Lab Assignment 3: Tuition savings calculator

With the help of an RESP (Registered Education Savings Plan), a parent can start putting aside money for a child's post-secondary education. Suppose that a young couple deposits **$2,000** to start the savings plan, and each month thereafter they contribute some fixed amount (**($200/month**). Assume that a fixed interested rate of **6.25%** per year compounded monthly is applied.

Each month the balance increases according to the following formula:

New balance = Old balance + (Old balance * Monthly interest rate)
+ Monthly contribution (1)

Since the university tuition fee increases over the years, you are asked to further analyze whether the saving is enough to cover the tuition fee for a 4-year study program when the child turns 18 and enters post-secondary education. You will help them analyze this for 3 different types University programs: Arts, Science, and Engineering, for which the average tuition fees in the current year are as follows: **$5550**, **$6150**, and **$6550** respectively. The predicted average percentage of tuition increase *each year* is **7%**.

Each year the new fee due to the annual increase is calculated as follows:

New cost = Old cost + (Old cost * Annual increase rate) (2)

Version 0: Getting Started

Download and unzip the files in the **V0GetStarted.zip** file into your Current Folder. Open the lab3V0.py file in the Editor Window and inspect it before running it. The program simply generates a counter variable that increments by 1 and stores it as a vector using a ***for loop***. A simple linear plot is also produced. The command output and plot should match the image provided in the zip file.

Version 1: Savings/Tuition Calculation

Download and unzip the files in the **V1Saving.zip** file into your Current Folder. In Version 1, you are asked to calculate the balance of the saving account in each month over 18 years, so the total saving by the end of the 18th year is known. Then calculate the *predicted tota*l tuition fee by the start of the 18th year, which should be the total fee over a 4-year university program (i.e. from year 19 to year 22). A **`for`** loop must be used for the above calculations. Display the final savings amount and tuition cost for the 3 programs numerically in the command console and graphically through a plot. Please use the provided lab3V1.py file to write your program. Remove or modify the exiting code to satisfy the above requirements. You can perform the following steps below

1. Calculate the monthly balance of the savings plan for the next 18 years, based on the formula in equation (1). You must save the amount in a **list/ndarray** format.

2. Calculate the predicted tuition fee per year for the next 22 years, and then add up the fees over the last 4 college years (from the year 19 to year 22) to get the predicted total education cost for a 4-year program.

3. Display the final savings amount and tuition cost to **2 decimal spots by using the print** statement. To visualize the results, you are also asked to plot the annual savings with respect to the number of years; plot a horizontal line (threshold) indicating the total tuition fee for each program on the same plot. There should be 3 horizontal lines (different colors for each) with each representing a different program. Finally, add a suitable title, label all axis and provide a legend.

The output in the command window should match the image in the CommandOutputV1.jpg file that was unzipped. Your plot should also match (as close as possible) to the one in the image. Include a section comment header called `Saving Calculation`, `Tuition Calculation`, and `Plot` and summarize what the sections does.

Version 2: Savings optimization

Download and unzip the files in the **V2Optimization.zip** file into your Current Folder. For Version 2, in addition to all requirements for Version 1, add a user input option to allow the user to select which program information will be displayed. To select a program, the user will enter **1 for Arts, 2 for Science, and 3 for Engineering**.-After a correct input is received, a **match** or **if elif statement** can be used to tie the input to the program selected. Your program will verify and determine if the savings amount calculated in version 1 met the program tuition cost. A simple output as to whether or not the savings amount in the command window is sufficient. You will also optimize and determine the minimum monthly contribution amount (to the **nearest dollar**) that is required to satisfy the program tuition cost. Since each program tuition is different, the monthly amount required will be also be different.

Assume that the same first contribution of $2,000 is used as in Version 1. Based on the **`for`** loop structure constructed in Version 1, reuse it while adding a conditional **`while`** loop.

1. Assume that the monthly payment starts at 1 dollar with an increment of $1 each time and repeat the savings calculation until the saving goal is reached. For this, the students are asked to use a **`while`** loop.

2. Use the input function to get the user input. A match statement or if elif statements can be used to determine user program selection.

3. Check the savings against the program tuition and output a message indicating sufficient funds or not

4. Save the ‘minimum’ monthly contribution and print it on the command window.

Test Case 1 – 1
Test Case 2 – 2
Test Case 3 – 3

The output in the command window should match the images for the 3 different programs selected in the CommandOutputV2.pdf file that was unzipped. Along with the section comment headers from version 1, add an additional section comment header called `Optimization` and summarize what the sections does.

Code Requirements/Submission details

1. You are required to use at least **one for** and **one while loop** in this assignment for Version 2. Failure to do this will result in a possible maximum mark of **'Min. Pass'**.

2. You must use the **input** instruction to prompt the user to enter either 1, 2 or 3. No error check is required

3. The savings/tuition amount must have **2 decimal places**. The monthly contribution must have **0 decimal places**.

4. File naming convention.

lab3<part>_<ccid>.py
For example **lab3V1_whoy.py** or **lab3V2_whoy.py**

5. The assignment is due

Version 1 – Wednesday, Feb 15, 2023 at 4:30PM MST
Version 2 – Tuesday, Feb 28, 2023 at 4:30PM MST

6. Please consult the schedule posted on Eclass for due dates. Late submission for Version 1 will not be accepted. Late submission for Version 2 will result in a **reduction of 10%** per business day (1 day max). Late submission penalties can be waived at the Lab instructor's discretion with valid proof of documentation (ie, doctor's note). Non valid reasons will not be accepted such as but not limited to - forgetting due date, computer/laptop crashing, submitting wrong assignment, etc). It is your responsibility to verify that the submitted file is correct and uploaded properly to the Eclass submission link.

Additional Note/Hints

1. For Version 2, continuing from your Version 1 code, you can use an input statement together with a match statement (or a bunch of if elif) for selecting from the 3 academic programs.