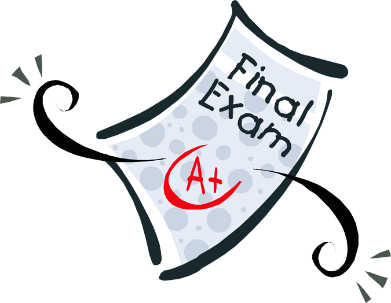
**CS161**

**Lab 02**

**Due Tuesday 10/18/22**

The interactive program focuses on **control statements**, such as **if – elif – else**, **loops**, and **functions**. The program reads input from the user for a student's grades on **homework** and **three exams**, and uses them to compute the student's **total score**. Based on the total score, the program will print out the corresponding **letter grade** and **encouraging comments** for the student.

NOTE: this assignment is adapted from professor Sisi Virasak’s CS161 lab#2 (2021).



## How to Submit Lab #2

1. Create a python program and name it **gradanator.py** (e.g. gradanator\_nw.py)**.**
2. Include a header comment at the top of the program that contains:
   1. File name.
   2. Author name.
   3. Date completed.
   4. Briefly explain what the program does.

Example:

"""File: program\_name.py

Author: Harry Potter

Date completed: 01/02/2030

Description: what does this program do?

"""

1. Please leave short comments throughout your program to explain what some block/line of code does, if needed. (*for learning purpose; also, this helps me see what you understand or where you may have confusions* 😊)
2. Take a **screenshot** of successful execution of your program, you can name it **gradanator** and save it as **png** or **jpeg** format.
3. Put **gradanator.py** and **gradanator.jpeg** (or png) in a folder named **week02\_initials** (replace “initials” with your own initials. e.g. week02\_nw), compress the folder and upload the zipped folder to Moodle.

## Introduction

Below is one example log of execution from the program. Your output should match the examples exactly given the same input. Be mindful of spacing, such as after input prompts and between output sections.

|  |
| --- |
| This program reads exam/homework scores  and reports your overall course grade.  Exam 1:  Weight (0-100)? 10  Score earned? 78  Total points = 78 / 100  Weighted score = 7.8  Exam 2:  Weight (0-100)? **10**  Score earned? **84**  Total points = 84 / 100  Weighted score = 8.4  Final Exam:  Weight (0-100)? 30  Score earned? 100  Total points = 100 / 100  Weighted score = 30.0  Homework:  Weight (0-100)? 50  Number of assignments? 3  Assignment 1 score? **14**  Assignment 1 max? **15**  Assignment 2 score? **17**  Assignment 2 max? **20**  Assignment 3 score? **19**  Assignment 3 max? **25**  How many sections did you attend? 10  Section points = 30 / 34  Total points = 80 / 94  Weighted score = 42.6  Overall percentage = 88.8  Your grade will be at least: B  *<< your custom grade message here >>* |

NOTE: This program behaves differently depending on the user input (user input is bold and underlined in the example.)

## Implementations

1. The program begins with an introduction message that briefly explains the program to the user.
2. Each category is weighted – points for each category is scaled up to a fraction of the one hundred percent (100%) grade for the course. As the program begins reading each category, it first prompts for the category's **weight** and the **score earned** for the category, followed by the **calculated total points** and **weighted score**.
3. The user begins by entering **scores** earned on Exam 1 and the **weight** of Exam 1.

(Note: the total score of an exam is 100 points.)

Then output the **total points** and **weighted scores** for Exam 1. The weighted score equal to the user's score multiplied by the exam's weight. For example,

|  |
| --- |
| Total points: **78**  Weight for Exam 1: **10**  Weighted score: 78 / 100 \* 10 = 7.8 |

(NOTE: user input is bold and underlined in the example.)

1. Next, the program prompts the user to input data about Exam 2 and then about the Final Exam. The behavior for each is the same as the behavior for Exam 1.
2. Next, the user enters information about the student’s homework, including the **weight** and **how many assignments** were given. For each assignment, the user enters a **score** and **max points** for that assignment. Use a **cumulative sum**. See example,

|  |
| --- |
| Homework:  Weight (0-100)? **50**  Number of assignments? **3**  Assignment 1 score? **14**  Assignment 1 max points? **15**  Assignment 2 score? **17**  Assignment 2 max points? **20**  Assignment 3 score? **19**  Assignment 3 max points? **25** |

(NOTE: user input is bold and underlined in the example.)

Hint: **use a loop** instead of repeatedly writing the same code to get user input and calculate the grade.

1. Part of the homework score comes from **sections attended**. To simplify the formula, assume that each section attended is worth **3 points**, up to a **maximum of 34 points**. For example,

|  |
| --- |
| How many sections did you attend? **10**  Attendance points = 30 / 34 |

(NOTE: user input is bold and underlined in the example.)

1. After collecting all the data for the homework assignments and attendance, the program prints out the **total points** and **weighted scores** for the homework.
2. Once the program has read the user information for the homework and three exams, it prints out the student's **overall percentage** earned in the course, which is the **sum of the weighted scores** (round the score to **show one digit after the decimal point**) from the four categories, see the formula below:

formula to calculate grade

1. The program prints out a **letter grade** the student will get in the course, based on the following scale.

|  |  |
| --- | --- |
| **90% and above:** | **A** |
| **89.99% - 80%:** | **B** |
| **79.99% - 70%:** | **C** |
| **69.99% - 60%:** | **D** |
| **under 60%:** | **F** |

1. After showing the user their letter grade, print a **custom message** of your choice about the grade they get. This message should be **different for each grade range** shown above. It should be at least 1 line of any non-offensive text you like.

### Development Strategy and Hints:

* A student can receive extra credit on an *individual assignment*, but **the total points for each homework are capped at the maximum possible**. For example, a student can receive a score of 22/20 on one homework assignment, but if their total homework score for all assignments is 63/60, this score should be capped at 60/60.
* **Section points** for attendance are capped at **34**.
* **Exam scores** are capped at **100**.
* You can use [**min()**](https://www.w3schools.com/python/ref_func_min.asp) functions to constrain numbers to be within a particular bound.
* Otherwise, you may assume the user enters **valid input**.

- When prompted for a value, the user will enter an integer in the proper range;

- The user will enter a number of homework assignments ≥ 1;

- The sum of the four categories’ weights will be exactly 100;

- The weight of each category will be a non-negative number.

* All weighted scores and grades are printed with **no more than 1 digit after the decimal point**. You can achieve this with the [**round()**](https://www.w3schools.com/python/ref_func_round.asp) function. The following code prints variable x rounded to the nearest tenth:

x = 1.2345

print("x rounded to the nearest tenth is **" + round(x, 1)**)

**Create functions:**

To make your program more organized and make the code reusable.

1. Create functions ***exam\_score(****parameters****),*** ***homework\_score(****parameters****)***, and ***course\_grade(****parameters****)***. You will decide what parameters each function takes, and how many parameters each function takes (the “parameters” in the parenthesis is just a placeholder here).
2. After defining the functions, call the functions to implement the behaviors of the program.
3. Below is an example of how we can call the functions created (this example doesn’t include the function definition part of the program; this is just to show how the program should behave):

score\_one = exam\_score("Exam 1")

score\_two = exam\_score ("Exam 2")

score\_three = exam\_score ("Final Exam")

score\_four = homework\_score()

course\_grade(score\_one, score\_two, score\_three, score\_four)

1. In the script above, we first call ***exam\_score*** function three times to calculate the weighted score of each exam. We pass the name of each exam as the parameter to the function every time. We assign each return value of ***exam\_score*** function to variables ***score\_one***, ***score\_two***, and ***score\_three***.
2. Then, we call ***homework\_score()*** function to calculate the weighted score of homework, and assign the ***return value*** to variable ***score\_four***.
3. Lastly, we call the ***course\_grade*** function and pass in the four scores as arguments. This function gives us the letter grade of the student and the corresponding encouraging message.

**Guidelines**

1. Your functions will need to make appropriate use of parameters and return values.
2. Each function should perform a coherent task and should not do too large a share of the overall work.
3. Avoid lengthy “chaining” of function calls, where each function calls the next, no values are returned, and control does not come back to main (we will talk more about main function soon, for now, just regard the example above (step 3 in “Create Functions”) as the main function part).
4. Give meaningful names to functions (already given in the instructions above) and variables, and use proper indentation and whitespace.
5. Declare variables in the smallest scope needed.
6. Include docstring in your function (see [PEP 257](https://www.python.org/dev/peps/pep-0257/))
7. Also, see [PEP 8](https://www.python.org/dev/peps/pep-0008/) for code style.
8. Tackle parts of the program (exam 1, exam 2, final exam, and homework) one at a time, rather than writing the entire program at once.
9. Write a bit of code, get it to run, and test what you have so far. If you try to write large amounts of code without attempting to run it, you may encounter a large list of errors and/or bugs.

**Before you submit your program, check that it runs without any errors.**