CSE 8A Programming Assignment 2

Due Date: Tue. Oct 20th, 11:59 PM (PDT)

Learning goals:

- Practice writing functions
- Learn to use if-elif-else statements
- Use comparisons between numbers
- Manipulate lists using indexing

Logistics:

All information for the following can be found on our course syllabus.

- **Pair programming** - This programming assignment can be done individually or with a partner. Working with a partner is highly encouraged to get the most out of this class. Make sure to read the guide on pair programming.
- **Academic integrity** - Please adhere to all academic integrity guidelines on the syllabus.

Submission:

- You will submit the following files to Gradescope:
    - **pa2.py** - Contains all your code
    - **Pa2-writeup.pdf** - Contains all written portions of the assignment
    - **Pa2-video.mp4** - Your recorded video
- You can find the template of the write-up here
- You are also required to complete the weekly reflection
- Instructions on how to submit can be found below at submission instructions.

Part 1: Implementation (6 Points)

Save all of the following code in a python file named (pa2.py).

In this programming assignment, you will be implementing a function to convert a numerical grade between 0 and 100 into a letter grade. Possible letter grades are “F”, “D”, “C”, “B”, and “A”. An if-elif-else statement will be very helpful in writing this function.

Part 1.1: Implement a grade calculator

First implement the following function. The name of your function should match exactly as shown, including cases (all lowercase). Our autograder on gradescope will not be able to test your function if it is not named properly.

Function Name: grade_calculator
Parameter: number_grade - A number that represents the grade in integer or float form, between 0 and 100.
Return: A string that represents the number grade as a letter grade, which must be “F”, “D”, “C”, “B”, or “A”. In case the argument provided is not valid (for example, -1 or 1000), the string **“**`Input Invalid!`**”** should be returned.
Description: Given a numerical grade, return the equivalent letter grade.
Example: Argument: number_grade = 97.4 Returns: “A”

Use the following grade schema, where *x* represents the numerical grade:

0 <= *x* < 60 : “F”
60 <= *x* < 70 : “D”
70 <= *x* < 80 : “C”
80 <= *x* < 90 : “B”
90 <= *x* <= 100 : “A”

Think about how you will catch cases where the input is invalid. The input will always be an integer or float, but it may be outside of the bounds of 0 and 100.

You may want to test your function using a similar technique to Part 1.2 of PA 1. Make sure you test edge cases to ensure behavior is as expected. Some edge cases that may present problems are 0, -1, and 79.99. If edge cases present unexpected output, ask yourself why and make sure you adjust your code accordingly.

Part 1.2: Manipulating lists

A toy car company is performing tests of their latest wind up cars. For each test, they wind up the car and record its location on a track before release, when its coil stops unwinding, and when it finally comes to a stop. The wind up cars never go backwards, so their location only increases over time. They store this data in a list of three integers, which are of course always strictly increasing. Strictly increasing order means each number will be greater than the previous number, with equality not permitted. In other words, given list i, i[0] < i[1] < i[2]. Some examples of such a list are [0, 6.2, 23] and [-1, 0, 10]. However, while entering this data into the company database, an employee makes a huge mistake. Some of the items in the list have

been entered in reverse order! For example, if the proper data was [0, 1, 2], it would have been entered in the database as [2, 1, 0]. Your job is to write a function that will fix one of these 3 item lists if it is in reverse order, and leave it as is if it is already in the correct order.

Function Name: fix_list
Parameter: data - A list with 3 elements (that are either floats or ints) that is either in strictly decreasing or strictly increasing order.
Return: A list with the same 3 elements as the parameter that is in strictly increasing order.
Description: Returns a strictly increasing 3 element list given a 3 element list that is either in strictly increasing or strictly decreasing order.
Examples: Parameter: [-1, 0, 2] Returns: [-1, 0, 2] Parameter: [6.4, 3, 2] Returns: [2, 3, 6.4]

You may notice that the middle number is irrelevant to your solution to this problem. In other words, you should be seeking to manipulate the first and last numbers, at index 0 and 2 (or 0 and -1, since they are equivalent for a list of length 3).

Python has some built-in functions to sort lists. Please refrain from using them and use only list indexing, concatenation, and slicing.

Star Points (optional):

*Save all of the following code in **pa2.py***
Read about star points on our course syllabus. To obtain a star point for this PA, you will be writing the following two functions in pa2.py.

Part 1.3: Implement a function that tells the number of days in a month. (Only for star points; Not required to get full credits on this PA)

Function Name: days_in_month
Parameter: month - A string that is the name of a month. Use the convention of a capital first letter and all other letter lowercase. For example, “June”, “March”, “January”

Return: An integer that represents the number of days in the given month (non leap year).
Description: Given the name of a month, return the number of days in that month.
Examples: Parameter: “February” Returns: 28 Parameter: “December” Returns: 31

Part 1.4: Manipulating lists 2: sorting strings
(Only for star points; Not required to get full credits on this PA)

Write a function similar to 1.2, with a twist. The elements of the list are going to be **strings**, and will be in order of strictly increasing or decreasing length. Some examples of lists in strictly increasing length order are [“pan”, “banana”,”xylophone”], and [“kayn”, “swain”, “morgana”].

Function Name: fix_string_list
Parameter: data - A list with 3 elements (that are all lowercase strings) that is either in order of strictly increasing or decreasing length.
Return: A list with the same 3 elements as the parameter that is in order of strictly increasing length.
Description: Given a list of 3 strings that may be in order of either strictly increasing or decreasing length, return a list of 3 strings that are in order of strictly increasing length.
Examples: Parameter: [“aloe”,”zebra”,”magnesium”] Returns: [“aloe”,”zebra”,”magnesium”] Parameter: [“Hydrogen”, “Carbon”, “Iron”] Returns: [“Iron”,”Carbon”,”Hydrogen”]

Part 2: Write up (3 Points)

You must report how you tested your code as well as answer a few short questions in **Pa2-writeup.pdf.** A template has been provided. See submission instructions below on how to make a copy of this template to your own drive. In particular, you must provide:

A. Report Bugs and Issues
Include in your report any known bugs or issues with your program.

B. Questions
Answer the following questions:

1. In the following code snippets **A** and **B**, what will the difference, if any, in output be if **var = 5**? What if **var = 15**? Explain why there is or isn't a difference.

A

```
if var < 10:
    print("A")
elif var < 20:
    print("B")
else:
    print("C")
```

B

```
if var < 10:
    print("A")
if var < 20:
    print("B")
else:
    print("C")
```

2. Present a small code example of why indentation is important when working with if, elif, and else statements. Avoid using code that was already written for this assignment.
3. What are some operations or functions that work with both strings **and** lists?

Part 3: Video (2 Points)

For this part, you will create a video recording explaining the code you have written. Your video should answer the questions below. If you are working with a partner, both partners should be in the video and each partner should have some speaking point. One partner will answer question 1 and the other will answer question 2. An example of a code walkthrough/execution can be found here: https://youtu.be/7uw_Vi_F-dY

1. Follow the execution of the function **grade_calculator**, line by line, with an input of **74.4**. **Explain** which lines of code will be executed for this input and **why**.

2. Follow the execution of the function **fix_list**, line by line, with an input of [4, 6.5, 11]. **Explain** which lines of code will be executed for this input and **why**.

The following things will be checked in your video while grading:

1. The student(s) **code is clearly visible** in the video. [Hint: Increase your font size to 18
   Windows: "Options" → "Configure Idle" → Size
   Mac: "IDLE"(in top menu bar) → "Preferences" → "Fonts/Tabs" → "Size"]
2. The student(s) clearly **answer(s) the questions**.
   a. Students submitting individually answer both questions.
   b. Students submitting in pairs each answer one of the questions.
3. Video is within **time limit** (max: **2 mins**)

Part 4: Weekly Reflection (1 Point)

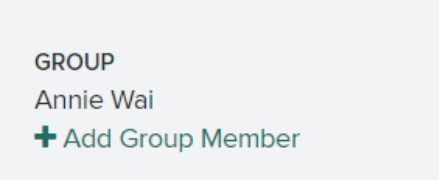
Fill out the reflection form here. This weekly reflection form is not optional, it counts towards 1 point of your assignment. All students have to individually submit their own weekly reflection regardless if you're working with a partner. Weekly reflections are not due when the PA is due. You may submit your weekly reflection for PA 2 anytime before **11:59pm on Friday Oct 23rd**.

Submission Instructions

Read all instructions carefully before submitting.

- You will need to submit **pa2.py, Pa2-writeup.pdf, and Pa2-video.mp4** on Gradescope, fill out the weekly reflection, and complete the degree planning assignment.
- To copy the writeup template and export as a pdf:
  - Click here to see a copy of the assignment format.
  - Click on "File" -> "Make a copy", and you will get a local copy of this Google Doc.
  - Fill in the Google Doc, making sure you keep the headings in about the same places.
  - Once you are done, in Google Docs, click on "File"-> "Download" -> "PDF Document", which will export it to a pdf.
- To record a video on zoom:
  - Start a zoom video meeting.
  - Choose "Join with Computer Audio"
  - If you're working with a partner, invite your partner into the meeting.
  - Share your computer's screen using "Share Screen"
  - Show your code on your computer's screen
  - Click "Record" > "Record on your computer".
  - Answer the required questions.
  - Once you are done, click "stop recording" and "End meeting".
  - Save your video file on your computer and name it as Pa2-video.mp4

- Sign into Gradescope and submit all files to PA 2. You should be able to drag and drop multiple files into the upload files window. Ask a teaching staff for help if you are unsure whether you've submitted properly.
- If you are working with a partner, **only one member will need to submit the files** in Gradescope. Do not both submit the files individually to Gradescope. It will be your responsibility to ensure both members are added in Gradescope.
- To add a group member on Gradescope:
    - First submit all files to PA 2.
    - This should take you to your submissions page. Otherwise, you can view your submission by clicking on the assignment.
    - Click on **"Add Group Member"** on the top right under your name.

    - Confirm you have added your partner. You should see both you and your partner's name under "Group" in the top right after submitting.
- You may submit multiple times until the deadline. We will be grading only your latest submission. So, please make sure that your latest submission is your best version!