Triangle Classifier

Lab 7

Jackson de Gruiter, jtd431
Dakota Hester, dh2306

"On my honor, as a Mississippi State University student, I have neither given nor received unauthorized assistance on this academic work."

CSE 1284 Introduction to Computer Programming

Lecture Section Time: MWF 10am
Instructor: Monika Jankun-Kelly

Lab Section: Friday
Lab Assistant: Tyler Narmore

Problem Statement

This program should classify a triangle as equilateral, isosceles, scalene, right and isosceles, or right and scalene based on the lengths of the sides inputted.

Program Design

import square root function from math module

set variable ‘epsilon’ to 10^-14

function:	isEquilateral
Parameters:	a, b, c
Returns:	True or False
Purpose:	Find out if triangle is equilateral

Algorithm:

- return ((side A is equivalent to side B) and (side B is equivalent to side C))

function:	isIsosceles
parameters:	a, b, c
returns:	True or False
Purpose:	Find out if triangle is isosceles

Algorithm:

- put side lengths in list
- sort list by value from least to greatest
- return ((lowest side value is equivalent to second lowest side value) and (lowest side value is not equivalent to greatest side value))

function:	isRight
parameters:	a, b, c

returns: True or False

Purpose: Determine if triangle is right and scalene or right and isosceles

Algorithm:

- put side lengths in list
- sort list by value from least to greatest
- find out if the square of both shorter legs adds up to the square of the hypotenuse (or is at least very very close to it) with the following line...
- return (the absolute value of (the shortest side squared + the second shortest value squared – the hypotenuse squared) is less than variable ‘epsilon’

Function: triangleClassifier

parameters: a, b, c

returns: ‘equilateral’, ‘isosceles’, ‘right and isosceles’, ‘right and scalene’, or ‘scalene’

Purpose: Determines the category of triangle that the side lengths create

algorithm:

- if function isEquilateral(side A, side B, side C) returns True:
    - return ‘equilateral’
- else if function isRight(side A, side B, side C) returns True:
    - if function isIsosceles(side A, side B, side C) returns True:
        - return ‘right and isosceles’
    - else:
        - return ‘right and scalene’
- else if function isIsosceles(side A, side B, side C) returns True:
    - return ‘isosceles’
- else:
    - return ‘scalene’

# The following code runs after all of the functions have been created

- user inputs side A
- user inputs side B
- user inputs side C
- print triangleClassifier(side A, side B, side C)

Testing

isEquilateral Function

Parameters	Expected Return	Actual Return	Test Passed?
3, 3, 3	True	True	Yes
3, 4, 3	False	False	Yes
1.00000000001, 1, 1	False	False	Yes
0, 0, 0	True	True	Yes

isIsosceles Function

Parameters	Expected Return	Actual Return	Test Passed?
3, 4, 3	True	True	Yes
2, 2, 2	False	False	Yes
1, 1, 1	False	False	Yes
1, 2, 2	True	True	Yes

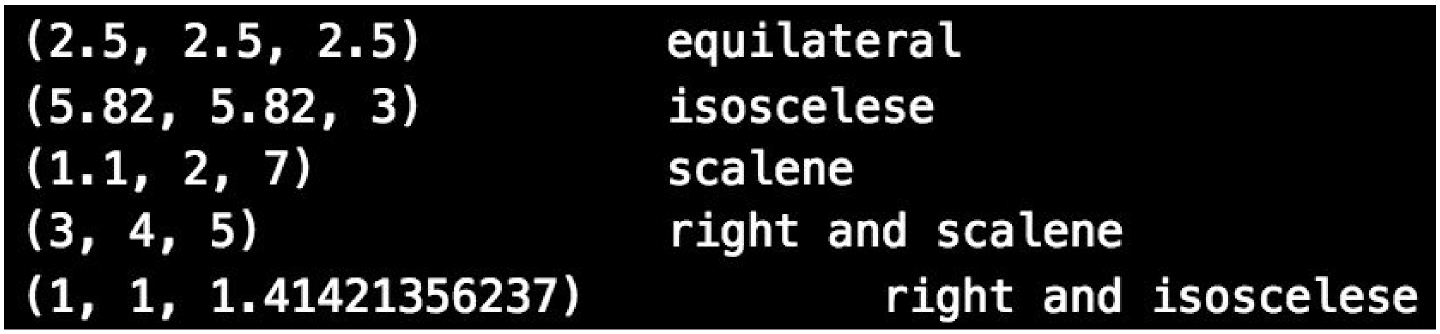
isRight function

Parameters	Expected Return	Actual Return	Test Passed?
1, √2, 1	True	True	Yes
3, 4, 5	True	True	Yes
1, 2, 2	False	False	Yes
2, 2, 2	False	False	Yes

triangleClassifier function

Parameters	Expected Return	Actual Return	Test Passed?
2.5, 2.5, 2.5	“equilateral”	“equilateral”	Yes
5.82, 5.82, 3	“isosceles”	“isosceles”	Yes
1.1, 2, 7	“scalene”	“scalene”	Yes
3, 4, 5	“right and scalene”	“right and scalene”	Yes
1, 1, √2	“right and isosceles”	“right and isosceles”	Yes

Execution Screenshot



```
(2.5, 2.5, 2.5)         equilateral
(5.82, 5.82, 3)         isoscelese
(1.1, 2, 7)             scalene
(3, 4, 5)               right and scalene
(1, 1, 1.41421356237)          right and isoscelese
```

Analysis and Conclusions

1. What was your most useful reference for this assignment?

We found that the most useful reference was the textbook. The chapter on functions has quite a bit of material that helped us through this

2. What test case do you have where you ran into a problem with a == b not giving the expected result? how did you solve this problem? [hint: when you do floating point math, it is not precise]

While testing our right triangle we discovered that triangle (1, 1, $\sqrt{2}$) was not considered a right triangle. The reason for this was that floating point values behave in ways that we didn't expect them to. The fix for this was returning true when the absolute value of the triangles hypotenuse squared was subtracted from each of the legs squared was less than a very small number (epsilon)

3. What did you do that best helped you prepare for this assignment?

Everything we needed to do this lab was explained in class, no preparation other than that was needed.

Code Appendix

```
from math import sqrt

################################################################################
# name:         isEquilateral
# parameters:   sideA, sideB, sideC  --- triangle side lengths
# returnValues: boolean - True if triangle is equilateral
# purpose:       Checks if triangle with given sides a,b,c is equilateral
def isEquilateral(sideA, sideB, sideC):
    # return whether or not all of the side lengths are equal
    return sideA == sideB and sideB == sideC

################################################################################
# name:         isIsosceles
# parameters:   sideA, sideB, sideC  --- triangle side lengths
# returnValues: boolean - True if triangle is isoscelese
# purpose:       Checks if triangle with given sides a,b,c is isoscelese
def isIsosceles(sideA, sideB, sideC):
    # create a list of sides from the parameters
    sides = [sideA, sideB, sideC]

    # sort the sides to make them easier to work with
    sides.sort()

    # return whether or not two of the sides are equal, but not the third
    return((sideA == sideB and sideA != sideC) or (sideA==sideC and sideA != 
sideB)) or (sideB == sideC and sideB != sideA )
```

```
################################################################################
# name:         isRight
# parameters:   sideA, sideB, sideC  --- triangle side lengths
# returnValues: boolean - True if triangle is right
# purpose:      Checks if triangle with given sides a,b,c is right
def isRight(sideA, sideB, sideC):
    # create a list of sides from the parameters
    sides = [sideA, sideB, sideC]

    # sort the side lengths to make them easier to work with
    sides.sort()

    # return whether or not leg1^2 + leg2^2 is almost equal to hypotenuse^2
    return abs(sides[0]**2 + sides[1]**2 - sides[2]**2) < epsilon

################################################################################
# name:         triangleClassifier
# parameters:   sideA, sideB, sideC  --- triangle side lengths
# returnValues: string --- type(s) of triangle (equilateral, right, isoscelese,
or scalene)
# purpose:      returns the type of triange given sides a,b,c
def triangleClassifier(sideA, sideB, sideC):
    # Is the triangle equlateral?
    if isEquilateral(sideA, sideB, sideC):
        return 'equilateral'

    # Is the triangle a right triangle? If so, what kind?
    elif isRight(sideA, sideB, sideC):
        if isIsosceles(sideA, sideB, sideC):
            return 'right and isoscelese'
        else:
            return 'right and scalene'

    # Is the triangle an Isosceles
    elif isIsosceles(sideA, sideB, sideC):
        return 'isoscelese'
    else:
        return 'scalene'

################################################################################
# All of our functions are now defined, below is the main program



# use this number as our machine epsilon
epsilon = 10 ** -14



# Below we create a list of triangle length lists
triangles = [
            [2.5, 2.5, 2.5],
            [5.82, 5.82, 3],
            [1.1, 2, 7],
            [3, 4, 5],
            [1, 1, sqrt(2)]]



# for each triangle defined above, print the side lengths and the classification

for triangle in range(len(triangles)): # for each triangle in the triangles list

    # The sides variable is a list of the side lengths in the current triangle
    sides = triangles[triangle]
    # Output should look something like this ---> (1, 2, 3)        scalene

    print("(" + str(sides[0]) + ", "+ str(sides[1]) + ", "+ str(sides[2])+ ")" +
"\t\t" + triangleClassifier(sides[0], sides[1], sides[2]))
```