Initiation à la programmation – TP14 – 24 novembre

Exercice 1 – listes de dictionnaires

On décide de représenter les résultats d'un étudiant dans un dictionnaire : les clés seront les matières et les valeurs seront les notes correspondantes.
Exemples :

```
e1={'alg': 9.0, 'python': 15, 'anglais': 7, 'cNum': 15}
e2={'alg': 8, 'python': 12, 'anglais': 12, 'cNum': 14, 'proba': 9, 'Archi': 11, 'Com': 12}
```

1. Travail sur un dictionnaire étudiant :
   Écrire une fonction qui cherche la meilleure note d'un étudiant et renvoie la ou les matières correspondantes ; s'il y a plusieurs notes identiques, on donnera toutes les matières correspondantes. On renverra dans tous les cas une liste.
2. En général, les notes n'ont pas le même coefficient. On aura donc un dictionnaire de toutes les matières possibles avec comme clé le nom de la matière et comme valeur le coefficient de celle-ci. Par exemple :

   ```
   coeffs={"alg":6,"python":6,"anglais":2,"cNum":4,"proba":6,"Archi":4, "Com":2}
   ```

   remarque : tous les étudiants n'ont pas forcément de notes dans toutes les matières.
   - Écrire une fonction sommeCoeff (etud) qui calcule la somme des coefficients des matières de l'étudiant etud.
   - Écrire la fonction moyenneCoeff(etud) qui étant donné un dictionnaire etud représentant un étudiant calcule sa moyenne pour les matières représentées en tenant compte des coefficients (qu'on ira chercher dans la variable globale coeffs)
   - En déduire la fonction recu(etud) qui teste si un étudiant est reçu : il sera reçu si la somme des coefficients des matières de l'étudiant est de 30 et si sa note calculée avec les coefficients est de 10 au moins. Si l'étudiant n'a pas assez de coefficients, on ne calculera pas sa note et on renverra « pas assez de notes ».

```
>>> e1
{'alg': 9.0, 'python': 15, 'anglais': 7, 'cNum': 15}
>>> noteMax(e1)
['python', 'cNum']
>>> moyenneCoeff(e1)
12.11
>>> recu(e1)
'pas assez de coeffs'

>>> e2
{'alg': 8, 'python': 12, 'anglais': 12, 'cNum': 14, 'proba': 9, 'Archi': 11, 'Com': 12}
>>> noteMax(e2)
['cNum']
>>> moyenneCoeff(e2)
10.73
>>> recu(e2)
```

```
True
```

3. Listes d'étudiants :
   On veut travailler avec des listes d'étudiants, c'est à dire des listes de dictionnaires.
   — Écrire une fonction qui compte le nombre d'étudiants reçus (en utilisant les fonctions précédentes et le dictionnaire coeffs).
   — Les professeurs veulent pouvoir modifier, pour une matière donnée, toutes les notes correspondantes dans une liste d'étudiants (par exemple un professeur qui se rendrait compte qu'il a mis ses notes sur 10 au lieu de 20 pourrait multiplier toutes ses notes par 2).
   Écrire la fonction ModifierLesDicos(listeEtud,mat, k) qui va multiplier toutes les notes correspondant à la matière mat par k. Attention tous les étudiants n'ont pas forcément de notes dans la matière mat.

```
>>> liste
[{'alg': 9.0, 'python': 15, 'anglais': 7, 'cNum': 15}, {'alg': 12, 'python': 12,
 'anglais': 8, 'cNum': 12, 'proba': 9, 'Archi': 18, 'Com': 11}, {'alg': 4,
 'python': 6, 'anglais': 12, 'cNum': 10, 'proba': 5, 'Archi': 11, 'Com': 7},
 {'alg': 3, 'python': 17, 'anglais': 14, 'cNum': 14, 'proba': 4, 'Archi': 15},
 {'alg': 10, 'python': 12, 'anglais': 8, 'cNum': 12, 'proba': 11, 'Archi': 11,
  'Com': 18}, {'alg': 8, 'python': 10, 'anglais': 8, 'cNum': 12, 'proba': 7,
  'Archi': 12, 'Com': 11}]

>>> nbRecus(liste)
2

>>> Modifie(liste, "alg", 1.5)
[{'alg': 13.5, 'python': 15, 'anglais': 7, 'cNum': 15}, {'alg': 18.0, 'python': 12,
'anglais': 8, 'cNum': 12, 'proba': 9, 'Archi': 18, 'Com': 11}, {'alg': 6.0,
'python': 6,'anglais': 12, 'cNum': 10, 'proba': 5, 'Archi': 11, 'Com': 7}, {'alg': 4.5,
'python': 17,  'anglais': 14, 'cNum': 14, 'proba': 4, 'Archi': 15}, {'alg': 15.0,
'python': 12, 'anglais': 8,  'cNum': 12, 'proba': 11, 'Archi': 11, 'Com': 18},
{'alg': 12.0, 'python': 10, 'anglais': 8,   'cNum': 12, 'proba': 7, 'Archi': 12,
'Com': 11}]

>>> nbRecus(liste)
3
```

Exercice 2

On décide de coder des phrases de façon enfantine en remplaçant un caractère par un nombre.
On va commencer par générer un dictionnaire qui représentera un code puis on codera puis décodera des phrases.
Les exemples sont donnés à la fin de l'exercice.

1. On partira d'une chaîne de caractères contenant tous les caractères autorisés qu'on appellera `caracteresPermis`. Par exemple on peut prendre :

```
caracteresPermis="abcdefghijklmnopqrstuvwxyz'-_., !?"
```

   Écrire la fonction `creerCode(chainepermise)` qui renvoie un dictionnaire où les clés sont les lettres de la chaîne `chainepermise` et les valeurs des entiers (évidemment tous différents) aléatoires entre `0` et

`len(chainepermise)-1`. On pourra créer la liste des entiers de `0` à `len(chainepermise)-1` et utiliser la fonction `shuffle` du module `random`.

2. Écrire la fonction `Coder(message, dico)` qui étant donnés une chaîne de caractères `message` et un dictionnaire de code `dico`, va coder le message et renvoyer une chaîne de caractères composés des nombres correspondants aux lettres du message séparés par des `/`.
3. Écrire la fonction `Inverse(dico)` qui renvoie un dictionnaire "inverse" du dictionnaire dico : les clés sont devenues les valeurs et réciproquement.
4. En utilisant la fonction précédente, écrire la fonction `Decoder(message, dico)` qui décode un message qui a été codé avec le dictionnaire `dico`.
   On commencera par étudier l'effet de la commande `messageCode.split("/")` si `messageCode` est le message après codage.

```
>>> monCode=creerCode(permis)
>>> monCode
{'a': 16, 'b': 17, 'c': 31, 'd': 27, 'e': 33, 'f': 14, 'g': 5, 'h': 10, 'i': 26, 'j': 0, 'k': 21,
 'l': 20, 'm': 3, 'n': 23, 'o': 24, 'p': 2, 'q': 29, 'r': 13, 's': 4, 't': 19, 'u': 32, 'v': 7,
 'w': 11, 'x': 9, 'y': 28, 'z': 22, "'": 18, '-': 12, '_': 15, '.': 25, ' ': 30, ',': 6,
 '!': 1, '?': 8}
>>> new=Coder("python est un langage formidable, j'adore coder en python!", monCode)
>>> new
'2/28/19/10/24/23/30/33/4/19/30/32/23/30/20/16/23/5/16/5/33/30/14/24/13/3/26/27/16/17/20/33/6/30/
0/18/16/27/24/13/33/30/31/24/27/33/13/30/33/23/30/2/28/19/10/24/23/1/'
>>> Decoder(new,monCode)
"python est un langage formidable, j'adore coder en python!"
>>> Inverse(monCode)
{16: 'a', 17: 'b', 31: 'c', 27: 'd', 33: 'e', 14: 'f', 5: 'g', 10: 'h', 26: 'i', 0: 'j', 21:
'k', 20: 'l', 3: 'm', 23: 'n', 24: 'o', 2: 'p', 29: 'q', 13: 'r', 4: 's', 19: 't', 32: 'u', 7:
 'v', 11: 'w', 9: 'x', 28: 'y', 22: 'z', 18: "'", 12: '-', 15: '_', 25: '.', 30: ' ', 6: ',',
  1: '!', 8: '?'}
```

Exercice 3 – Inversion de dictionnaires

Il peut être très utile « d'inverser » un dictionnaire (annuaires inversés, recherches statistiques...)

1. Premier cas un dictionnaire « simple » :
   On suppose avoir un dictionnaire où les valeurs sont uniques par exemple :{'chien' :'dog','rouge' :'red','deux' :'two'}.
   Écrire la fonction inverse1 qui renvoie le dictionnaire comprenant les paires val :cle
   Pour l'exemple ci-dessus on obtiendrait {'dog' :'chien, 'red' : 'rouge', 'two' :'deux'}
2. Deuxième cas : On travaille avec un dictionnaire avec valeurs répétées (on suppose quand même que les valeurs du dictionnaire peuvent être des clés).
   Par exemple : Si d= {'Jean' :12,'Pierre' :4,'Tom' :12,'Marie' :12,'Alain' :0,'Laurent' :2}, l'inversion simple n'est pas possible, car les clés se répéteraient. Quelle serait la valeur attachée à 12 dans ce cas ?
   On implémentera la stratégie suivante : si une valeur est unique, on inverse les clés et les valeurs. Sinon, on combine les valeurs identiques en une seule clé à laquelle on associe une liste de valeurs (les anciennes clés associées à cette valeur) ici on obtiendrait le dictionnaire :
   d1={12 :['Jean','Tom','Marie'],2 :Laurent',4 :'Pierre',0 :'Alain'}.

Écrire la fonction inverse2 correspondante.

```
>>> dico
{'red': 'rouge', 'blue': 'bleu', 'yellow': 'jaune', 'green': 'vert'}
>>> inverse(dico)
{'rouge': 'red', 'bleu': 'blue', 'jaune': 'yellow', 'vert': 'green'}

>>> dic
{'jean': 12, 'pierre': 12, 'marc': 12, 'anne': 17, 'marie': 7, 'paul': 8, 'pascal': 7}
>>> inverse2(dic)
{12: ['jean', 'pierre', 'marc'], 17: ['anne'], 7: ['marie', 'pascal'], 8: ['paul']}
>>>
```