Initiation à la programmation – TP13 – 18 novembre

Objectif : Dictionnaires en python

Exercice 1

On travaille sur un dictionnaire qui comporte des clés qui sont des strings représentants des noms d'ingrédients et des valeurs qui sont les prix de ces ingrédients donc des réels.

On aura par exemple le dictionnaire :

```
dicoprix={"jambon":3,"sauce_tomate":1.5, 'poivrons':2, 'oignons':1,"champignons":2,
 "mozarella":1.5, "creme_fraiche":1.5, "chevre":2, "tomates":2, "lardons":2.5,
 'saumon':4, 'merguez':3}
```

1. Écrire la fonction `combien(dico)` qui renvoie le nombre d'ingrédients présents dans le dictionnaire `dico`.
2. Écrire la fonction `prix_moyen(dico)` qui renvoie la moyenne des prix des ingrédients du dictionnaire `dico`.remarque : on pourra utiliser round(x,2) pour arrondir avec 2 chiffres après la virgule.
3. Écrire la fonction `moinscher(dico)` qui renvoie le nom de l'ingrédient le moins cher du dictionnaire `dico`. Si plusieurs ingrédients ont ce même prix "moins cher", on ne donnera qu'un seul d'entre eux.
4. Écrire la fonction `dollars(dico)` qui modifie `dico` pour que les prix soient en dollars. On prendra comme taux de conversion 1 euro correspond à 1.12 dollars.

```
>>> combien(dicoprix)
12
>>> prixmoyen(dicoprix)
2.17
>>> moinscher(dicoprix)
oignons
>>> dollars(dicoprix)
>>>dicoprix
{"jambon":3.36,"sauce_tomate":1.68, 'poivrons':2.24, 'oignons':1.12,"champignons":2.24,
 "mozarella":1.68, "creme_fraiche":1.68, "chevre":2.24, "tomates":2.24, "lardons":2.80,
 'saumon':4.48, 'merguez':3.36}
```

5. Écrire la fonction `prixpizza(l,dicoprix )` qui étant donnés une liste d'ingrédients l qui correspond aux ingrédients d'une pizza et un dictionnaire de prix renvoie le prix de la pizza correspondante : pour cela on fera le total des prix des ingrédients et on le multipliera par 1.5 le total obtenu pour tenir compte de la pâte, du four etc.... On suppose que tous les ingrédients des listes sont présents dans le dictionnaire dicoprix (sans tests).

```
>>> prixPizza(['jambon', 'mozarella', 'sauce_tomate', 'champignons'],dicoprix)
12.0
```

6. Le patron gère plusieurs succursales et il se rend compte qu'il n'y a pas le même dicoprix dans toutes. Écrire la fonction `gestion(dicoprix1,dicoprix2 )` qui étant donnés deux dictionnaires va fabriquer un nouveau dictionnaire de prix : dans le nouveau dictionnaire il doit y avoir tous les ingrédients présents dans l'un des deux dictionnaires et quand un ingrédient est dans les deux dictionnaires, on doit garder le prix le moins élevé des deux.

```
>>> dic1={"jambon":3,"sauce_tomate":1.6, 'oignons':1,"champignons":2,
 "mozarella":1.5, "creme_fraiche":1.5, "chevre":2, "tomates":2, "lardons":2.5,
 'saumon':4, 'merguez':3}
>>> dic2={ "poivrons":2, "jambon":3.1,"sauce_tomate":1.5, 'poivrons':2, 'oignons':1,
"roquefort":2, "mozarella":1.5, "chorizo":2.6, 'saumon_fume':3.5, 'merguez':3}
>>> gestion(dic1,dic2)
{"jambon":3,"sauce_tomate":1.5, 'oignons':1,"champignons":2, "mozarella":1.5,
"creme_fraiche":1.5, "chevre":2, "tomates":2, "lardons":2.5, 'saumon':4, 'merguez':3,
"poivrons":2, "roquefort":2,"chorizo":2.6, 'saumon_fume':3.5}
```

7. On considère un deuxième dictionnaire : les clés sont des strings qui sont des noms de pizzas et les valeurs associées sont des listes d'ingrédients . On aura comme exemple :

```
monDicoPizzas={"reine":["jambon", "mozarella", "sauce_tomate", "champignons"],
"vesuvio":['merguez','jambon', 'mozarella', 'poivrons', 'oignons' ],
"cabri":["chevre", "lardons", "creme_fraiche", "mozarella"],
"napoli":["jambon", "tomates", "mozarella", "sauce_tomate",
"champignons", "poivrons", "oignons"], "neptune":['saumon', 'creme_fraiche','champignons']}
```

Écrire la fonction `possible(dicopizzas, val, dicoprix)` qui renvoie tous les noms des pizzas qu'on peut acheter si on a la somme val.

```
>>> possible(mespizzas, 12, dicoprix)
['napoli', 'reine', 'vesuvio']
```

8. Ecrire la fonction `inter(l1,l2)` qui teste si les listes l1 et l2 ont au moins un élément commun.
   En déduire la fonction `sansAllergie(monDicoPizzas, listeing)` qui renvoie les pizzas qui ne contiennent aucun élément de listeing dans leur composition.
9. Finalement on décide d'avoir une fonction qui peut fabriquer un troisième dictionnaire. Écrire la fonction `fusion(dicoprix, dicopizzas)` qui renvoie le dictionnaire où chaque pizza est associée à son prix et ses ingrédients (la valeur associée à un nom de pizza sera un couple avec la liste des ingrédients et le prix).

```
>>> fusion(monDicoPizzas,dicoprix)
{'reine': (['jambon', 'mozarella', 'sauce_tomate', 'champignons'], 12.0),
 'vesuvio': (['merguez', 'jambon', 'mozarella', 'poivrons', 'oignons'], 15.75),
```

```
'cabri': (['chevre', 'lardons', 'creme_fraiche', 'mozarella'], 11.25),
'napoli': (['jambon', 'tomates', 'mozarella', 'sauce_tomate',
'champignons', 'poivrons', 'oignons'], 19.5),
'neptune': (['saumon', 'creme_fraiche', 'champignons'], 11.25)}
```

Exercice 2

On utilise un dictionnaire pour gérer des données : les clés seront les prénoms des personnes, les valeurs seront des listes comprenant 4 valeurs : le sexe (1 ou 2), l'âge (un entier), la taille en mètres, le poids en kilos. On supposera que tous les prénoms sont différents.

1. Écrire la fonction `age(prenom, dico)` qui étant donnés un dico et un prénom renvoie l'âge de la personne du dico portant ce prénom ou 0 s'il n'y a personne portant ce prénom dans le dico.
2. Écrire la fonction `nbFemmes(dic)` qui compte combien il y a de femmes dans le dictionnaire dic.
3. On considère qu'une personne est en surpoids si son IMC est supérieure à 25 (l'IMC est obtenue en divisant le poids en kilos par la taille en mètres au carré).
   — Écrire la fonction `imc(taille,poids)` qui fait le calcul de l'IMC et le renvoie.
   — Écrire une fonction qui renvoie les prénoms des personnes en surpoids qui sont présentes dans le dictionnaire mis en paramètre.
4. Écrire la fonction `plus1(dico)` qui ajoute 1 an à toutes les personnes du dictionnaire dico.

```
>>> d
{'lise': [2,34, 1.78, 57], 'lea': [2,45, 1.76, 89], 'luc': [1,45, 1.76, 87],
'anne': [2,55, 1.55, 45], 'marie': [2,67, 1.67, 56], 'louise': [2,45, 1.56, 78],
'pierre': [1,32, 1.87, 67]}
>>> age("lea", d)
45
>>> age("lealea", d)
0
>>>nbFemmes(d)
5
>>> qui_surpoids(d)
['lea', 'luc', 'louise']
>>>plus1(d)
{'lise': [2 , 35, 1.78, 57], 'lea': [2 ,46, 1.76, 89], 'luc': [1 ,46, 1.76, 87],
'anne': [2,56, 1.55, 45], 'marie': [2,68, 1.67, 56], 'louise': [2,46, 1.56, 78],
'pierre': [1,33, 1.87, 67]}
```

Exercice 3

On travaille sur des dictionnaires dont les clés sont des noms de villes et les valeurs des nombres représentants le nombre d'habitants de la ville.

Par exemple :

```
>>> ex1= {'Lyon': 491268, 'Strasbourg': 272222, 'Angers': 148803, 'Rennes': 208033,
'Bordeaux': 239399, 'Nice': 344064, 'Reims': 180752, 'Toulouse': 447340, 'Metz': 119962,
'Amiens': 133327, 'Brest': 140547,  'Limoges': 137758, 'Grenoble': 157424,
'Caen': 108793, 'Nantes': 287845}
```

1. Écrire la fonction `moyenne(d)` qui étant donné un dictionnaire d renvoie le nombre d'habitants en moyenne des villes du dictionnaire d.
2. Écrire la fonction `plushabitants(nb, d)` qui étant donné un dictionnaire d et un nombre nb renvoie la liste des couples (ville, nombre d'habitants) pour les villes ayant au moins nb habitants.
3. Écrire la fonction `milliers(d)` qui étant donné un dictionnaire de ce type d modifie le dictionnaire pour que le nombre d'habitants soit indiqué en milliers. On pourra utiliser la fonction round qui fait un arrondi entier.
4. Écrire la fonction `maximini(d)` qui étant donné un dictionnaire d de ce type renvoie le couple composé de la ville ayant le plus d'habitants et celle en ayant le moins dans le dictionnaire d.

```
>>> moyenne(ex1)
227835.8

>>> plushabitants(250000,ex1)
[['Lyon', 491268], ['Strasbourg', 272222], ['Nice', 344064], ['Toulouse', 447340],
['Nantes', 287845]]

>>> ex1
{'Lyon': 491268, 'Strasbourg': 272222, 'Angers': 148803, 'Rennes': 208033,
'Bordeaux': 239399, 'Nice': 344064,
'Reims': 180752, 'Toulouse': 447340, 'Metz': 119962, 'Amiens': 133327, 'Brest': 140547,
 'Limoges': 137758, 'Grenoble': 157424, 'Caen': 108793, 'Nantes': 287845}

>>> milliers(ex1)
{'Lyon': 491, 'Strasbourg': 272, 'Angers': 149, 'Rennes': 208, 'Bordeaux': 239,
'Nice': 344, 'Reims': 181, 'Toulouse': 447, 'Metz': 120, 'Amiens': 133, 'Brest': 141,
 'Limoges': 138, 'Grenoble': 157, 'Caen': 109, 'Nantes': 288}
>>> maximini(ex1)
('Lyon', 'caen' )
```

5. On dispose de plus d'un dictionnaire qui a pour clés des noms de villes et comme valeurs le numéro du département correspondant. Attention ce dictionnaire comporte un certain nombre de villes mais pas forcément toutes celles des dictionnaires.

   Écrire la fonction `affichageavecdep(d, depart)` qui fait l'affichage du dictionnaire par ordre alphabétique des villes avec le numéro de département si le numéro de département est dans le dictionnaire departements (sinon on n'affiche pas la ville).

   On rappelle que le dictionnaires e sont pas ordonnés mais que si on a une liste de string maliste , alors la commande maliste.sort() triera la liste dans l'ordre alphabétique.

```
>>> departements={'Lyon': 69, 'Strasbourg': 67, 'Angers': 45, 'Rennes': 35,
'Bordeaux': 33, 'Nice': 6, 'Reims': 51, 'Metz': 57, 'Amiens': 80, 'Brest': 29,
'Grenoble': 38, 'Caen': 14, 'Nantes': 44}

>>> affichage2(ex1, departements)
Amiens  133327  habitants   numero :80
```

```
Angers 148803 habitants numero:45
Bordeaux 239399 habitants numero :33
Brest 140547 habitants numero :29
Caen 108793 habitants numero :14
Grenoble 157424 habitants numero :38
Lyon 491268 habitants numero :69
Metz 119962 habitants numero :57
Nantes 287845 habitants numero :44
Nice 344064 habitants numero :6
Reims 180752 habitants numero :51
Rennes 208033 habitants numero :35
Strasbourg 272222 habitants numero :67
```