Python unittest Coverage

Summary: in this tutorial, you'll learn how to use the Python unittest coverage command to generate a test coverage report.

What is a test coverage

Test coverage is a ratio between the number of lines executed by at least one test case and the total number of lines of the code base:

```
test coverage = lines of code executed / total number of lines
```

The test coverage is also known as code coverage.

The test coverage is often used to assess the quality of a test suite. If the test coverage is low e.g., 5%, it is an indicator that you're not testing enough.

However, the reverse may not be true. For example, 100% test coverage is not a guarantee that you have a good test suite. In other words, a test suite with high coverage can still be of poor quality.

Unittest coverage example

We'll use the following project structure to demo the `unittest` coverage. Note that you can get the source code from this tutorial (https://www.pythontutorial.net/python-unit-testing/python-run-unittest/) .

```
D:\python-unit-testing
├── shapes
|   ├── circle.py
|   ├── shape.py
|   └── square.py
└── test
    ├── test_circle.py
    ├── test_square.py
    └── __init__.py
```

To generate a coverage report, you need to carry out two steps:

First, run the coverage module to generate the coverage data:

```
python -m coverage run -m unittest
```

Second, turn the coverage data into a report:

```
python -m coverage report
```

Output:

```
Name                    Stmts   Miss  Cover
-----------------------------------------
shapes\circle.py            9      0   100%
shapes\shape.py             4      0   100%
shapes\square.py            9      0   100%
test\__init__.py            0      0   100%
test\test_circle.py        14      0   100%
test\test_square.py        14      0   100%
-----------------------------------------
TOTAL                      50      0   100%
```

To generate the coverage report in HTML format, you change the option of the coverage module to HTML like this:

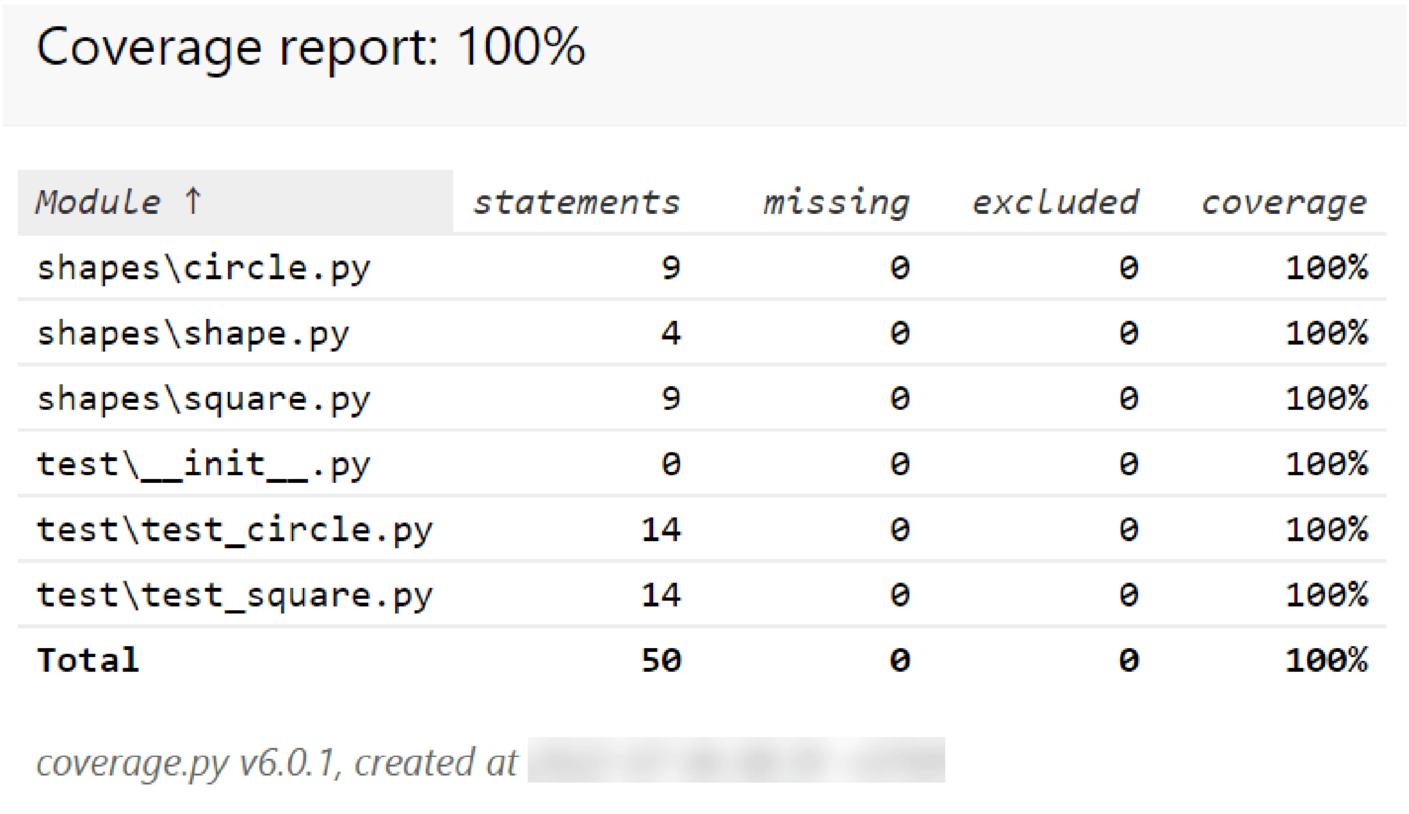
```
python -m coverage html
```

Output:

```
Wrote HTML report to htmlcov\index.html
```

The output indicates the location of the HTML coverage report `htmlcov\index.html` under the project folder.

If you open the index.html file of the `htmlcov` folder, it'll look like the following:



Coverage report: 100%

Module ↑	statements	missing	excluded	coverage
shapes\circle.py	9	0	0	100%
shapes\shape.py	4	0	0	100%
shapes\square.py	9	0	0	100%
test__init__.py	0	0	0	100%
test\test_circle.py	14	0	0	100%
test\test_square.py	14	0	0	100%
Total	**50**	**0**	**0**	**100%**

coverage.py v6.0.1, created at [redacted]

Examining unittest coverage detailed report

First, add the `perimeter()` method to the `Circle` class as follows:

```
import math
from .shape import Shape
```

```
class Circle(Shape):
    def __init__(self, radius: float) -> None:
        if radius < 0:
            raise ValueError('The radius cannot be negative')

        self._radius = radius

    def area(self) -> float:
        return math.pi * math.pow(self._radius, 2)

    def perimeter(self) -> float:
        return 2 * math.pi * self._radius
```

Next, gather the coverage data by running the following command:

```
python -m coverage run -m unittest
```

Then, generate the coverage report by executing the following command:

```
python -m coverage report
```

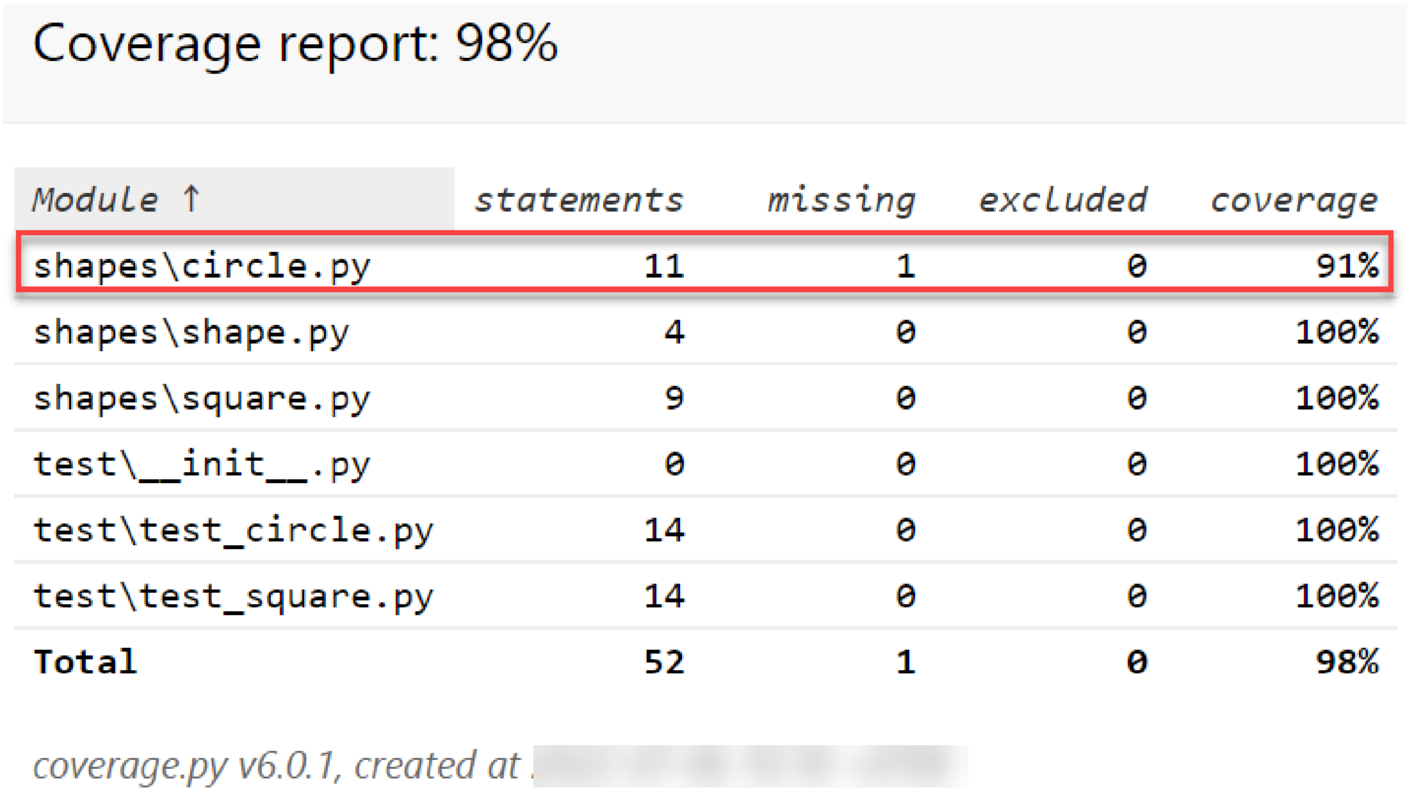
Output:

```
Name                      Stmts   Miss  Cover
-----------------------------------------
shapes\circle.py             11      1    91%
shapes\shape.py               4      0   100%
shapes\square.py              9      0   100%
test\__init__.py              0      0   100%
test\test_circle.py          14      0   100%
test\test_square.py          14      0   100%
-----------------------------------------
TOTAL                        52      1    98%
```

The coverage now is 98% in total and 91% in the `shape\circle.py` module. This is because the `perimeter()` method is not tested.

After that, generate the coverage report in HTML format:

```
python -m coverage html
```



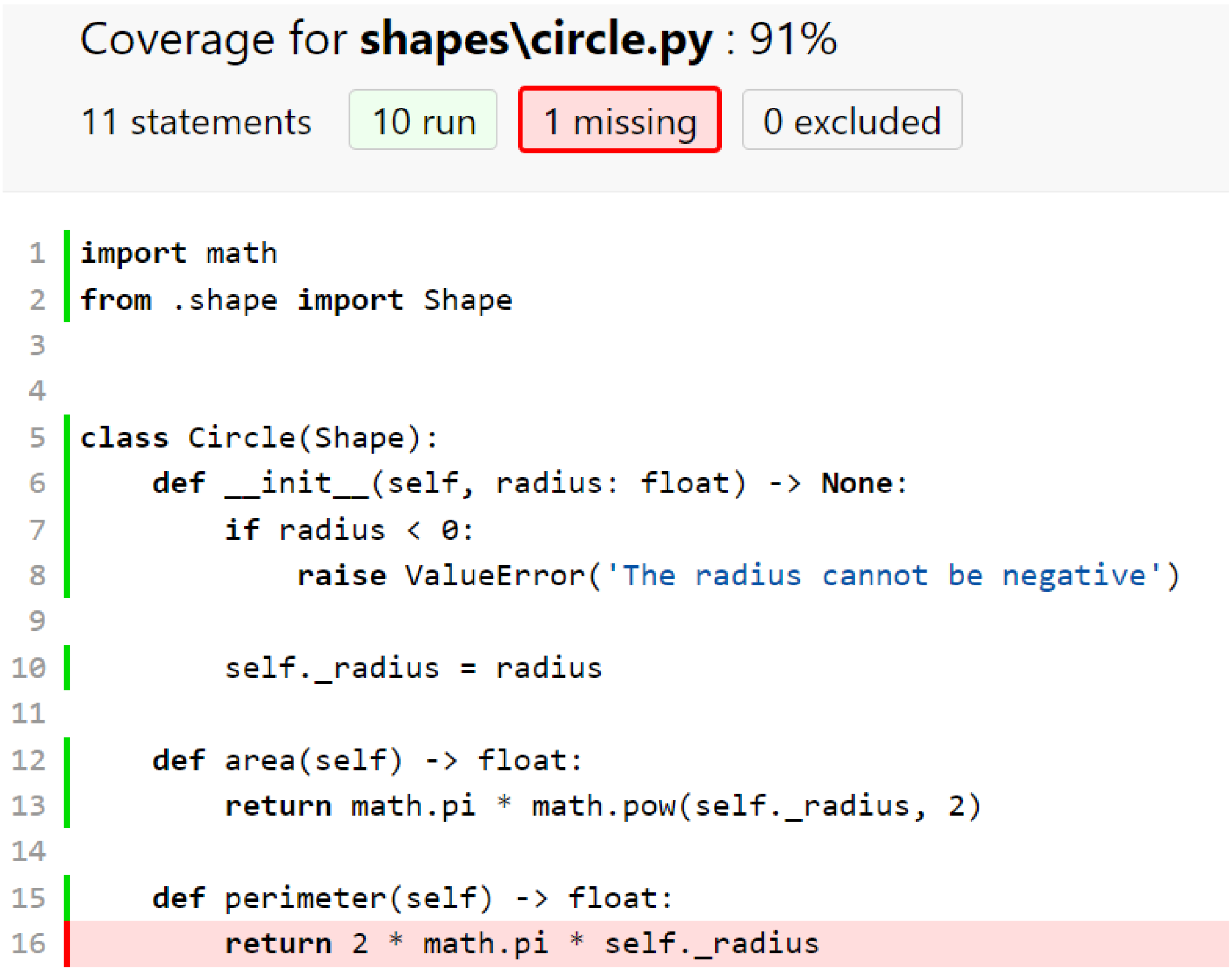
Coverage report: 98%

Module ↑	statements	missing	excluded	coverage
shapes\circle.py	11	1	0	91%
shapes\shape.py	4	0	0	100%
shapes\square.py	9	0	0	100%
test__init__.py	0	0	0	100%
test\test_circle.py	14	0	0	100%
test\test_square.py	14	0	0	100%
Total	**52**	**1**	**0**	**98%**

coverage.py v6.0.1, created at [illegible]

The `circle.py` has 11 statements. The test executes 10 of them and misses one statement. Therefore, the test coverage is 10/11 ~ 91%.

Finally, click the `circle.py` module for the detailed report:



Coverage for **shapes\circle.py** : 91%

11 statements | 10 run | 1 missing | 0 excluded

```python
import math
from .shape import Shape


class Circle(Shape):
    def __init__(self, radius: float) -> None:
        if radius < 0:
            raise ValueError('The radius cannot be negative')

        self._radius = radius

    def area(self) -> float:
        return math.pi * math.pow(self._radius, 2)

    def perimeter(self) -> float:
        return 2 * math.pi * self._radius
```

Summary

- Use the `python -m coverage run -m unittest` command to gather coverage data and the `python -m coverage report` command to generate a coverage report.
- Use the `python -m coverage html` to generate the test coverage report in HTML format.