3. WRITTEN RESPONSES

3 a.

3.a.i.

The overall purpose of the program is to provide information about a triangle based on inputted side lengths.

3.a.ii.

The video demonstrates the right triangle identification functionality of the code, which includes the display of a ratio of side lengths and a table of trigonometric values.

3.a.iii.

The video demonstrates user inputs of the side lengths and program outputs of side lengths, classification by side lengths, angle measurements, classification by angle measurements, and trigonometric values of each angle.

3 b.

3.b.i.

```
47      sideIndex = ['A:B:C = ' + sideRatio, 'Equilateral', 'Isosceles', 'Scalene']
```

3.b.ii.

```
48      if (A**2 + B**2 == C**2) or(C**2 + B**2 == A**2) or (A**2 + C**2 == B**2) :
49          return 'Classification by Side: ' + sideIndex [0]
50      if (A==B) and (B==C):
51          return 'Classification by Side: ' + sideIndex [1]
52      elif (A==B) or (B==C) or (A==C):
53          return 'Classification by Side: ' + sideIndex [2]
54      else:
55          return 'Classification by Side: ' + sideIndex [3]
```

3.b.iii.

The list is named sideIndex.

3.b.iv.

The data in the list contains the possible classifications by side lengths of the triangle.

3.b.v.

If sideIndex was not used, the return statements would include a classification along with the 'Classification by Side: '. Without sideIndex, line 53, for example, would appear as: return 'Classification by Side: Isosceles'. This would be more difficult to manage than a list because in order to change an output, the coder would have to go to the exact place where that specific classification is returned, rather than being able to access all the classifications from a single list. sideIndex allows all of the possible classifications to be easily accessible in one place

3 c.

3.c.i.

```
30  def ratioCalculate (A, B, C):
31      intA = int (A)
32      intB = int (B)
33      intC = int (C)
34      if ((math.gcd (intA, intB) == 1) or (math.gcd (intC, intB) == 1) or (math.gcd (intA, intC) == 1)) and (intA == A) and (intB == B) and (intC == C):
35          return str (intA) + ':' + str (intB) + ':' + str (intC)
36      constant = 10**((max (len (str (A)), len (str(B)), len (str(C))))-2)
37      A = int (constant*A)
38      B = int (constant*B)
39      C = int (constant*C)
40      for i in range (min(A, B, C), 0, -1):
41          remainders = str (A%i) + ':' + str (B%i) + ':' + str (C%i)
42          if remainders == '0:0:0' :
43              return str (int(A/i)) + ':' + str (int(B/i)) + ':' + str (int(C/i))
```

3.c.ii.

```
46      sideRatio = ratioCalculate (A, B, C)
```

3.c.iii.

ratioCalculate takes in three inputs and returns a ratio of whole numbers which corresponds to the ratio of the inputs in the order of A:B:C. The function is used to calculate the ratio of the side lengths of the triangle and is used in storing a value in sideRatio, which is used in the list sideIndex. If the triangle is a right triangle, the item in the list sideIndex which includes the sideRatio value is displayed.

3.c.iv.

ratioCalculate is defined as a function with inputs A, B, and C.

- intA equals the integer version of A
- intB equals the integer version of B
- intC equals the integer version of C

If the greatest common denominator of intA and intB, intC and intB, or intA and intC is equal to 1 and intA equals

- A and intB equals B and intC equals C,

the function returns a concatenation by +'s of the string version of intA, ":", the string version of intB, ":", and

- the string version of intC.

After the conditional, a variable, constant, equals 10 to the power of the quantity of 2 less than the maximum of the

- length of the string version of A, the length of the string version of B, and the length of the string version of C.
- A equals the integer version of the quantity constant times A.
- B equals the integer version of the quantity constant times B.
- C equals the integer version of the quantity constant times C.
- An iterative loop for variable i decreases by 1 from the maximum of A, B, and C to 0 but not including 0.

Within this loop, a remainders variable equals a concatenation by +'s of the string version of the quantity A

- mod i, ":", the string version of the quantity B mod i, ":", and the string version of the quantity C mod i.
- Within the same loop, a conditional checks if remainders equals "0:0:0".

If the condition is satisfied, the function returns a concatenation by +'s of the string version of the quantity A divided by i, ":", the string version of the quantity B divided by i, ":", and the string version

- of the quantity C divided by i.

3 d.

3.d.i.

First call:

ratioCalculate (3.0, 4.0, 5.0)

Second call:

ratioCalculate (1.2, 2.3, 3.4)

3 d.ii.

Condition(s) tested by first call:

((math.gcd (intA, intB) == 1) or (math.gcd (intC, intB) == 1) or (math.gcd (intA, intC) == 1)) and (intA == A) and (intB == B) and (intC == C)

is tested. This checks if the integer values of the inputs are equal to their actual values and can not be further simplified.

Condition(s) tested by second call:

((math.gcd (intA, intB) == 1) or (math.gcd (intC, intB) == 1) or (math.gcd (intA, intC) == 1)) and (intA == A) and (intB == B) and (intC == C) is tested.

This checks if the integer values of the inputs are equal to their actual values and can not be further simplified.

remainders == "0:0:0" is tested. This checks if A, B, and C can be divided by a certain value of i and not leave any remainders.

3.d.iii.

Results of the first call:

"3:4:5"

Results of the second call:
“12:23:34”