NYU Tandon School of Engineering

CS-UY 1114 Fall 2022

Homework 02

Due: 11:59pm, Thursday, September 29, 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: `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 (depending on the file name)
Date due: 2022-09-29, 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**, Python modules and boolean expressions. 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, selection statements (i.e. `if`, `elif`, `else`), `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. **Gravity Don't Pull Me (`hw2_q1.py`)**
2. **Harmonic Analysis (`hw2_q2.py`)**
3. **Oh No! The Pokémon Broke Free! (`hw2_q3.py`)**
4. **Collective Timetables (`hw2_q4.py`)**
5. **Curb Your (Graduation) Privilege (`hw2_q5.py`)**

Question 1: *Gravity Don't Pull Me*

The formula for calculating the force of gravity between two objects in outer space is as follows:

$$F = G\frac{m_1 m_2}{r^2}$$

Figure 1: Newton's law of universal gravitation, where F represents the gravitational force, m_1 represents the mass of the first object, m_2 the mass of the second object, r the distance between the *centers* of the masses, and G the gravitational constant. This latter value is rather small, and for this problem you can approximate it to be `6 * 10 ** -11`. Because this number is so small, your results may vary slightly from run to run. This is pretty much okay as long as the magnitudes are accurate (i.e. the power of 10).

Ask the user for five pieces of data:

- The mass and the radius of the first object.
- The mass and the radius of the second object.
- The distance from the surface of the first object to the surface of the second object.

You may assume that the user will always enter positive numbers that make physical sense.

Then, print out the value of the force of gravity **exactly** as it is shown below:

```
Enter the mass of the first object: 1000000000
Enter the radius of the first object: 5000000
Enter the mass of the second object: 2000000000
Enter the radius of the second object: 7000000
Enter the distance between the surfaces of both objects: 123456789
The force of gravitation between these two objects is 6.5400292869222625e-
09 N.
```

Question 2: *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:

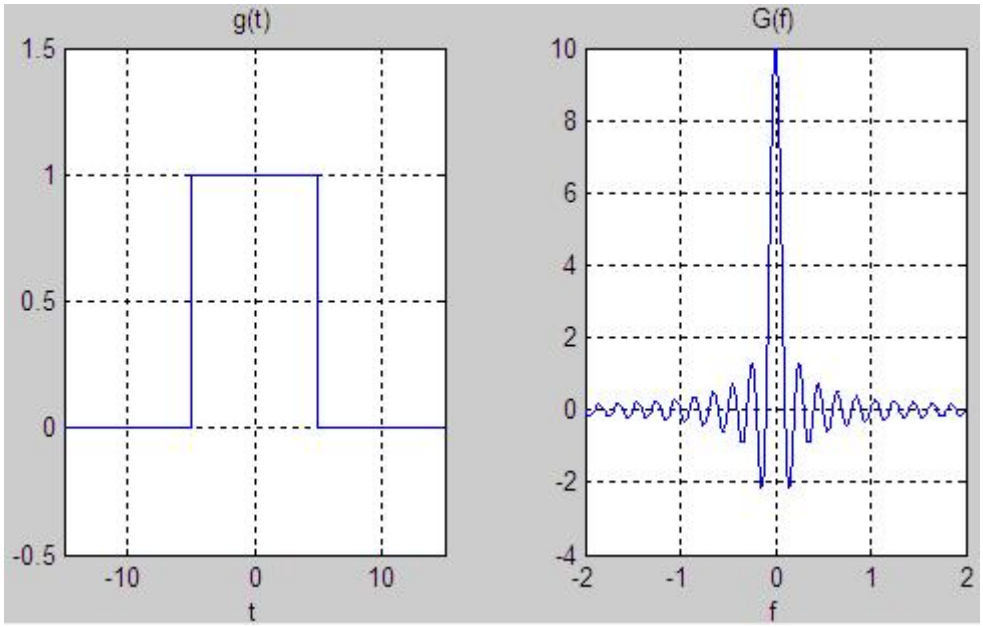

Figure 1: A "natural" waveform, such as the one on the right, can be simplified into a very simple **pulse, or square, wave** which, as you can imagine, is a lot easier to model using data.

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: 0.34
Enter a value for the duration of the sound wave, T: 0.5002
The amplitude spectrum of this Fourier transform is: 0.996
```

Question 3: *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 `⌊ ⌋` 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 pseudorandom 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 `True` if the Pokémon was caught, or `False` if it broke free.

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 4: *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: 2
Please enter the number of hours Semi has worked: 12
Please enter the number of minutes Semi has worked: 15
Please enter the number of days Daniel has worked: 3
Please enter the number of hours Daniel has worked: 15
Please enter the number of minutes Daniel has worked: 20
The total time both of them worked together is: 6 days, 3 hours and 35
minutes.
```

Problem 5: *Curb Your (Graduation) Privilege*

Note: As a reminder, selection statements (i.e. `if`, `elif`, `else`) are *not* allowed on this homework assignment.

Let's say you are tasked with writing the program that determines whether NYU students can graduate or not. Our interface will be simple—it basically prints `True` if a student meets the requirements needed to graduate, and `False` if not. The requirements are as follows:

A student may graduate if:

- They have accumulated 64 credits **and** are approved seniors, or:
- They have accumulated 40 credits **and** have special permission from their advisor, or:
- They have special permission from the dean to graduate, regardless of how many credits they have.

For this program, you may assume that the user will **always** enter the character `'y'` for "yes", or the character `'n'` for "no".

Your program's output *must* behave as follows:

- *Example 1*:

```
Do you have permission from the dean? [y/n] n
Do you have permission from your advisor? [y/n] n
Do you hold senior status? [y/n] y
How many credits do you have? 20
This student can graduate: False
```

- *Example 2*:

```
Do you have permission from the dean? [y/n] n
Do you have permission from your advisor? [y/n] n
Do you hold senior status? [y/n] y
How many credits do you have? 65
This student can graduate: True
```

- *Example 3*:

```
Do you have permission from the dean? [y/n] n
Do you have permission from your advisor? [y/n] y
Do you hold senior status? [y/n] n
How many credits do you have? 50
This student can graduate: True
```

These are, of course, not the only combinations possible, so make sure to test a few more.