Tutorial 2

Swinburne University of Technology

Software Testing and Reliability (SWE30009)

Semester 2, 2022

Lecturer: Prof Tsong Yueh Chen

Tutor: Dr Hung Q Luu

Acknowledgement of Country

We respectfully acknowledge the Wurundjeri People of the Kulin Nation, who are the Traditional Owners of the land on which Swinburne's Australian campuses are located in Melbourne's east and outer-east, and pay our respect to their Elders past, present and emerging.

We are honoured to recognise our connection to Wurundjeri Country, history, culture, and spirituality through these locations, and strive to ensure that we operate in a manner that respects and honours the Elders and Ancestors of these lands.

We also respectfully acknowledge Swinburne's Aboriginal and Torres Strait Islander staff, students, alumni, partners and visitors.

We also acknowledge and respect the Traditional Owners of lands across Australia, their Elders, Ancestors, cultures, and heritage, and recognise the continuing sovereignties of all Aboriginal and Torres Strait Islander Nations.

Part A

Assignment 1

Important notices

- This is an individual assignment.
- Worth 20% of the total unit score.
- Due date: 11:00pm, Monday, 21 August 2023.

Program under test

```
Input A, B // A and B are real variables
A = A - B
C = A * 2
Output C  // C is a real variable
```

Only four feasible arithmetic operators:

+ (addition),
- (subtraction),
* (multiplication)
/ (division)

Testing objective

- Detect ANY **possible incorrect use** of arithmetic operators.

Example of incorrect program

Expected program

```
Input A, B // A and B are real variables
A = A - B
C = A * 2
Output C  // C is a real variable
```

Incorrect program

```
Input A, B // A and B are real variables
A = A + B
C = A * 2
Output C  // C is a real variable
```

Task 1

Explain and show all details on

how to design the test cases

for the above testing objective.

Task 2

Suppose you use test case (A=3, B=1) to test the above program.

- Is this test case able to **achieve the required testing objective**?
- Provide your answer with justifications.

Task 3

From your design in Task 1

- What is (or are) the **concrete test case (or cases)** that can achieve the above testing objective?
- Explain and justify your concrete test case (or cases).

Task 4

Given B=1

- Find **all possible values of A** so that the concrete test cases (A,B) **cannot** achieve the above testing objective?
- Explain and justify the correctness of your solution.

Requirement about report

- Must be self-contained and complete.
- Must use 12-point font size on A4 papers.
- Must have no more than 3 pages.
- Must contain full name & student number on the first line of first page.
- Coverage page is not required.

Requirement about submission

- Must use the submission for Assignment in the unit's Canvas.
- Must submit the Assignment 1 report as a single PDF file.
- Must have the filename specified in this format "Assignment1-YourStudentID-YourSurname.pdf"
  - Example: Assignment1-12345678-Nash.pdf
- Submission must be before the due date

Marking criteria

- A maximum of **40 marks for Task 1**.
- A maximum of **10 marks for Task 2**.
- A maximum of **20 marks for Task 3**.
- A maximum of **20 marks for Task 2**.
- A maximum of **10 marks is for presentation**, structure, comprehensibility and completeness of report, as well as compliance with requirements.

Marking penalty

- Penalty will be applied for late submission and plagiarism.
- Refer to the Unit Outline for the policy on late submission and plagiarism.

Hints

You may want to refer to “Lecture 2” for all tasks.

For Task 3, you can use the trial method and may construct one or more than one concrete test cases.

For Task 4, you can develop a program to find them automatically, or prove the exhaustiveness of test cases.

Guideline

Start assignments as early as possible – but only after Lecture 2.

Make sure you fully understand Lecture 2 before doing the assignment.

Part B

Practice exercises in Week 1

Test objectives

- The aim or purpose of testing

Testing objectives

Consider the following program:

```
Input A, B // A and B are integer variables
C = (A – B) * B
Output C
```

Suppose the following testing objectives are applied:

1. One and only one incorrect arithmetic operator
2. Incorrect use of arithmetic operators

Task

Question 1:

What are the constraints for test cases in order to achieve testing objectives (1) and (2)?

Question 2:

How do you construct your concrete test cases from the constraints of test cases for testing objectives (1) and (2)?

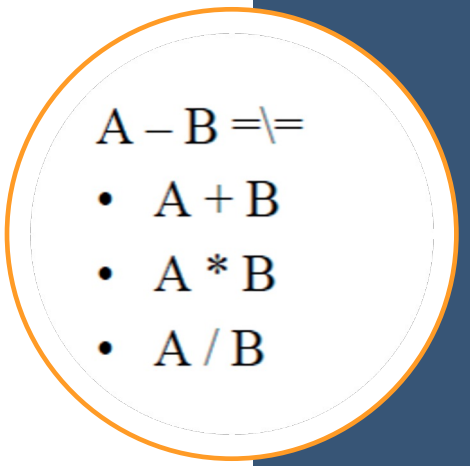
Testing objectives

- Frankly speaking, we seek to create one or multiple test cases that can help distinguish **“good” program** (correctly coded) with all **“bad” programs** (buggy or incorrectly implemented).

- Solution
  - Easy: Trial and error
  - Medium: Automated search
  - Hard: Constraint solving

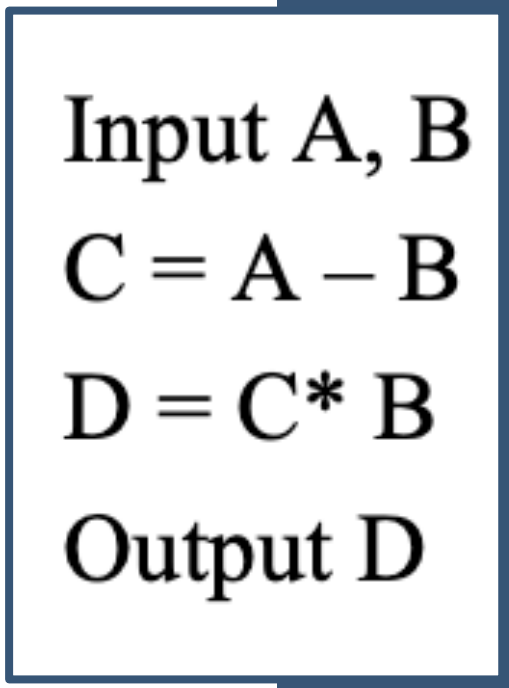
Which test case is better?

- Test case 1: A = 4, B = 2
- Test case 2: A = 3, B = 1
- Test case 3: A = -1, B = 1
- Test case 4: A = 1, B = 1



Which test case is better?

- Test case 1: A = 4, B = 2
- Test case 2: A = 3, B = 1



Hint

- D = (A + B) * B
- D = (A / B) * B
- D = (A * B) * B
- D = (A - B) + B
- D = (A - B) - B
- D = (A - B) / B
- …

D = (A – B) * B

Input A, B

C = A – B

D = C* B

Output D

Part C

Testing triangle program

Program under test: triangle.py

- Reads in 3 integers **a,b,c** as the lengths of a triangle

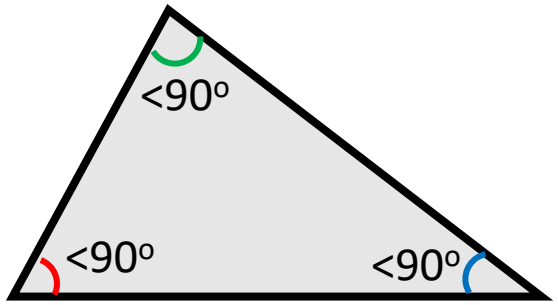
- Output the **text** displaying the type of triangle (or not)
  - Right-angled triangle
  - Isosceles triangle
  - Equilateral triangle
  - Scalene triangle
  - Not a triangle

Triangles

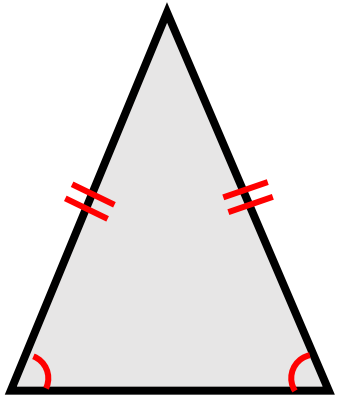
Source:
https://www.architectureanddesign.com.au/features/list/louvre-pyramid-im-pei

Source:
https://en.wikipedia.org/wiki/Egyptian_pyramids#/media/File:All_Gizah_Pyramids.jpg

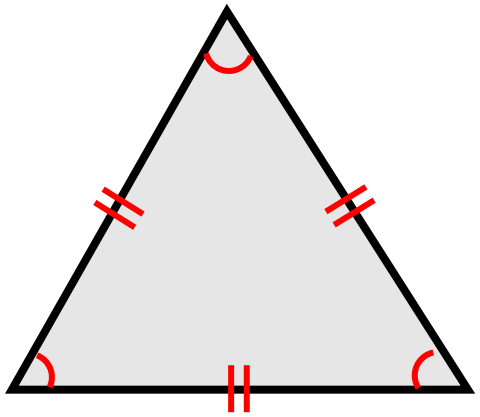
Triangle types



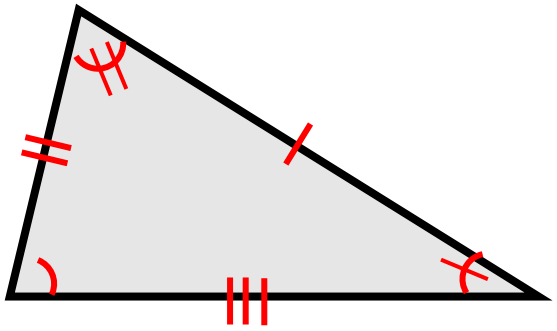
acute triangle



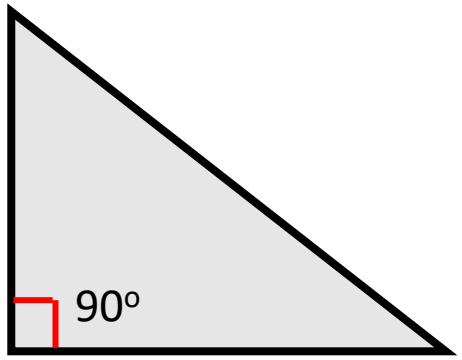
isosceles triangle



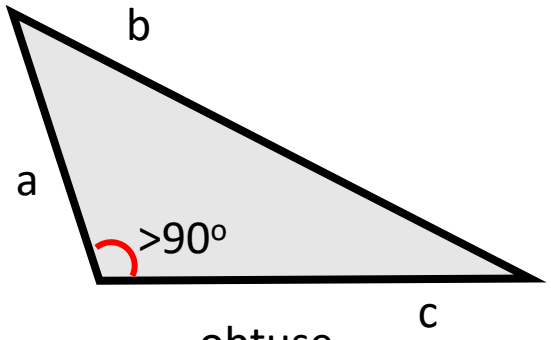
equilateral triangle



scalene triangle



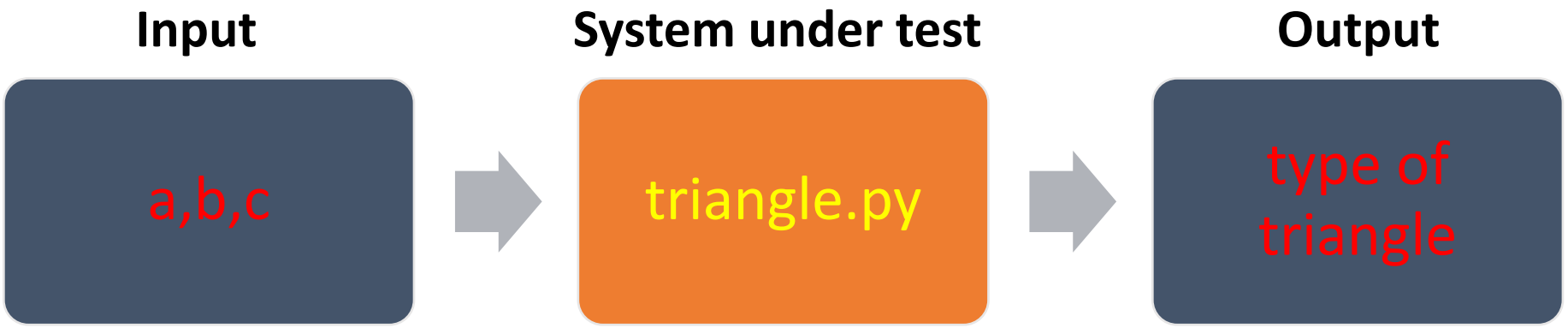
right triangle



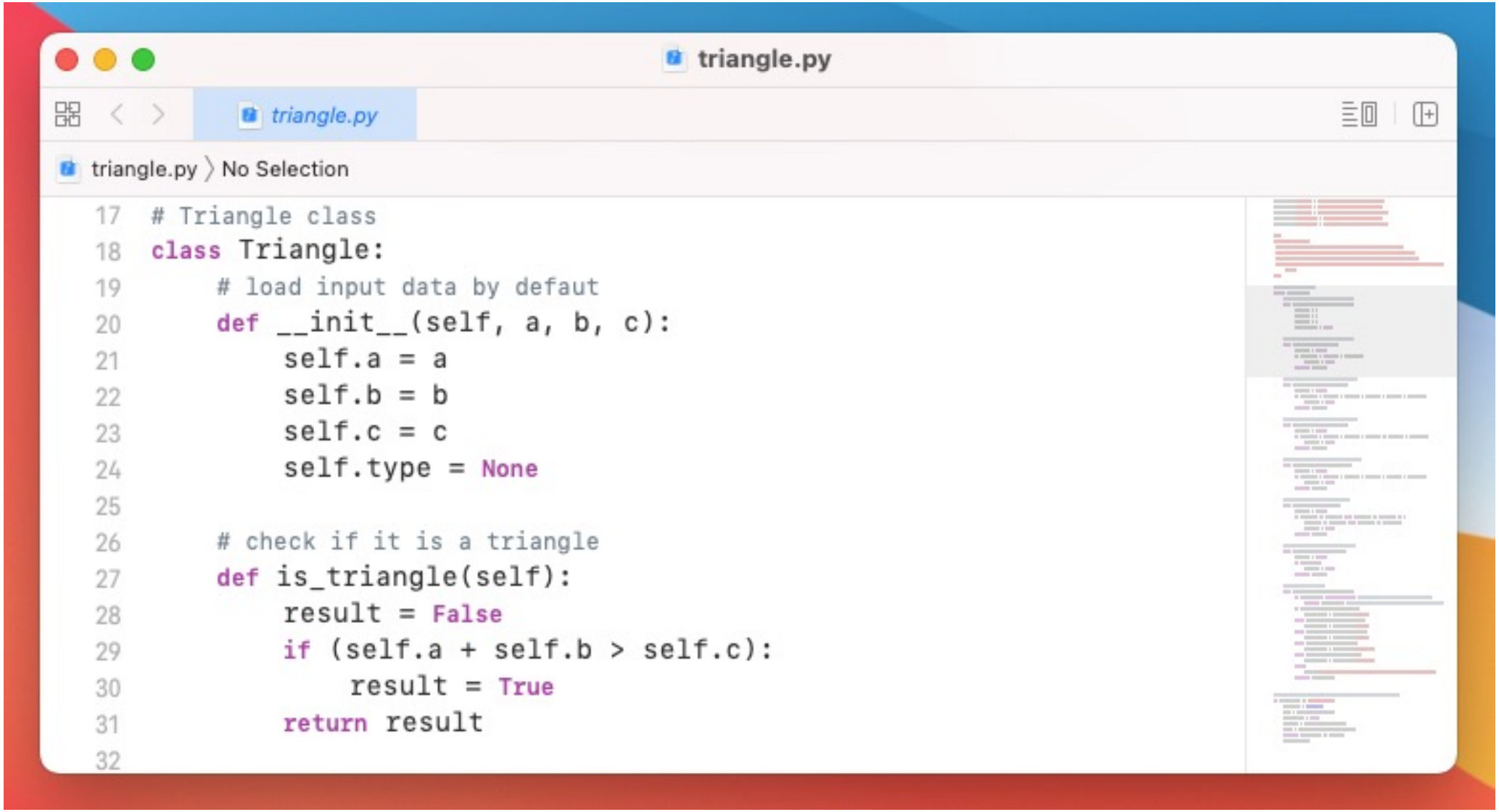
obtuse triangle

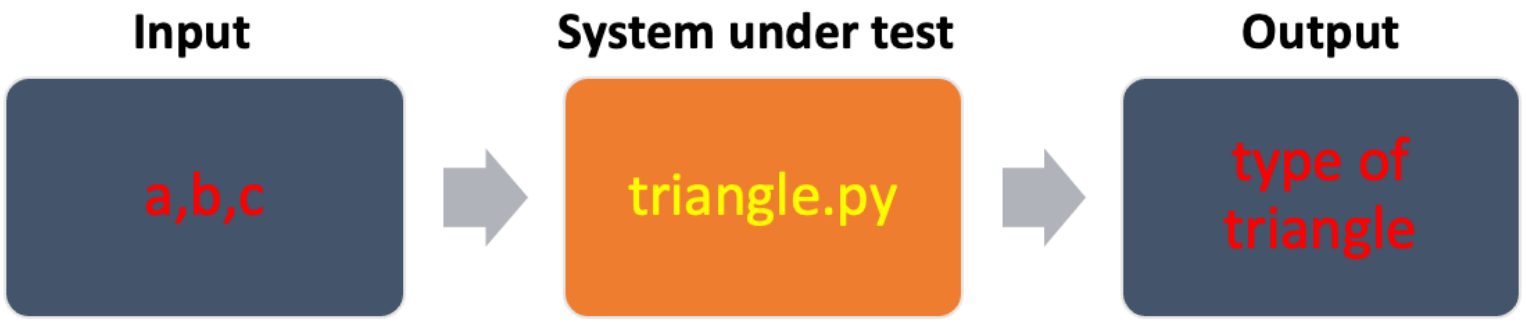
pythagoras theorem: $a^2 + b^2 = c^2$

System under test



System under test






```
expected = "right-angled triangle"
```

Discussion

- Valid test cases

<< Our focus in this unit

- Invalid test cases

Design test cases

Test ID	Input Data			Expected Output	Reason for the test
	a	b	c		
1	3	4	9	NaT	Test the situation when a triangle cannot be formed
2	3	4	5	RAT	Test the situation when a right-angled triangle can be formed
3	3	4	6	OAT	Test the situation when a obtuse-angled triangle can be formed
4	-3	4	5	InvInp	Test the situation when invalid input is entered
5	4	4	4	EquiT	Test the situation when a equilateral triangle can be formed
6	12	14	16	AAT	Test the situation when a acute-angled triangle can be formed
7	3	3	7	NaT	Test the situation when a triangle cannot be formed

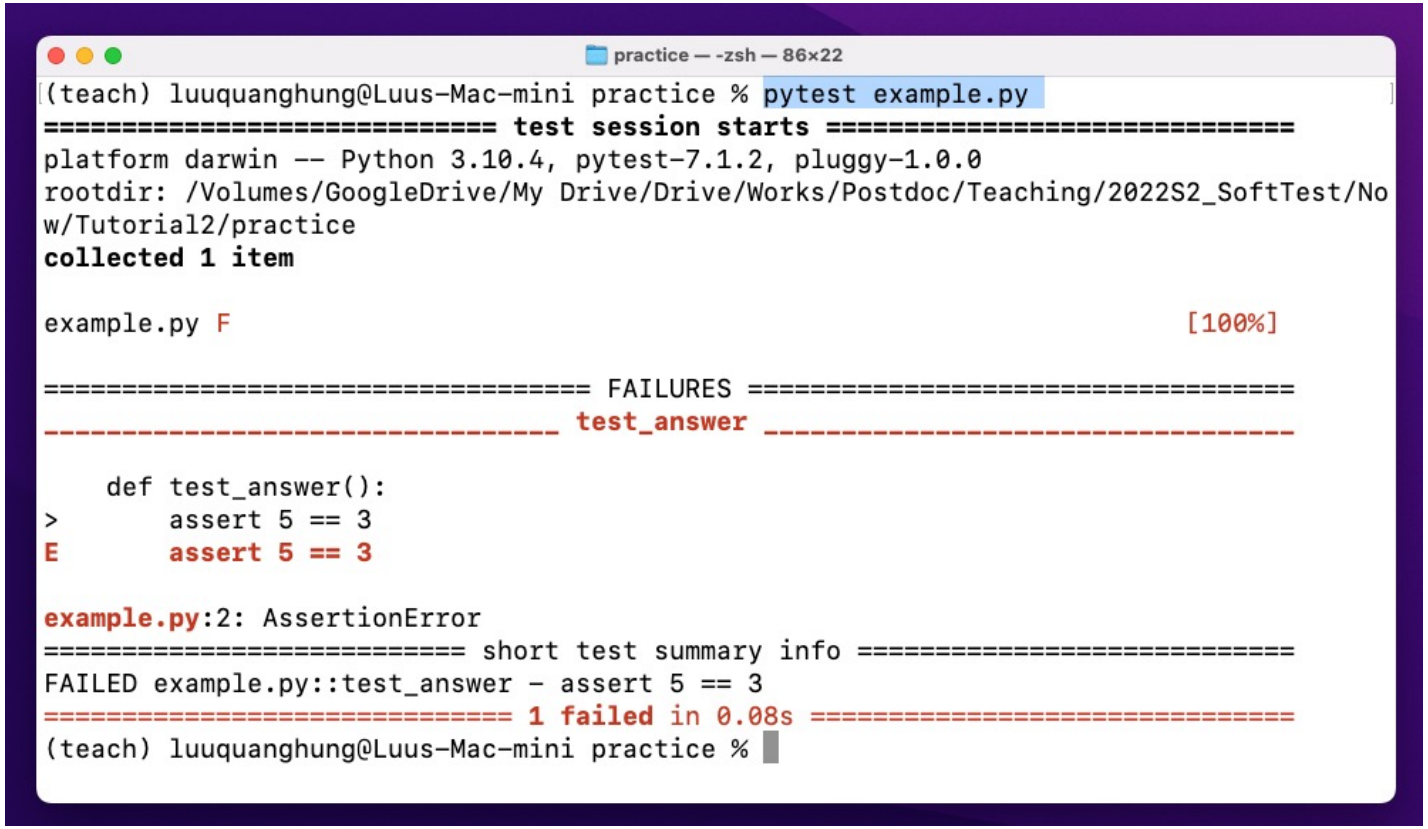
Pytest

- A compact and easy framework to conduct testing:
  - https://docs.pytest.org/en/7.1.x/

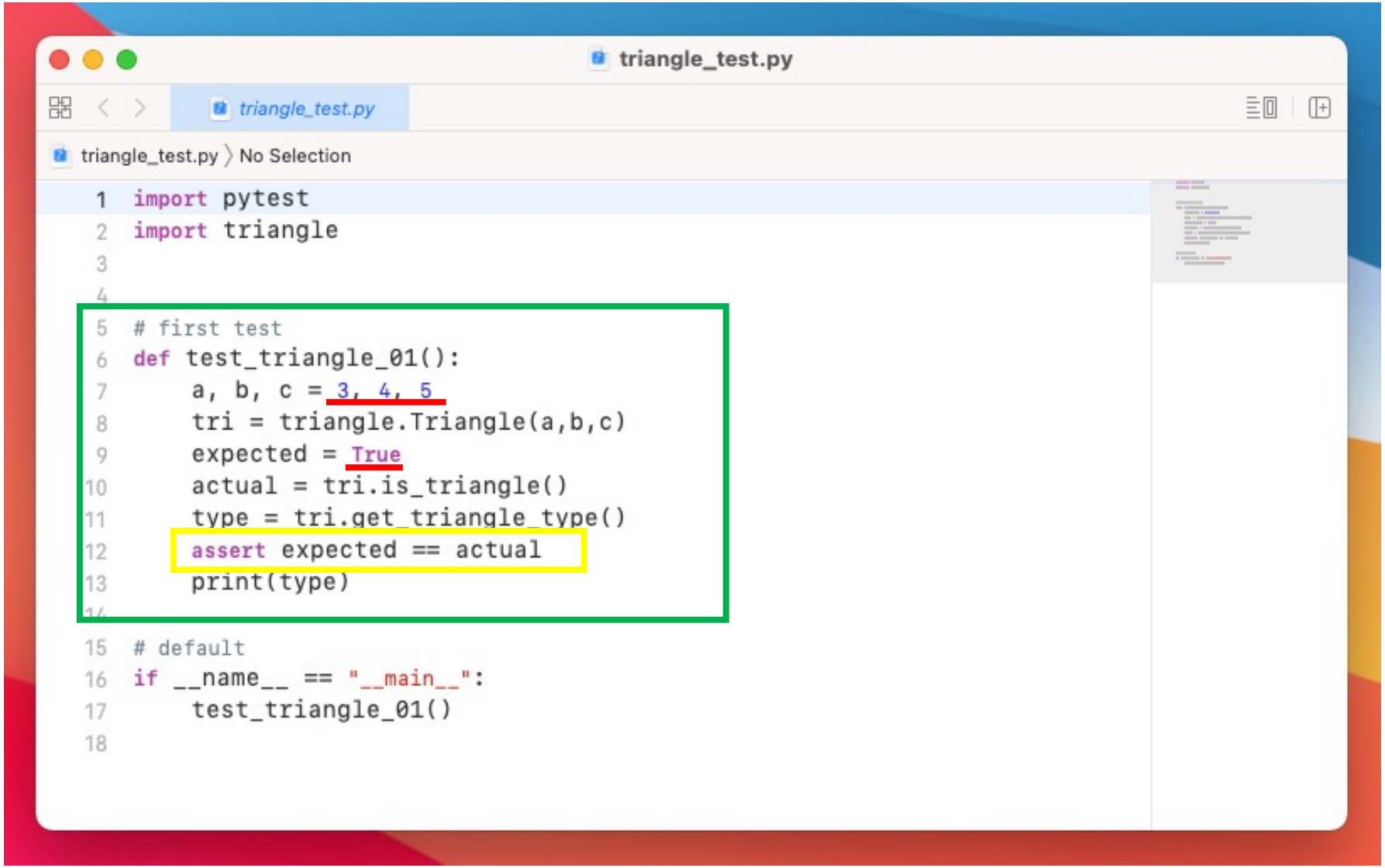
- Instruction
  - Test cases in Pytest is written in Python
  - Each test case is wrapped in a function whose name starting with test

```
def test_answer():
    assert 5 == 3
```

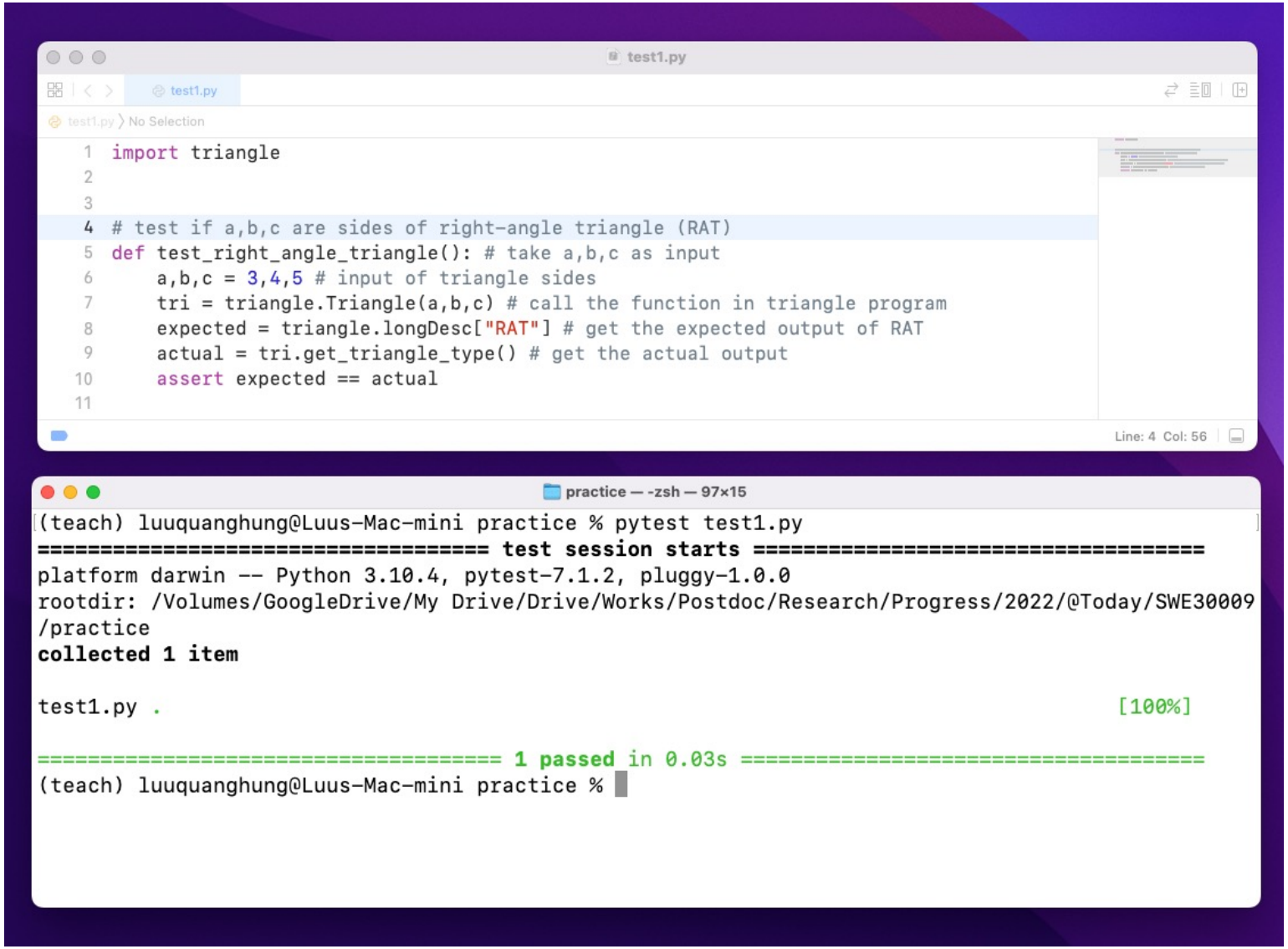
  - Execution in Terminal



How to implement the test



How to implement the test



PyTest asserts

- `assert <True/False Statement>`
  - Implement the testing with comparative True/False statement
- `assert tri.is_triangle() == True`
  - Equality comparison
- `assert 5 > 3`
  - Inequality comparison
- `assert 4 in [1,2,3,4,5,6]`
  - Other True/False comparison

Thank you