# HW 8 · Python 4 · Team

## Introduction

**Assignment Goals**

The purpose of this assignment is to learn how to read and write files in Python.

**Successful Completion**

1. This assignment has five tasks and is worth 5 points. **Complete tasks 2 and 5 to earn 4 points. Complete two of the remaining tasks to earn 1 additional point.**
2. Read *Notes Before You Start*, on **Page 1**.
3. Read each task carefully. You are responsible for following all instructions within each task. The deliverables list within each task contains everything you are expected to submit.
4. When your work is complete, confirm your deliverables are submitted to Gradescope.
5. Late submissions will be accepted up to 24 hours after the due date. There is no late penalty for the first homework assignment.

## Notes Before You Start

**Guidelines for Team Tasks**

- You should work with your classmates but remember that you and your team members will be held responsible for all material.
- You may work collaboratively with your classmates and even submit solutions that are similar as long as all of the collaborators are clearly listed, and their contributions are properly referenced.
- Each team is responsible for submitting their assignment to Gradescope. Team tasks will only need to be submitted by one member of the team. That team member will need to assign all the other team members to the submission. Click here for help with submitting team assignments.
- You can resubmit to Gradescope as many times as you need. Only your final submission is graded.
- If you cannot copy-paste into Gradescope, then you may need to refresh or update your browser. Use a search engine to find a solution for your specific browser.

**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

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

**Submissions**

| | |
|---|---|
| Gradescope Assignments | HW 8 – PY 4 Team |
| Task Type | Team |
| Deliverables | `Py4_Team_teamnumber.pdf`<br>`Py4_Task1_teamnumber.py`<br>`Py4_Task1_output.txt` |

**Task Instructions**
**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:
Purdue

Enter your first name:
John

Enter your age in whole years:
218

Enter the days elapsed since your last birthday:
332
```

The output file should have the following three lines:

```
John Purdue
You are 218.90898587542455 years old.
You are 6908094666 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 #2 [Required]

**Learning Objectives**
Determine the output of a program through hand tracking; looping structures; nested loops

**Background**
When determining what a program does or checking for bugs it is often helpful to be able to determine by hand, or with debugging tools, what the output of a program is. In this task you will work through two pieces of code using different starting values to determine the output.

**Submissions**

| | |
|---|---|
| Gradescope Assignments | HW 8 – PY 4 Team |
| Task Type | Team |
| Deliverables | Py4_Team_teamnumber.pdf |

**Team Hand Tracking Exercises**
Submit the answers to the following exercises in your `Py4_Team_teamnumber.pdf` file.

A) What is the output of the following code when run after all the variables have been cleared with the command reset?

```
x=3
y=6
while i<10:
    for z in [x,y]:
        i+=z
    if x<3:
        x+=1
print(i)
```

B) What is the output of the following code when run after all the variables have been cleared with the command reset?

```
i=1
x=3
y=6
while i<10:
```

```
        for z in [x,y]:
            i+=z
        if x<3:
            x+=1
print(i)
```

C) The following commands were entered into the console window of Python. What appears on the screen after they are executed?

```
x=[3,6,-2,1]
y=0

for index in x:
    if index <= 3:
        y=y+index
print(f'y={y}')
```

D) The following commands were entered into the console window of Python. What appears on the screen after they are executed?

```
x=[3,6,-2,1]
y=0

for index in x:
    if index <= 3:
        y=y+index
    print(f'y={y}')
```

## Task #3

**Learning Objectives**
Practice File I/O and looping in Python.

**Submissions**

| | |
|---|---|
| Gradescope Assignments | HW 5 – PY 1 Team |
| Task Type | Team |
| Deliverables | `Py1_Task1_`*`teamnumber`*`.py` |

**Task Instructions**
**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`*. Be sure to add the flowchart associated with this task in the previously created pdf file.

## Task #4

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

**Submissions**

| | |
|---|---|
| Gradescope Assignments | HW 8 – PY 4 Team |
| Task Type | Team |
| Deliverables | `Py4_Team_`*`teamnumber`*`.pdf`<br>`Py4_Task4_`*`teamnumber`*`.py` |

**Task Instructions**
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 ($\varepsilon$), and an absorbency ($A$), and then find the concentration for each absorbency value. The input file containing the raw data is named `Py4_Task4_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 output:

```
The name of the substance is Glucose Oxidase
For 0.9863 absorbency value, the concentration is 0.0000508
For 0.6868 absorbency value, the concentration is 0.0000354
For 0.4462 absorbency value, the concentration is 0.0000230
```

Name your main program: `Py4_Task4_`*`teamnumber`*`.py` Be sure to add the flowchart associated with this task in the previously created pdf file.

**Hint**: Look up `strip()`, `split()` functions

## Task #5 [Required]

**Learning Objectives**
Plotting data and images with `matplotlib`

**Submissions**

| | |
|---|---|
| Gradescope Assignments | HW 8 – PY 4 Team |
| Task Type | Team |
| Deliverables | `Py4_Task5_`*`teamnumber`*`.py`<br>`Py4_task5_worldmap_`*`teamnumber`*`.png`<br>`Py4_task5_USAmap_`*`teamnumber`*`.png` |

**Task Instructions**
For this task, you will be plotting an image of a map projection of Earth and plotting atop it the locations of several thousand of the largest cities on the planet.

To start, import `Py4_Task5_map.png` into Python using the `matplotlib.image.imread` function. Note that this import converts the image into an *N* by *M* by *4* NumPy array, where *N* is the number of rows of pixels, *M* is the number of columns, and *4* represents the 4 RGBA elements of the image. Images are typically represented as RGB (in which case it would be an *N* by *M* by *3* array) or RGBA arrays, where the RGB values give a scaling from 0 to 1 of the respective amounts of red, green, and blue for that pixel, and A value represents the transparency (from 0 to 1, where 0 is fully transparent and 1 is fully opaque) of that pixel.

Using your script, determine what the RGBA value for the first pixel (i.e., the pixel on the upper left corner, corresponding to the $0^{th}$ row and $0^{th}$ column), and add a comment in your code listing that value.

Now, plot your image using the `matplotlib.pyplot.imshow` function. Note that if you plot it by using the basic command, your x and y-axes show the indices of each pixel. Since we want to plot data, represented by coordinates, onto this map, you should plot the image and use the `extent` option to make the x- and y-axes the longitudes and latitudes that this map spans. The map spans from -180 to 180 degrees in longitude and from -80 to 80 degrees in latitude.

The data you will be plotting atop this map are the locations of some of the largest 40000 cities in the world. Import `Py4_Task5_worldcities.csv`. Note that the first row are column headings and that, since this is a csv, the delimiter is a comma. Using a scatterplot, plot the locations of these cities atop the map image, color each point by the base-10 logarithm of the city's population, and use the `coolwarm` colormap for the coloring. The matplotlib documentation for the scatter plot function is available at this hyperlink. Make sure to use an appropriate size for the scatter points (i.e., ensure that they are large enough to be visible, but not so big that they all significantly overlap).

First, create one plot of the entire world. Use appropriate x- and y-axis labels. Pick an appropriate title and **make sure that your team number is listed in the title**. Save this plot as `Py4_Task5_worldmap_`*`teamnumber`*`.png` with a resolution of 200dpi. Set `bbox_inches='tight'` in your `savefig` command so that the plot saves without extra white space.

Finally, create a plot made up of two subplots arranged vertically. The top subplot should be the same map as previously but with the x- and y-axes limited to show the contiguous USA. The bottom subplot should have the axes limited to show Indiana. Hint: look up the coordinate spans of these regions on the Google maps. Note that the coordinate ranges do not need to be exact but should have the majority of the map dominated by these regions.

Use appropriate x- and y-axis labels. Pick an appropriate title and **make sure that your team number is listed in the title**. Use the `matplotlib.pyplot.tight_layout()` function right before your `savefig` statement so that the plot labels do not overlap. Save this plot as `Py4_Task5_USAmap_`*`teamnumber`*`.png` with a resolution of 200dpi. Set `bbox_inches='tight'` in your `savefig` command so that the plot saves without extra white space.