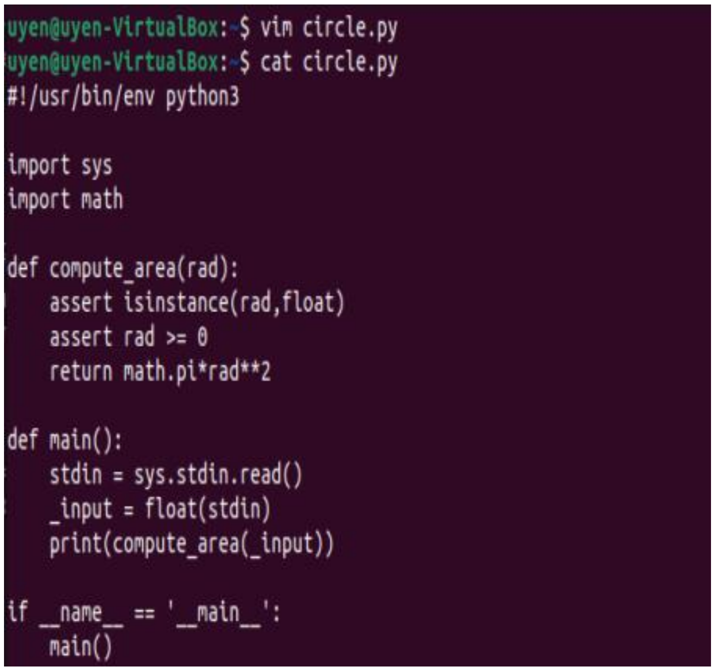
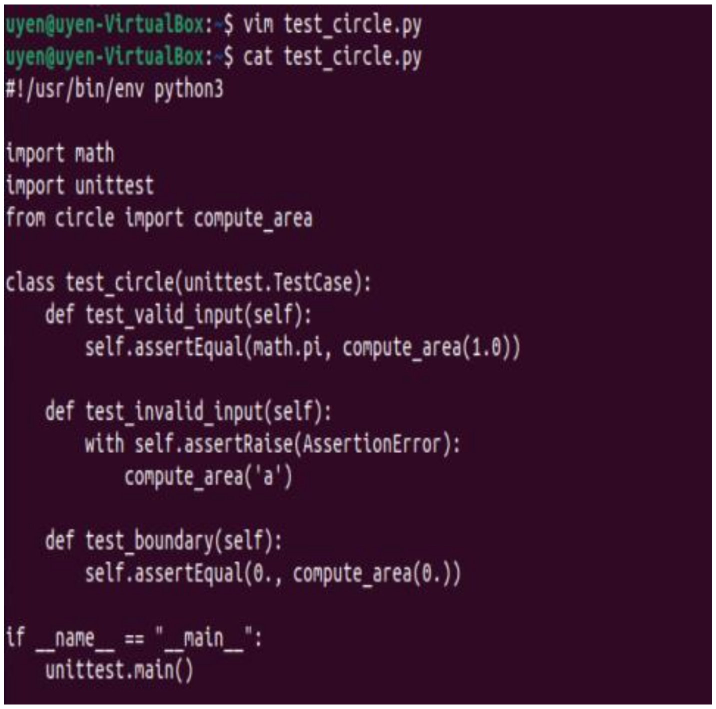
1. Write a python file “circle.py” with a program to compute an area of a circle, given its radius. The program should read the circle radius from a standard input, and write the area to a standard output. The code that actually computes the area should be encapsulated in a function “compute_area”. The code to read input, compute the area, and print the result should be wrapped into a “main” function.



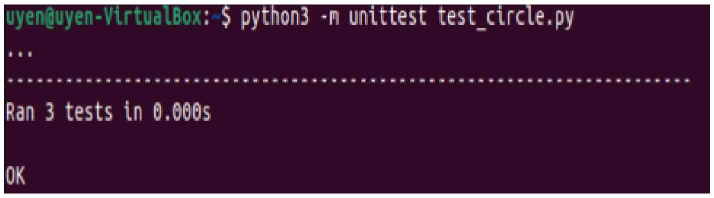
```
uyen@uyen-VirtualBox:~$ vim circle.py
uyen@uyen-VirtualBox:~$ cat circle.py
#!/usr/bin/env python3

import sys
import math

def compute_area(rad):
    assert isinstance(rad,float)
    assert rad >= 0
    return math.pi*rad**2

def main():
    stdin = sys.stdin.read()
    _input = float(stdin)
    print(compute_area(_input))

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

2. Create a test-case for the encapsulated function, along with several test methods. Write at least three different tests, providing a random valid input (“test_valid_input”), random invalid input (“test_invalid_input”), and a boundary condition (“test_boundary”).



```
uyen@uyen-VirtualBox:~$ vim test_circle.py
uyen@uyen-VirtualBox:~$ cat test_circle.py
#!/usr/bin/env python3

import math
import unittest
from circle import compute_area

class test_circle(unittest.TestCase):
    def test_valid_input(self):
        self.assertEqual(math.pi, compute_area(1.0))

    def test_invalid_input(self):
        with self.assertRaise(AssertionError):
            compute_area('a')

    def test_boundary(self):
        self.assertEqual(0., compute_area(0.))

if __name__ == "__main__":
    unittest.main()
```

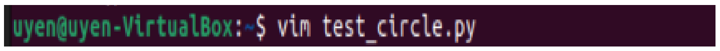
3. Use the "unittest" Python package from the command-line to run all of the tests, or each individual test, or tests matching a regex.



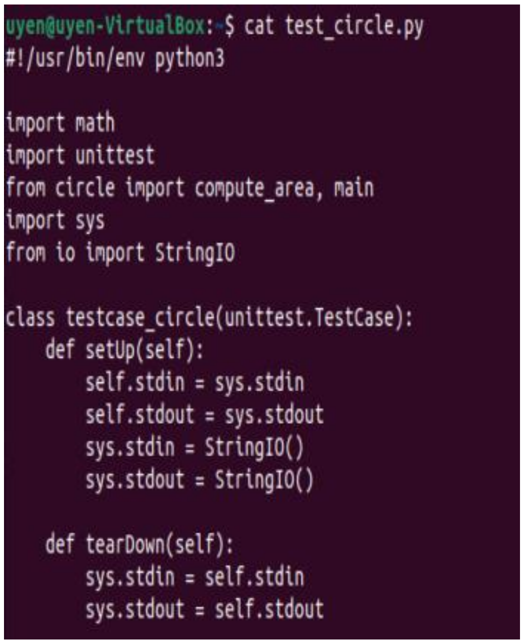
```
uyen@uyen-VirtualBox:~$ python3 -m unittest test_circle.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```

4. Create a test-case for the "main" function, that would test both how the circle area is computed and how the program interacts with the console.

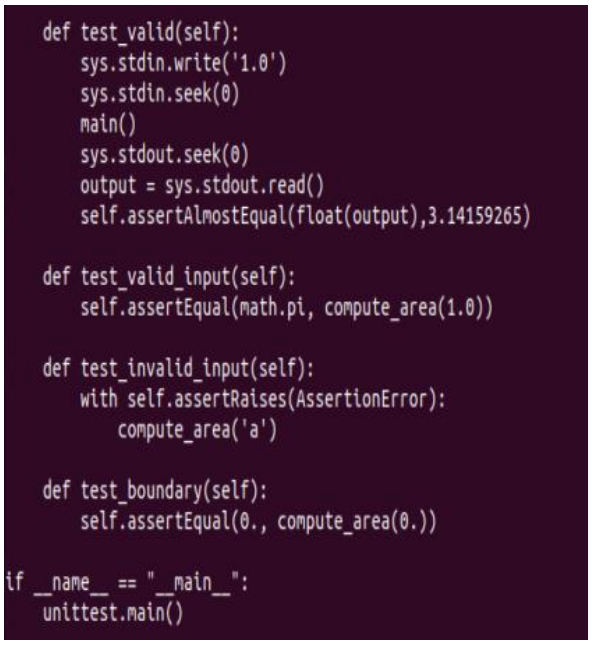
Note: you need to emulate console IO. To this end, use the "setUp/tearDown" instance methods in the "unittest.TestCase" class to substitute "sys.stdin" and "sys.stdout" for "StringIO" objects. The rest of the test case should be the same.



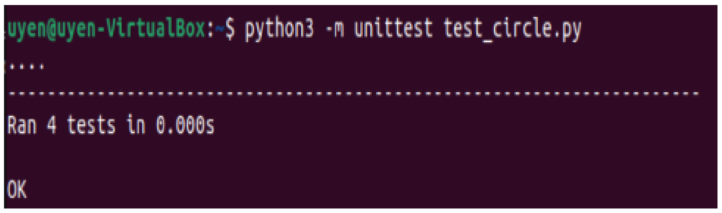
```
uyen@uyen-VirtualBox:~$ vim test_circle.py
```



```
uyen@uyen-VirtualBox:~$ cat test_circle.py
#!/usr/bin/env python3

import math
import unittest
from circle import compute_area, main
import sys
from io import StringIO

class testcase_circle(unittest.TestCase):
    def setUp(self):
        self.stdin = sys.stdin
        self.stdout = sys.stdout
        sys.stdin = StringIO()
        sys.stdout = StringIO()

    def tearDown(self):
        sys.stdin = self.stdin
        sys.stdout = self.stdout
```

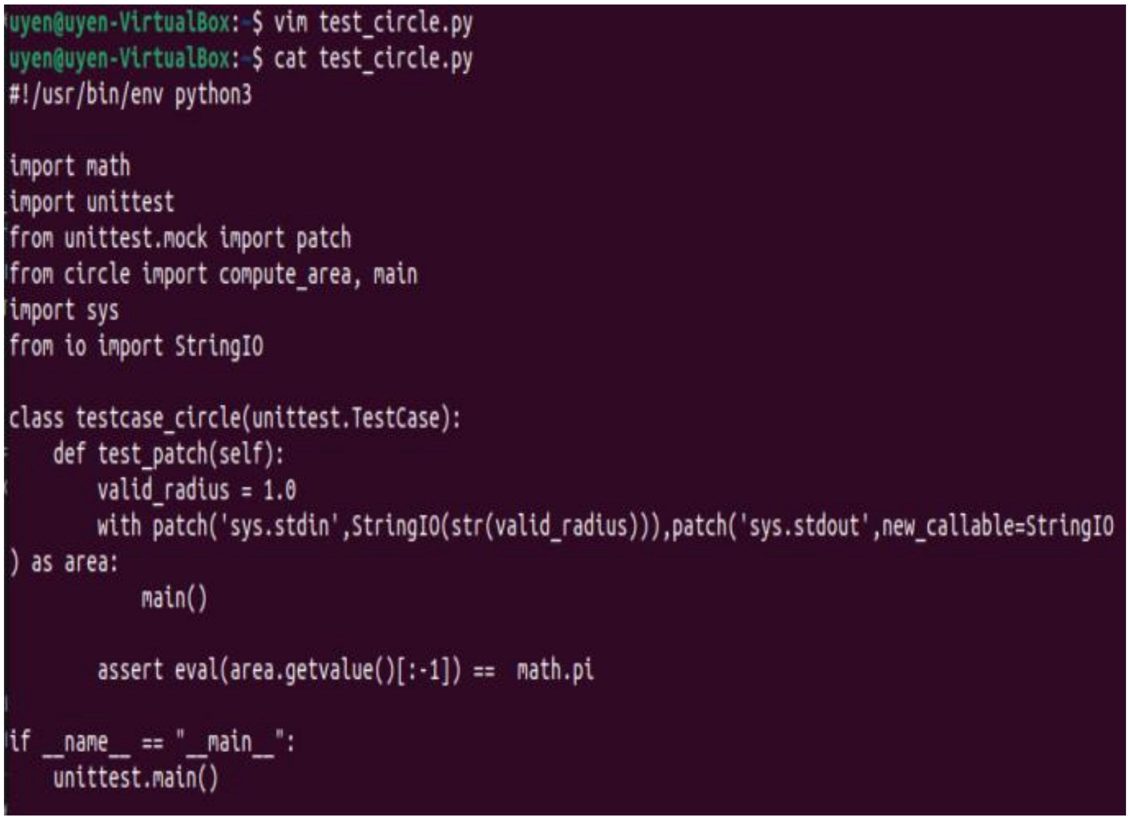


```
    def test_valid(self):
        sys.stdin.write('1.0')
        sys.stdin.seek(0)
        main()
        sys.stdout.seek(0)
        output = sys.stdout.read()
        self.assertAlmostEqual(float(output),3.14159265)

    def test_valid_input(self):
        self.assertEqual(math.pi, compute_area(1.0))

    def test_invalid_input(self):
        with self.assertRaises(AssertionError):
            compute_area('a')

    def test_boundary(self):
        self.assertEqual(0., compute_area(0.))

if __name__ == "__main__":
    unittest.main()
```

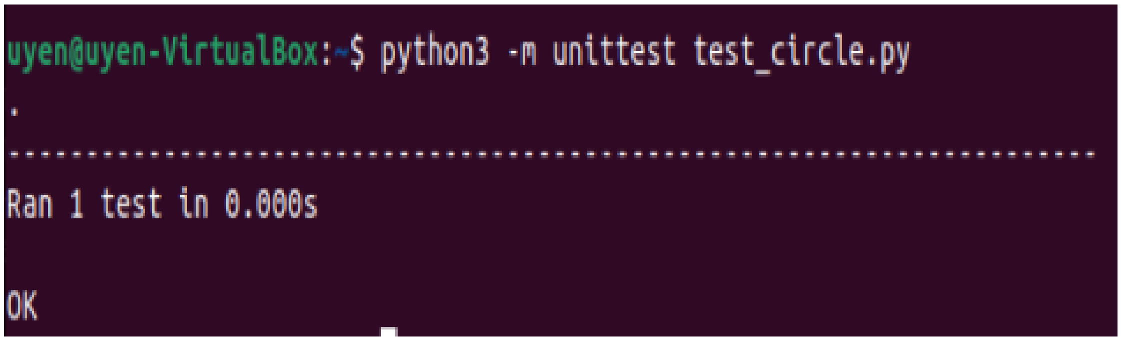


```
uyen@uyen-VirtualBox:~$ python3 -m unittest test_circle.py
....
----------------------------------------------------------------------
Ran 4 tests in 0.000s

OK
```

5. Modify the test-case for the “main” function using the “unittest.mock” package. Put the invocation of the “main” function into “unittest.mock.patch” context manager to manage mocking of the standard input and standard output.



```
uyen@uyen-VirtualBox:~$ vim test_circle.py
uyen@uyen-VirtualBox:~$ cat test_circle.py
#!/usr/bin/env python3

import math
import unittest
from unittest.mock import patch
from circle import compute_area, main
import sys
from io import StringIO

class testcase_circle(unittest.TestCase):
    def test_patch(self):
        valid_radius = 1.0
        with patch('sys.stdin',StringIO(str(valid_radius))),patch('sys.stdout',new_callable=StringIO
) as area:
            main()

        assert eval(area.getvalue()[:-1]) ==  math.pi

if __name__ == "__main__":
    unittest.main()
```



```
uyen@uyen-VirtualBox:~$ python3 -m unittest test_circle.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```