lettre_freq_max(frequences_lettres('alea jacta est'))
'a'
```

```
>>> lettre_freq_max(frequences_lettres("l'élève"))
'l'
```

Remarque : s'il y a plusieurs lettres de fréquence maximale, alors on n'en retourne qu'une choisie arbitrairement.

Question 3 (en TME)

Dans cette question, nous aimerions effectuer notre petit test statistique sur un véritable texte.

Pour cela, nous allons tout d'abord définir une fonction `chargement_texte` permettant de lire un fichier texte et de placer le résultat dans une chaîne de caractères.

Remarque : nous n'étudions pas le chargment et la sauvegarde des fichiers dans ce cours, donc on utilisera cette fonction en suivant simplement sa spécification.

```
def chargement_texte(fichier):
    """ str -> str
    Hypothèse : le fichier est présent sur le disque
    Retourne la chaîne de caractères correspondant au contenu
    du fichier."""

    # contenu : str
    contenu = ''  # contenu du fichier

    with open(fichier, 'r') as f:
        contenu = f.read()

    return contenu
```

On récupérera alors un fichier texte (encodage `UTF-8`) de langue française pour en étudier le contenu.

On peut par exemple récupérer un texte intégral via le *Projet Gutemberg*, à l'adresse suivante : http://www.gutenberg.org

Pour le TME, on peut choisir son propre texte mais attention à ce qu'il ne soit pas trop volumineux. Pour les exemples on a choisi *Quatrevingt treize* de Victor Hugo que l'on trouvera dans :

`/Vrac/1I001/quatrevingt-treize.txt`

Donner deux expressions Python permettant de :

1. récupérer le dictionnaire des fréquences des lettres présentes dans votre texte d'exemple.
2. trouver la lettre dont la fréquence est la plus grande.

Question 4

On souhaite maintenant connaître les lettres qui ne dépassent pas une fréquence donnée dans un texte. Donner une définition de la fonction `lettres_freq_inf` qui étant donné un dictionnaire de fréquences `Freqs` et une fréquence `fseuil` retourne l'ensemble des lettres de fréquence inférieure ou égale à `fseuil`.

Par exemple :

```
>>> lettres_freq_inf(frequences_lettres('alea jacta est'), 1)
{'c', 'j', 'l', 's'}
```

```
>>> lettres_freq_inf(frequences_lettres("l'élève"), 2)
{'e', 'l', 'v', 'è', 'é'}
```

Remarque : on fera l'hypothèse que la fréquence de seuil est strictement positive. En effet, nous n'étudions pas l'absence d'un lettre dans le texte.

Question 5 (en TME)

Donner une expression Python permettant d'obtenir l'ensemble des lettres utilisées moins de 100 fois dans votre texte.