INFO-H-600 - Python
TP 4 - Sets, dictionnaries and files

Ex. 1. Write a function `word_in_file(word, fname)` that detects if a word given as argument is in a file which name is also provided as argument (`fname`). We suppose that the file contains one word per line. There is no supposition on the order of the words in the file.

Example: Content of the `test.txt` file

```
analphabete
croisiere
bricabrac
```

```
>>> word_in_file("hello", "test.txt")
False
>>> word_in_file("croisiere", "test.txt")
True
```

Ex. 2. Write a function `longest_word(fname)` that finds the longest word in a file (the name `fname` of the file is given as argument). The result must be returned as a tuple containing the longest word and its length. As for the previous exercices, we suppose that each line containes only on word per line with no information about the order of the words.

Example: Content of the `test2.txt` file

```
zero
un
deux
trois
quatre
cinq
```

```
>>> longest_word("test2.txt")
("quatre", 6)
```

Ex. 3. Write a function that return a dictionnary composed of the number of occurencts of each word of 2 letters (AA, AC, AG, ...) contained in a sequence of nucleotide.

Example :

```
>>> dico = occurrences2letters("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. 4. Write a function that prints the dictionnary in the following manner:

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

Bonus : print the keys in alphabetic order:

Ex. 5. Write a function that returns a datastructure composed of each word contained in the file `dico.txt` available on the UV. (If you choose to use a dictionnary, the value associated to the keys in this dictionnary has no importance in this exercise). The file `dico.txt` is composed of lines containing on word in capital case. Each word is present only once in the file.

Then, write a function that checks if a word given in parameter is in this dictionnary.

```
dico = load_dico("dico.txt")
print(is_in_dico("hello",dico)) #-> True
```

Ex. 6. Write a function that returns a dictionnary containing the number of occurrences of each word in the file `hamlet.txt` which can be found on the UV.

Before doing this, you should *clean the text* which means removing the punctuation symbols and setting all the words in upper case. Please find in the documentation which functions can help you performing this job (modul `string`).

Then, create a function that determines which word is the most used in this text.

```
dico = occurrences_words("hamlet.txt")
print(most_used_word(dico)) # -> THE
```

Ex. 7. Given the file `address.dat` starting as follows:

```
LastName : Jean
Office : UB4.131
FirstName : Dupont
Phone : 026503770
====================
LastName : John
FirstName : Dupond
```

The entries are separated by a line of 20 = signs. Each entry is composed of a field of the form `key : value` separated by a new line symbol. The fields `LastName` and `FirstName` are mandatory and the fields `Office` and `Phone` are optional. The order of the fields has no importance.

Write a program that allows to:

- load the address book in a list of dictionnaries,
- show the address book,
- add an entry in the address book,
- delete an entry in the address book,
- save the address book in the file `address.dat`

Bonus:

Ex. 8. Write a function `words_in_files(words, fnames)` that detects if one of the words given as argument is in on a the files which names are also provided as argument (`fnames`). We suppose that the files contain one word per line. There is no supposition on the order of the words in the file. Please compare the speed of execution with large files and long lists of words. What datastructure is most suited for this kind of job ?

Example: Content of the `test1.txt` file

```
analphabete
croisiere
bricabrac
```

Content of the `test2.txt` file

```
ball
pepper
science
```

```
>>> word_in_file(["hello", "disco"], ["test1.txt", "test2.txt"])
False
>>> word_in_file(["ball", "ten"], ["test1.txt", "test2.txt"])
True
```

INFO-H-600 - Programmation
TP 4 - Sets, dictionnaries and files
Corrections

Solution to the exercise 1:

```
def word_in_file(word,fname):
    f = open(fname)
    line = f.readline()
    while line != '' and line.strip() != word:
        line = f.readline()
    return line != ''
```

Solution to the exercise 2:

```
def longest_word(fname):
    f = open(fname)
    max_pair = ("", 0)
    for line in f.readlines():
        word = line.strip()
        if len(word) > max_pair[1]:
            max_pair = (word, len(word))
    return max_pair
```

Solution to the exercise 3:

```
def occurrences2letters(sequence):
    dico = {}
    for i in range(len(sequence)-1):
        key = sequence[i]+sequence[i+1]
        dico[key] = dico.get(key,0) + 1
    return dico
```

Solution to the exercise 4:

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

Sorted version:

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

or

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

Solution to the exercise 5:

```
def loadDico(fileName):
    f = open(fileName)
    list_words = f.read().split()
    dico = {}
    for word in list_words:
        dico[word] = None
    return dico

def isInDico(word, dico):
    return word.upper() in dico

dico = loadDico("dico.txt")
print(isInDico("bonjour", dico))
```

Another solution using a `set` :

```
def loadFile(fileName):
    f = open(fileName)
    list_words = f.read().split()
    res = set()
    for word in list_words:
        res.add(word)
    return res
```

Solution to the exercise 6:

```
def words(text):
    import string
    itab = string.punctuation             # punctuation symbols
    otab = " " * len(string.punctuation) # to replace by space
    tab = str.maketrans(itab, otab)       # translation table
    text = text.translate(tab)            # perform the translation
    text = text.upper()
    return text.split()

def occurrenceswords(fileName):
    f = open(fileName)
    text = f.read()
    wordlist = words(text)
    dico = {}
    for word in wordlist:
        dico[word] = dico.get(word, 0) + 1
    return dico

def wordMostUsed(dico):
    maxi = 0
    wordMax = ""
    for word in dico:
        if dico[word] > maxi:
            maxi = dico[word]
            wordMax = word
    return (wordMax, maxi)

dico = occurrenceswords("hamlet.txt")
print(wordMostUsed(dico))
```

Solution to the exercise 7:

```
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("====================")
    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 = input("Enter a first name (mandatory) : ")
    lastName = input("Enter a last name (mandatory) : ")
    office = input("Enter office (optional) : ")
    phone = 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(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(input(" > "))
        if choice == 1:
            actionDelete(addressList)
        elif choice == 2:
            actionAdd(addressList)
        elif choice == 3:
            actionSave(addressList)

menu()
```

Solution to the exercise 8:

```
def load_dico_list(dico):
    f = open(dico, 'r')
    words = []
    for line in f:
        words.append(line.strip())
    return words

def load_dico_set(dico):
    f = open(dico, 'r')
    words = set()
    for line in f:
        words.add(line.strip())
    return words


def words_in_files(words, fileNames):
    files_words = []
    for f in fileNames:
        # comment the line wanted to compare lists and sets
        files_words.append(load_dico_list(f))
        # files_words.append(load_dico_set(f))

    for word in words:
        for file_words in files_words:
            if word in file_words:
                return True
    return False

# for simple cases the difference is not so obvious
print(words_in_files(["sdsf", "ball", "ekfsdf"], ["dico.txt", "dico_fr.txt"]))
# for larger cases sets are significantly faster for this operation !
print(words_in_files(list(range(1000)), ["dico.txt", "dico_fr.txt"]))
```