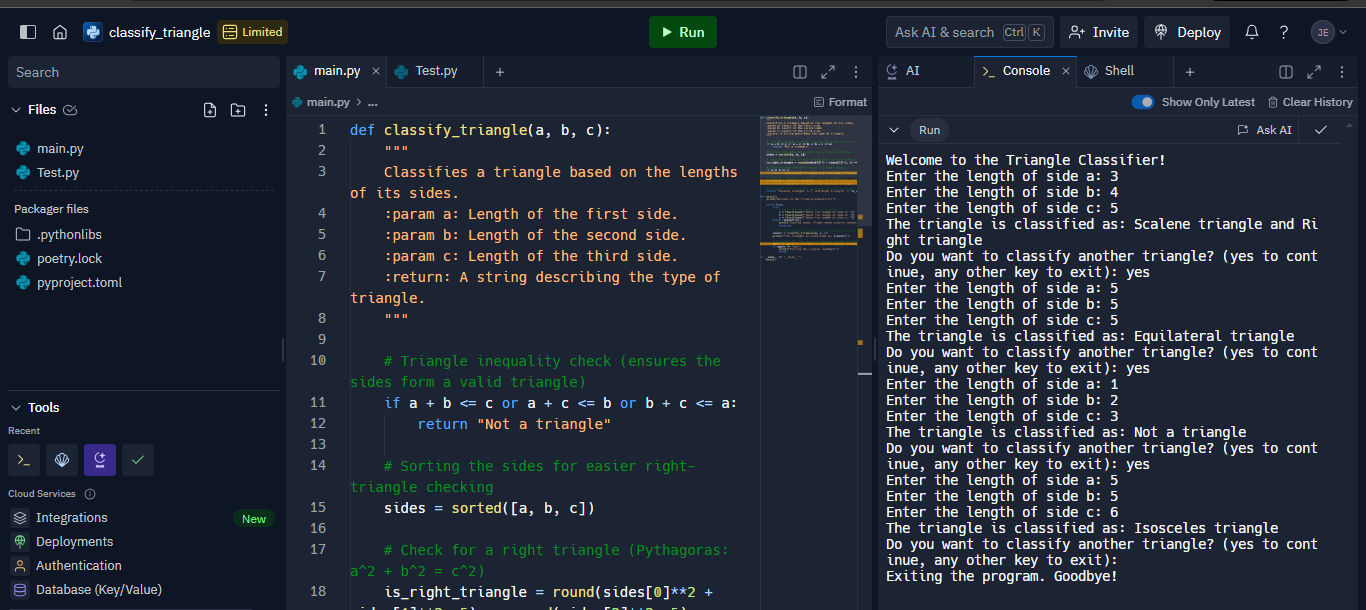
**HW 01: Testing triangle classification**

* **equilateral** triangles have all three sides with the same length
* **isosceles** triangles have two sides with the same length
* **scalene** triangles have three sides with different lengths
* **right** triangles have three sides with lengths, a, b, and c where a2+ b2= c2

Your assignment is to write a program in Python to classify triangles and use an automated test platform, e.g. UnitTest or PyTest, and write test cases to test your implementation of classifying triangles.  The goal is for you to gain experience using automated test tools and to think through the issues associated with testing a "system".

**Python code for Classification of Triangle output:**



**Unit Testing for the Classification of Triangle:**

* **Test Case: test\_equilateral**

Description: Tests if classify\_triangle correctly identifies an equilateral triangle (all sides equal).

Example Inputs: (3, 3, 3)

Expected Output: "Equilateral triangle"

* **Test Case: test\_isosceles**

Description: Tests if classify\_triangle correctly identifies an isosceles triangle (two sides equal).

Example Inputs: (5, 5, 8)

Expected Output: "Isosceles triangle"

* **Test Case: test\_scalene**

Description: Tests if classify\_triangle correctly identifies a scalene triangle (no sides equal).

Example Inputs: (4, 5, 6)

Expected Output: "Scalene triangle"

* **Test Case: test\_right\_scalene**

Description: Tests if classify\_triangle correctly identifies a right scalene triangle (Pythagorean theorem satisfied).

Example Inputs: (3, 4, 5)

Expected Output: "Scalene triangle and Right triangle"

* **Test Case: test\_right\_isosceles**

Description: Tests if classify\_triangle correctly identifies a right isosceles triangle (two equal sides and Pythagorean theorem satisfied).

Example Inputs: (1, 1, math.sqrt(2))

Expected Output: "Isosceles triangle and Right triangle"

* **Test Case: test\_not\_a\_triangle**

Description: Tests if classify\_triangle correctly identifies an invalid triangle where the sum of two sides is not greater than the third.

Example Inputs: (1, 2, 3)

Expected Output: "Not a triangle"

* **Test Case: test\_zero\_length\_sides**

Description: Tests if classify\_triangle correctly identifies a triangle with zero length sides as invalid.

Example Inputs: (0, 0, 0)

Expected Output: "Not a triangle"

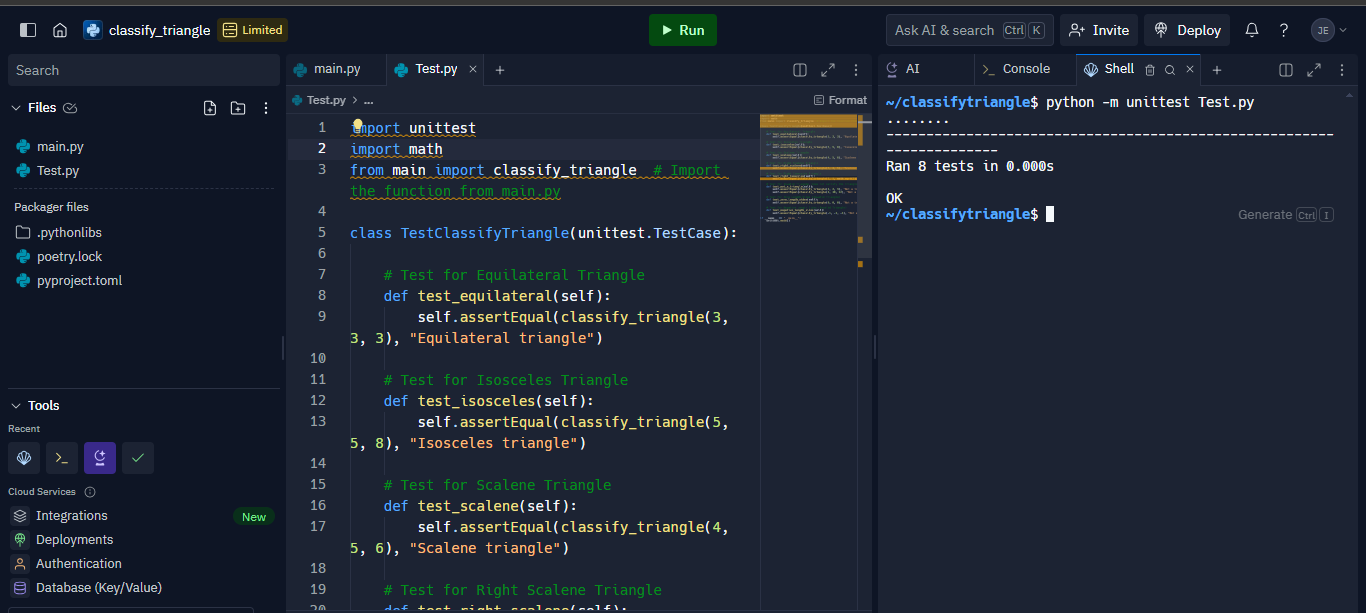
* **Test Case: test\_negative\_length\_sides**

Description: Tests if classify\_triangle correctly identifies a triangle with negative length sides as invalid.

Example Inputs: (-1, -1, -1)

Expected Output: "Not a triangle"

**Unit Testing Output:**

****

**Describe your experience with this assignment, specifically:**

1. What challenges did you encounter with this assignment, if any?

One of the biggest challenges I faced with this task was ensuring that the test cases were straightforward and clear while still covering a range of scenarios. Initially it was tempting to focus on the types of triangles equilateral, isosceles and scalene. However I quickly realized the importance of edge cases such as sides with zero length, sides with negative lengths and invalid triangles that don’t meet the triangle inequality. Another challenge was dealing with accuracy, when working with right angled triangles especially those involving square roots like isosceles right triangles.

1. What did you think about the requirements specification for this assignment?

The main goal of the requirements document which aimed to classify different types of triangles was clearly communicated. I would have appreciated more guidance on the expected handling of edge cases such as invalid or degenerate triangles where the sum of two sides equals the third. Overall the project allowed room for creativity, especially when it came to designing the test scenarios.

1. What challenges did you encounter with the tools?

Using unit test was pretty easy but I did run into some difficulties when it came to comparing points specifically for right triangles with square roots. In those situations it was tough to sidestep precision problems unless I used Pythons math.isclose() function. Exporting test case data to a Word document using libraries was a great learning opportunity even though it took some trial and error to get the formatting just right, particularly when differentiating between table and non-table formats.

1. Describe the criteria you used to determine that you had sufficient test cases, i.e. how did you know you were done?

I checked a few things to make sure I had enough test cases:

* Key features: I made sure to cover the three main triangle types equilateral, isosceles, and scalene. To ensure reliability,
* I included edge cases like invalid triangles (when the sides don't form a real triangle) sides with zero length, and sides with negative length.
* Right triangles: I checked that the program could handle situations with right triangles, both scalene and isosceles.
* In the end, I'd dealt with all common triangle types, handled edge cases well, and made sure the program did what it should with both normal and unusual inputs.