MODERN DATA Science

Assignment - 1

Question 2

Saurabh Dharmadhikari
Saurabh.dharma01@gmail.com

Table of Content:

Question 2.1:

You are given three sets of lists and each list contains three numbers, you will need to write a code with functionality of inputting the three numbers from each list and find out the answers for below (three) questions. list1 = [1, 3, 5], list2 = [12, 35, 37], list3 = [2, 2, 2.8]

- **Define a function to check whether the given three numbers could form a triangle?**

- **Define a function to check whether the given three numbers could form the right triangle?**

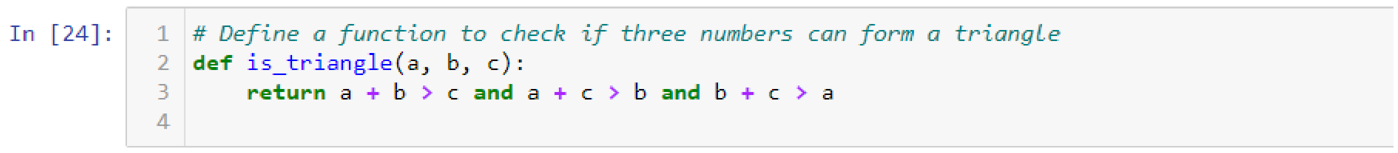
- **Write the code or function to check whether the given three numbers could form the isosceles triangle?**

One possible solution you may try to first define a function to check whether the three input numbers could meet the triangle rule (The sum of the length of the two sides of a triangleis greater than the length of the third side). Then you need to check whether the length of the side could meet the condition of right triangle and isosceles triangle.

Run your code with each list by inputting the numbers and return the results to answer above three questions (you could run your code three times and save the results).

Answer:

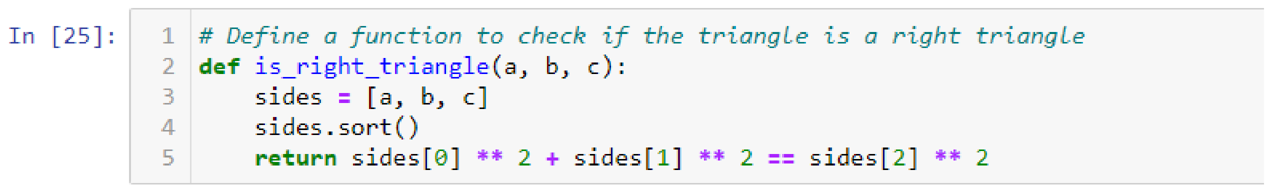
Firstly, we need to address the fundamental question of whether the three provided numbers can indeed form a triangle. In geometry, the Triangle Inequality Theorem states that the sum of the lengths of any two sides of a triangle must be greater than the length of the third side. The code achieves this by defining the is_triangle function. This function evaluates whether the given three numbers (representing the lengths of the sides of a triangle) satisfy the Triangle Inequality Theorem. If they do, it returns True, indicating that a triangle can be formed, otherwise, it returns False.



In [24]:

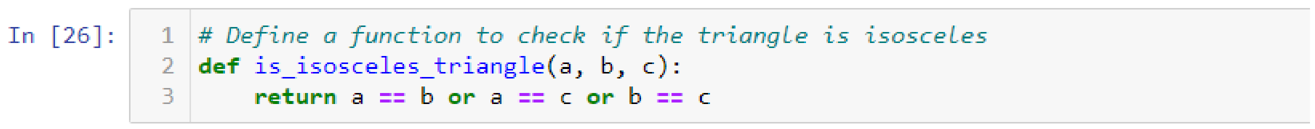
```python
# Define a function to check if three numbers can form a triangle
def is_triangle(a, b, c):
    return a + b > c and a + c > b and b + c > a

```

Next, we investigate whether the triangle is a right triangle. A right triangle is a type of triangle where one of the angles is exactly 90 degrees. To determine this, the code defines the is_right_triangle function. It checks whether the squares of the three side lengths satisfy the Pythagorean Theorem: the sum of the squares of the two shorter sides is equal to the square of the longest side. If this condition holds, the triangle is classified as a right triangle. As shown below:



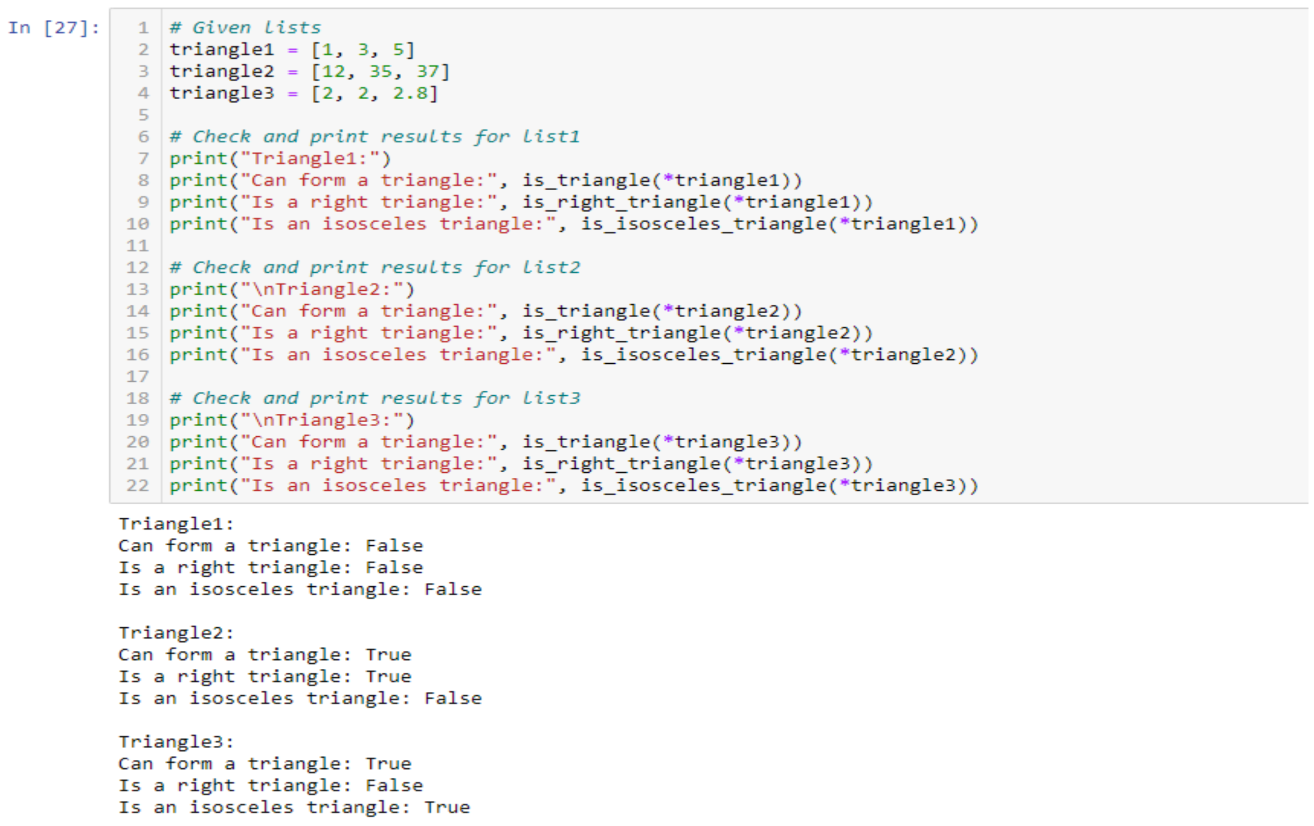
```
In [25]:
# Define a function to check if the triangle is a right triangle
def is_right_triangle(a, b, c):
    sides = [a, b, c]
    sides.sort()
    return sides[0] ** 2 + sides[1] ** 2 == sides[2] ** 2
```

Lastly, we examine whether the triangle is an isosceles triangle. An isosceles triangle is one in which at least two sides have the same length. The is_isosceles_triangle function checks if any two sides of the triangle are equal in length. If so, it identifies the triangle as an isosceles triangle.



```
In [26]:
# Define a function to check if the triangle is isosceles
def is_isosceles_triangle(a, b, c):
    return a == b or a == c or b == c
```

We apply these functions to the lists of length of sides of triangles given to us to check and validate if they actually can form triangle or not or if they are right angled in nature or isosceles.



```
In [27]:
# Given lists
triangle1 = [1, 3, 5]
triangle2 = [12, 35, 37]
triangle3 = [2, 2, 2.8]

# Check and print results for list1
print("Triangle1:")
print("Can form a triangle:", is_triangle(*triangle1))
print("Is a right triangle:", is_right_triangle(*triangle1))
print("Is an isosceles triangle:", is_isosceles_triangle(*triangle1))

# Check and print results for list2
print("\nTriangle2:")
print("Can form a triangle:", is_triangle(*triangle2))
print("Is a right triangle:", is_right_triangle(*triangle2))
print("Is an isosceles triangle:", is_isosceles_triangle(*triangle2))

# Check and print results for list3
print("\nTriangle3:")
print("Can form a triangle:", is_triangle(*triangle3))
print("Is a right triangle:", is_right_triangle(*triangle3))
print("Is an isosceles triangle:", is_isosceles_triangle(*triangle3))
```

```
Triangle1:
Can form a triangle: False
Is a right triangle: False
Is an isosceles triangle: False

Triangle2:
Can form a triangle: True
Is a right triangle: True
Is an isosceles triangle: False

Triangle3:
Can form a triangle: True
Is a right triangle: False
Is an isosceles triangle: True
```

Conclusion:

Thus, from the above results we can see that list of dimensions provided for triangle 1 cannot form a triangle. Triangle 2 is a right-angle triangle and triangle 3 is an isosceles triangle.

Question 2.2:

The problem for Question 2.2 is how to calculate the area of the triangle. When you find out the list(s) in Question 2.1 which could form the triangle, could you also please calculate their area (round to integer)? There is one method to calculate the area of given shape - Heron's formula as below:

$A = s(s - a)(s - b)(s - c)$ (1)

where $s = (a+b+c)/2$

You are required to define the function find_area(a,b,c) for this problem, and you will need to run the find_area() and print the results for the list(s) in Question 2.1 which could form the triangle.

Answer:

The formula is expressed as follows:

A= sqrt[s(s−a)(s−b)(s−c)]

Where:

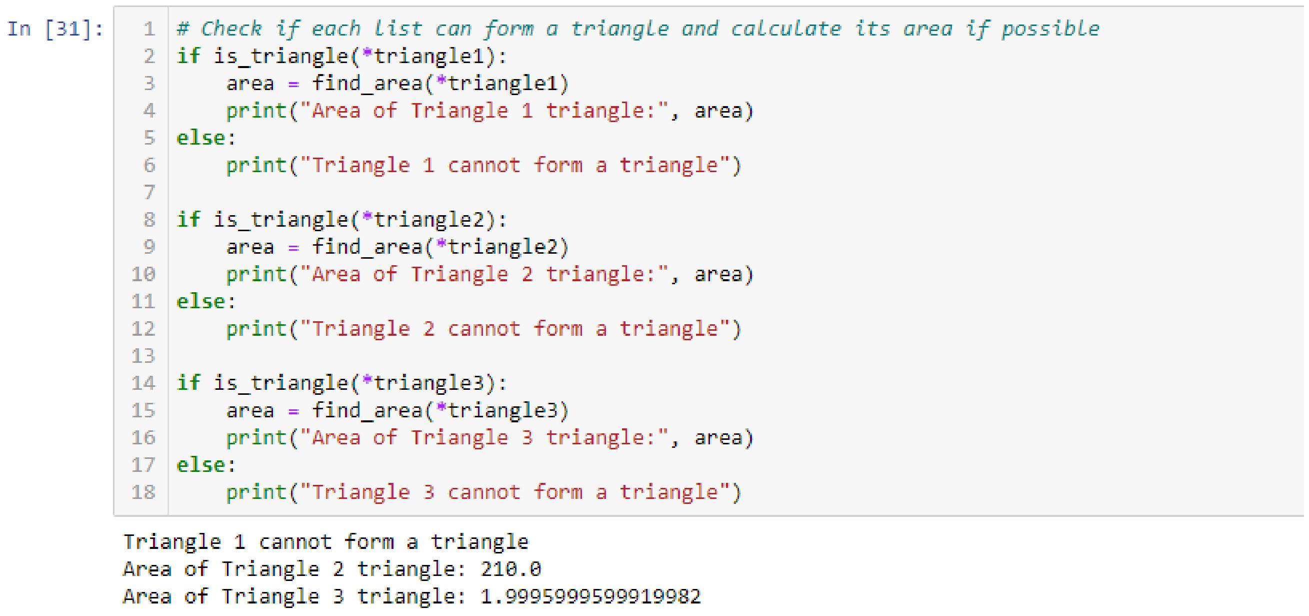
- A is the area of the triangle.
- s is the semi parameter of the triangle, calculated as s = (a+b+c)/2.
- a, b and c are the length of the sides of the triangle.

First, we import math. The code then defines the find_area function, which takes the lengths of the three sides as input and uses Heron's Formula to calculate the area.

```
In [27]:
import math
```

```
In [30]:

# Define a function to calculate the area of a triangle using Heron's formula
def find_area(a, b, c):
    # Calculate the semi-perimeter (s)
    s = (a + b + c) / 2

    # Calculate the area using Heron's formula
    area = math.sqrt(s * (s - a) * (s - b) * (s - c))

    return (area)  # Round the area to an integer
```

The code then applies the find_area function to the triangles that were determined to be valid in Question 2.1. For each valid triangle, the code calculates and prints its area.



```
In [31]:
# Check if each list can form a triangle and calculate its area if possible
if is_triangle(*triangle1):
    area = find_area(*triangle1)
    print("Area of Triangle 1 triangle:", area)
else:
    print("Triangle 1 cannot form a triangle")

if is_triangle(*triangle2):
    area = find_area(*triangle2)
    print("Area of Triangle 2 triangle:", area)
else:
    print("Triangle 2 cannot form a triangle")

if is_triangle(*triangle3):
    area = find_area(*triangle3)
    print("Area of Triangle 3 triangle:", area)
else:
    print("Triangle 3 cannot form a triangle")
```

```
Triangle 1 cannot form a triangle
Area of Triangle 2 triangle: 210.0
Area of Triangle 3 triangle: 1.9995999599919982
```

Conclusion:

- Area for triangle 1 can not be calculated because it can not be formed with the dimensions provided. For Triangle 2 area is 210.
- Triangle 3 area is 1.999. It is very close to 2.

Bibliography

GeeksforGeeks. (2023, May 4). "Python Functions." GeeksforGeeks. https://www.geeksforgeeks.org/python-functions/

BYJU's. (2021, September). "Heron's Formula." BYJU's. https://byjus.com/maths/heron-formula/

GeeksforGeeks. (2022, June 7). "Calculating Areas of Different Shapes Using Python." GeeksforGeeks. https://www.geeksforgeeks.org/calculating-areas-of-different-shapes-using-python/