Software Testing，Lab 4，March 16，2023

Tasks：

1. **Install pytest testing framework.** You can install it with `pip install pytest` and check its version with `pytest --version`

   We recommend that you use anaconda to set up a virtual environment for this experiment. Be aware of **compatibility issues between pytest and python versions.**
   The documentation of pytest can be found here: https://docs.pytest.org/en/7.2.x/.

2. **Implement the two functions** mentioned in Lab3 in Python.

   Given the length value (integer) of 3 sides of a triangle. Implement 2 functions respectively,

   a) classifying the triangle

   given 3 length of sides(integers), output the shape of triangle made up by given sides. (Output a String, the shape could be “SCALENE”,“EQUILATERAL”,“ISOSCELES”,“INVALID”.)

   b) calculating the area of valid triangle.

   given 3 length of sides(integers), if these 3 sides can make up a valid triangle, output the area of the triangle (double or float), otherwise, return 0. (reference : Heron's formula)

3. **Design test cases for these functions** according to your previous study (MC/DC,boundary value, equivalence partitioning, etc.), guarantee the sufficiency and diversity of your test set.

   Note that **pytest’s Advanced assertion introspection will intelligently report intermediate values of the assert expression so you can avoid the many names of JUnit legacy methods .**

   Try to have fun with Advanced assertion introspection and design some fascinating test cases **if you are willing to（Not required**).

   Design setup and teardown functions with pytest and figure out their execution order.

   - setup_function、teardown_function
   - setup_class、teardown_class
   - setup_method、teardown_method

   Design test functions wit mark

   - `pytest.mark.skip` (reason=None) 无条件跳过测试用例
   - `pytest.mark.skipif` (condition, *, reason=None) 如果条件为真，则跳过测试用例
   - `pytest.mark.xfail`(condition=None, *, reason=None, raises=None, run=True, strict=False) 将测试用例标记为预期失败
   - etc

4. **Conventions for pytest test discovery**

   - your test files should be named 'test_*.py' or '*_test.py'
   - From those test files, pytest collect test items:
     - test prefixed test functions or methods outside of class (test_)
     - test prefixed test functions or methods inside Test prefixed test classes (without an **init** method) (Class Test*)

So manage your files and project structure carefully so that pytest can automatically collect your test cases.

As a suggestion, we recommend you to put tests into an extra directory outside your actual application code (often a good idea). An example:

```
pyproject.toml
src/
    mypkg/
        __init__.py
        app.py
        view.py
tests/
    test_app.py
    test_view.py
    ...
```

5. **Analyse the report provided by pytest. Collect these informations:**
    - The number of test cases executed
    - The number of failed test cases and their reason
    - The execution order of setup and teardown functions
    - The function of mark

    Use pytest plugins to manage your testing process and gather your testing results
    - pytest-html 生成html报告
    - pytest-timeout 限制超时时间

Requirements

1. Finish the tasks above individually.
2. Post your experiment report to “智慧树” , the following information should be included in your report:
    - The brief description that you install pytest and its plugins.
    - Steps for constructing your test suite and a brief introduction for it
    - Steps for testing process.
    - The testing results gathered from pytest's report
    - Your discussion based on provided report.

Submission deadline:

23:59 March 22, 2023

1. Install

pytest仅需一行命令`pip install pytest`即可安装，在之前的学习中已经安装，此处不再赘述。

```
PS D:\py-learn> pytest --version
pytest 6.2.4
```

2.Implement the two functions mentioned in Lab3 in Python.

使用Python重新编写实验三中UpgradedTriangle的代码，代码定义了一个名为"UpgradedTriangle"的三角形类。它有四种类型：INVALID（无效）、SCALENE（不等边）、EQUILATERAL（等边）和ISOSCELES（等腰）。

其中，classify()方法用于判断给定的三条边长是否可以组成一个有效的三角形，并根据三边长的不同情况返回相应的三角形类型。如果三边长中有任何一条小于等于0，或者任意两边之和小于第三边，则被认为是无效的三角形。

另外，area()方法用于计算三角形的面积，需要传入三角形的三条边长。如果给定的三边长不能组成一个有效的三角形，则返回0。否则，该方法将使用海伦公式计算三角形面积并返回结果。具体如下：

```python
import math


class UpgradedTriangle(object):
    class Type(object):
        INVALID, SCALENE, EQUILATERAL, ISOSCELES = range(4)

    @classmethod
    def classify(cls, a, b, c):
        if a <= 0 or b <= 0 or c <= 0:
            return cls.Type.INVALID
        if a + b <= c or a + c <= b or b + c <= a:
            return cls.Type.INVALID
        if a == b and b == c:
            return cls.Type.EQUILATERAL
        if a == b or b == c or a == c:
            return cls.Type.ISOSCELES
        return cls.Type.SCALENE

    @classmethod
    def area(cls, a, b, c):
        if cls.classify(a, b, c) == cls.Type.INVALID:
            return 0
        p = (a + b + c) / 2.0
        return math.sqrt(p * (p - a) * (p - b) * (p - c))
```

3. Design test cases

pytest不同范围的前/后置方法

- 1、模块级：setup_module/teardown_module 运行于模块的前后
- 2、函数级：setup_function/teardown_function 运行于每个函数的前后，即py文件中类外面的每个函数的前后运行一次
- 3、类级：setup_class/teardown_class 运行于测试类的始末，只在类的前后执行一次
- 4、方法级：setup_method/teardown_method 运行于测试类中的每个方法的始末，即类中的每个方法前后运行一次
- 5、类里面：setup/teardown 在类里面执行一次

整体全部的顺序：

setup_module->setup_function->test_one->teardown_function->setup_function->test_two->teardown_function->setup_class->setup_method->setup->test_three->teardown->teardown_method->setup_method->setup->test_four->teardown->teardown_method->teardown_class->teardown_module

使用pytest测试框架，包含一个TestTriangle类，该类包含两个测试方法，即test_triangle_area和test_triangle_classify。这两个方法分别测试UpgradedTriangle类的area和classify方法的输出是否符合预期。

其中，使用了pytest.mark.parametrize装饰器，对输入参数进行了多组测试，使用了pytest.mark.skip和pytest.mark.xfail来标记跳过的测试和期望失败的测试。

在setup_class、teardown_class、setup_method和teardown_method函数中，设置了测试用例的前置和后置条件。

```
import UpgradedTriangle as ut
import pytest


class TestTriangle:
    @classmethod
    def setup_class(cls):
        print("Setting up the TestTriangle class")

    @classmethod
    def teardown_class(cls):
        print("Tearing down the TestTriangle class")

    @staticmethod
    def setup_method():
        print("Setting up a test method")

    @staticmethod
    def teardown_method():
        print("Tearing down a test method")

    @pytest.mark.parametrize("a, b, c, expected", [
```

```
        (0, 1301, 1, 0),
        (3, 4, 5, 6),
        (5, 12, 13, 30),
        (6, 8, 10, 24),
        (1, 1, 2, 0),
        (-1, 2, 3, 0),
        (3, -1, 4, 0),
        (5, 6, -1, 0),
        (3, 4, 7, 0),
        (5, 6, 10, 11.399013115177997),
        (8, 8, 8, 27.712812921102035),
        (4, 4, 7, 6.777720855862979),
        (9, 12, 15, 54),
        (5, 5, 8, 12),
        (12, 16, 21, 95.4512310030625),
        pytest.param(-1, -2, 3, 0, marks=pytest.mark.skip(reason="Negative input
value")),
    ])
    def test_triangle_area(self, a, b, c, expected):
        assert ut.UpgradedTriangle.area(a, b, c) == expected

    @pytest.mark.parametrize("a, b, c, expected", [
        (0, 1301, 1, ut.UpgradedTriangle.Type.INVALID),
        (1108, 1, 1, ut.UpgradedTriangle.Type.INVALID),
        (1, 0, -665, ut.UpgradedTriangle.Type.INVALID),
        (1, 1, 0, ut.UpgradedTriangle.Type.INVALID),
        (1, 2, 450, ut.UpgradedTriangle.Type.INVALID),
        (1187, 1146, 1, ut.UpgradedTriangle.Type.INVALID),
        (784, 784, 1956, ut.UpgradedTriangle.Type.INVALID),
        (1, 450, 1, ut.UpgradedTriangle.Type.INVALID),
        (-1, 1, 1, ut.UpgradedTriangle.Type.INVALID),
        (582, 582, 582, ut.UpgradedTriangle.Type.EQUILATERAL),
        (1640, 1956, 1956, ut.UpgradedTriangle.Type.ISOSCELES),
        (1146, 1, 1146, ut.UpgradedTriangle.Type.ISOSCELES),
        (5, 5, 8, ut.UpgradedTriangle.Type.ISOSCELES),
        (1663, 1088, 823, ut.UpgradedTriangle.Type.SCALENE),
        (3, 4, 6, ut.UpgradedTriangle.Type.SCALENE),
        pytest.param(5, 5, 5, ut.UpgradedTriangle.Type.ISOSCELES,
                     marks=pytest.mark.xfail(reason="Expected Equilateral triangle,
got Isosceles instead")),
        pytest.param(-3, -1, -1, ut.UpgradedTriangle.Type.INVALID,
                     marks=pytest.mark.skip(reason="Negative input value"))
    ])
    def test_triangle_classify(self, a, b, c, expected):
        assert ut.UpgradedTriangle.classify(a, b, c) == expected


if __name__ == '__main__':
    pytest.main()
```

4. Analyze Result

执行如下命令安装pytest-html并且生成报告。

```
conda install pytest-html
pytest --html=report.html
```

report.html

Report generated on 16-Mar-2023 at 18:02:40 by pytest-html v3.1.1

Environment

Packages	{"pluggy": "0.13.1", "py": "1.10.0", "pytest": "6.2.4"}
Platform	Windows-10-10.0.19044-SP0
Plugins	{"anyio": "2.2.0", "html": "3.1.1", "metadata": "1.11.0"}
Python	3.9.7

Summary

31 tests ran in 0.26 seconds.

(Un)check the boxes to filter the results.

☑ 30 passed, ☑ 2 skipped, ☐ 0 failed, ☐ 0 errors, ☑ 1 expected failures, ☐ 0 unexpected passes

Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[0-1301-1-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[3-4-5-6]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[5-12-13-30]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[6-8-10-24]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[1-1-2-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[-1-2-3-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[3--1-4-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[5-6--1-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[3-4-7-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[5-6-10-11.399013115177997]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[8-8-8-27.712812921102035]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[4-4-7-6.777720855862979]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[9-12-15-54]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[5-5-8-12]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_area[12-16-21-95.4512310030625]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[0-1301-1-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[1108-1-1-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[1-0--665-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[1-1-0-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[1-2-450-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[1187-1146-1-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[784-784-1956-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[1-450-1-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[-1-1-1-0]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[582-582-582-2]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[1640-1956-1956-3]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[1146-1-1146-3]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[5-5-8-3]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[1663-1088-823-1]
Passed *(show details)*	UpgradedTriangle_test.py::TestTriangle::test_triangle_classify[3-4-6-1]

从结果中不难发现：

30个测试用例通过。

2个测试用例被跳过，原因是输入的边中有负值，是无效的测试用例。

```
    pytest.param(-3, -1, -1, ut.UpgradedTriangle.Type.INVALID,
                 marks=pytest.mark.skip(reason="Negative input value"))

    pytest.param(-1, -2, 3, 0, marks=pytest.mark.skip(reason="Negative input
value")),
```

1个测试用例属于预期错误，原因是对于三条边都是5的三角形，应该输出是等边三角形而不是等腰三角形。

```
    pytest.param(5, 5, 5, ut.UpgradedTriangle.Type.ISOSCELES,
                      marks=pytest.mark.xfail(reason="Expected Equilateral triangle,
got Isosceles instead")),
```