| | |
|---|---|
| Assignment Date | 18 November 2022 |
| Student Name | G.Karthika |
| Student Roll Number | 815819104011 |
| Maximum Marks | |

```
class Square:
    def __init__(self, side):
        """ creates a square having the given side
        """
        self.side = side

    def area(self):
        """ returns area of the square
        """
        return self.side**2

    def perimeter(self):
        """ returns perimeter of the square
        """
        return 4 * self.side

    def __repr__(self):
        """ declares how a Square object should be
printed
        """
        s = 'Square with side = ' + str(self.side) +
'\n' + \
        'Area = ' + str(self.area()) + '\n' + \
        'Perimeter = ' + str(self.perimeter())
        return s


if __name__ == '__main__':
    # read input from the user
    side = int(input('enter the side length to create a
Square: '))

    # create a square with the provided side
    square = Square(side)

    # print the created square
    print(square)
```

**Note:** For more information about the function __repr__(), refer this article.

Now that we have our software ready, let's have a look at the directory structure of our project folder and after that, we'll start testing our software.

```
---Software_Testing
   |--- __init__.py (to initialize the directory as python package)
   |--- app.py (our software)
   |--- tests (folder to keep all test files)
           |--- __init__.py
```

```
def test_area(self):
    # testing the method Square.area().

    sq = Square(2)    # creates a Square of side 2
units.

    # test if the area of the above square is 4
units,
    # display an error message if it's not.

    self.assertEqual(sq.area(), 4,
        f'Area is shown {sq.area()} for side =
{sq.side} units')
import unittest
from .. import app

class TestSum(unittest.TestCase):

    def test_area(self):
        sq = app.Square(2)

        self.assertEqual(sq.area(), 4,
            f'Area is shown {sq.area()} rather than
9')

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

```
import unittest
from .. import app

class TestSum(unittest.TestCase):

    def test_area(self):
        sq = app.Square(2)
        self.assertEqual(sq.area(), 4,
            f'Area is shown {sq.area()} rather
than 9')

    def test_area_negative(self):
        sq = app.Square(-3)
        self.assertEqual(sq.area(), -1,
            f'Area is shown {sq.area()} rather
than -1')

    def test_perimeter(self):
        sq = app.Square(5)
        self.assertEqual(sq.perimeter(), 20,
            f'Perimeter is {sq.perimeter()} rather
than 20')

    def test_perimeter_negative(self):
        sq = app.Square(-6)
        self.assertEqual(sq.perimeter(), -1,
            f'Perimeter is {sq.perimeter()} rather
than -1')

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

```
.F.F
======================================================================
FAIL: test_area_negative (__main__.TestSum)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "tests_unittest.py", line 11, in test_area_negative
    self.assertEqual(sq.area(), -1, f'Area is shown {sq.area()} rather
than -1 for negative side length')
AssertionError: 9 != -1 : Area is shown 9 rather than -1 for negative
side length

======================================================================
FAIL: test_perimeter_negative (__main__.TestSum)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "tests_unittest.py", line 19, in test_perimeter_negative
    self.assertEqual(sq.perimeter(), -1, f'Perimeter is {sq.perimeter()}
rather than -1 for negative side length')
AssertionError: -24 != -1 : Perimeter is -24 rather than -1 for negative
side length

----------------------------------------------------------------------
Ran 4 tests in 0.001s

FAILED (failures=2)
```