ES112: Computing
Lab 06 — 22 October 2018

Topic: Dictionaries and File Handling

1. You are given a dictionary **grades_data** containing the ids of students as keys and a list of marks of 3 subjects associated with each id. Return a list containing tuples of the ids of the students and sum total of marks associated with each id, sorted in descending order according to total marks.

   Sample input.

   ```
   {57:[25,23,35],12:[10,10,20],50:[45,48,50]}
   ```

   Expected output.

   ```
   [(50, 143), (57, 83), (12, 40)]
   ```

   Function Signature

   ```
    def solve01(grades_data):
   ```

   - **grades_data** is a dictionary.
   - The expected return type is a list.

2. You are given a string **`elections`** containing some election data in the following format: the string is a comma-separated list of chunks, where a chunk consists of a string, followed by a space, followed by a number. This corresponds to the name of the candidate and the number of votes they got in some state. Count the results of the elections: sum the number of votes for each candidate.

   Return a string which has the same format but has every candidate's name appearing exactly once, and in alphabetical order, and the number would be the total number of votes received by the candidate.

   Sample input

   ```
   McCain 10, McCain 5, Obama 9, Obama 8, McCain 1
   ```

   Expected output

   ```
   McCain 16, Obama 17
   ```

   Function Signature

   ```
    def solve02(elections):
   ```

   - **`elections`** is a string.
   - The expected return type is a string.

3. Some text is given as a string. For each word of the text count the number of its occurrences before it, and return this as a list of numbers.

   Sample input

   ```
   message = "one two one two three two four three"
   ```

   Expected output

   ```
   "0 0 1 1 0 2 0 1"
   ```

   Function Signature

   ```
    def solve03(message):
   ```

   - **message** is a string.
   - The expected return type is a string.

4. You are given a list of integers **L**. Return a list of integers from **L** which have even frequency in **L** in the ascending order.

   Sample input.

   ```
   L = [1, 2, 3, 2, 1, 3]
   ```

   Expected output.

   ```
   [1,2,3]
   ```

   Sample input.

   ```
   L = [1, 1, 3, 1]
   ```

   Expected output.

   ```
   []
   ```

   Function Signature

   ```
   def solve04(L):
   ```

   - **L** is a list of integers.
   - The expected return type is *list*.

5. You are given two strings **A** and **B** corresponding to sentences. A sentence is a string of space separated words. Each word consists only of lowercase and uppercase letters.

   A word is *uncommon* if it appears exactly once in one of the sentences, and does not appear in the other sentence. Return the count of all uncommon words. Do not consider the case sensitivity — for example, "This" and "this" are considered the same.

   Sample input.

   ```
   A: This is a string
   B: this is another str
   ```

   Expected output.

   ```
   4
   ```

   Function Signature

   ```
   def solve05(A,B):
   ```

   - **A** and **B** are strings.
   - The expected return type is an int.

6. You are given a list consisting of tuples of size two, each element of which corresponds to a string. These tuples correspond to pairs of words. Every word is a synonym of the other word in its pair. All the words in the dictionary are different.

   Apart from the list, there's one more word given. Find a synonym for it.

   Sample input

   ```
   [("Hello", "Hi"), ("Bye", "Goodbye"), ("List", "Array")], "Goodbye"
   ```

   Expected output

   ```
   "Bye"
   ```

   Function Signature

   ```
   def solve06(synonyms,query):
   ```

   - **synonyms** is a list of tuples while **query** is a string.
   - The expected return type is also a string.
   - Quotes shown in the sample inputs and outputs are only for clarity.

7. You are given a list **A** of strings, each containing a file extension in lowercase and the corresponding MIME type separated by a space (for example, "txt text/plain"). You are also given a list **B** of file names. The file extension is the part of the file name after the last period (`.`) and is **not case sensitive** (i.e. **TXT**, **tXt**, and **txt** are the same).

   Return a list containing the correct MIME type for each file name. If the MIME type cannot be determined (either because the file name has no periods, or because the extension is not present in the list **A**), then the corresponding list element should be **None**.

   Sample input

   ```
   A = ["txt text/plain", "pdf application/pdf", "doc application/msword"]
   B = ["myfile.txt", "another.document.pdf", "helloworld", "figure.png",
   "report-1.doc", "article.TXT"]
   ```

   Expected output

   ```
   ["text/plain", "application/pdf", None, None, "application/msword", "text/plain"]
   ```

   Function Signature

   ```
   def solve07(A, B):
   ```

   - **A** and **B** are lists.
   - The expected return type is a list.
   - MIME types contain no spaces.

8. You are given an undirected graph with **n** vertices numbered **0** through **n-1**. For each valid **i**, there is an undirected edge connecting two different vertices **x[i]** and **y[i]**. No two edges connect the same pair of vertices.

   A triangle is a set of three distinct vertices such that each pair of those vertices is connected by an edge. Formally, three distinct vertices u,v,w are a triangle if the graph contains the edges (u,v), (v,w), and (w,u). You are given the description of the graph: the integer **n** and the lists of integers **x** and **y**. You are allowed to add edges to this graph. You may add as many edges as you want, and each of them may connect any two vertices. Your goal is to produce a graph that contains at least one triangle. Compute and return the smallest number of edges you need to add.

   Sample input 1

   ```
   n = 3
   x = [], y = []
   ```

   Expected output 1

   ```
   3
   ```

   Sample input 2

   ```
   6
   x = [0,0,2], y = [3,1,4]
   ```

   Expected output 2

   ```
   1
   ```

   Sample input 3

   ```
   n = 4
   x = [0,2,1,2], y = [3,0,2,3]
   ```

   Expected output 3

   ```
   0
   ```

   Function Signature

   ```
    def solve08(n,x,y):
   ```

   - **n** is an integer and **x,y** are lists of integers. The expected return type is also an integer.
   - **x** will have between **0** and **n*(n-1)/2** elements, inclusive and **y** will have the same number of elements as **x**.
   - No two edges will connect the same pair of vertices.

9. *(Not for grading on Gradescope.)*

   The virus attacked the filesystem of the supercomputer and broke the control of access rights to the files. For each file there is a known set of operations which may be applied to it:

   - write W,
   - read R,
   - execute X.

   The data regarding the access rights is available in a file called **`file_access.txt`**. You can download this file from the webpage for this lab assignment.

   The first line contains the number N âĂŤ the number of files contained in the filesystem. The following N lines contain the file names and allowed operations with them, separated by spaces. The next line contains an integer M âĂŤ the number of operations to the files. In the last M lines specify the operations that are requested for files. One file can be requested many times.

   You need to recover the control over the access rights to the files. For each request your program should return **`OK`** if the requested operation is valid or **`Access denied`** if the operation is invalid. These responses should be written to a file, separated by newlines. Write this output to a file called **`access_responses.txt`**.

   Sample input

```
4
helloworld.exe R X
pinglog W R
nya R
goodluck X W R
5
read nya
write helloworld.exe
execute nya
read pinglog
write pinglog
```

   Expected output

```
OK
Access denied
Access denied
OK
OK
```

10. *(Not for grading on Gradescope.)*

    In this problem, you will implement an interactive English dictionary.

    First, download the following data file:

    data.json

    from the webpage for this lab assignment.

    Then load the data from this file into a regular Python dictionary.

    Now use this dictionary to ask the user for a word and:

    1. If the word is present in the dictionary, output the meaning,
    2. If the word is missing from the dictionary, then check if there are any close matches:
        (a) If close matches are found then display them and ask the user to choose one of them and display an appropriate output; otherwise,
        (b) If no close matches are found then exit after displaying an appropriate error message.

    To check for close matches, you can use the **`get_close_matches()`** function from the difflib library. You can find the documentation here:

    `https://docs.python.org/3/library/difflib.html`

    To help you get started, you can use this template python file provided on the webpage for this lab.