Engineering 13300
Python 4: Python File I/O

This assignment has three sections.

Team Hand Tracking - The first section has Team Exercises that are intended to be done with your team without actually using Python. Work the problems together and answer the questions and present your answer to one of the instructional team. After you find the correct answers, you can move to the second section. There are no files to turn in but you must complete this section before advancing to the next tasks.

Team Tasks - The second section has five team tasks (Tasks 1-5) that you may work together and develop code together as long as you document who contributed.

Individual Tasks - The third section has two task (Tasks 6-7) that are individual tasks. You can work together to conceptually help each other but each person should work on their own code.

Team Hand Tracking

What output will appear in the Python Command Window when the following code is executed?

```
matr = [[0,1,3,5,7,9,2,4],
        [3,4,6,8,9,0,1,2]]
n = 8
k = 0
print('n b k new_term')
while n > 3:
    b = 4
    for i_c in range (b,n,2):
        new_term = n*matr[k][i_c]*n
        print(f'{n} {b} {k} {new_term}')
    n = n - 2
    if n <= 6:
        k=1
```

Team Tasks

Recall the guidelines for team activities:

1. You should work as a team; **all** team members will be held responsible for all material. You may work together and contribute to one program and submit similar codes as long as the contributors to the development of the solution are documented.
2. Each student is responsible for submitting their own assignment.

Assignment Background:

Computers can process large amounts of information which they access from files stored on external devices or user input from keyboard. Large databases of information often contain thousands or millions of records. A computer can access the file on a device and then search through it to locate specific information or alter part of the file with new information. This process requires accessing external hardware which is controlled by the operating system. Therefore, Python has functions that can request that the operating system access information from a device and store information to that device. The following tasks explore various methods for accessing information stored in files or from keyboard, processing it in some way, then outputting it to an external device (computer screen, hard drive etc).

Task 1 (of 7) [Team]

Learning Objectives: Read in input data from a user at the keyboard in Python; Practice writing data to an output file.

Part A:

Please answer the following questions:

1. How would you open a file name `test.txt` to read only? To write only? To read and write?
2. What is the general command structure for writing to a file?
3. How do you output a string to a file? An integer? A real number with three decimal points?

Save your answers in a PDF named: `Py4_Team_`*`teamnumber`*`.pdf`

Part B:

Your task is to write a program that prompts the user to input their last name, first name, an integer representing their age in years, and an integer representing the number of days elapsed since their most recent birthday. Each of these inputs will be followed by a return (i.e. should be done in **separate** lines). Given this data, your program should calculate the user's total present age in years since their last birthday then use a user-defined function to calculate and return the user's age in whole seconds. Your program should then output the user's name (first then last name), a real number representing their total present age in years, and the number of whole seconds that have elapsed in this time to a file named `Py4_Task1_output.txt`. Assume that there are ***365.242199*** days in a year.

The input of the program should follow this example as closely as possible:

```
Enter your last name:
Armstrong
Enter your first name:
Neil
Enter your age in whole years:
90
Enter the days elapsed since your last birthday:
5
```

The output file should have the following three lines:

```
Neil Armstrong
You are 90.0136895463 years old.
You are 2838671713 seconds old.
```

First, create a flowchart of the algorithm you will use for this program. Save your flowchart as a separate page in the previously created PDF file. Then, write a Python program that implements your algorithm. Name your main program: `Py4_Task1_`*`teamnumber`*`.py`

Task 1 Files:

1. `Py4_Team_`*`teamnumber`*`.pdf`
2. `Py4_Task1_`*`teamnumber`*`.py`
3. `Py4_Task1_output.txt`

Task 2 (of 7) [Team]

Learning Objectives: Practice using while loops and File I/O in Python

Create a Python script that reads in a file containing names and grades. The number of lines in the file is not specified. The input should come from a text file named `Py4_Task2_input.txt` that you construct.

Example of `Py4_Task2_input.txt`:

```
Alex 85
Brian 68
Casey 89
David 72
```

In your Python script, create a function called `Avg_Std` that will take an input of list of grades, and calculate the average and the standard deviation of the grades. Recall from your statistics course that the average $\bar{x}$ is calculated using,

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Where N is the number of data points (number of students in this example) and x_i represents value of each data point (grades of each student). The formula for standard deviation σ, is given by,

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

Create a second function called `Hi_Lo` that will take an input of list of grades and return the highest and lowest grades.

Output the results to a file called `Py4_Task2_output.txt`. Given the sample input file above, the output file should print the results in the following manner:

```
Exam 1:
Average = 78.5
Standard Deviation = 8.73
High Score = 89.00
Low Score = 68.00
```

Name your main program: `Py4_Task2_`*`teamnumber`*`.py`

Task 2 Files:

1. `Py4_Task2_`*`teamnumber`*`.py`
2. `Py4_Task2_input.txt`
3. `Py4_Task2_output.txt`

Task 3 (of 7) [Team]

Learning Objectives: Practice File I/O and looping in Python

Part A:
Please answer the following questions:

1. How would you read in a string?
2. How would you read in an integer and a float? How is this different from a string?
3. How could you read in a large list of data very quickly?

Include these in your previously created PDF: `Py4_Team_`*`teamnumber`*`.pdf`

Part B:
In this task, you will create a Python script in which you will practice reading files in Python and writing them to a new output file.

Construct a text file called `Py4_Task3_input.txt` that has the following lines:

```
4
Independence Day 04 July 2021
Halloween 31 October 2021
Thanksgiving 22 November 2021
Christmas 25 December 2021
```

The first line of the file represents the number of lines in the data file.

Write a loop that reads in each additional line (one line at a time) and stores this line first as a string, then converts it to a list, before moving on to reading the next line. Hint: the string for the second line should be `Independence Day 04 July 2021` whereas the list for the second line should be `['Independence', 'Day', '04', 'July', '2021']`.

Create a second looping structure that outputs each line as a string, then as a list, before moving on to the next line. Name your output file `Py4_Task3_output.txt`.

Name your main program `Py4_Task3_`*`teamnumber`*`.py`

Task 3 Files:

1. `Py4_Team_`*`teamnumber`*`.pdf`
2. `Py4_Task3_`*`teamnumber`*`.py`
3. `Py4_Task3_input.txt`
4. `Py4_Task3_output.txt`

Task 4 (of 7) [Team]

Learning Objectives: Practice using loops and File I/O in Python

Create a Python script that will read data from an input file `Py4_Task4_input.txt` containing dates consisting of a month and a year.

Example of `Py4_Task4_input.txt`:

```
January 2013
December 2023
October 2045
March 1995
August 2020
April 2010
September 1970
```

Create a function called `Sort_Dates` that will use a nested conditional structure in order to sort the dates into three categories: future, present, and past. Use the current month and year as the present date. Dates do not need to be sorted within categories. Note: unlike some other programming languages, Python does not automatically recognize names of months as being in any particular order. Therefore, you may wish to define some sort of data structure that allows you to "look up" the position of a given month in the standard order of months.

In your main program, output the dates sorted into the three categories as well as the number of dates in each category to a file called `Py4_Task4_output.txt`

```
Future months: 2
December 2023
October 2045

Present months: 1
August 2020

Past months: 4
January 2013
March 1995
April 2010
September 1970
```

Name your main program: `Py4_Task4_`*`teamnumber`*`.py`

Task 4 Files:

1. `Py4_Task4_`*`teamnumber`*`.py`
2. `Py4_Task4_input.txt`
3. `Py4_Task4_output.txt`

Task 5 (of 7) [Team]

Learning Objectives: Understand how to read different data types from a file

Background:
In many biological systems it is difficult to simply measure the concentration of any one substance by itself. To accomplish such measurements methods of tagging with enzymes that produce color when they catalyze a reaction have been developed. Beer's Law is used to relate the absorbance of a sample to the concentration of the products produced by these enzymes. So, almost any protein concentration can be found by simply taking absorbance readings if it has first been tagged with an enzyme.

Beer's Law simply states that the rate of photon absorbance, A, is directly proportional to the concentration of the absorbers, c. The proportionality constant is the product of the path length the light must travel, b, and the molar extinction coefficient of the substance, ε.

Beer's Law:

$$A = \varepsilon c b$$

Write a Python script that will open a file containing the name of the substance that was tagged, the path length (b), the molar extinction coefficient of the absorbers (ε), and a list of absorbencies (A), and then find the concentration for each absorbency value. The input file containing the raw data is named `Py4_Task5_input.txt` and is located on Brightspace under the Python 4 module.

Within your main program, create a function named `Absorb_Calc` that takes relevant inputs to calculate the concentration for each value of absorbency. Make sure your program can calculate concentrations for any number of absorbency values given. Your program should then output the name of the substance and a list of the concentrations associated with that substance as follows to the screen. (You do not need to worry about units for this specific task, and you do not need to output results to a text file.).

Example:

```
Bovine Serum Albumin Concentrations
.0000356
.0000474
.0000267
```

Name your main program: `Py4_Task5_`*`teamnumber`*`.py`

Task 5 Files:

1. `Py4_Task5_`*`teamnumber`*`.py`

Individual Tasks

Guidelines for Tasks 6–7:
Tasks 6–7 are individual tasks. You may seek help from classmates, the instructional team or others but the work you submit should be your own. If you collaborate with others and use information developed together or by someone else, ALWAYS document and reference that material.

Task 6 (of 7) [Individual]

Learning Objectives: Utilize file I/O functions in Python; Construct files using formatted output using Python; Read in input data from a user at the keyboard in Python.

Background:

For many reasons, engineers write formal reports in the real world. In a documentation of experiments or designs, if each line matches a number, like line numbering in Python programs, reviewing and discussing these reports with co-workers or clients can be easier.

Task:

Develop a flow diagram and write a program in Python that will ask the user for an input file name, and then for an output file name. For both files, user must include a proper extension such as ".txt" for text files. The input file contains a guideline with an unknown number of lines, each line representing a step that can be performed for all workplaces to take to reduce risk of exposure to coronavirus. Your job is to create a new file with the given output file name, and then write the file such that each line of the input file is written into it with step numbers appended to the beginning of the line. Note that each line may contain several sentences.

Example:

Input:

```
Enter the filename of the input file: Py4_Task6_input.txt
Enter the filename of the output file:
Py4_Task6_output.txt
```

Contents of `Py4_Task6_input.txt`:

```
Encourage workers to stay home if sick.
Encourage respiratory etiquette, including covering coughs and
sneezes.
Provide a place to wash hands or alcohol-based hand rubs containing at
least 60% alcohol.
Limit worksite access to only essential workers, if possible.
Establish flexible worksites (e.g., telecommuting) and flexible work
hours (e.g., staggered shifts), if feasible.
Discourage workers from using other workers' phones, desks, or other
work tools and equipment.
Regularly clean and disinfect surfaces, equipment, and other elements
of the work environment.
Use Environmental Protection Agency (EPA)-approved cleaning chemicals
with label claims against the coronavirus.
Follow the manufacturer's instructions for use of all cleaning and
disinfection products.
Encourage workers to report any safety and health concerns.
```

Contents of `Py4_Task6_output.txt`:

```
Step 1: Encourage workers to stay home if sick.
Step 2: Encourage respiratory etiquette, including covering coughs and
sneezes.
Step 3: Provide a place to wash hands or alcohol-based hand rubs
containing at least 60% alcohol.
Step 4: Limit worksite access to only essential workers, if possible.
Step 5: Establish flexible worksites (e.g., telecommuting) and
flexible work hours (e.g., staggered shifts), if feasible.
Step 6: Discourage workers from using other workers' phones, desks, or
other work tools and equipment.
Step 7: Regularly clean and disinfect surfaces, equipment, and other
elements of the work environment.
Step 8: Use Environmental Protection Agency (EPA)-approved cleaning
chemicals with label claims against the coronavirus.
Step 9: Follow the manufacturer's instructions for use of all cleaning
and disinfection products.
Step 10: Encourage workers to report any safety and health concerns.
```

Task 6 Files:

1. `Py4_Ind_`*`username`*`.pdf`
2. `Py4_Task6_`*`username`*`.py`
3. `Py4_Task6_output.txt`

Task 7 (of 7) [Individual]

Learning Objectives: Utilize file I/O functions in Python.

Background:

Repeating the same word within a short chunk may lead to a less engaging read. To avoid overusing words while writing paper, calculating occurrences of the same word is a good way to check.

Task:

Write a flow chart and a python program that will ask the user for a search word and the name of text file. Assume that the input file is named `Py4_Task7_text.txt` and that it will contain a large amount of words, likely in the form of an excerpt from a text. The program is supposed to search the input file for the search word and compute the percentage of times the word occurs in the document, e.g. if there are 200 words and the word occurs 3 times, the percentage would be 1.50%. Note that this percentage should be truncated to the hundredth place.

It is important to note that capitalization does not matter for this search, e.g. "Butterfly" is equivalent to "butterfly". Also, words may be followed by characters including but not limited to commas, periods, and semi-colons, e.g. "butterfly," will count as "butterfly". The search word will always be a single word, which means occurrences of the search word should not be counted if they appear inside another word. For example, "butterfly" should not be counted for "butter" or "fly".

Hint:

The following functions may be useful: `lower()`, `isalnum()`, `isalpha()`

Example:

Input:

```
Enter the filename of the text file: Py4_Task7_text.txt
Enter the search word: GARDEN
```

Contents of `Py4_Task7_text.txt`

```
Ode to a butterfly
Thou spark of life that wavest wings of gold,
Thou songless wanderer mid the songful birds,
With Nature's secrets in thy tints unrolled
Through gorgeous cipher, past the reach of words,
Yet dear to every child
In glad pursuit beguiled,
Living his unspoiled days mid flowers and flocks and herds!
Thou winged blossom, liberated thing,
What secret tie binds thee to other flowers,
Still held within the garden's fostering?
Will they too soar with the completed hours,
Take flight, and be like thee
Irrevocably free,
Hovering at will o'er their parental bowers?
Or is thy lustre drawn from heavenly hues,
A sumptuous drifting fragment of the sky,
```

```
Caught when the sunset its last glance imbues
With sudden splendor, and the tree-tops high
Grasp that swift blazonry,
Then lend those tints to thee,
On thee to float a few short hours, and die?
Birds have their nests; they rear their eager young,
And flit on errands all the livelong day;
Each fieldmouse keeps the homestead whence it sprung;
But thou art Nature's freeman,—free to stray Unfettered through the
wood,
Seeking thine airy food,
The sweetness spiced on every blossomed spray.
The garden one wide banquet spreads for thee,
O daintiest reveller of the joyous earth!
One drop of honey gives satiety;
A second draught would drug thee past all mirth.
Thy feast no orgy shows;
Thy calm eyes never close,
Thou soberest sprite to which the sun gives birth.
And yet the soul of man upon thy wings
Forever soars in aspiration; thou
His emblem of the new career that springs
When death's arrest bids all his spirit bow.
He seeks his hope in thee
Of immortality.
Symbol of life, me with such faith endow!
By Thomas Wentworth Higginson. Edmund Clarence Stedman, ed. (1833–
1908). An American Anthology, 1787–1900. 1900.
```

Note: highlighting is used to clarify the example – there is no highlighting in the text file itself. The word "garden's" is a possessive and therefore does not count.

Output:

```
The search word occurred 0.33% of the time.
```

Task 7 Files:

1. Py4_Ind_*username*.pdf
2. Py4_Task7_*username*.py

Summary of Files to submit

1. Py4_Team_*teamnumber*.pdf
2. Py4_Task1_teamnumber.py
3. Py4_Task1_output.txt
4. Py4_Task2_*teamnumber*.py
5. Py4_Task2_input.txt
6. Py4_Task2_output.txt
7. Py4_Task3_*teamnumber*.py
8. Py4_Task3_input.txt
9. Py4_Task3_output.txt
10. Py4_Task4_*teamnumber*.py
11. Py4_Task4_input.txt
12. Py4_Task4_output.txt
13. Py4_Task5_*teamnumber*.py
14. Py4_Ind_*username*.pdf
15. Py4_Task6_*username*.py
16. Py4_Task6_output.txt
17. Py4_Task7_*username*.py