CS 121 – Lab #3 – Practice if statements and while loops

Today's lab consists of two short programs, similar to the exercises we practiced on in class. On your USB drive or account, create a new folder called lab03, and do all of your work in this folder. Open the Spyder Python system.

Part 1: Triangle program

Let's create a program called triangle.py. It will ask the user to enter the lengths of the three sides of a triangle. These will be real numbers, not necessarily integers. The program needs to perform these tasks:

- Store the three input values into the variables first, second and third.
- Determine which side is the longest. Store the longest side into a new variable called c. The other two sides will be stored in variables a and b.
- Determine if the triangle is obtuse, right or acute, and output this answer.
- Determine if the triangle is equilateral, isosceles or scalene, and output this answer.

Here are some definitions to assist you in this program:

Type of triangle	Definition
obtuse	$c^2 > a^2 + b^2$
right	$c^2 = a^2 + b^2$
acute	$c^2 < a^2 + b^2$
equilateral	a, b and c are all equal
isosceles	Among a, b, c, two are equal
scalene	a, b and c are all different

Let's test your program on some input data. Fill in the blanks in the following table. The first three columns are input, and the last two columns are output. Try to find input values for which you get different pairs of output values.

First side	Second side	Third side	Obtuse/right/acute?	Eq./Isos./Scalene?

Part 2: Divisor program

Do you remember how to tell if a number is prime? Let's suppose the number is called n. We test all possible divisors from 1 to n, and count how many of these numbers are actually divisible into n. If we count 2 such divisors, then the number is prime.

For the purpose of this program, let's make a small modification. It turns out that there is no need to test if n is divisible by n. Of course it is. So, let's revise the algorithm: Test just the possible divisors from 1 to n – 1. Count how many divide into n. If we count exactly one, then the number is prime.

It turns out that the divisors less than n are called the *proper divisors* of n.

Create new Python program and call it divisor.py. Begin by typing the following code, which implements our new procedure for determining if n is prime.

```
# divisor.py

n = int(input("Please enter a positive integer:  "))

# Count the number of proper divisors.
count = 0
divisor = 1
while divisor < n:
    if n % divisor == 0:
        count += 1
    divisor += 1

if count == 1:
    print("Prime number")
else:
    print("NOT a prime number")
```

Make the following modifications to this program.

1. In addition to counting the proper divisors, find their sum. Before the loop, create a variable called sum, and set it to zero. Then, inside the loop, find the appropriate place to reset the sum once you have discovered a new divisor. After the loop, print this sum.

2. We are now ready to classify the number n as being abundant, perfect or deficient. Write an if-statement that implements these definitions.

Term	Definition
abundant	The sum of proper divisors is greater than n
perfect	The sum of proper divisors equals n
deficient	The sum of proper divisors is less than n

3. Test your program with 28, 2520 and 2521. Try to find an input number n that is both "not prime" and "deficient."