INFO-H-100 - Programmation
TP 13 — Dictionnaires et fichiers

Lorsque l'exercice demande d'écrire une fonction, écrivez la fonction demandée et testez-la avec plusieurs valeurs pertinentes.

Ex. 1. Ecrire une fonction qui renvoie un dictionnaire composé du nombre d'occurrences de tous les mots de 2 lettres (AA, AC, AG, ...) qui existent dans une séquence nucléotidique.

Exemple :

```
>>> dico = occurrences2lettres("ACCTAGCCATGTAGAATCGCCTAGGCTTTAGCTAGCTCTAGCTAGCTG")
>>> print dico
{'AA': 1, 'AC': 1, 'GT': 1, 'AG': 7, 'CC': 3, 'CA': 1, 'CG': 1,
 'TT': 2, 'GG': 1, 'GC': 7, 'AT': 2, 'GA': 1, 'TG': 2, 'CT': 8,
 'TC': 2, 'TA': 7}
```

Ex. 2. Ecrire une fonction qui affiche un dictionnaire de la façon suivante :

```
Key : GT - Value : 1
Key : CT - Value : 8
...
```

Bonus : afficher les clés dans l'ordre alphabétique.

Ex. 3. Ecrire une fonction qui renvoie un dictionnaire composé de chaque mot contenu dans le fichier `dico.txt` disponible sur la page web des TPs (la valeur associée à la clé n'a pas d'importance dans cet exercice). Le fichier `dico.txt` est composé de lignes contenant chacune un mot en majuscule. Chaque mot est unique.

Ensuite, écrire une fonction qui vérifie qu'un mot donné en paramètre est bien dans ce dictionnaire.

```
dico = chargerDico("dico.txt")
print estDansDico("bonjour",dico) #-> True
```

Ex. 4. Ecrire une fonction qui renvoie un dictionnaire contenant le nombre d'occurrences de chaque mot du fichier `hamlet.txt` disponible sur la page web des TPs.

Avant cela, veillez à nettoyer le texte, c'est à dire enlever chaque caractère de ponctuation et mettre en majuscule tous les mots. Renseignez-vous dans la documentation pour trouver les fonctions nécessaires à ce travail (module `string`).

Créez ensuite une fonction qui détermine le mot le plus utilisé dans ce texte.

```
dico = occurrencesMots("hamlet.txt")
print motPlusFrequent(dico) # -> THE
```

Ex. 5. Soit le fichier `address.dat` suivant :

```
LastName : Verhaegen
FirstName : Boris
Office : UB4.131
Phone : 026503766
====================
LastName : Silovy
FirstName : Alain
```

Les entrées sont séparées par une ligne de 20 signes =. Elles sont composées de champs de forme `cle : valeur` séparés par des retour à la ligne. Les champs `LastName` et d'un `FirstName` sont obligatoires et les champs `Office` et `Phone` sont optionnels. L'ordre des entrées et des champs n'a pas d'importance.

Ecrire un programme qui permet :
– de charger le carnet d'adresse dans une liste de dictionnaires,

– d’afficher le carnet d’adresse
– d’ajouter une entrée dans le carnet d’adresse,
– de supprimer une entrée dans le carnet d’adresse et
– de sauvegarder le carnet d’adresse dans le fichier `address.dat`

INFO-H-100 - Programmation
TP 13 — Dictionnaires et fichiers
Corrections

Solution de l'exercice 1:

```
def occurrences2lettres(sequence):
    dico = {}
    for i in range(len(sequence)-1):
        cle = sequence[i]+sequence[i+1]
        dico[cle] = dico.get(cle,0) + 1
    return dico
```

Solution de l'exercice 2:

```
def afficheDico(dico):
    for cle in dico:
        print "Key : "+str(cle)+" - Value : "+str(dico[cle])
```

Version triée :

```
def afficheDico(d):
    for c in sorted(d):
        print "Key: " + str(c) + " - Value: " + str(d[c])
```

ou

```
def afficheDico(dico):
    cles = dico.keys()
    cles.sort()
    for cle in cles:
        print "Key : "+str(cle)+" - Value : "+str(dico[cle])
```

Solution de l'exercice 3:

```
def chargerDico(nomFich):
    f = open(nomFich)
    liste = f.read().split()
    dico = {}
    for mot in liste:
        dico[mot] = None
    return dico

def estDansDico(mot,dico):
    return mot.upper() in dico

dico = chargerDico("dico.txt")
print estDansDico("bonjour",dico)
```

Autre solution avec un `set` :

```
def chargerDico(nomFich):
    f = open(nomFich)
    liste = f.read().split()
    dico = set()
    for mot in liste:
        dico.add(mot)
    return dico
```

Solution de l'exercice 4:

```
def mots(texte):
    import string
    itab = string.punctuation            #les signes de ponctuation...
    otab = " " * len(string.punctuation) #a remplacer par des blancs
    tab = string.maketrans(itab, otab)
    texte = texte.translate(tab)
    texte = texte.upper()
    return texte.split()
```

```
def occurrencesMots(nomFich):
    f = open(nomFich)
    texte = f.read()
    listeMots = mots(texte)
    dico = {}
    for mot in listeMots:
        dico[mot] = dico.get(mot, 0) + 1
    return dico

def motPlusFrequent(dico):
    maxi = 0
    motMax = ""
    for mot in dico:
        if dico[mot] > maxi:
            maxi = dico[mot]
            motMax = mot
    return (motMax, maxi)

dico = occurrencesMots("hamlet.txt")
print motPlusFrequent(dico)
```

Solution de l'exercice 5:

```
def getEntryDico(entry):
    lines = entry.split("\n")
    dico = {}
    for line in lines:
        if line != '':
            key,value = line.split(" : ")
            dico[key]=value
    return dico

def loadAddressFile():
    f = open("address.txt")
    t = f.read()
    t = t.strip()
    entries = t.split("\n====================\n")
    addressList = []
    for entry in entries:
        dico = getEntryDico(entry)
        addressList.append(dico)
    return addressList

def addressToString(address):
    s = ""
    for key,value in address.items():
        s+=key+" = "+value+" "
    return s

def showBook(addressList):
    for i in range(len(addressList)):
        s = "["+str(i)+"]"+addressToString(addressList[i])
        print s

def actionAdd(addressList):
    firstName = raw_input("Enter a first name (mandatory) : ")
    lastName = raw_input("Enter a last name (mandatory) : ")
    office = raw_input("Enter office (optional) : ")
    phone = raw_input("Enter phone (optional) : ")
    dico = {"FirstName":firstName, "LastName":lastName}
    if office != "":
        dico["Office"] = office
    if phone != "":
        dico["Phone"] = phone
    addressList.append(dico)

def actionDelete(addressList):
    toDel = int(raw_input("Item to delete : "))
    del addressList[toDel]

def actionSave(addressList):
    f = open("address.txt","w")
    for i in range(len(addressList)):
        for key,value in addressList[i].items():
            f.write(key+" : "+value+"\n")
```

```
        if i < len(addressList)-1:
            f.write("====================\n")
    f.close()

def menu():
    addressList = loadAddressFile()
    text = """1) Delete item\n2) Add item\n3) Save\n4) Exit"""
    choice = 0
    while choice != 4:
        showBook(addressList)
        print text
        choice = int(raw_input(" > "))
        if choice == 1:
            actionDelete(addressList)
        elif choice == 2:
            actionAdd(addressList)
        elif choice == 3:
            actionSave(addressList)

menu()
```