Lab Exercise #8

Assignment Overview

This lab exercise provides practice with dictionaries in Python.

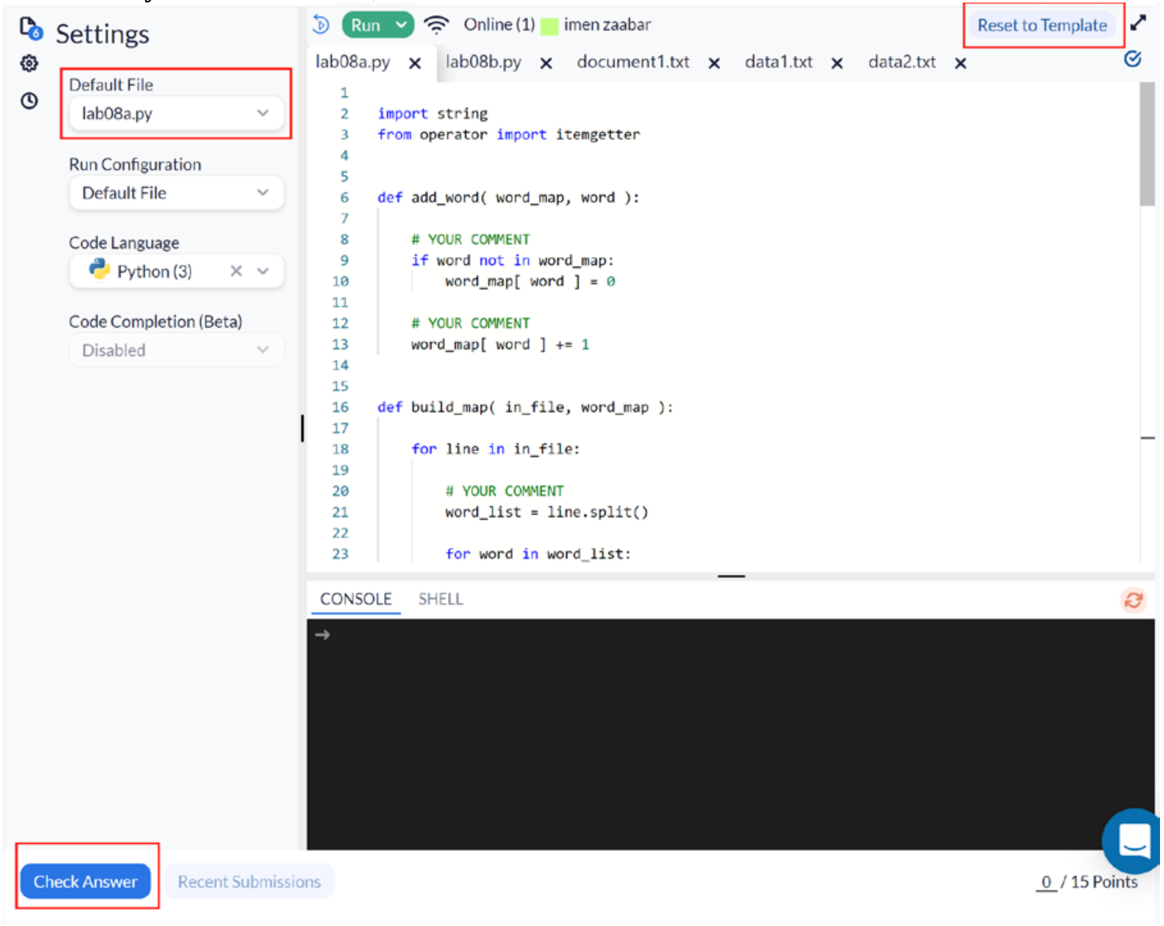
★ **On-line students: you can do it on your own or with a partner. Best place to find a partner is Piazza.** If you are working with a partner, two people should work at one computer. Occasionally switch the person who is typing. Talk to each other about what you are doing and why so that both of you understand each step.

Instructions for automatic testing:

There are 2 parts for this lab: (1) modifying an existing code, (2) write your own program.
You need to work on Spyder or any professional IDE that you are comfortable with. Once you are ready to check your code, submit to Coding Rooms.

When submitting to Coding Rooms:

1. first, you need to click on the "**Reset to template**" button to clear/reset to the provided starter codes in the coding window. Then, you can copy and paste your code into the corresponding Coding Rooms IDE (Coding window) in *lab08a.py* and *lab08b.py*
2. To check your answer: Make sure that your default file is *lab08a.py*, click on the gear icon and from the drop down choose `lab08a.py`
3. Click on the "**Check Answer button**". You can click on that button as many times as you want.
4. To submit your final version, click on the "**Submit**" button.



Default File: lab08a.py
Run Configuration: Default File
Code Language: Python (3)
Code Completion (Beta): Disabled

```

import string
from operator import itemgetter


def add_word( word_map, word ):

    # YOUR COMMENT
    if word not in word_map:
        word_map[ word ] = 0

    # YOUR COMMENT
    word_map[ word ] += 1


def build_map( in_file, word_map ):

    for line in in_file:

        # YOUR COMMENT
        word_list = line.split()

        for word in word_list:
```

A. Modify a program that uses Dictionaries.

Consider the file named "`lab08a.py`". That file contains the skeleton of a Python program to do a simple analysis of a text file: it will display the number of unique words which appear in the file, along with the number of times each word appears. Case does not matter: the words "pumpkin", "Pumpkin" and "PUMPKIN" should be treated as the same word. (The word "map" is used in identifiers because sometimes a dictionary is called a "map.")

Execute the program (which currently uses "document1.txt" as the data file) and inspect the output.

a. Replace each of the lines labeled "YOUR COMMENT" with meaningful comments to describe the work being done in the next block of statements. Use more than one comment line, if necessary.

b. Add doc strings to each function to describe the work being done in the function.

c. The program currently processes the empty string as a word. Revise the program to exclude empty strings from the collection of words.

d. The program currently processes words such as "The" and "the" as different words. Revise the program to ignore case when processing words.

e. The program currently always uses "document1.txt" as the input file. Revise the program to prompt the user for the name of the input file.

f. Revise the program to display the collection of words sorted by greatest frequency of occurrence to least frequency, and sorted alphabetically for words with the same frequency count. Since the `sorted` function and the `sort` method are stable sorts, you can first sort the words alphabetically, and then sort them by frequency (with `reverse=True`). (You do the two sorts in that order because you do the primary key last, frequency is the primary key in this case.) By default sorting is done on the first item in a list or tuple. To sort on other items use `itemgetter` from the `operator` module. See documentation here, focus on the `students_tuple` example:
https://docs.python.org/3/howto/sorting.html

g. Test the revised program. There are two sample documents available: "document1.txt" (The Declaration of Independence) and "document2.txt" (The Gettysburg Address).

★ **On-line students should submit the completed program (named "`lab08a.py`") for grading via the Coding Rooms system.**

B. Write a program using Dictionaries

Given two files named exactly `“data1.txt”` and `“data2.txt”` (no error checking needed) of names and scores print out the combined scores in alphabetical order by name. Note that the files will be formatted the same, but some names will be in both files and some names will only be in one file. For names in both files the scores need to be summed. The file format will be one header line followed by lines that have a name (string) and a number (int) separated by some unknown number of spaces. For output use this format string `"{:10s} {:<10d}"`

Requirements:

(1) You must use a dictionary.
(2) You must use at least two functions that have a dictionary as an argument, e.g. read a file, print results.

For example, if `data1.txt` contains

```
Name    Score
Joe     20
Mary    70
Rich    50
Jose    90
```

and `data2.txt` contains

```
Name         Score
Sarah        80
Ming         20
Joe          65
Rich         30
```

the output will be:

```
Name       Total
Joe        85
Jose       90
Mary       70
Ming       20
Rich       80
Sarah      80
```

★On-line students should submit the completed program (named “`lab08b.py`”) for grading via the Coding Rooms system.