CS 115 - Introduction to Programming in Python

Lab 01

Lab Objectives: Input/output, data, expressions, branching

Instructions: For this assignment, you can use your favorite IDE (Spyder or Jupyter recommended). Upload your solutions as a single .zip file to the Lab01 assignment for your section in Moodle before the end of your lab session. Use the following naming convention: **SS_Lab01_Surname_FirstName.zip** where SS is the section number 01, 02, 03, ..., & and Surname is your family name, & FirstName is first name. You must attend the lab Zoom session. You must show and explain your solutions to your TA during your lab session and must answer their questions to get your grade by the end of your lab session (the week of Oct 11).

Students who do not attend the lab Zoom session but submit will get 0.

1. Write a program, `Lab01_yourname_Q1.py`, that prompts the user to enter the coefficients of a quadratic polynomial `f(x) = a * x ** 2 + b * x + c` where `a`, `b`, and `c` are all int values. Then it inputs an *x* value (float) from the user, calculates and prints the `f(x)` for that entered *x* value.

Sample Run 1: (User inputs are red)	**Sample Run 2:**
Enter a: 3 Enter b: 4 Enter c: 5 Enter x: 6 f(x) = 3 * x ** 2 + 4 * x + 5 f(6.0) = 137.00	Enter a: 3 Enter b: 4 Enter c: 5 Enter x: 6.8 f(x) = 3 * x ** 2 + 4 * x + 5 f(6.8) = 170.92

2. Write a program, `Lab01_yourname_Q2.py`, that inputs three integers from the user and reports how many of those three integers are odd.

Sample Run 1: (User inputs are red)	**Sample Run 2:**
Enter first integer: 1 Enter second integer: 3 Enter third integer: 9 3 of the 3 numbers are odd.	Enter first integer: 4 Enter second integer: 0 Enter third integer: 28 0 of the 3 numbers are odd.
Sample Run 3: Enter first integer: 4 Enter second integer: 3 Enter third integer: 7 2 of the 3 numbers are odd.	**Sample Run 4:** Enter first integer: 4 Enter second integer: 32 Enter third integer: 29 1 of the 3 numbers are odd.

3. Write a program, `Lab01_yourname_Q3.py`, that prompts the user for four integers: starting hour, starting minute, ending hour, and ending minute to decide if the time interval will be sufficient for eating lunch. Assume that one needs at least 35 minutes to eat lunch. Each pair of hours and minutes represents a time on the 24-hour clock. For example, 13:24 will be represented as 13 and 24.

- If the ending time is earlier than the starting time, it should display 'Starting time must be before ending time...'.
- If the gap between the two times is long enough to eat lunch (that is, if the ending time is at least 35 minutes after the starting time), it should display 'You can have lunch!'
- Otherwise, it should display 'Not enough time for lunch :('.

Assume that all input values are valid: the hours are both between 0 and 23, and the minutes are between 0 and 59. You may also assume that both times represent times in the same day, e.g. the first time won't represent a time yesterday while the second time represents a time today.

Sample Run 1: (User inputs are red) Enter starting hour: 11 Enter starting minute: 05 Enter ending hour: 11 Enter ending minute: 45 You can have lunch!	**Sample Run 2:** Enter starting hour: 13 Enter starting minute: 05 Enter ending hour: 12 Enter ending minute: 10 Starting time must be before ending time...
Sample Run 3: Enter starting hour: 11 Enter starting minute: 40 Enter ending hour: 12 Enter ending minute: 15 You can have lunch!	**Sample Run 4:** Enter starting hour: 12 Enter starting minute: 50 Enter ending hour: 13 Enter ending minute: 10 Not enough time for lunch :(
Sample Run 5: Enter starting hour: 12 Enter starting minute: 50 Enter ending hour: 12 Enter ending minute: 10 Starting time must be before ending time...	**Sample Run 6:** Enter starting hour: 15 Enter starting minute: 20 Enter ending hour: 15 Enter ending minute: 40 Not enough time for lunch :(