Lab #8 Grading Rubric and Instructions

This is the eighth lab for the 'Introduction to Computer Programming' course.

Please see the syllabus for information about when the work in this lab is due.

Lab Objective

The purpose of this lab is to give you practice with using files, 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.

CITATION: Some of the exercises found here are completely original to the instructor.

- The abbreviation 'MH' will be used to indicate these exercises. This citation will be placed next to the exercise title.
    - ex: [MH]

Lecture Notes/ Supplemental Readings:

- 'Check off' your notes in your Engineering Notebook for the following material with the TA/ Instructor. These notes should already be done before the start of the lab period.
    - Runestone chapters 11 and 12
        - **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.
        - **NOTE**: You only need to take notes on the Runestone readings if you did *not* take lecture notes in class. The contents of these chapters are what was covered in the lectures.
    - `The_Runtime_Stack.pptx`
        - **Available**: `'Files' -> 'Supplimental_Slides_and_Files'` folder on Canvas

`removeNonASCII.py`: [MH]

- Write a script that does the following:
    - Use a `main()` function in the way demonstrated in class.
        - Ex: `if __name__ == "__main__":` etc.
    - In `main()`, take input as a string for a `.txt` filename.

        - Use the included `testFile.txt` file to help you test this script. It should be placed in the same folder you write your `removeNonASCII.py` script.
            - Notice the non-ASCII characters between each word in `testFile.txt`. When you successfully complete this task, you will be able to produce a file that does not include these characters.
            - **NOTE**: Do not worry if you print out the contents of `testFile.txt` to the terminal, but the output does not exactly match the contents of the file as they appear in VS Code - it should still work.
    - Make a new function wherein you pass in your filename string as a parameter, and then, in that new function, open that file and import its contents with the `.read()` method.
    - Return the output of `.read()` to a variable, and then return the contents of that variable back to `main()` where you will assign it to a new variable in `main()`.
    - Pass the contents of your newly read file into a new function called `removeNonASCII()`, which takes a string as its parameter.
    - Inside `removeNonASCII()`, create a new empty string variable called `clean`.
    - Iterate through the parameter string, one character at a time, and append that character to your new `clean` string only if the character under consideration is an ASCII character.
        - **HINT**: Investigate how the `ord()` function works (https://docs.python.org/3/library/functions.html#ord), and the decimal range of ASCII characters (https://www.asciitable.com/) in order to do this.
    - Once you have removed all non-ASCII characters from the parameter string and placed them in your `clean` string, return the `clean` string and assign that output to a variable back in `main()`.
    - In `main()`, pass your 'cleaned up' file content and your original file name to a new function which will write this content to a new file called `<Your File Name>_clean.txt`.
        - Ex: If your filename is `testFile.txt`, the new filename would be `testFile_clean.txt`.
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `removeNonASCII.py`.

`bookAnalysis.py`: [thinkcspy 12.7]

- Obtain a 'Plain Text (`.txt`)' book of your choice from https://www.gutenberg.org
    - 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.
    - Be sure there are no Unicode characters in your file.
        - 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.
        - You can use your `removeNonASCII.py` script (created above) to remove any non-ASCII characters from your `.txt` file.
    - Once you download and 'clean' 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.
    - **NOTE**: You *may* have to modify the code further to remove any missing punctuation marks or special characters.
        - Ex: `{`, `}`, `+`, `*`, etc.

```
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. The code then writes out the sorted list of words, as well as the number of times they appear in `alice.txt`, into a new `.txt` file.
- Modify the code in the following ways:
    - Use a `main()` function in the way demonstrated in class.
        - Ex: `if __name__ == "__main__":` etc.
    - In `main()`, take input as a string for a `.txt` filename.
    - Change the code found above so that it uses `with` statements for file reading/ writing instead of `f = open( ... )` and `out = open( ... )`.
    - Consider the code that goes through each line of the file and removes the punctuation and counts the words. Please this code in a parameterized function called `analyzeBook()`.
        - This function takes a file name string as input, counts the words in the `.txt` file, and then puts this information into a dictionary called `count`. Finally, `analyzeBook()` then returns the `count` dictionary as output, which should be assigned to a variable in the function that calls `analyzeBook()`.

        - Meaning - this function should be able to count the words of *any* book - not just `alice.txt`.
    - Consider the code that sorts the dictionary keys and outputs the word counts to a file. Please this code in another function called `outputAnalysis()`.
        - This function should take the previously created `count` dictionary, as well as the filename string, as input.
        - The function should sort the keys in the `count` 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).
                - Ex: If your filename is `alice.txt`, the new filename would be `alice_analysis.txt`.
- See the **Example Script** section below for an example outline of what this entire script should look like.
- **HINT**: Try developing your solution using a small/ trivial `.txt` input file before trying it with your book. If you successfully completed the `removeNonASCII.py` task, you should have a file called `testFile_clean.txt` that can serve this purpose.
- 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`.

`filesExercise.py`: [MH]

- You have been provided with a file called `filesProblem.pptx` which will outline what your code needs to calculate.
- You have also been provided with files called `scores.txt` and `students.txt`, which will be used in the calculation. Another file, called `grades_example.txt` will demonstrate what your output should look like when you save it to a file called `grades.txt`
- Follow the instructions in `filesProblem.pptx` and make a Python script which makes all necessary calculations to produce the `grades.txt` file.
- 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.
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `filesExercise.py`.

`wordSwap.py`: [MH]

- 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.'
        - This 'value' word can theoretically be the same word as the 'key' word.
    - 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`.

Attendance:

- If you have completed all of your tasks for the lab, you may work on any of the 'Additional Resources for Study' found in the Canvas announcement of the same name.
    - **NOTE**: If you leave early, you will not receive the 'attendance points' for the lab.

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

`testFile.txt`

`filesProblem.pptx`

`scores.txt`

`students.txt`

`grades_example.txt`

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():
    title = input("Please Enter Your Title:")
    countDict = analyzeBook(title)
    outputAnalysis(countDict, title)

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

Example Output

Running: `python wordSwap.py`

```
Enter a Sentence: That cute cute cat!
{'That': 'cute', 'cute': 'cat!', 'cat!': 'That'}
cute cat! cat! That

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

- (**Lecture Notes/ Supplemental Readings**) Has the student taken notes on the listed material, and shown their notes in their Engineering Notebook to the TA/ Instructor?: _____ / 20
- (**removeNonASCII.py**) Has the student completed the task above, and saved their work to a file called `removeNonASCII.py`?: _____ / 10
- (**bookAnalysis.py**) Has the student completed the task above, and saved their work to a file called `bookAnalysis.py`?: _____ / 20
- (**filesExercise.py**) Has the student completed the task above, and saved their work to a file called `filesExercise.py`?: _____ / 30
- (**wordSwap.py**) Has the student completed the task above, and saved their work to a file called `wordSwap.py`?: _____ / 10
- (**Attendance**) Did the student attend the full lab meeting in person?: _____ / 10

TOTAL _____ / 100