Lab 1: Control Flow, If Statements, For Loops, and While Loops:

Background:

The Collatz Conjecture is a famous, unsolved problem in number theory. We'll be exploring it today to build some skills with Python's control flow features. The idea is as follows:

- Take a number x and
  - If it is even, divide it by two.
  - If it is odd, triple it and add one.
- Repeat.

Every number we have tried always ends up reaching one. There may be some number out there that ends up stuck in a loop and never reaches one, but so far, we have not found any instances of this. Today we'll build a python program that explores the Collatz Conjecture for various inputs.

Here's an example calculation: 10 → 5 → 16 → 8 → 4 → 2 → 1.

Skills:

First, we'll need to learn a few new things about python syntax:

- Booleans:

```
#Assign a variable:
y = 8675309

myBool = y > 1

print(myBool)
```

```
True
```

- if, elif, else conditions:

```
y = 10

if y < 10:
    print("y is less than ten.")
elif y < 100:
    print("y is less than one hundred but greater than or equal to ten.")
else:
    print("y is 100 or greater")
```

```
y is less than one hundred but greater than or equal to ten.
```

- modular arithmetic:

```
#print the remainder of a division:

print(61 % 12)
```

```
1
```

- for loops – loops through a predetermined number of values.

```
for i in range(0, 10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

- while loop – loops until a condition is no longer met.

```
x = 5
while x > 0:
    print(x)
    x = x - 1

print("the last x value is: ", x)
```

```
5
4
3
2
1
the last x value is:  0
```

Problems:

1. The Collatz Conjecture is essentially a "while" loop. That is, we can tell python: while x is not equal to one, do some operations. Then, if x eventually equals one, the program stops and the number we have chosen does not break the Collatz Conjecture. Write a while loop to perform the calculations for some value x – that is: compute x/2 or 3x+1 depending on if the number x is even or odd respectively and repeat this until x = 1. Print out each number along the way.

2. Before continuing with your program, consider the following: Why are while loops potentially dangerous?

3. It is interesting to ask how many operations are needed for a given choice of x to reach 1 in the Collatz Conjecture process. Create a new variable before your while loop that counts the number of times the while loop is entered. Every time the while loop is entered, add one to the counter.

4. Now let's step it up a notch. Trying one number at a time requires the user to sit around and enter a bunch of numbers one by one to test them. Instead, let's write a for loop to test the numbers 10 through 100 for us all in one go. For each number you start with, print out the original number as well as the number of steps that were required to get the number down to one. This is what is called a "nested loop," because you will have one loop command inside of another loop command. Which number from 10-100 inclusive has the highest number of steps needed to arrive at one (we call this the Stopping Time)?

5. Now let's visualize the result. Modify your for loop to test the numbers 2 through 1000. Create an array of all numbers you start with, and another array that records the number of steps that were required to get the number down to one. To plot, use the following code:

```
plt.plot(x, y, 'r.')
plt.xlabel("This is the horizontal axis")
plt.ylabel("This is the vertical axis")
plt.show()
```