Part 4: Running the Script

Bookmark this page

Part 4: Running the Script

Running *python test.py* will invoke the SquarerTest.test_positive_numbers method and also the SquarerTest.test_negative_numbers method. If our Squarer.calc method is working as we expect it to, if it conforms to the behavior we defined in the *squares* dictionary, then nothing will be printed to the console. On the other hand, if the SquarerTest finds discrepancies in the behavior of Squarer, then it will print those discrepancies to the console.

Since *squarer.py* has an error that the other developer just introduced, here is the result of running *python test.py*:

```
$ python test.py
Squared 3 and got 27 but expected 9
Squared 12 and got 8916100448256 but expected 144
Squared 100 and got
10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000 but expected 10000
Squared -1 and got -1.0 but expected 1
Squared -2 and got 0.25 but expected 4
Squared -3 and got -0.037037037037037035 but expected 9
Squared -12 and got 1.1215665478461509e-13 but expected 144
Squared -100 and got 1e-200 but expected 10000

$
```

Let's go back into squarer.py and fix our mistake:

```
# squarer.py
class Squarer(object):

    @staticmethod
    def calc(operand):
        # return operand*operand   # OLD
        # return operand**operand  # WRONG
        return operand*operand
```

Has this actually fixed our code? Let's run our test script to find out:

```
$ python test.py

$
```

Running test.py produces no errors: test.py was not able to find any discrepancies between the expected behavior that we defined for Squarer's calc method and the method's actual behavior.

Suppose you wanted to revise squarer.py one more time. You want to try another implementation of the calc method that uses the power operator:

```
# squarer.py
class Squarer(object):

    @staticmethod
    def calc(operand):
        # return operand*operand   # OLD
        # return operand**operand  # WRONG
        # return operand*operand   # OLD
        return operand**2
```

By putting our test into a script file that we can run from the command line, we've made it so that it only takes a few seconds to test whether we've introduced an error into our code. We run our test script:

seconds to test whether we've introduced an error into our code. We run our test script:

Previous

Next

```
$ python test.py

$
```

The test script has not found any discrepancies between the *expected* and *actual* performance of Squarer.calc: it looks like *operand**2* is an acceptable way to implement squaring!

Help Center Contact Us Privacy Terms