Introduction to CS for Data Science

HW1 - Control Flow

1. Dealing with errors

When writing software, you will inevitably run into errors. While some errors generated by the Python interpreter are informative, most are rather cryptic. One way to get used to, and understand these error messages, is to force common programming errors intentionally and then read and figure out the result error massages. The script 'debugging.py' includes some operations you saw in class. You are required to make some changes to this script and observe how the Python interpreter reacts to these changes. Although you are using PyCharm and can fix some of the mistakes automatically, make sure you run the script after every change you make and observe and understand the errors. You need to submit a fixed script including comments for each change you made in the script itself.

2. Leap years

Write a script that determines if a given year is a leap year. Open the file 'leap.py', use the provided structure and complete the script (including implementation and some additional tests).

Also fill in the test_leap_year function with two tests using the assert keyword for the years '1900' (which is not a leap year) and '2000' (which a leap year).

3. Approximating π

Consider the following remarkable formula for approximating the constant π:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$$

The more terms this summation has, the more accurate is the approximation. Write a program that prints an approximation of π using n terms.

4. Perfect numbers

Write a script that determines if a given number is a perfect number.

Also fill in the test_perfect function with two tests using the assert keyword for the numbers '6' (which is a perfect number) and '5541' (which is not a perfect number).

5. Random walker

A robot moves aimlessly, starting at the origin $(0,0)$, and moves one meter in a random direction (up, down, left, right), with probability 25% for each direction. How far will the robot be after n random steps? What is the average distance after performing T such tests?

a. Write a function named 'random_walker' that takes as input a number n and simulates the motion of a random walker for n steps. Print the location at each step, including the starting point $(0,0)$. At the end of the walk, print the square of the final Euclidean distance from the origin.
Reminder: Euclidean distance between 2d points $x = (x_0, x_1)$ and $y = (y_0, y_1)$ is defined as:

$$\left|\left|x - y\right|\right|_2 = \sqrt{(x_0 - y_0)^2 + (x_1 - y_1)^2}$$

b. Write a function named 'random_walker_sim' that takes as input two numbers, n and t. The first input determines the number of steps in each experiment and the second input determines how many times you will run this experiment. For example, if the inputs are 10, 100, you will run 100 experiments, each consisting of 10 random steps. This program prints the average squared distance of the robot.

c. After experimenting with your code, you might be able to formulate a hypothesis regarding the average square distance that the robot makes after taking n random steps.

6. Expand

Write a script that expands a compressed DNA sequence. For example, the string '**G2T11C4A5T1**' should be expanded to '**GGTTTTTTTTTTTCCCCAAAAAT**'. Your script should print the expanded DNA sequence.

Also fill in the test_expand function with a test using the assert keyword for the given DNA sequence.

4. **Read the following submission guidelines carefully:**
    1. This exercise may be submitted in pairs.
    2. All submissions must be zipped and submitted as an archive file with both your IDs as a filename. For example, compress all relevant files using zip and name it '123456789_987654321.zip'.

Good luck.