NYU Tandon School of Engineering

CS-UY 1114 Fall 2022

Homework 03

Due: 11:59pm, Thursday, October 06, 2022

Submission instructions

1. You should submit your homework on **Gradescope**.
2. For this assignment you should turn in 5 separate `.py` files named according to the following pattern: `hw3_q1.py`, `hw3_q2.py`, etc.
3. Each Python file you submit should contain a header comment block as follows:

```
"""
Author: [Your name here]
Assignment / Part: HW3 - Q1 (depending on the file name)
Date due: 2022-10-06, 11:59pm
I pledge that I have completed this assignment without
collaborating with anyone else, in conformance with the
NYU School of Engineering Policies and Procedures on
Academic Misconduct.
"""
```

No late submissions will be accepted.

REMINDER: *Do not use any Python structures that we have not learned in class.*

For this specific assignment, you may use everything we have learned up to, **and including**, selection statements (i.e. `if`, `elif`, `else`). Please reach out to us if you're at all unsure about any instruction or whether a Python structure is or is not allowed.

Do **not** use, for example, `for`- and `while`-loops, user-defined functions (*except* for `main()` if your instructor has covered it during lecture), strings and string methods, file i/o, exception handling, dictionaries, lists, tuples, and/or object-oriented programming.

Problems

1. **Are You Experienced? (`hw3_q1.py`)**
2. **It's Super Effective! (`hw3_q2.py`)**
3. **Oh, Oh, Telephone Line, Give Me Some Time (`hw3_q3.py`)**
4. **Why, This Car Could Be Systematic, Programmatic, Quadratic! (`hw3_q4.py`)**
5. **What Is This, A Math Class? (`hw3_q5.py`)**

Question 1: *Are You Experienced?*

Let's say that we are developing a video-game where the user's level is determined by their current experience points (XP). For this problem, the user will enter their current XP as input and the program will output their current level.

XP	Level
Below 18.0	1
18.0 — 24.9	2
25.0 — 29.9	3
30.0 — 39.9	4
40.0 — 50.0	5

Figure 1: Player levels with their respective *XP* range equivalents. Note that experience points can be any number between and including *0.0* and *50.0*.

Your program should function **exactly** as follows:

```
Enter this user's current XP: 44.2
Level 5 Player (XP: 44.2)
```

```
Enter this user's current XP: 83.1719
ERROR: Please enter a valid XP value.
```

```
Enter this user's current XP: 7
Level 1 Player (XP: 7.0)
```

```
Enter this user's current XP: 26.4
Level 3 Player (XP: 26.4)
```

Question 2: *It's Super Effective!*

In the *Pokemon™* game series, there is always a chance that your Pokemon will land a critical hit when attacking. This basically means that, based on your Pokemon's stats, your attack **might** roughly double in damage. The "might" here is actually based on probability:

> Whether a move scores a critical hit is determined by comparing a 1-byte **random number (0 to 255)** against a 1-byte threshold value (also 0 to 255); if the random number is less than the threshold, the Pokémon scores a critical hit.

— ***Critical hit***, *Bulbapedia*.

Basically, we'll have two values:

- A random value, `R` from 0 to 255.
- A threshold value, `T` from 0 to 255. If this value is higher than `R`, the Pokemon lands a critical hit; we can calculate this value as follows:

```
T = (Pokemon_Speed / 2)
```

Figure 2: Threshold formula, where `Pokemon_Speed` is equal to the Pokemon's speed stat.

If it is determined that the Pokemon has landed a critical hit, the damage multiplier (that is, the number by which the Pokemon's attack will be multiplied by) will be equal to:

```
M = (2L + 6) / (L + 6)
```

Figure 3: Multiplier formula, where `L` is equal to the Pokemon's level.

With this knowledge in hand, write a program that will ask the user for the Pokemon's level and speed stat (both integer values), and print out the move's **damage multiplier**. You'll want think about what would happen if the Pokemon *doesn't* land a critical hit as well. You may assume that the Pokemon's level and speed stat will always be positive integers.

See some possible executions below:

```
What is this Pokémon's level? 90
What is this Pokémon's speed? 150
The Pokémon's move will be 1.94x stronger!
```

```
What is this Pokémon's level? 60
What is this Pokémon's speed? 200
The Pokémon's move will be 1.91x stronger!
```

NOTE: When testing this program, you may want to keep in mind that procedures that involve randomness may need to be tested several times in order to observe different behavior. Try changing your test values towards such that they are more likely to land a critical hit.

Question 3: *Oh, Oh, Telephone Line, Give Me Some Time.*

***Note**: As a reminder, you cannot use lists, tuples, or any container object in this homework assignment.*

Write a program that computes the cost of a long-distance call. The cost of the call is determined according to the following rate schedule:

- Any call started between 5:00 A.M. and 9:00 P.M., Monday through Thursday, is billed at a rate of $0.55 per minute.
- Any call starting before 5:00 A.M. or after 9:00 P.M., Monday through Thursday, is charged at a rate of $0.35 per minute.
- Any call started on a Friday, Saturday, or Sunday is charged at a rate of $0.10 per minute.

The input will consist of:

- The day of the week
- The time the call started
- The length of the call in minutes

The output will be the cost of the call.

A few things to keep in mind:

- The time must be input as a 4-digit number, representing the time in **military, or 24-hour, time**. For example, the time 1:30 P.M. corresponds to the input `1330`.
- The day of the week must be read as one of the following three character strings:
  - `"Mon"`
  - `"Tue"`
  - `"Wed"`
  - `"Thr"`
  - `"Fri"`
  - `"Sat"`
  - `"Sun"`
- The number of minutes will be input as a positive integer.

You can assume the user will always enter valid inputs. For example, an execution *could* look like this:

```
Enter the day the call started at: Thr
Enter the time the call started at (hhmm): 2350
Enter the duration of the call (in minutes): 22
This call will cost $7.7
```

Question 4: *Why, This Car Could Be Systematic, Programmatic, Quadratic!*

Write a program that asks the user to input three floating-point numbers: `a`, `b`, and `c` (just this once, **only these three** single-letter variables will be permitted). These are the parameters of the **quadratic equation**. Classify the equation as one of the following:

- **Infinite number of solutions**: For example, `a = 0`, `b = 0`, `c = 0` has an infinite number of solutions.
- **No solution**: For example, `a = 0`, `b = 0`, `c = 4` has no solution.
- **No real solution**: For example, `a = 1`, `b = 0`, `c = 4` has no real solutions.

- **One real solution**: In cases there is a solutions, please print the solutions.
- **Two real solutions**: In cases there are two real solutions, please print the solutions.

Hint: If a ≠ 0 and there are real solutions to the equation, you can get these solutions using the following formula:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Figure 4: The **quadratic formula**.

The number of solutions depends on whether the discriminant (b^2 - 4ac) is positive, zero, or negative.

For example, an execution *could* look like:

```
Please enter value of a: 1
Please enter value of b: 4
Please enter value of c: 4
This equation has 1 solution: x = -2.0
```

Question 5: *What Is This, A Math Class?*

Write a program that asks the user to input the lengths of the three sides of a triangle. Then, classify the triangle into one of the following:

- **Equilateral triangle**
- **Isosceles right triangle**
- **Isosceles triangle that is not a right triangle**
- **A triangle that is not an isosceles and not an equilateral (i.e. scalene)**

Your program's execution **must** be formatted in the following way:

```
Length of the first side: 30
Length of the second side: 30
Length of the third side: 30
30.0, 30.0, 30.0 form an equilateral triangle
```

A few things to note here:

- You may assume that the inputs entered by the user will always form a valid triangle, since some combinations of inputs don't.
- Since floating points are only approximations to real world values, in some cases, different approximations represent the same real value. Therefore, we are more tolerant when checking if two floats are equal to one another; if they are close enough to one another we consider them to be equal. In this question, you may assume that two floating points are equal if they are far from each other **by not more than 0.00001**. For example 43.25678976 and 43.256786371 are considered equal (because 43.25678976 – 43.256786371 = 0.000003389).