NYU Tandon School of Engineering

CS-UY 1114 Spring 2023

Homework 02

Due: 11:59pm, Thursday, February 16th, 2023

Submission instructions

1. You should submit your homework on **Gradescope**.
2. For this assignment you should turn in 6 separate `.py` files named according to the following pattern: `hw2_q1.py`, `hw2_q2.py`, etc.
3. Each Python file you submit should contain a header comment block as follows:

```
"""
Author: [Your name here]
Assignment / Part: HW2 - Q1 (etc.)
Date due: 2023-02-16, 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**, variables, types, mathematical and boolean expressions, user IO (i.e. `print()` and `input()`), number systems, and the `math` / `random` modules, and 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), string methods, file i/o, exception handling, dictionaries, lists, tuples, and/or object-oriented programming.

Failure to abide by any of these instructions will make your submission subject to point deductions.

Problems

1. **Harmonic Analysis (`hw2_q1.py`)**
2. **Oh No! The Pokémon Broke Free! (`hw2_q2.py`)**
3. **Collective Timetables (`hw2_q3.py`)**
4. **It's Super Effective! (`hw2_q4.py`)**
5. **Are You Experienced? (`hw2_q5.py`)**
6. **Oh, Oh, Telephone Line, Give Me Some Time (`hw2_q6.py`)**

Question 1: *Harmonic Analysis*

For engineers working at music platform companies such as Apple Music and Spotify, one of their most crucial jobs is turning the natural sound waves of an artist's music into data (that is, actual ones and zeroes) that they can stream straight into your phone with the best quality possible.

The problem with natural sound waves is that it is pretty much impossible to replicate all of their nuanced peaks and valleys *exactly* using programmatic methods. Oftentimes, what ends up happening is that engineers convert complex waveforms into much simpler ones:

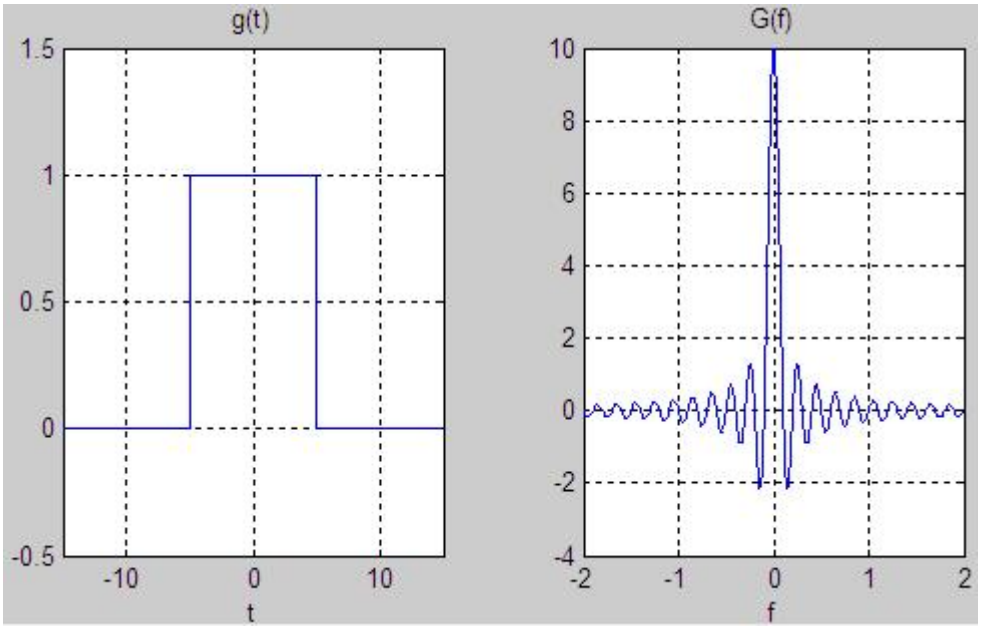

Figure 1: A "natural" waveform, such as the one on the right, can be simplified into a very simple **pulse, or square, wave** which, as you can imagine, is a lot easier to model using data.

This process is known as a reverse **Fourier Transform**, and it may be one of the most important developments in modern mathematics, as far as applicability goes. Anyway, the solution to the Fourier Transform in figure one is actually relatively simple:

$$f(w) = \frac{2 \times sin(wT)}{w}$$

Figure 2: A Fourier Transform solution, where `F(w)` is the referred to as the amplitude spectrum (i.e. how "loud" the square wave will be), `w` is referred to as the real frequency variable (which you can think of as the frequency of the natural sound wave), and `T` represents the duration of the soundwave. This is not strictly true but we're simplifying a lot here to make the problem easier.

Write a program that will:

1. Ask the user to input a value for the frequency (`w` in figure 1).
2. Ask the user to input a value for the duration of the sound wave (`T` in figure 1).
3. Calculate the value of the amplitude spectrum (`F(w)` in figure 1).
4. Display the result rounded to the third decimal place.

For example, your execution *could* look as follows:

```
Enter a value for the frequency, w: 4.2
Enter a value for the duration of the sound wave, T: 20
```

```
The amplitude spectrum of this Fourier transform is: 0.349
```

Question 2: *Oh No! The Pokémon Broke Free!*

In the video game series **Pokémon™**, the probability of catching a Pokémon with a regular PokéBall is determined by using the following formula:

$$f = \left\lfloor \frac{HP_{max} \times 255 \times 4}{HP_{current} \times Ball} \right\rfloor$$

Figure 3: Where HP_{max} represents the maximum amount of health points the Pokémon could have, $HP_{current}$ represents the amount of health points the Pokémon currently has, and *Ball* represents a randomly-generated value between 0 and 255. Notice that the result of this division is always *rounded down* (because of those $\lfloor \ \rfloor$ braces).

After calculating f using the formula in figure 3, generate one final pseudo-random number between 0 and 255. If f is greater than or equal to that pseudo-random number, then you have caught the Pokémon. Otherwise, it will break free.

Write a program that will ask the user to enter the value of HP_{max}. From there, your program must generate the following pseudo-random values:

1. The Pokémon's current health, which can be any value between and including 1 and HP_{max}.
2. The PokéBall's *Ball* value which, again, is a value between and including 0 and 255.

Based on these inputs and the calculations from above, your program will print `"You've caught the Pokémon!"` if the Pokémon was caught, or `"Oh no! The Pokémon broke free!"` if it broke free.

In other words, two *possible* executions of this program are the following:

```
Enter the max health of this Pokémon: 140
Oh no! The Pokémon broke free!
```

```
Enter the max health of this Pokémon: 120
You've caught the Pokémon!
```

Your program need not interact with the player the same way as above, but it *must* print the messages specified above depending on the user input. Keep in mind that, since this program uses random behaviour, it may take a while for you to get a specific outcome.

Note that this is a simplified version of how the **catch rate** is actually calculated in generation I Pokémon games. We'll revisit this problem once you have enough know-how to calculate it exactly as the game does.

Question 3: *Collective Timetables*

Suppose Semi and Daniel, two of our indefatigable CAs, each worked for a certain amount of time, and we wanted to calculate the total time both of them worked.

Write a program that reads a number of days, hours, and minutes minutes each of them worked, and prints the total time both of them worked together as days, hours, and minutes.

For example, an execution *could* look like this:

```
Please enter the number of days Semi has worked: 4
Please enter the number of hours Semi has worked: 5
Please enter the number of minutes Semi has worked: 31
Please enter the number of days Daniel has worked: 6
Please enter the number of hours Daniel has worked: 0
Please enter the number of minutes Daniel has worked: 15
The total time both of them worked together is: 10 days, 5 hours and 46 minutes
```

Question 4: *It's Super Effective!*

Coming back to the *Pokemon™* game series, when you battle other Pokémon, there is always a chance that any 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:

- Critical Hit:

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

- No Critical Hit:

```
What is this Pokémon's level? 40
What is this Pokémon's speed? 100
The Pokémon's move will be 1x 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 5: *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 15.0	1
15.0 — 24.9	2
25.0 — 29.9	3
30.0 — 39.9	4
40.0 — 60.0	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 *60.0*.

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 6: *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 5:30 A.M. and 9:00 P.M. (inclusive on both ends), Monday through Thursday, is billed at a rate of $0.55 per minute.
- Any call starting before 5:30 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): 500
Enter the duration of the call (in minutes): 22
This call will cost $7.7
```