Engineering 14100
Python 4 ACT: Python File I/O

Recall the guidelines for activities:

1. You should work as a team; **all** team members will be held responsible for all material.
2. You should work on this Task using one computer for the entire team unless otherwise directed.
3. The time estimate given is approximately three times the amount required for an experienced user to complete the task. If you are not making progress, take action to get unstuck.
4. Do not write on the activity sheets and be sure to return them at the end of class.

Assignment Background:
Computers can process large amounts of information which they accesses 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 5)

Learning Objectives: Read in input data from a user at the keyboard in Python; Practice writing data to an output file.
Computer Operator: Person with the most pets

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_ACT_`*`login`*`.pdf`

Part B:
Write a Python 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 minutes. 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 minutes that have elapsed in this time to a file named `Py4_ACT_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:
    Purdue
    Enter your first name:
    John
    Enter your age in whole years:
    201
    Enter the days elapsed since your last birthday:
    77
```

The output file should have the following three lines:

```
    John Purdue
    You are 201.210819013 years old.
    You are 105826582 minutes old.
```

Name your main program: `Py4_ACT_Task1_`*`login`*`.py`

Task 1 Files:

1) `Py4_ACT_`*`login`*`.pdf`
2) `Py4_ACT_Task1_`*`login`*`.py`
3) `Py4_ACT_Task1_output.txt`

Task 2 (of 5)

Learning Objectives: Practice using while loops and File I/O in Python
Computer Operator: Person with the oldest sibling.

Create a Python script that reads in a file containing names and grades. The number of lines in the file is not specified.

Create a text file named `Py4_ACT_Task2_input.txt`

Example of `Py4_ACT_Task2_input.txt`:

```
Mark 65
Sally 78
John 90
Karen 46
```

In your Python script, create a function called `Avg_Std` that will calculate the average and the standard deviation of the grades. The formula for standard deviation is:

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

Create a second function called `Hi_Lo` that will return the highest and lowest grades.

Output the results to a file called `Py4_ACT_Task2_output.txt` that will follow this format:

```
Exam 1:
Average = XX.XX
Standard Deviation = XX.XX
High Score = XX.XX
Low Score = XX.XX
```

Name your main program: `Py4_ACT_Task2_`*`login`*`.py`

Task 3 Files:

1. `Py4_ACT_Task2_`*`login`*`.py`
2. `Py4_ACT_Task2_input.txt`
3. `Py4_ACT_Task2_output.txt`

Task 3 (of 5)

Learning Objectives: Practice File I/O and looping in Python
Computer Operator: Person who can name all of the schools in the Big Ten the fastest

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_ACT_login.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_ACT_Task3_input.txt` that has the following lines:

```
3
Halloween 31 October 2013
Thanksgiving 22 November 2013
Christmas 25 December 2013
```

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 them first as a string, then as a list, before moving on to reading the next line. Hint: string input would see the first line as `“Halloween 31 October 2013”` whereas a list input would see it as `[“Halloween”, 31, “October”, 2013].`

Create a second looping structure that outputs each line as a string, then a list, before moving on to the next line. Name your output file `Py4_ACT_Task3_output.txt.`

Name your main program `Py4_ACT_Task3_`*`login`*`.py`

Task 2 Files:
1. `Py4_ACT_`*`login`*`.pdf`
2. `Py4_ACT_Task3_`*`login`*`.py`
3. `Py4_ACT_Task3_input.txt`
4. `Py4_ACT_Task3_output.txt`

Task 4 (of 5)

Learning Objectives: Practice using loops and File I/O in Python
Computer Operator: Person with the oldest cell phone.

Create a Python script that will read data from an input file `Py4_ACT_Task4_input.txt` containing dates consisting of a month and a year.

Example of `Py4_ACT_Task4_input.txt`:

```
May 2009
December 2000
April 2018
May 2005
June 2025
October 2015
```

Create a function called `Sort_Dates` that will sort the dates into three categories: future, present, and past. Use the current month and year as the present date.

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_ACT_Task4_output.txt`

```
Future dates: 2
April 2018
June 2025

Present dates: 1
October 2015

Past dates: 3
May 2009
December 2000
May 2005
```

Name your main program: `Py4_ACT_Task4_login.py`

Task 4 Files:

1) `Py4_ACT_Task4_login.py`
2) `Py4_ACT_Task4_input.txt`
3) `Py4_ACT_Task4_output.txt`

Task 5 (of 5)

Learning Objectives: Understand how to read different data types from a file
Computer Operator: Person who lives in the highest floor number of Shreve.

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).

This input file containing the raw data is named `Python4_Task5_input.txt` and is located on Blackboard under the Activities Folder.

Within your main program, create a function named `Absorb_Calc` that will calculate the concentration for each of the absorbency. 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).

Example:

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

Name your main program: `Py4_ACT_Task5_`*`login`*`.py`

Task 5 Files:

1) `Py4_ACT_Task5_login.py`

Bonus Activity Submissions

Instructions: Complete and submit **ALL** Task materials associated with this Activity (see 'Submit Files' below). You are allowed to combine the work you and your team completed during the Activity with materials you individually (or as a team) complete outside of class. The Bonus Activity Submission will not be graded and returned to you like a typical assignment. Instead, it will be reviewed, and the bonus point awarded, for its completeness, i.e., for completing ALL the Tasks associated with the Activity. Submitting an incomplete Bonus Activity (something less than all of the Tasks) will be considered an act of **Academic Dishonesty** for which the penalty will be forfeiture of the opportunity to earn future Bonus Activity Submission points.

There are two options for completing the materials for the Bonus Activity Submission:

As an Individual: Combine the work you and your team completed in class with materials you have individually completed outside of class. When submitting an individual Bonus Activity Submission you will append your electric signature (i.e., your typed name) at the top of the file that represents your individual work. Your electronic signature indicates that this is your individual work and you have not collaborated with other individuals (other than the teaching team) to obtain the final materials being submitted – working with other individuals/groups (e.g., discussing ideas and concepts, helping find errors, talking about potential solutions to errors) is permissible up to the point where the work represents a coloration (i.e., working with another person or group to achieve an answer). Any work previously completed by your team should include each team member's electronic signature. The significance of an electronic signature by an individual for team work is stated below.

As a Team or Ad Hoc Group: Combine the work you and your team completed in class with materials your team (or ad hoc group) completed outside of class **(For the Bonus Activity Submission ONLY**: you are allowed to work with any other members of the class to complete the assignment). However, you should exercise care when appending your electronic signature to ensure you are in full compliance). When submitting a Bonus Activity that has been worked on as a team (or ad hoc group) each person will append his/her electronic signature (i.e., his/her typed name) at the top of the file that represents the collaborative work. The electronic signature of each individual implies he/she: was an active participant in the preparation of the materials; and has a general understanding of **ALL** the materials being submitted. Even for work submitted as a team, each individual who wishes to receive credit must submit the team's file (with all appropriate signatures) to their own individual assignment drop box.

Submit Files: Submit *all* files electronically via Blackboard to the appropriate box on time.

1. `Py4_ACT_`*`login`*`.pdf`
2. `Py4_ACT_Task1_`*`login`*`.py`
3. `Py4_ACT_Task1_output.txt`
4. `Py4_ACT_Task2_`*`login`*`.py`
5. `Py4_ACT_Task2_input.txt`
6. `Py4_ACT_Task2_output.txt`
7. `Py4_ACT_Task3_`*`login`*`.py`
8. `Py4_ACT_Task3_input.txt`
9. `Py4_ACT_Task3_output.txt`
10. `Py4_ACT_Task4_login.py`
11. `Py4_ACT_Task4_input.txt`
12. `Py4_ACT_Task4_output.txt`
13. `Py4_ACT_Task5_login.py`