Corrections exercices - Structures de données, interface et implémentation

Exercice 1 Dans chaque cas, indiquer quelle structure de données vous semble la plus adaptée, et donner une exemple de création et d'interaction avec la structure de données.

1. Une personne avec son nom, prénom, adresse email.
   Dictionaire

```
personne = {'nom':'Collet','prenom':'Mikael','mail':'xxx@yyy.com'}
nom = personne['nom']
```

2. Les notes de NSI d'un élève au cours d'un trimestre. **tableau (liste)**

```
note = [12, 15, 17, 15.5]
```

3. Les notes de NSI de tous les élèves d'une classe au cours d'un trimestre.
   tableau de dictionnaire

```
note_classe = [{'nom':'Collet','prenom':'Mikael','note':[12, 15, 17, 15.5]},
               {'nom':'Turing','prenom':'Alan','note':[20, 18, 17.5, 19.5]}]

nom_eleve1 = note_classe[1]['nom']
note_eleve1 = note_classe[1]['note']
```

4. Une grille de sudoku
   Tableau de tableau

```
grille = [[1, 2, 3, 4, 5, 6, 7, 8, 9],
          [1, 2, 3, 4, 5, 6, 7, 8, 9],
          [1, 2, 3, 4, 5, 6, 7, 8, 9],
          [1, 2, 3, 4, 5, 6, 7, 8, 9],
          [1, 2, 3, 4, 5, 6, 7, 8, 9],
          [1, 2, 3, 4, 5, 6, 7, 8, 9],
          [1, 2, 3, 4, 5, 6, 7, 8, 9],
          [1, 2, 3, 4, 5, 6, 7, 8, 9],
          [1, 2, 3, 4, 5, 6, 7, 8, 9]]
```

Exercice 2

1. Voici un programme python :
   Quels seront les contenus des listes `lst1`, `lst2` et `lst3` à la fin de l'exécution de ce programme ?
   lst1 : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
   lst2 : [1, -1, 0, 3, 5, 8]
   lst3 : [1, 0, 5, 9, 2, -8]
2. Voici la définition python d'un tableau :

```
matiere = ["Nsi", "Mathematiques", "Anglais", "Philosophie", "Histoire", "
    Physique", "Svt"]
```

(a) Qu'affiche l'instruction `print(len(matiere))` ? **7**

(b) Qu'affiche l'instruction `print(len(matiere[1]))` ? **13**

(c) Quel sera le contenu des listes `lst1` et `lst2` à la fin de l'exécution du code suivant ?

```
lst1=[]
for i in range(2,5):
        lst1.append(matiere[i])

lst2=[]
for m in matiere:
    if len(m) < 5:
        lst2.append(m)
```

lst1 : ['Anglais', 'Philosophie', 'Histoire']
lst2 : ['Nsi', 'Svt']

(d) Quel sera le contenu de la variable `x` à la finc de l'exécution du code suivant ?

```
x = 0
for v in matiere:
    if 'a' in v:
        x+=1
```

a = 2

3. Quel est le contenu des tableaux associés à `mon_tab1`, `mon_tab2` et `mon_tab3` après l'exécution du programme ci-dessous ?

```
mon_tab1 = [p for p in range(0, 5)]
mon_tab2 = [v for v in mon_tab1[1:]]
mon_tab3 = [v for v in mon_tab1[-1:]]
```

mon_tab1 : [0, 1, 2, 3, 4]
mon_tab2 : [1, 2, 3, 4]
mon_tab3 : [4]

4. Quel est le contenu du tableau associé à `mon_tab` après l'exécution du programme ci-dessous ?

```
l = [1, 7, 9, 15, 5, 20, 10, 8]
mon_tab = [p for p in l if p > 10]
```

mon_tab : [15, 20]

Exercice 3

1. Quelle est la valeur associée au nom a après l'exécution du programme ci-dessous ?

```
m = [[1, 3, 4],
     [5, 6, 8],
     [2, 1, 3],
     [7, 8, 15]]
a = m[1][2]
```

a = 8

2. Quel est le contenu du tableau associé au nom mm après l'exécution du programme ci-dessous ?

```
m = [1, 2, 3]
mm = [m, m, m]
m[0] = 100
```

mm = [[100, 2, 3], [100, 2, 3], [100, 2, 3]]

3. Soit le code suivant :

```
m = [[1, 3, 4],
     [5, 6, 8],
     [2, 1, 3],
     [7, 8, 15]]
nb_colonne = 3
nb_ligne = 4
a = 0
for i in range(0, nb_ligne):
  for j in range(0, nb_colonne):
    if i>j:
      a += m[i][j]
print(a)
```

Quelle est la valeur de `a` affichée après l'exécution de ce programme ?
a = 38

Exercice 4

1. Écrivez un programme Python qui permettra de créer un dictionnaire avec les couples clé/valeur suivants : nom → Doe ; prenom → John ; age → 42

```
personne = {'nom':'Doe','prenom':'John','age':42}
nom = personne['nom']
```

2. Soit les données ci-contre :
Proposez une structure de données permettant de stocker ces informations (on attend un programme Python).

Prénom	Nom	Age
George	Orwell	122
Pierre	Martin	23
Alan	Turing	113
Mikaël	Collet	46

```
info = [{'nom':'Orwell','prenom':'George','age':122},
        {'nom':'Martin','prenom':'Pierre','age':23},
        {'nom':'Turing','prenom':'Alan','age':113},
        {'nom':'Collet','prenom':'Mikael','age':46}]
```

3. Ecrivez un programme Python permettant de lister l'ensemble des noms "stockés" dans le dictionnaire précédent (on attend une structure de type boucle)

```
for e in info:
        print(e['nom'])
```

Exercice 5

1. Soit le programme Python suivant :

```
inventaire = {'pommes': 430, 'bananes': 312,'oranges' : 274, 'poires' : 137}
stock = 0
for fruit in inventaire.keys():
  if fruit != 'bananes':
    stock = stock + inventaire[fruit]
```

   Quelle est la valeur de la variable stock après l'exécution de ce programme ?
   841

2. Soit le programme Python suivant :

```
P = [{"nom":"Turing","prenom":"Alan","age":28},{"nom":"Lovelace","prenom":"Ada
    ","age":27}]
```

   Qu'obtient-on si on tape P[1]['age'] dans une console Python ?
   27

3. Soit le programme Python suivant :

```
def ajoute(stock,element,quantite):
  if element in stock:
    stock[element] = stock[element] + quantite
  else:
    stock[element] = quantite

stock = { 'clous': 14, 'vis': 27, 'boulons': 8, 'ecrous': 24 }
ajoute(stock,'vis',5)
ajoute(stock,'chevilles',3)
```

   Quelle est la valeur de la variable stock à la fin de cette exécution ?

```
stock = { 'clous': 14, 'vis': 32, 'boulons': 8, 'ecrous': 24, 'chevilles':3 }
```

Exercice 6 Écrire une fonction `ajoute_dictionnaires` qui prend en paramètres deux dictionnaires `d1` et `d2` dont les clés sont des nombres et renvoie le dictionnaire `d` défini de la façon suivante :

- les clés de `d` sont celles de `d1` et celles de `d2` réunies ;
- si une clé est présente dans les deux dictionnaires `d1` et `d2`, sa valeur associée dans le dictionnaire `d` est la somme de ses valeurs dans les dictionnaires `d1` et `d2` ;
- si une clé n'est présente que dans un des deux dictionnaires, sa valeur associée dans le dictionnaire `d` est la même que sa valeur dans le dictionnaire où elle est présente.

Exemples :

```
>>> ajoute_dictionnaires({1: 5, 2: 7}, {2: 9, 3: 11})
{1: 5, 2: 16, 3: 11}
>>> ajoute_dictionnaires({}, {2: 9, 3: 11})
{2: 9, 3: 11}
>>> ajoute_dictionnaires({1: 5, 2: 7}, {})
{1: 5, 2: 7}
```

```
def ajoute_dictionnaires(d1, d2):
    d = {}
    for k in d1:
        if k in d2 :
            d[k] = d1[k] + d2[k]
        else :
            d[k] = d1[k]
    for k in d2:
        if k not in d:
            d[k] = d2[k]
    return d
```

Exercice 7 Un ingénieur en télécommunication doit créer une interface afin de transmettre des messages, selon un protocole donné, entre deux éléments d'un réseau de télécommunication. Il a dans un premier temps défini l'interface suivante :

- `createMessage(sender, receiver, payload)` : crée un élément de type `Message` à partir de trois paramètres `sender` (un nombre entier identifiant l'émetteur du message), `receiver`(un nombre entier identifiant le destinataire du message) et `payload` (une chaine de caractère qui représente le contenu du message).
- `getSender(msg)` : accès à l'émetteur du message `msg` (renvoie un entier)
- `getReceiver(msg)` : accès au destinataire du message `msg` (renvoie un entier)
- `getPayload(msg)` : accès au contenu du message `msg` (renvoi une chaine de caractère)
- `setSender(msg, sender)` : modifie l'émetteur du message `msg`
- `setReceiver(msg, receiver)` : modifie le destinataire du message `msg`
- `setPayload(msg,payload)` : modifie le contenu du message `msg`
- `getMsgSize(msg)` : retourne la taille du contenu du message `msg` en octet.
- `sendMsg(msg)` : envoi le contenu du message vers le destinataire du message . Le contenu du message est encodé selon l'encodage UTF-8.

Vois devez maintenant utiliser cette interface et créer un programme en python qui doit effectuer les tâches suivantes :

1. L'émetteur numéro 1979 doit envoyer le message "Hello World !" au destinataire 1984.
2. Puis le même émetteur envoie le même message au destinataire 2025.
3. Puis le même émetteur envoie le message "Big brother is watching you !" au destinataire 1984.

Remarque : Un message ne peut être envoyé que si la taille du message ne dépasse pas 20 octets.

```
msg = creerMessage(1979, 1984, 'Hello world !')
if getMsgSize(msg) < 20:
  sendMsg(msg)
setReceiver(msg,2025)
if getMsgSize(msg) < 20:
  sendMsg(msg)
setPayload(msg,'Big brother is watching you !')
if getMsgSize(msg) < 20:
  sendMsg(msg)
```

Exercice 8

1. Expliquer pourquoi la structure de donnée `tuple` ne peut pas être utilisée pour implémenter l'interface de l'exercice 7.
   La structure tuple n'est pas modifiable

2. Proposer deux implémentations python pour représenter le message suivant :
   - sender : 1979
   - receiver : 1984
   - payload : ”Big brother is watching you !”

   (a) liste : msg = [1979, 1984, ’Big brother is watching you !’]

   (b) dictionnaire : msg = {’sender’ :1979, ’receiver’ :1984, ’payload’ :’Big brother is watching you !’}

3. Pour chacune des deux repésentations précédentes, proposer une implémentation en python des 4 fonctions suivantes :
   - `createMessage(sender, receiver, payload)`
   - `getSender(msg)`
   - `setReceiver(msg, sender)`
   - `getMsgSize(msg)`

1. version liste :

```
def createMessage(sender, receiver, payload):
        return [sender, receiver, payload]

def getSender(msg):
        return(msg[0])

def setReceiver(msg, receiver):
        msg[1] = receiver

def getMsgSize(msg):
        return len(msg[2].encode('utf-8'))
```

2. version dictionnaire :

```
def createMessage(sender, receiver, payload):
        return {'sender':sender, 'receiver':receiver, 'payload':payload}

def getSender(msg):
        return(msg['sender'])

def setReceiver(msg, receiver):
        msg['receiver'] = receiver

def getMsgSize(msg):
        return len(msg['payload'].encode('utf-8'))
```