Deakin University

SIT742 – Mid Term Assessment

Modern Data Science

Task 1

Submitted by

Suraj Mathew Thomas
S223509398
Attempt # 1
Date 17TH September 2023

Target Grade : HD

Task Details - Task 1

Report explaining the solution to 2.1 and 2.2.

Since the answer to question 1.7 turned out to be odd (41), Version I of question 2 is chosen to be attempted and explained.

Answer 2.1

Objective

Since there are 3 lists given in the question where each lists consists of 3 elements, we will need to write a function or piece of code that will work on any list with 3 elements to determine whether

- The given elements in the list, which can be taken as the sides of a triangle, does actually form a triangle.
- If it does form a triangle, does it form a right angled triangle or an isosceles triangle.

Logic & Solution

The logic or formula's that are used to check for a triangle is to see if the sum of two sides is greater than the 3rd side. For example, if the 3 elements are (a, b, c), we can conclude the formation of a triangle only if it meets the below condition.

$$a + b > c \text{ AND}$$
$$a + c > b \text{ AND}$$
$$b + c > a$$

Now if the elements constitute to the formation of a triangle, we need to next see if the triangle that is formed will be a right-angled triangle or an isosceles triangle.

The condition for checking a right-angled triangle is that the sum of the squares of two smaller side should be equal to the square of the third side or the hypotenuse. The condition that it should satisfy is the Pythagoras theorem.

$$a^2 + b^2 = c^2$$

The condition for checking an isosceles triangle is to see if any of the two sides are equal in length.

$$a == b \text{ OR}$$
$$a == c \text{ OR}$$
$$b == c$$

Approaches Used

1) **Single Function** – Only one function is written where through the control statements [If-else], we first check if the given values will form a triangle. If it forms a triangle, we next go on to check with more control statements on the type of triangle, whether it is a right-angled triangle or an isosceles triangle. We then call the function separately for the 2 lists.

```
#One function to check the trianlge formation possibility and the type of triangle.
def check_triangle_formation(a, b, c):
    if a + b > c and a + c > b and c + b > a: #This is the condition to check if a triangle can be formed. The sum of two sides should be greater than the 3rd sid
        print("A Triangle can be formed with",[a,b,c])
        #Nested Control Statements [ If - Else ] to check the trinagle type
        if a**2+b**2==c**2: #Considering a triangle can be formed, the next check is to see if it is a right angled triangle by checking the condition.
          print("A right angled triangle can be formed with",[a,b,c])
        else:
          print("A right angled triangle cannot be formed with",[a,b,c])
        #Nested Control Statements [ If - Else ] to check the trinagle type
        if a==b or a==c or b==c: #This is the condition to check if any two sides are equal then they form an isoceles triangle.
          print("An isoceles triangle can be formed with",[a,b,c])
        else:
          print("An isoceles triangle cannot be formed with",[a,b,c])
    else:
        print("A Triangle cannot be formed with",[a,b,c])

[ ] check_triangle_formation(1,3,5)
    check_triangle_formation(12,35,37)
    check_triangle_formation(2,2,2.8)

    A Triangle cannot be formed with [1, 3, 5]
    A Triangle can be formed with [12, 35, 37]
    A right angled triangle can be formed with [12, 35, 37]
    An isoceles triangle cannot be formed with [12, 35, 37]
    A Triangle can be formed with [2, 2, 2.8]
    A right angled triangle cannot be formed with [2, 2, 2.8]
    An isoceles triangle can be formed with [2, 2, 2.8]
```

2) **Individual Functions** – Another approach is writing individual function. The first function is to check whether the given inputs satisfy the logic of forming a triangle.

The second function checks whether the triangle formed is a right angled triangle. The third function checks whether the triangle formed is an isosceles triangle. Each of the function is individually called.

```
[ ]  # Individual functions
     # 1 - To check whether a triangle can be formed with the given sides
     def check_triangle(x,y,z):
       if x + y > z and x + z > y and z + y > x:
         print("A Triangle can be formed with",[x,y,z])
       else:
         print("Triangle cannot be formed")
```

```
[ ]  check_triangle(1,3,5)
     check_triangle(12,35,37)
     check_triangle(2,2,2.8)
```

```
Triangle cannot be formed
A Triangle can be formed with [12, 35, 37]
A Triangle can be formed with [2, 2, 2.8]
```

```
[ ]  # 2 - To check whether a triangle can be a right angled triangle
     def check_right_triangle(p,q,r):
       if p**2 + q**2 == r**2:
         print("A Right Angled Triangle can be formed with",[p,q,r])
       else:
         print("A Right Angled Triangle cannot be formed with",[p,q,r])
```

```
     check_right_triangle(12,35,37)
     check_right_triangle(2,2,2.8)
```

```
A Right Angled Triangle can be formed with [12, 35, 37]
A Right Angled Triangle cannot be formed with [2, 2, 2.8]
```

```
[ ]  # 3 - To check whether a triangle can be an isoceles triangle
     def check_iso_triangle(e,f,g):
       if e == f or e == g or f == g:
         print("An Isoceles Triangle can be formed with",[e,f,g])
       else:
         print("An Isoceles Triangle cannot be formed with",[e,f,g])
```

```
[ ]  check_iso_triangle(12,35,37)
     check_iso_triangle(2,2,2.8)
```

```
An Isoceles Triangle cannot be formed with [12, 35, 37]
An Isoceles Triangle can be formed with [2, 2, 2.8]
```

3) Class and Functions – Using the concepts of inheritance and polymorphism, we build another approach to solve this by defining the main class (Triangle) and sub-classes like Formation, Right_Tri, Iso_Tri. We call each of these subclasses by using a loop.

```
class Triangle:
    '''Triangle Check.'''
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c
        print('Initialized Triangle Formation Check:')

    def tell(self):
        '''Tell my details.'''
        print('Side 1: %d Side 2: %d Side 3: %d'  % (self.a, self.b, self.c))

class Formation(Triangle):
    '''Checking if a triangle can be formed first.'''
    def __init__(self, a, b, c):
        Triangle.__init__(self, a, b, c)
        if a + b > c and a + c > b and c + b > a:
            print("A Triangle can be formed with",[a,b,c])
        else:
            print("Triangle cannot be formed")

    def tell(self):
        Triangle.tell(self)
        print('Side 1: %d Side 2: %d Side 3: %d'  % (self.a, self.b, self.c))

class Right_Tri(Triangle):
    '''Checking if a right triangle can be formed.'''
    def __init__(self, a, b, c):
        Triangle.__init__(self, a, b, c)
        if a**2 + b**2 == c**2:
          print("A Right Angled Triangle can be formed with",[a,b,c])
        else:
          print("Right Angled Triangle cannot be formed")

    def tell(self):
        Triangle.tell(self)
        print('Side 1: %d Side 2: %d Side 3: %d'  % (self.a, self.b, self.c))

class Iso_Tri(Triangle):
    '''Checking if an isoceles triangle can be formed.'''
    def __init__(self, a, b, c):
        Triangle.__init__(self, a, b, c)
        if a == b or a == c or b == c:
          print("An Isoceles  Triangle can be formed with",[a,b,c])
        else:
          print("An Isoceles Triangle cannot be formed")

    def tell(self):
        Triangle.tell(self)
        print('Side 1: %d Side 2: %d Side 3: %d'  % (self.a, self.b, self.c))
```

```
[ ]  t = Formation(1, 3, 5)
     r = Right_Tri(1, 3, 5)
     i = Iso_Tri(1, 3, 5)
     print('\n')
     members = [t, r, i]
     for member in members:
         member.tell()
```

```
Initialized Triangle Formation Check:
Triangle cannot be formed
Initialized Triangle Formation Check:
Right Angled Triangle cannot be formed
Initialized Triangle Formation Check:
An Isoceles Triangle cannot be formed


Side 1: 1 Side 2: 3 Side 3: 5
Side 1: 1 Side 2: 3 Side 3: 5
Side 1: 1 Side 2: 3 Side 3: 5
Side 1: 1 Side 2: 3 Side 3: 5
Side 1: 1 Side 2: 3 Side 3: 5
Side 1: 1 Side 2: 3 Side 3: 5
```

```
[ ]  t = Formation(12, 35, 37)
     r = Right_Tri(12, 35, 37)
     i = Iso_Tri(12, 35, 37)
     print('\n')
     members = [t, r, i]
     for member in members:
         member.tell()
```

```
Initialized Triangle Formation Check:
A Triangle can be formed with [12, 35, 37]
Initialized Triangle Formation Check:
A Right Angled Triangle can be formed with [12, 35, 37]
Initialized Triangle Formation Check:
An Isoceles Triangle cannot be formed


Side 1: 12 Side 2: 35 Side 3: 37
Side 1: 12 Side 2: 35 Side 3: 37
Side 1: 12 Side 2: 35 Side 3: 37
Side 1: 12 Side 2: 35 Side 3: 37
Side 1: 12 Side 2: 35 Side 3: 37
Side 1: 12 Side 2: 35 Side 3: 37
```

```
[ ]  t = Formation(2, 2, 2.8)
     r = Right_Tri(2, 2, 2.8)
     i = Iso_Tri(2, 2, 2.8)
     print('\n')
     members = [t, r, i]
     for member in members:
         member.tell()
```

```
Initialized Triangle Formation Check:
A Triangle can be formed with [2, 2, 2.8]
Initialized Triangle Formation Check:
Right Angled Triangle cannot be formed
Initialized Triangle Formation Check:
An Isoceles  Triangle can be formed with [2, 2, 2.8]


Side 1: 2 Side 2: 2 Side 3: 2
Side 1: 2 Side 2: 2 Side 3: 2
Side 1: 2 Side 2: 2 Side 3: 2
Side 1: 2 Side 2: 2 Side 3: 2
Side 1: 2 Side 2: 2 Side 3: 2
Side 1: 2 Side 2: 2 Side 3: 2
```

Answer 2.2

Objective

Here the aim is to build a function that will help in the determination of the area of a triangle. We would need to take the lists from 2.1 where we found that it is possible to form and triangle and then find the area of it.

Logic & Solution

The logic that are used to find the area of a triangle is done using the Heron's formula.

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

where

$$s = \frac{a+b+c}{2}$$

Approaches Used

There are two approaches used here. One of the approaches is to use the formula directly without importing any library and the other method is where we invoke the math library.

1) Two functions are defined, one called within another. Here the first function is to calculate the area of the triangle using the Heron's Formula and the second function calls the Heron Area Calculation function to calculate and return the area value that has been calculated.

```
[ ] def Herons_Area_Calc(a, b, c):
        s = (a+b+c) / 2
        return (s*(s-a)*(s-b)*(s-c))**0.5 #alternatively we can import the match function and call the sqrt instead of **0.5


    def find_area(a,b,c):
        Area = Herons_Area_Calc(a, b, c)
        print("The area of a triangle with sides of", [a,b,c], 'is', round(Area,2))

[ ] #The lists that could form a triangle are [12,35,37] and [2,2,2.8]. Insert
    find_area(12, 35, 37)
    find_area(2, 2, 2.8)

    The area of a triangle with sides of [12, 35, 37] is 210.0
    The area of a triangle with sides of [2, 2, 2.8] is 2.0
```

2) This approach uses a single function that calculates and returns the area. The difference between the above approach and this one is the import math library. Here the sqrt from the math is used for the formula.

```
import math

def Area(a,b,c):
  p=a+b+c
  s=(p)/2
  A = math.sqrt((s*(s-a)*(s-b)*(s-c)))
  return A

[ ] Area(2,2,2.8)

    1.9995999599919982

[ ] Area(12,35,37)

    210.0
```

References

Videos

1) Dr. Angela Yu (2023) '100 Days of Code: The Complete Python Pro Bootcamp for 2023, Udemy [e-Learning] course, Section 5 and Section 6 – Python Loops and Python Functions

Course Material

2) Deakin Study Material (n.d) 'Week 1 Python Foundations & Big Data ', Modern Data Science Content SIT742, Deakin University
3) Deakin Study Material (n.d) 'Week 1 Reference Material - Python Foundations & Big Data ', Modern Data Science Content SIT742, Deakin University