Lab Week 10 Grading Rubric and Instructions

This lab is assigned for Week 10 of COM S 127: Introduction to Programming.

This lab is due by the end of the lab period seven (7) days after the one it is assigned in. See the syllabus for details.

Lab Objective

The purpose of this lab is to give students practice with using dictionaries and a try/ except statement when writing to a file.

Instructions/ Deliverables

NOTE: These tasks can be completed in any order you like. See the **Grading Items** section below for the point distribution.

CITATION: Many of the exercises found here could possibly be seen as adaptations of exercises found in the online textbook "How to Think Like a Computer Scientist: Interactive Edition" By Jeffrey Elkner, Peter Wentworth, Allen B. Downey, Chris Meyers, and Dario Mitchell.

- Available: https://runestone.academy/ns/books/published/thinkcspy/index.html?mode=browsing
- Accessed: 3-24-2023
- The abbreviation 'thinkcspy' and the chapter/ section number will be used to indicate where similar exercises can be found. This citation will be placed next to the exercise title.
    - ex: [thinkcspy 2.13] indicates a similar exercise can be found in chapter 2, section 13.

Reading:

- Read Runestone chapters 12 and 13, and show the TA the notes you took in your Engineering Notebook for each chapter once you are done.
    - NOTE: You do not need to complete any of the exercises at the end of the chapter. However, it would be helpful to you in the long term if you were to do so.

`wordSwap.py`

- Write a script that does the following:
    - Take in a complete (multi-word) sentence from the user as input.
    - Use the `.split()` method to break the sentence into individual words in a list.
    - Use a dictionary to hold the individual (unique) words as keys, and randomly assign a different word from the sentence as a value for that key.
    - Print out the dictionary containing the original sentence words and the words they will be swapped with.
    - Print out a new sentence, one word at a time, with each word delineated by spaces, such that the original word in the sentence (the key in the dictionary) is replaced with the randomly chosen word from the sentence (the value for the key).
        - Meaning - If the sentence contains the word 'cat', this word will be replaced by the same same randomly chosen replacement word every time the word 'cat' appears in the original sentence.
            - Ex: If the sentence "The cat cat cat" generates the dictionary `{'cat': 'The', 'The': 'cat'}`, we would print the sentence "cat The The The".
- Use a `main()` function in the way demonstrated in class.
    - Ex: `if __name__ == "__main__":` etc.
- Other than using a `main()` function, you can code your answer any way you like.
- See the **Example Output** section below for examples of what this script should do.
- Save your code, including your name, code creation date, lab number, and brief description of what your code

does, to a file called `wordSwap.py`.

`bookAnalysis.py` [thinkcspy 12.7]

- Obtain a 'Plain Text (.txt)' book of your choice from https://www.gutenberg.org
  - Be sure there are no Unicode characters in your file. You may have to click the "`more files`" link (it is very small and hard to see) next to the 'folder' icon on the page of the book you want to use. The filename will likely just be a number, like `71915-0.txt`, for example.
    - While the "Plain Text UTF-8" link will *probably* be just plain ascii text, there may be Unicode characters in the file that could be troublesome in that they might not work with the code below.
  - Once you download the `.txt` file you want to use, make sure to manually delete any information from https://www.gutenberg.org itself which is not a part of the book/ story.
    - Check both the start and end of the file.
    - The purpose of this is so that you only get an analysis of the book itself - not of any kind of website-related information.
- Consider the following code taken from then answer of exercise #3 in thinkcspy, reproduced in the code block below (https://runestone.academy/ns/books/published/thinkcspy/Dictionaries/Exercises.html?mode=browsing):
  - **NOTE**: This code has been slightly modified to ensure that `count.keys()` is a sortable list. The final `print()` statement, present in the code at the link above, has been removed.
  - **NOTE**: You *may* have to modify the code further to make it work with a contemporary version of Python.

```
f = open('alice.txt', 'r')

count = {}

for line in f:
    for word in line.split():

        # remove punctuation
        word = word.replace('_', '').replace('"', '').replace(',', '').replace('.', '')
        word = word.replace('-', '').replace('?', '').replace('!', '').replace("'", "")
        word = word.replace('(', '').replace(')', '').replace(':', '').replace('[', '')
        word = word.replace(']', '').replace(';', '')

        # ignore case
        word = word.lower()

        # ignore numbers
        if word.isalpha():
            if word in count:
                count[word] = count[word] + 1
            else:
                count[word] = 1

keys = list(count.keys())
keys.sort()

# save the word count analysis to a file
out = open('alice_words.txt', 'w')

for word in keys:
    out.write(word + " " + str(count[word]))
    out.write('\n')
```

- The code above analyzes a Plain Text book called 'alice.txt,' and counts all the different words contained in

the book. It then writes the total, sorted, list of words and the number of times they appear in a `.txt` file.

- Modify the code in the following ways:
  - Change the code so that it uses `with` statements for file reading/ writing instead of `f = open( ... )` and `out = open( ... )`.
  - Place the code that removes the punctuation and counts the words in a parameterized function called `analyzeBook()`.
    - This function takes a file name string as input, counts the words in the `.txt` file and puts this information into a dictionary called `count`, and then returns the `count` dictionary as output.
      - Meaning - this function should be able to count the words of *any* book - not just 'alice.txt'.
  - Place the code that sorts the dictionary keys and outputs the word counts to a file in another function called `outputAnalysis()`.
    - This function should take the previously created dictionary, as well as the title the book (sans any `.txt` extension), as input.
    - The function should then sort the keys in the input dictionary.
    - Finally, the function should output the words in the dictionary, and their counts, in sorted order, into a new text file.
      - The new text file should be named `<title>_analysis.txt`, where `<title>` is the title of the book passed to the function (sans any `.txt` extension).
    - **NOTE**: You *must* use a try/ except statement in this function when attempting to write the book analysis to the text file.
      - If the file writing/ output fails, the `outputAnalysis()` function should print an error message and return `False`.
      - If the file writing/ output succeeds, the `outputAnalysis()` function should just return `True`.
  - Finally, print out whether the output succeeded or not.
- Use a `main()` function in the way demonstrated in class.
  - Ex: `if __name__ == "__main__":` etc.
- See the **Example Script** section below for an example outline of what this should look like.
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `bookAnalysis.py`.
- **HINT**: Try developing your solution using a small/ trivial `.txt` input file before trying it with your book. In some cases, a book in `.txt` format might have a non-ascii character hidden somewhere in the text which makes it seem like your code is not working properly. If this is the case, either try a different book, or just explain what happened to the TA and show it working with your small/ trivial `.txt` input file.

Optional Readings

NOTE: These readings are not required. However, they may provide a bit of interest/ insight into the broader world of Computer Science. Please complete the rest of your lab tasks before doing these readings. You do not need to take notes on these in your Engineering Notebook.

Protect Yourself from AI Voice-Cloning Scams: Stay One Step Ahead - by: Reem Alattas

- Available: https://www.linkedin.com/pulse/protect-yourself-from-ai-voice-cloning-scams-stay-one-%D8%AF-%D8%B1%D9%8A%D9%85-%D8%A7%D9%84%D8%B9%D8%B7%D8%A7%D8%B3

Securing a Computer Science Internship in Your First Year - by: Indeed Editorial Team

- Available: https://www.indeed.com/career-advice/finding-a-job/first-year-student-computer-science-internships

The Ultimate Guide to Building a Coding Portfolio - by: Christina Payne and Marisa Upson

- Available: https://www.bestcolleges.com/bootcamps/guides/how-to-build-coding-portfolio/

The Horrible World Of Video Game Crunch - by: Jason Schreier

- Available: https://kotaku.com/crunch-time-why-game-developers-work-such-insane-hours-1704744577

Files Provided

None

Example Script

bookAnalysis.py

```
# Matthew Holman               3-18-2023
# Lab Week 10 - An example script layout

def analyzeBook(title):
    pass

def outputAnalysis(counts, title):
    pass

def main():
    countDict = analyzeBook("book.txt")
    succeeded = outputAnalysis(countDict, "book")
    print("Did the output succeed?:", succeeded)

if __name__ == "__main__":
    main()
```

Example Output

Running: `python .\wordSwap.py`

```
Enter a Sentence: The cute red cat jumped over the fence!
{'The': 'cute', 'cute': 'jumped', 'red': 'cat', 'cat': 'over', 'jumped': 'the', 'over': 'red',
cute jumped cat over the red The fence!

Enter a Sentence: asdf qwer asdf zxcv asdf zxcv zxcv
{'asdf': 'zxcv', 'qwer': 'asdf', 'zxcv': 'qwer'}
zxcv asdf zxcv qwer zxcv qwer qwer

Enter a Sentence: 1 2 3 1 1 1 2 3 2 3 4
{'1': '3', '2': '2', '3': '4', '4': '1'}
3 2 4 3 3 3 2 4 2 4 1
```

Grading Items

- (**Attendance**) Did the student attend the lab meeting, or make arrangements to attend virtually via WebEx?: _____ / 1
- (**Reading**) Has the student read chapter 12 and 13 of the Runestone textbook and shown their notes in their Engineering Notebook to the TA?: _____ / 2
- (**`wordSwap.py`**) Has the student completed the task above, and saved their work to a file called `wordSwap.py`?: _____ / 3
- (**`bookAnalysis.py`**) Has the student completed the task above, and saved their work to a file called `bookAnalysis.py`?: _____ / 4

TOTAL _____ / 10