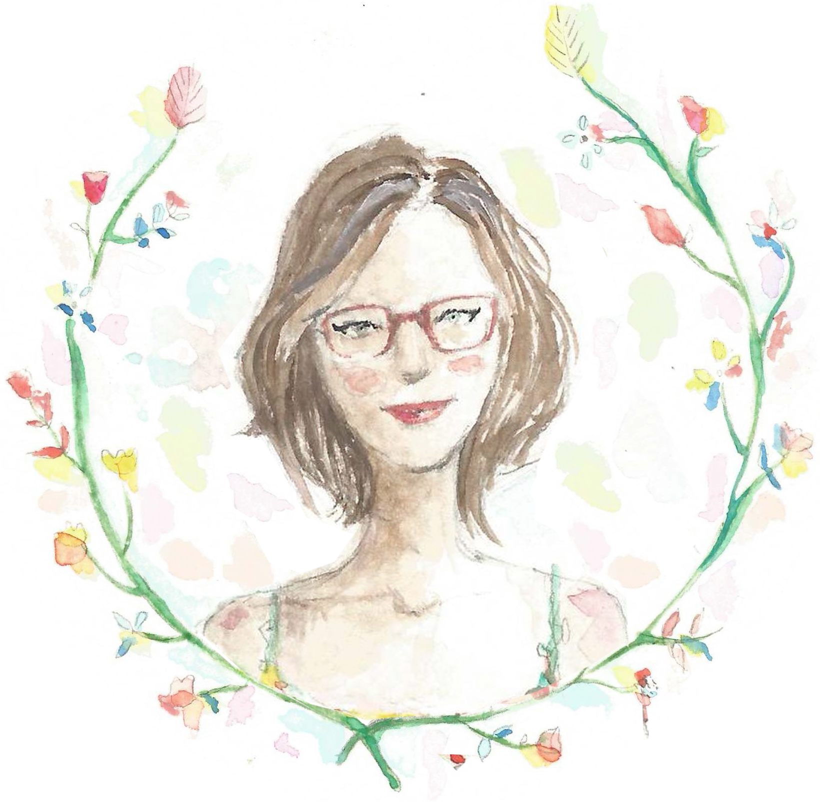
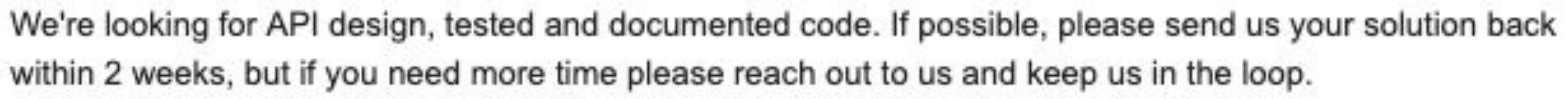
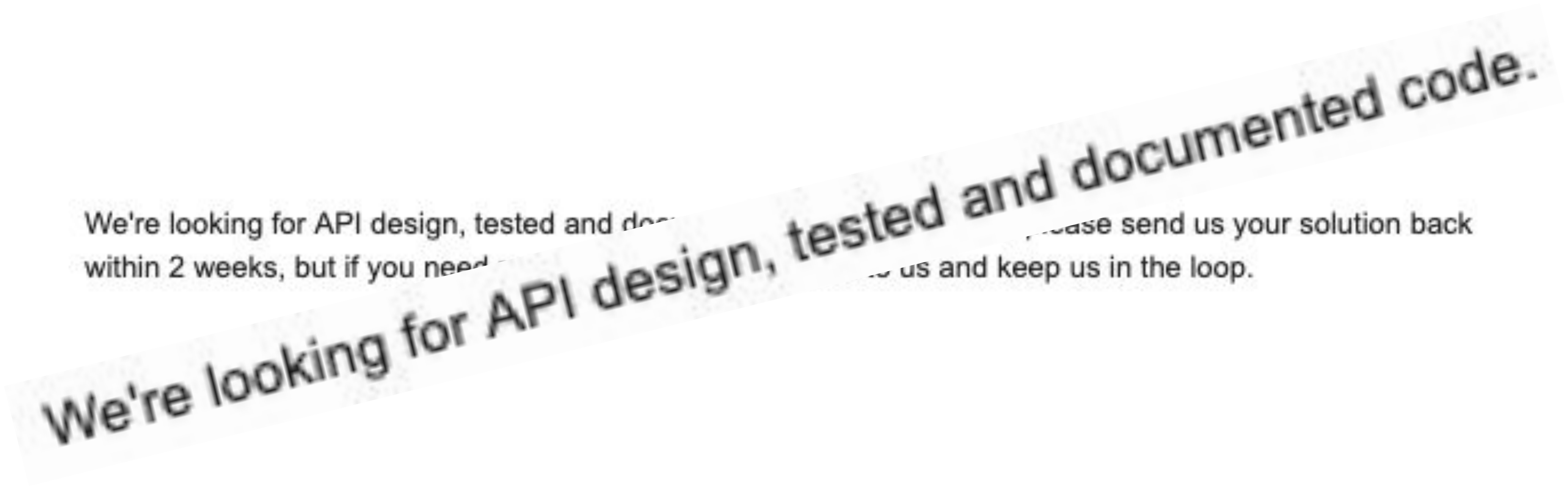
First steps into the testing world of Python

A report on learning Software Testing





We're looking for API design, tested and documented code. If possible, please send us your solution back within 2 weeks, but if you need more time please reach out to us and keep us in the loop.



We're looking for API design, tested and doc… …ease send us your solution back within 2 weeks, but if you need … us and keep us in the loop.

We're looking for API design, tested and documented code.

ented code.

We're looking for

tested

our solution back

within 2 weeks, b

We're look

Software Testing

what does it mean?

exploratory testing

Repetitive is boring, boring leads to mistakes and makes you look for a different job by the end of the week.

Ham Vocke

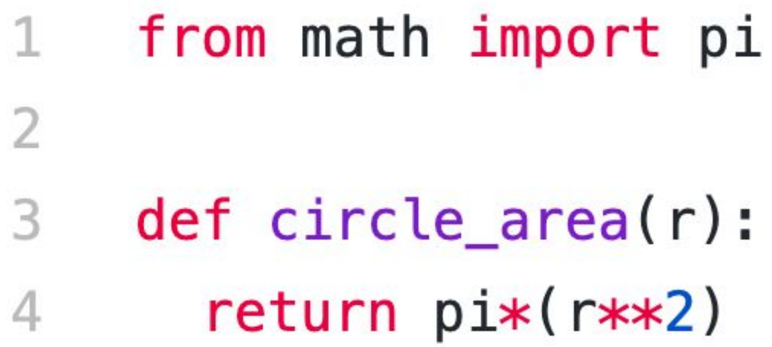
automated testing

where do I start?

unit testing

unittest

circles.py



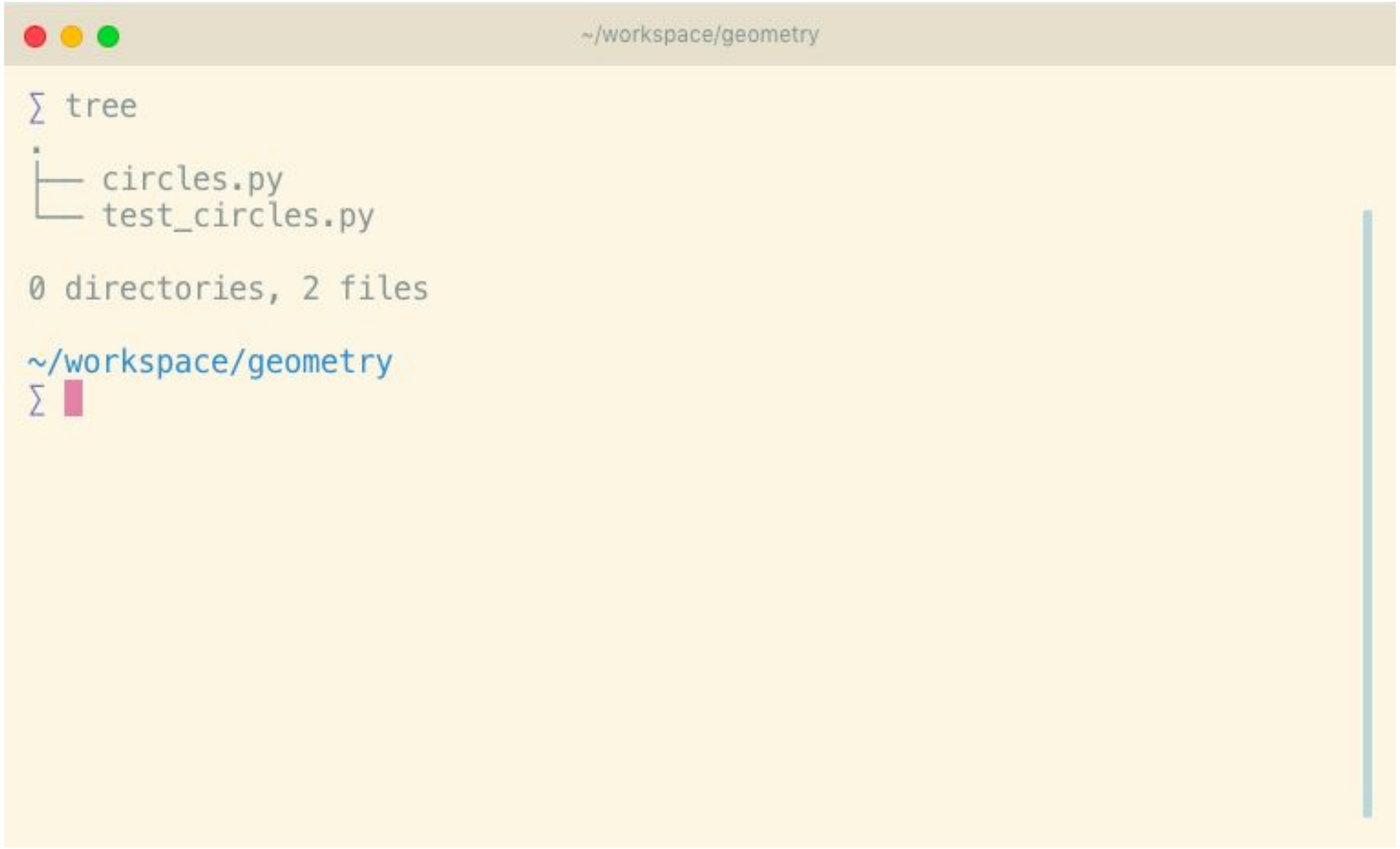
```python
from math import pi

def circle_area(r):
    return pi*(r**2)
```

circles.py

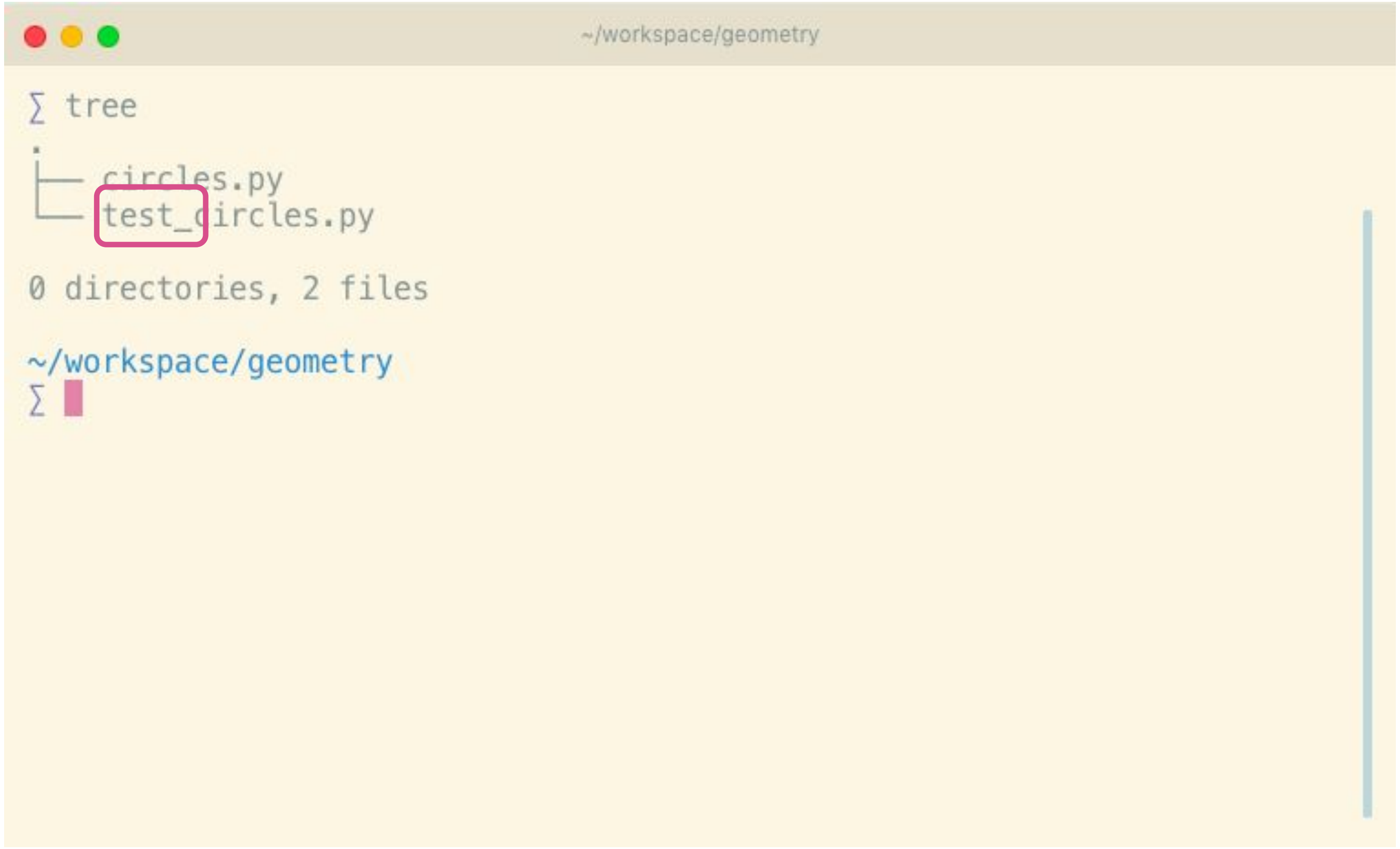
```python
from math import pi

def circle_area(r):
    return pi*(r**2)
```

circles.py

```python
from math import pi

def circle_area(r):
  return pi*(r**2)
```



```
~/workspace/geometry

∑ tree
.
├── circles.py
└── test_circles.py

0 directories, 2 files

~/workspace/geometry
∑ 
```



~/workspace/geometry

```
∑ tree
.
├── circles.py
└── test_circles.py

0 directories, 2 files

~/workspace/geometry
∑ 
```

what do I test?

Area of circle, using radius

The area of a circle is equal to its radius squared then multiplied by pi.

$$A = \pi(r^{**}2)$$

- ! Radius must be a real number.
- ! Radius cannot be negative.

how do I test?

test_circles.py

```python
import unittest
from math import pi
from circles import circle_area

class TestCircleArea(unittest.TestCase):
  def test_area_result(self):
    pass

  def test_negative_value(self):
    pass

  def test_parameter_type(self):
    pass
```

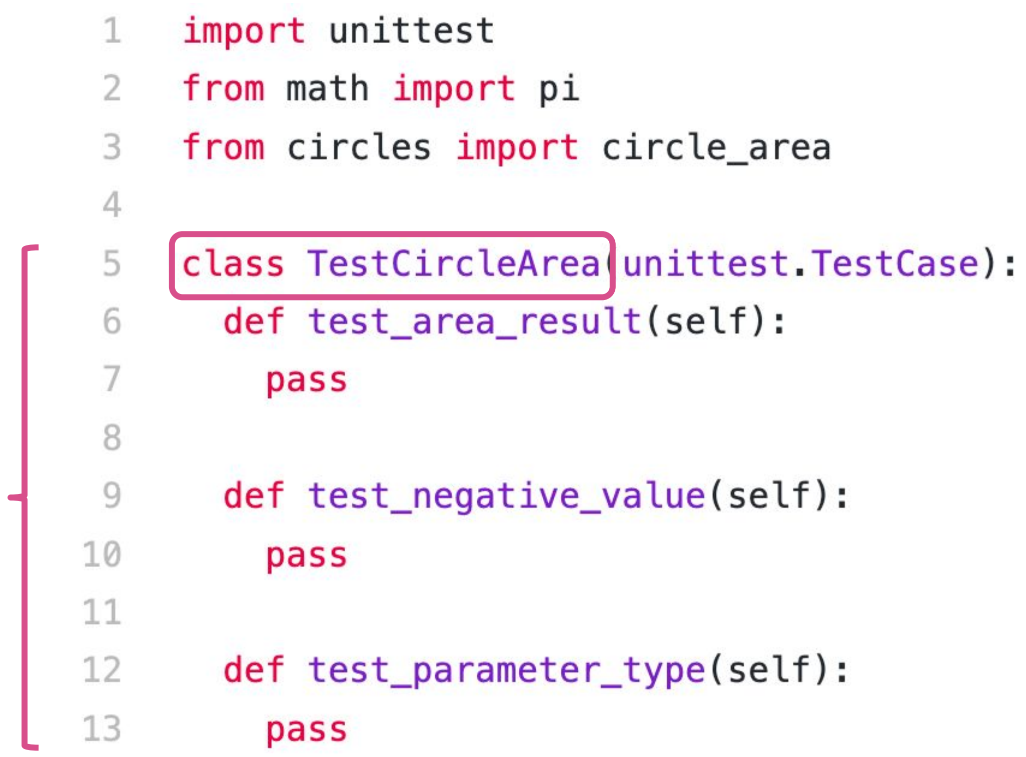
test_circles.py

```python
import unittest
from math import pi
from circles import circle_area

class TestCircleArea(unittest.TestCase):
  def test_area_result(self):
    pass

  def test_negative_value(self):
    pass

  def test_parameter_type(self):
    pass
```

test_circles.py

```python
import unittest
from math import pi
from circles import circle_area

class TestCircleArea(unittest.TestCase):
  def test_area_result(self):
    pass

  def test_negative_value(self):
    pass

  def test_parameter_type(self):
    pass
```

test_circles.py



```python
import unittest
from math import pi
from circles import circle_area

class TestCircleArea(unittest.TestCase):
  def test_area_result(self):
    pass

  def test_negative_value(self):
    pass

  def test_parameter_type(self):
    pass
```

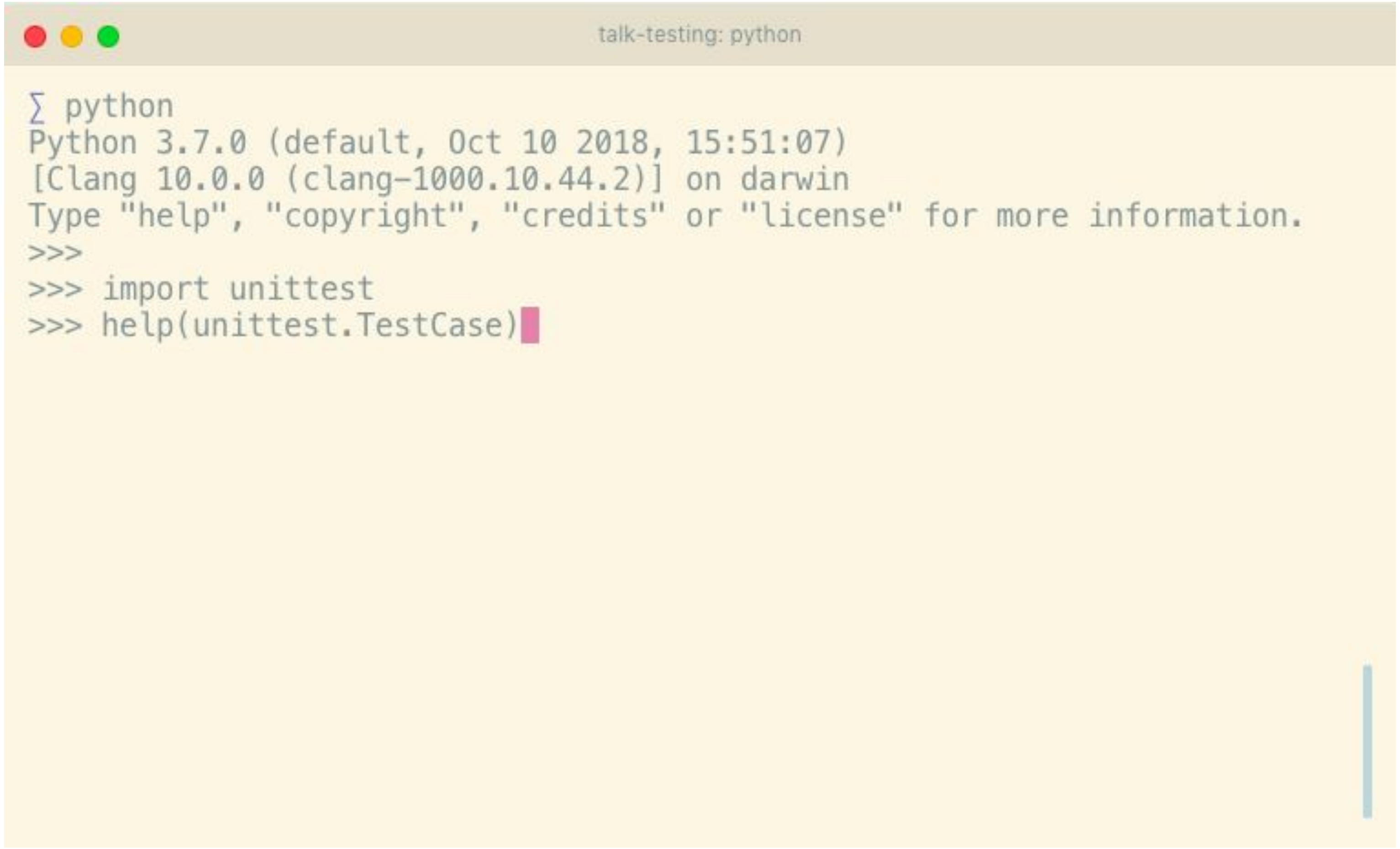
test_circles.py

```
import unittest
from math import pi
from circles import circle_area

class TestCircleArea(unittest.TestCase):
  def test_area_result(self):
    pass

  def test_negative_value(self):
    pass

  def test_parameter_type(self):
    pass
```

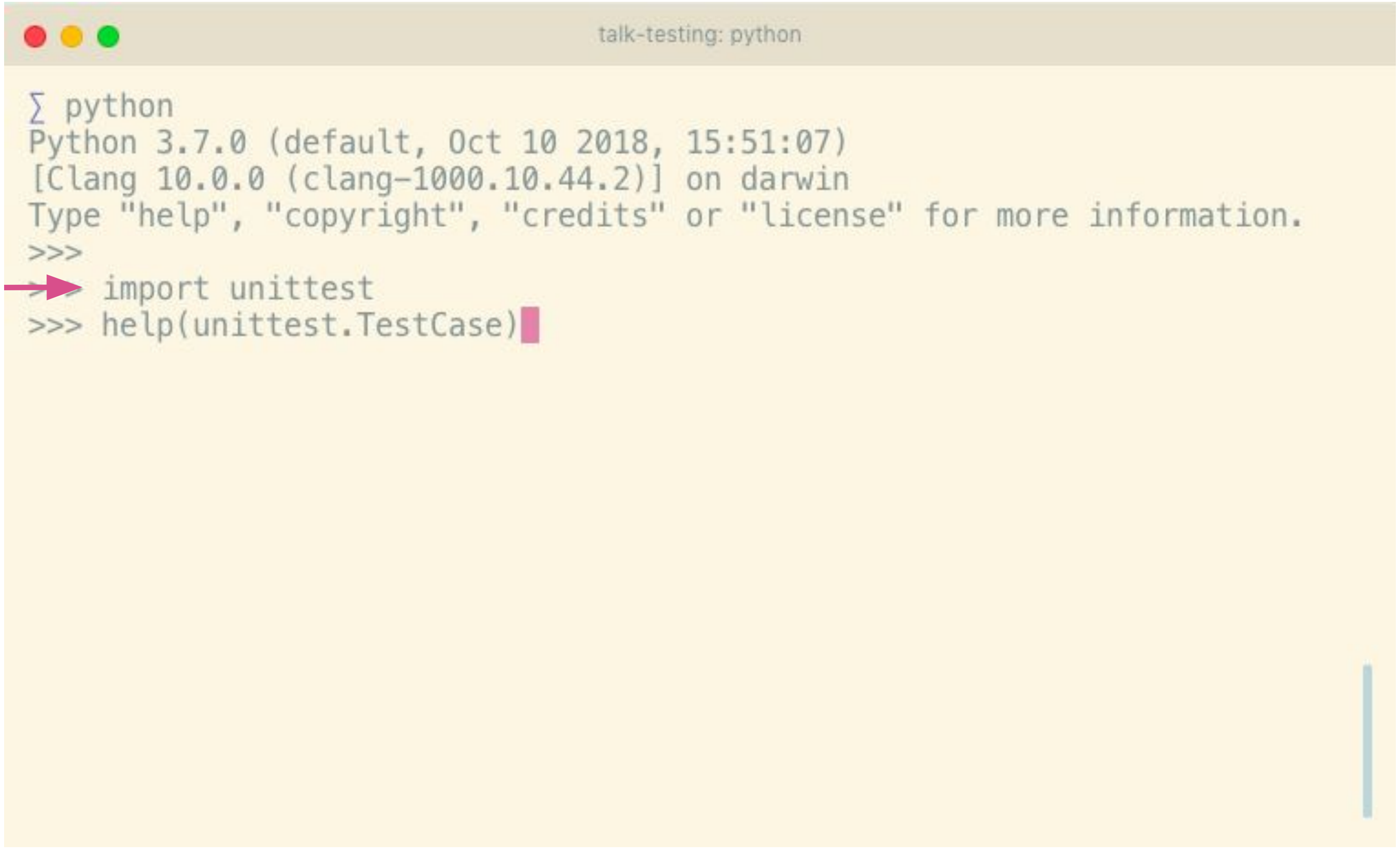
test_circles.py

```
import unittest
from math import pi
from circles import circle_area

class TestCircleArea(unittest.TestCase):
  def test_area_result(self):
    pass

  def test_negative_value(self):
    pass

  def test_parameter_type(self):
    pass
```

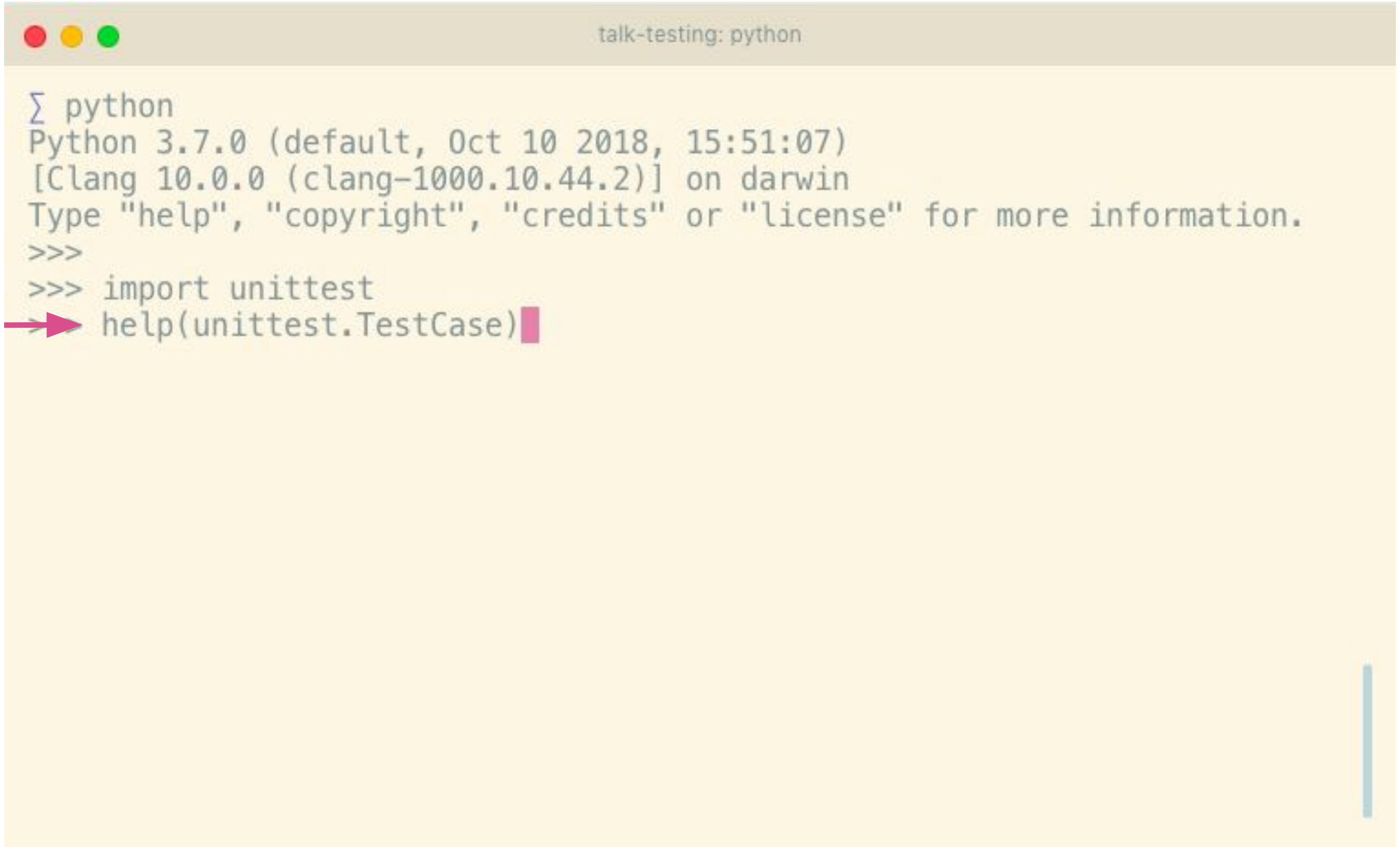
assert



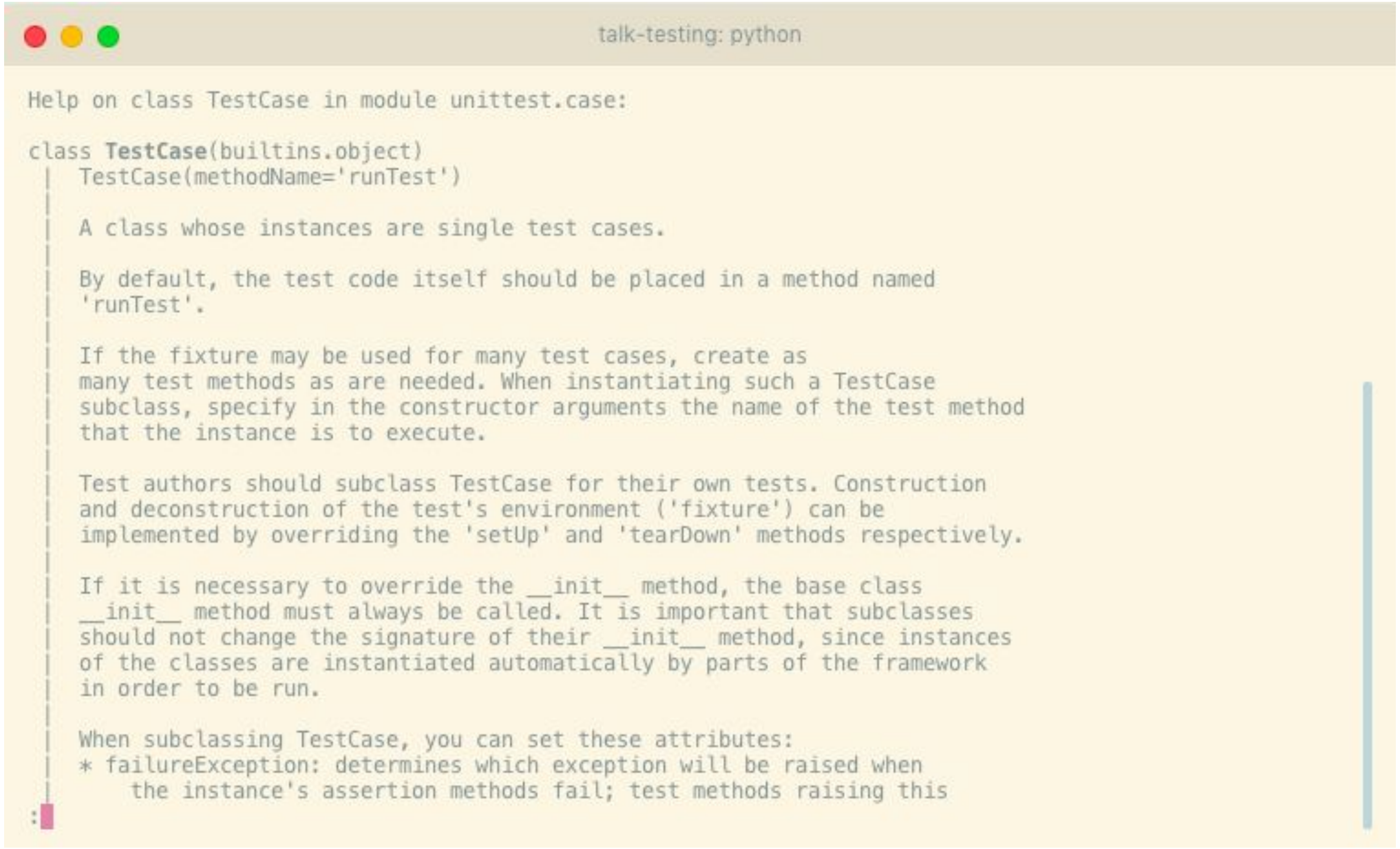
```
∑ python
Python 3.7.0 (default, Oct 10 2018, 15:51:07)
[Clang 10.0.0 (clang-1000.10.44.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import unittest
>>> help(unittest.TestCase)
```



```
∑ python
Python 3.7.0 (default, Oct 10 2018, 15:51:07)
[Clang 10.0.0 (clang-1000.10.44.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import unittest
>>> help(unittest.TestCase)
```

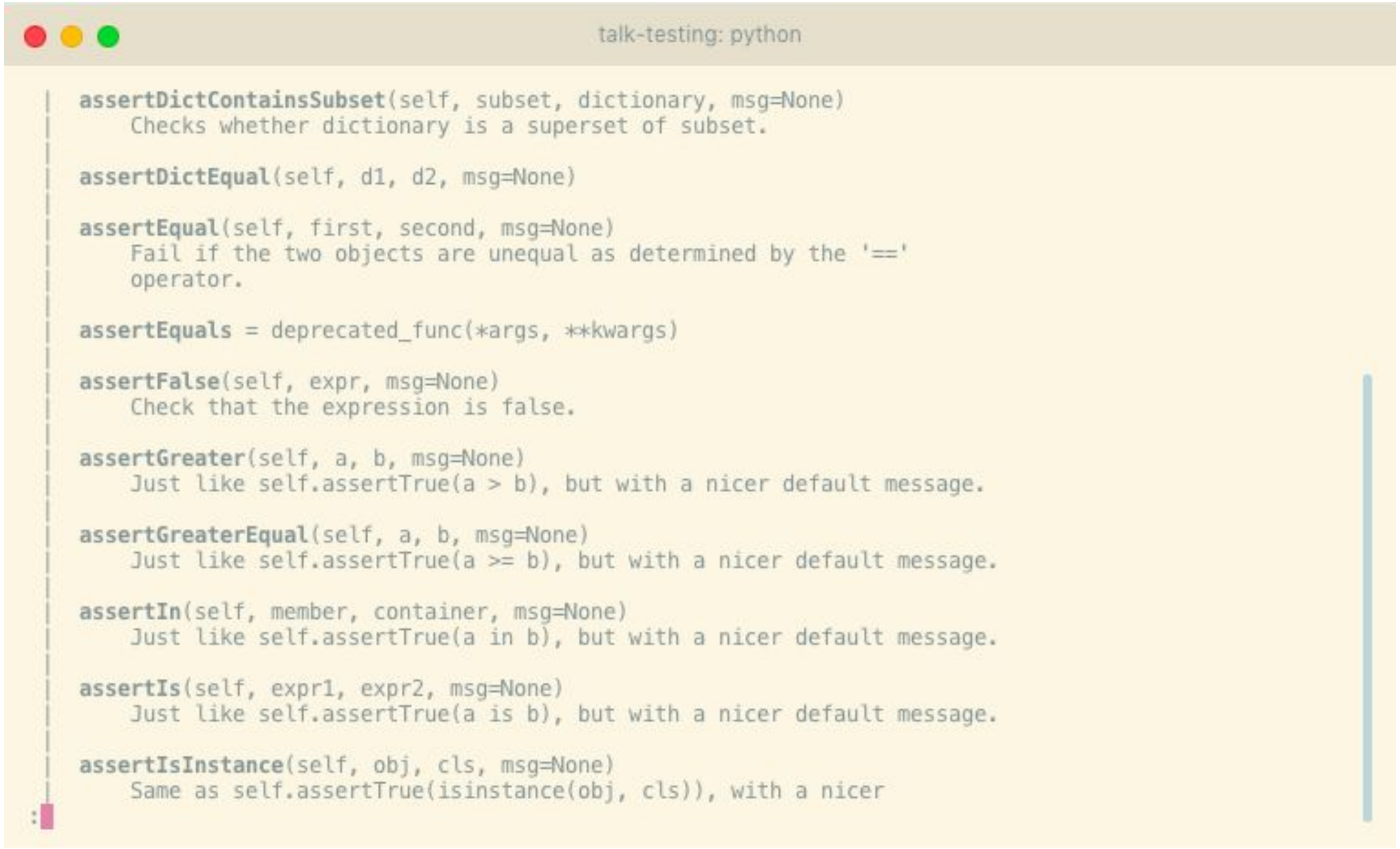


```
∑ python
Python 3.7.0 (default, Oct 10 2018, 15:51:07)
[Clang 10.0.0 (clang-1000.10.44.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import unittest
>>> help(unittest.TestCase)
```

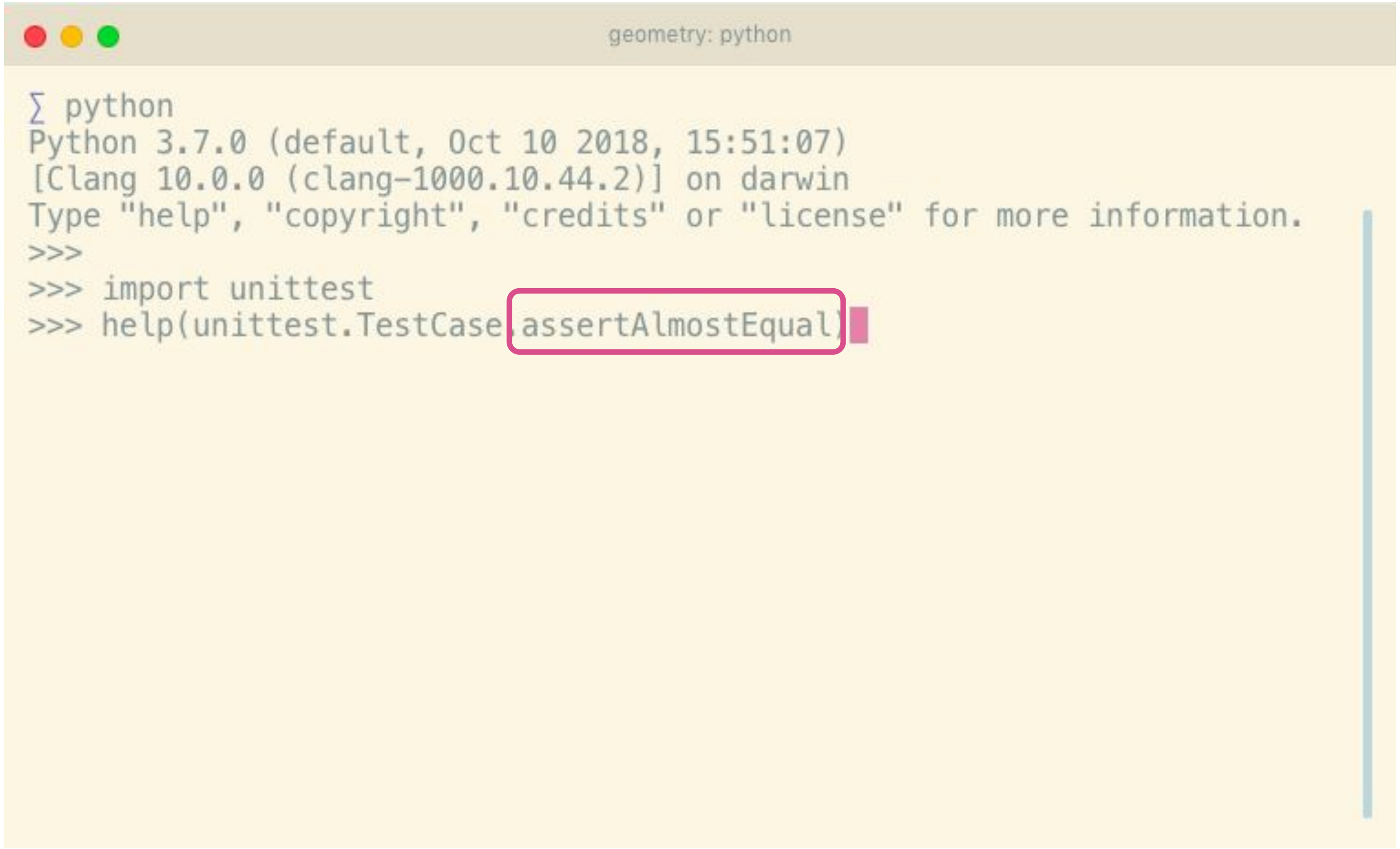


talk-testing: python

```
Help on class TestCase in module unittest.case:

class TestCase(builtins.object)
 |  TestCase(methodName='runTest')
 |
 |  A class whose instances are single test cases.
 |
 |  By default, the test code itself should be placed in a method named
 |  'runTest'.
 |
 |  If the fixture may be used for many test cases, create as
 |  many test methods as are needed. When instantiating such a TestCase
 |  subclass, specify in the constructor arguments the name of the test method
 |  that the instance is to execute.
 |
 |  Test authors should subclass TestCase for their own tests. Construction
 |  and deconstruction of the test's environment ('fixture') can be
 |  implemented by overriding the 'setUp' and 'tearDown' methods respectively.
 |
 |  If it is necessary to override the __init__ method, the base class
 |  __init__ method must always be called. It is important that subclasses
 |  should not change the signature of their __init__ method, since instances
 |  of the classes are instantiated automatically by parts of the framework
 |  in order to be run.
 |
 |  When subclassing TestCase, you can set these attributes:
 |  * failureException: determines which exception will be raised when
 |      the instance's assertion methods fail; test methods raising this
:
```

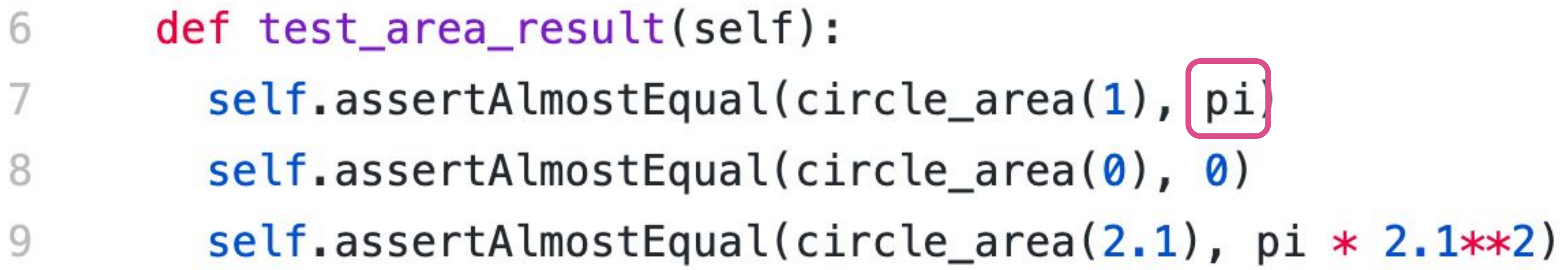


```
|  assertDictContainsSubset(self, subset, dictionary, msg=None)
|      Checks whether dictionary is a superset of subset.
|
|  assertDictEqual(self, d1, d2, msg=None)
|
|  assertEqual(self, first, second, msg=None)
|      Fail if the two objects are unequal as determined by the '=='
|      operator.
|
|  assertEquals = deprecated_func(*args, **kwargs)
|
|  assertFalse(self, expr, msg=None)
|      Check that the expression is false.
|
|  assertGreater(self, a, b, msg=None)
|      Just like self.assertTrue(a > b), but with a nicer default message.
|
|  assertGreaterEqual(self, a, b, msg=None)
|      Just like self.assertTrue(a >= b), but with a nicer default message.
|
|  assertIn(self, member, container, msg=None)
|      Just like self.assertTrue(a in b), but with a nicer default message.
|
|  assertIs(self, expr1, expr2, msg=None)
|      Just like self.assertTrue(a is b), but with a nicer default message.
|
|  assertIsInstance(self, obj, cls, msg=None)
|      Same as self.assertTrue(isinstance(obj, cls)), with a nicer
:
```

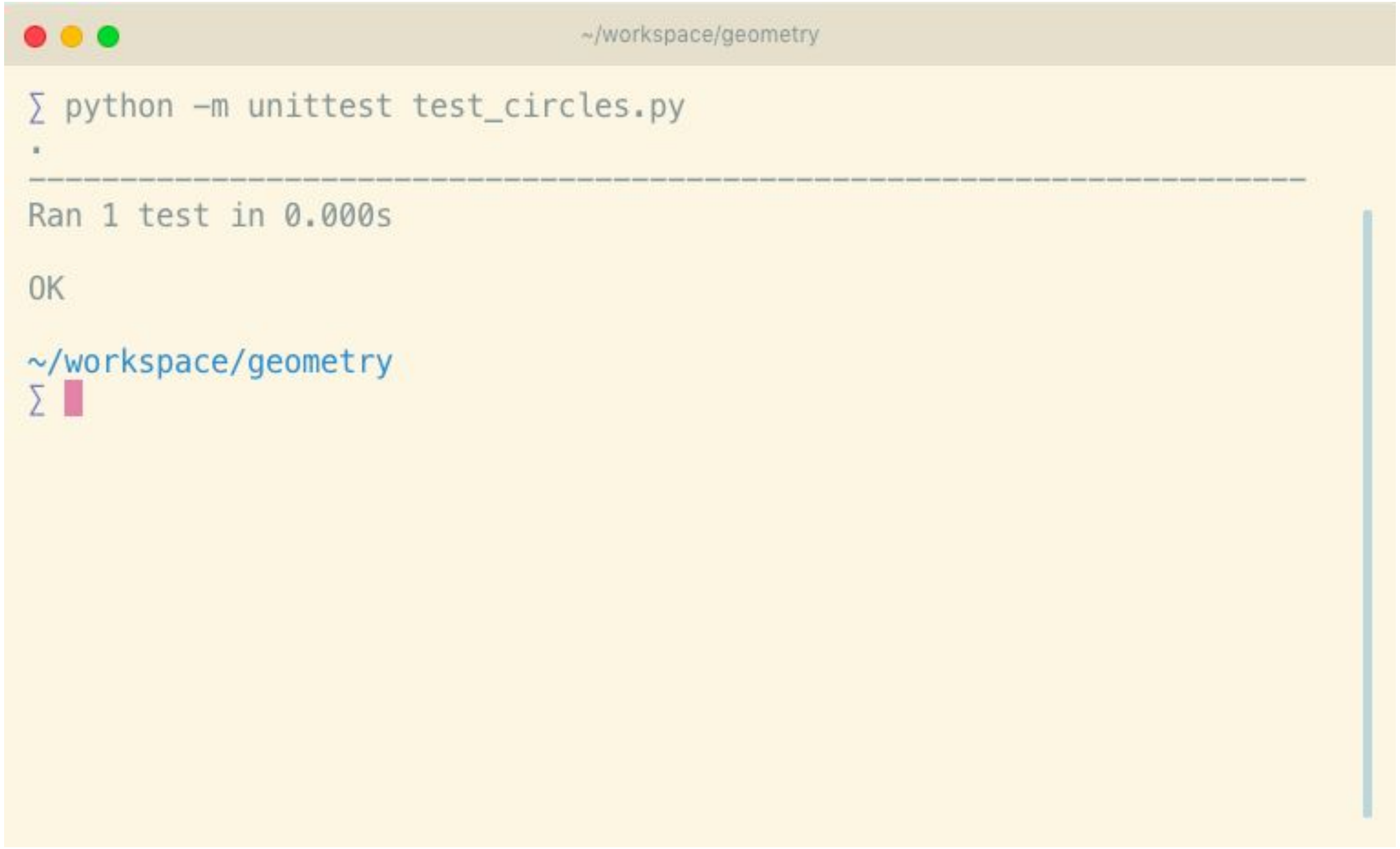


```
∑ python
Python 3.7.0 (default, Oct 10 2018, 15:51:07)
[Clang 10.0.0 (clang-1000.10.44.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import unittest
>>> help(unittest.TestCase.assertAlmostEqual)
```

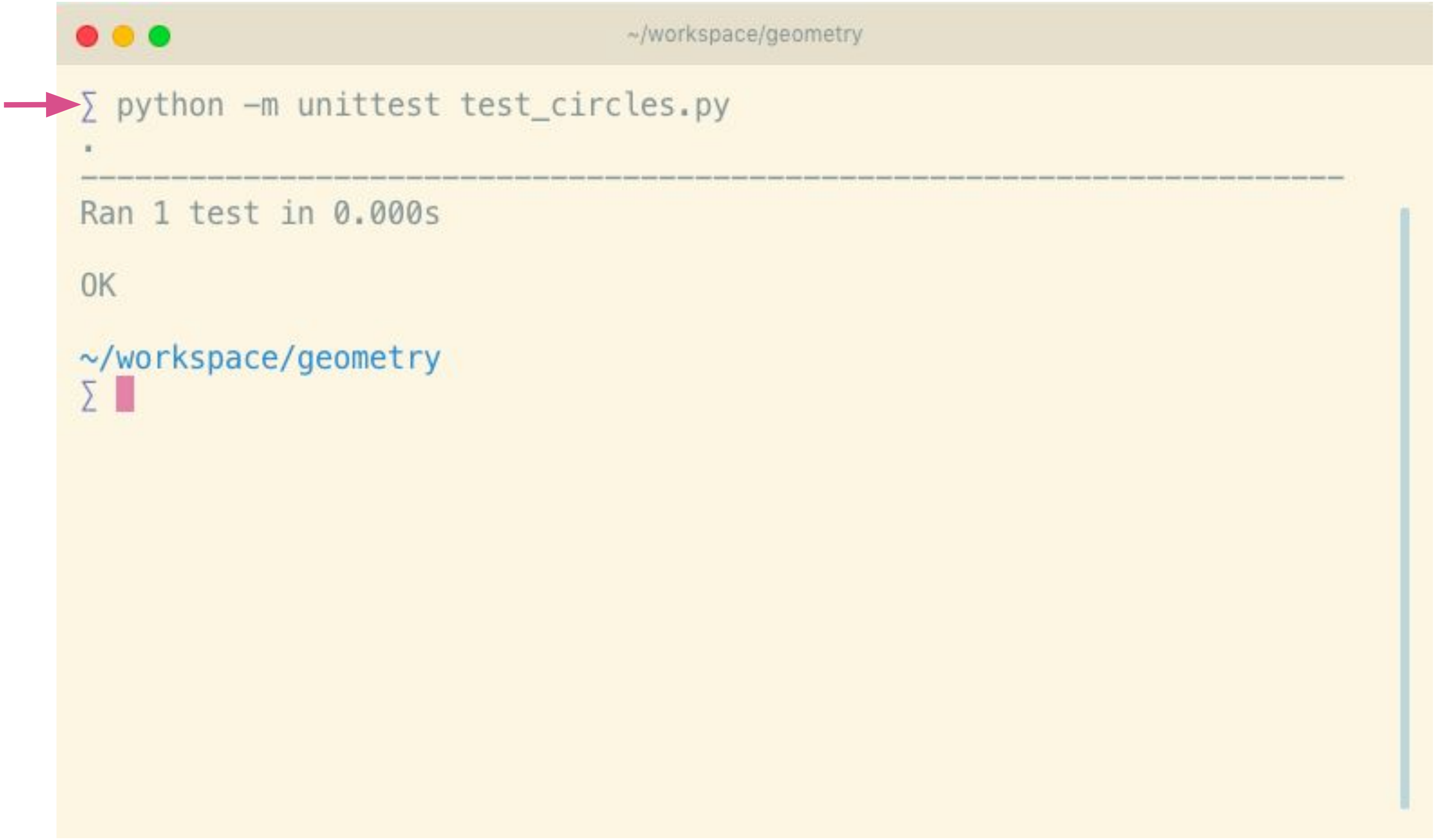
```python
  def test_area_result(self):
    self.assertAlmostEqual(circle_area(1), pi)
    self.assertAlmostEqual(circle_area(0), 0)
    self.assertAlmostEqual(circle_area(2.1), pi * 2.1**2)
    def test_area_result(self):
      self.assertAlmostEqual(circle_area(1), pi)
      self.assertAlmostEqual(circle_area(0), 0)
      self.assertAlmostEqual(circle_area(2.1), pi * 2.1**2)
```



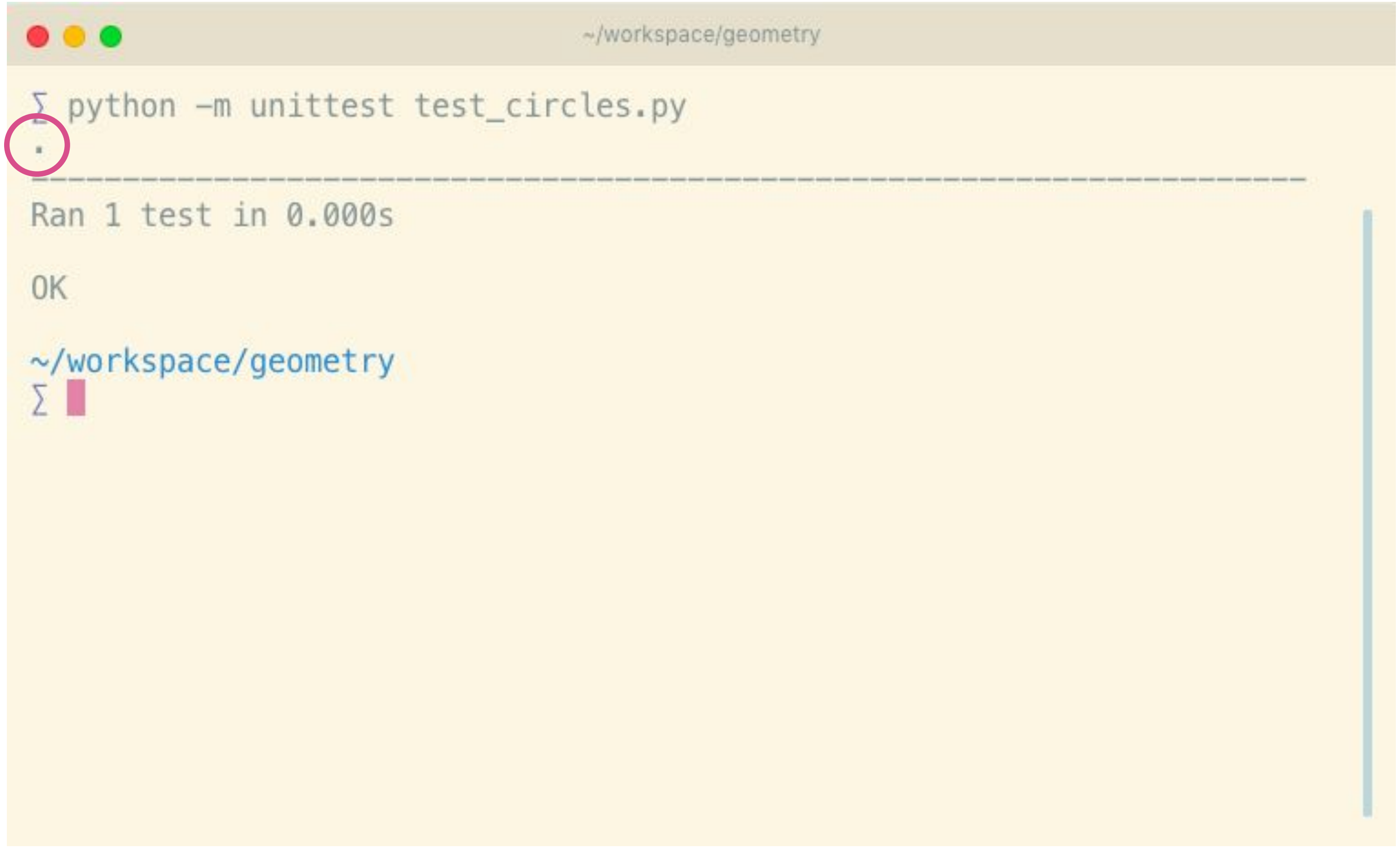
```
  def test_area_result(self):
    self.assertAlmostEqual(circle_area(1), pi)
    self.assertAlmostEqual(circle_area(0), 0)
    self.assertAlmostEqual(circle_area(2.1), pi * 2.1**2)
```


~/workspace/geometry

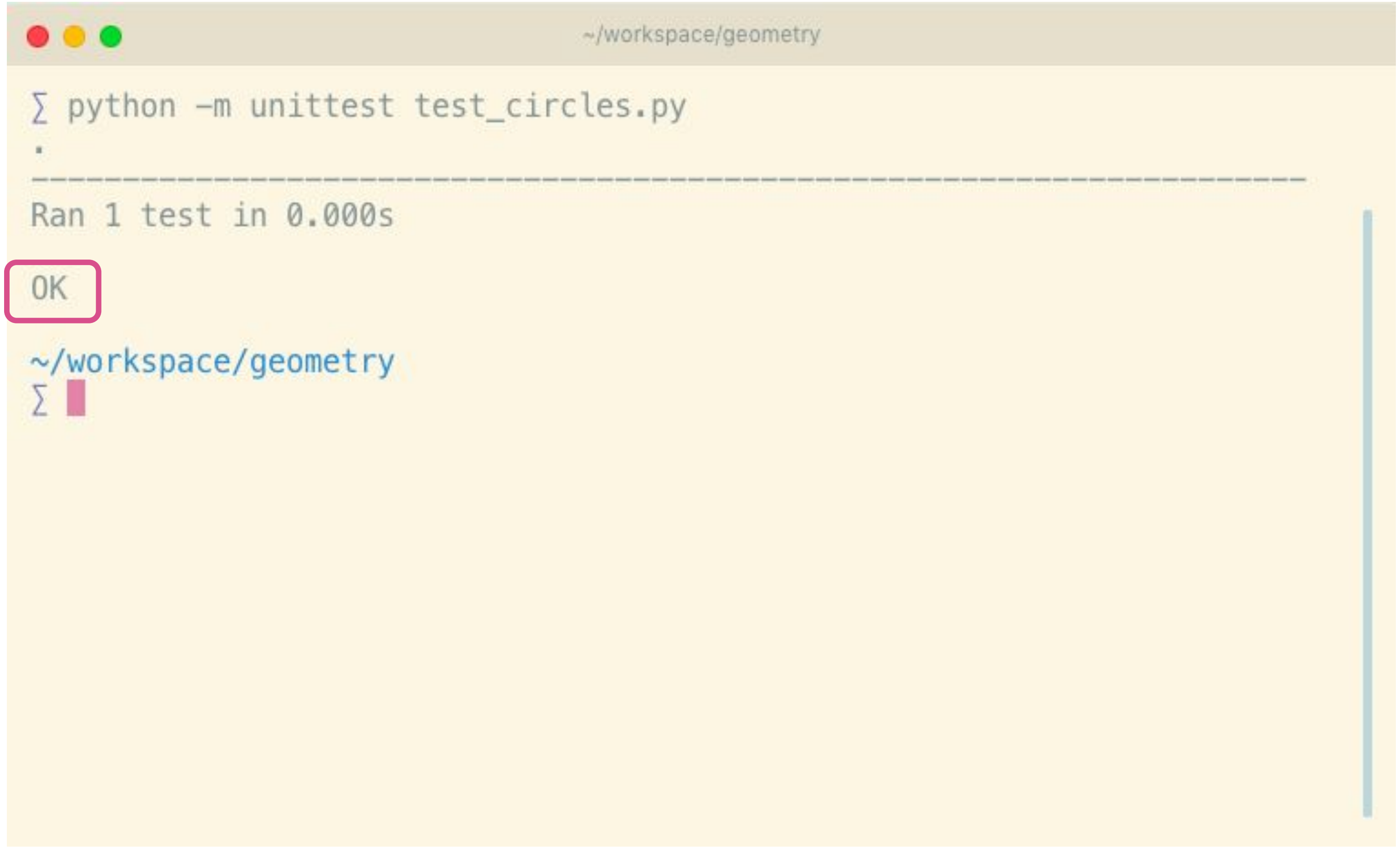
```
∑ python -m unittest test_circles.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK

~/workspace/geometry
∑ 
```



```
∑ python -m unittest test_circles.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK

~/workspace/geometry
∑ 
```



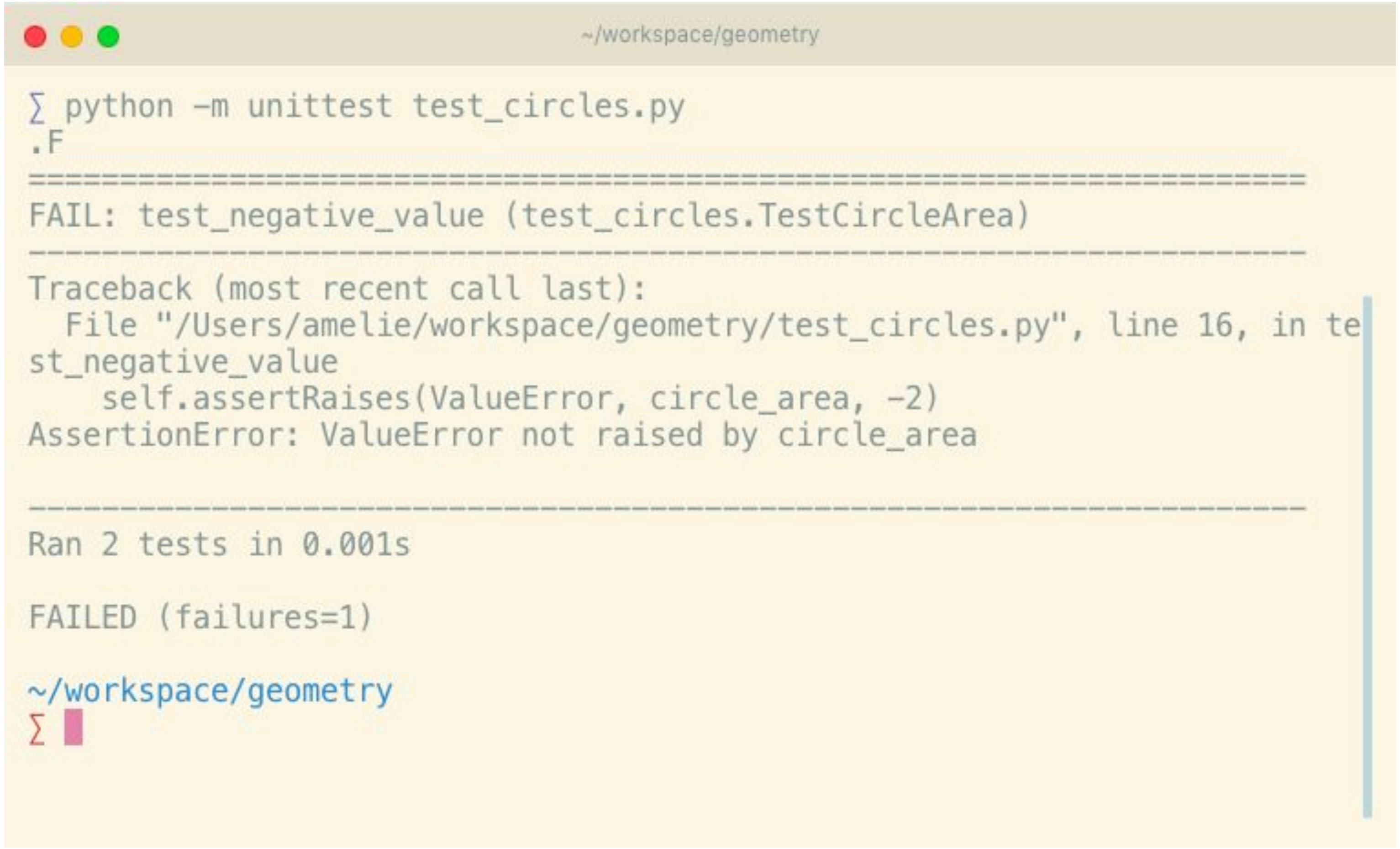
```
∑ python -m unittest test_circles.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK

~/workspace/geometry
∑
```



~/workspace/geometry

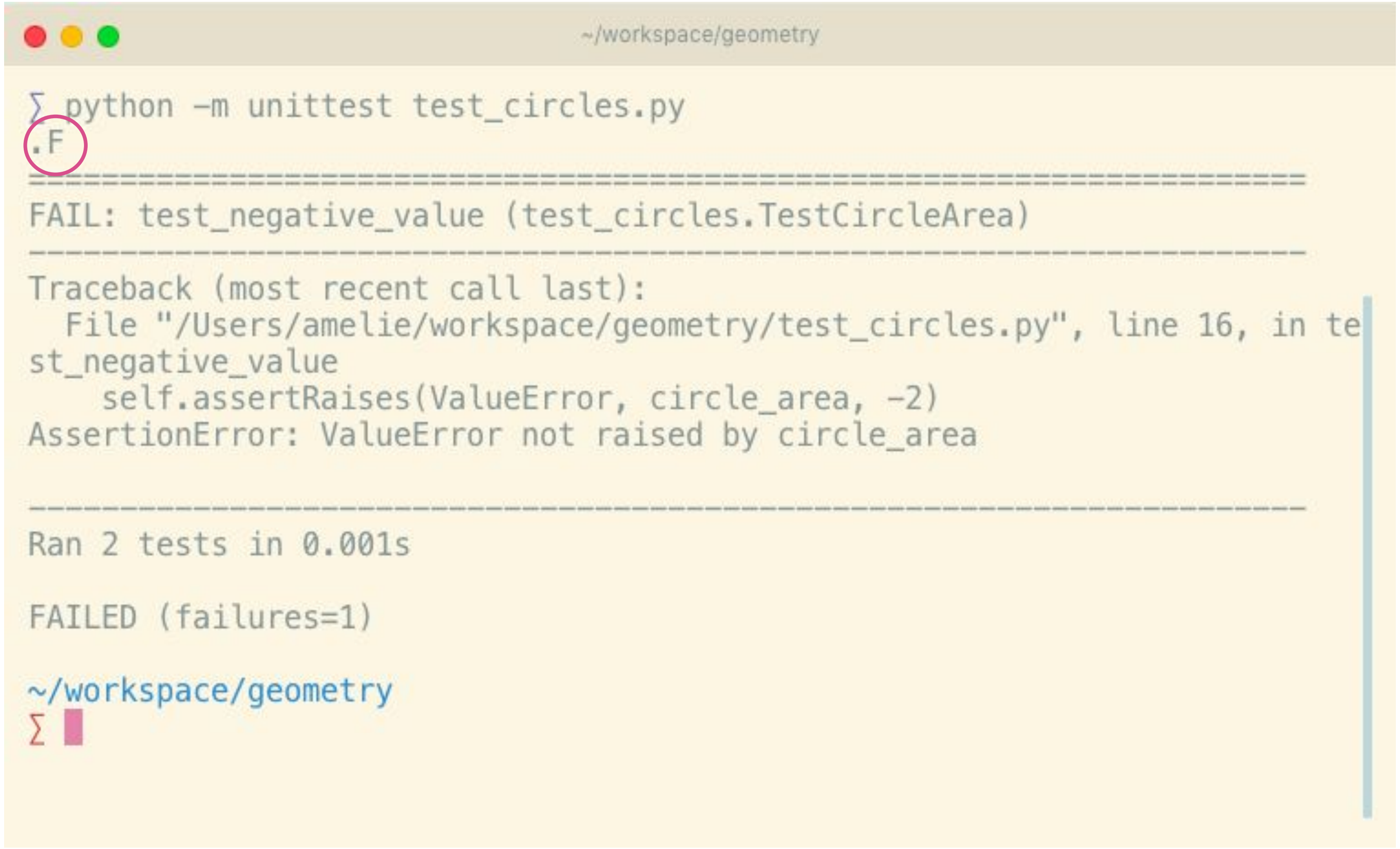
```
∑ python -m unittest test_circles.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK

~/workspace/geometry
∑
```

```python
    def test_negative_value(self):
      self.assertRaises(ValueError, circle_area, -2)
```

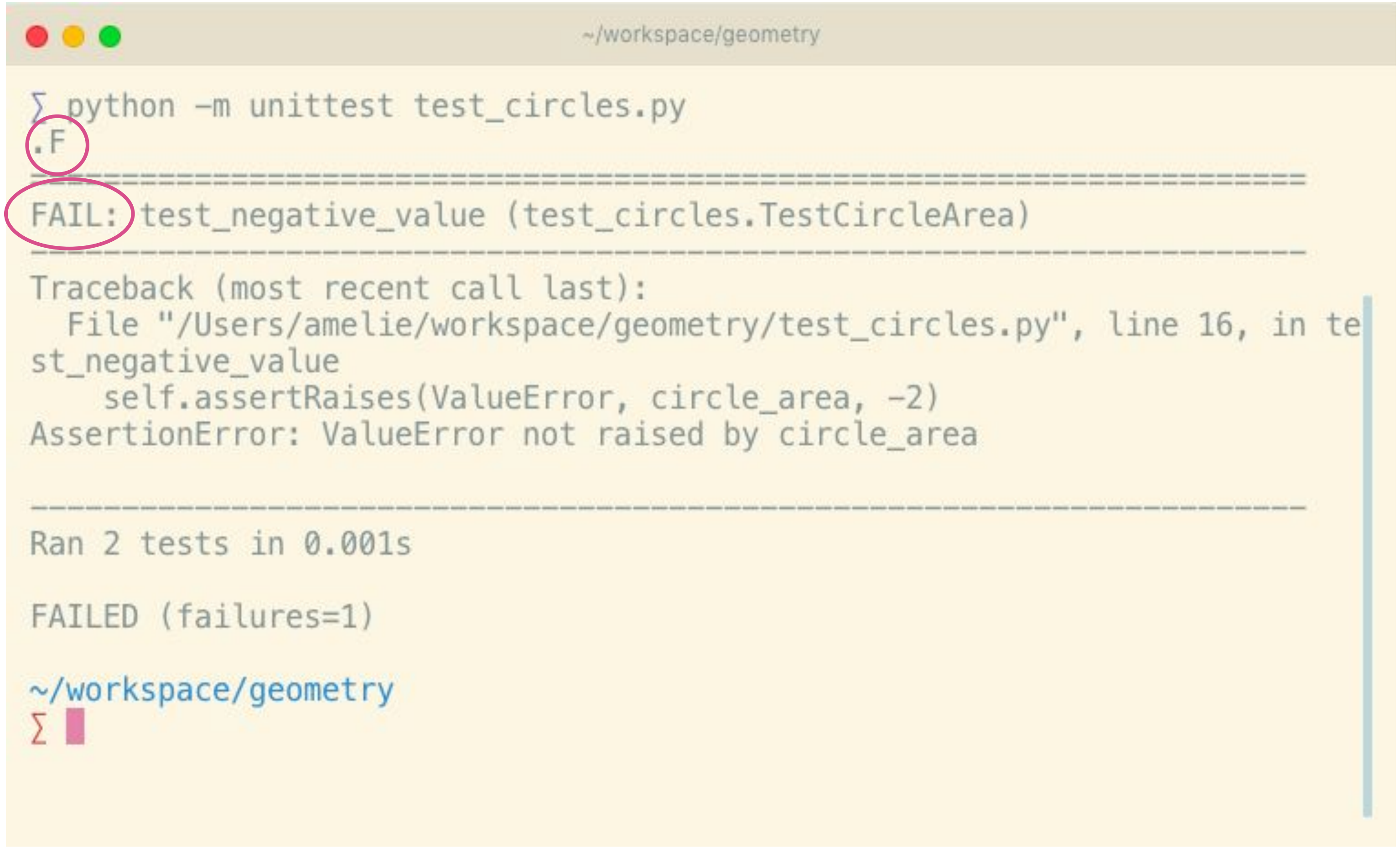


~/workspace/geometry

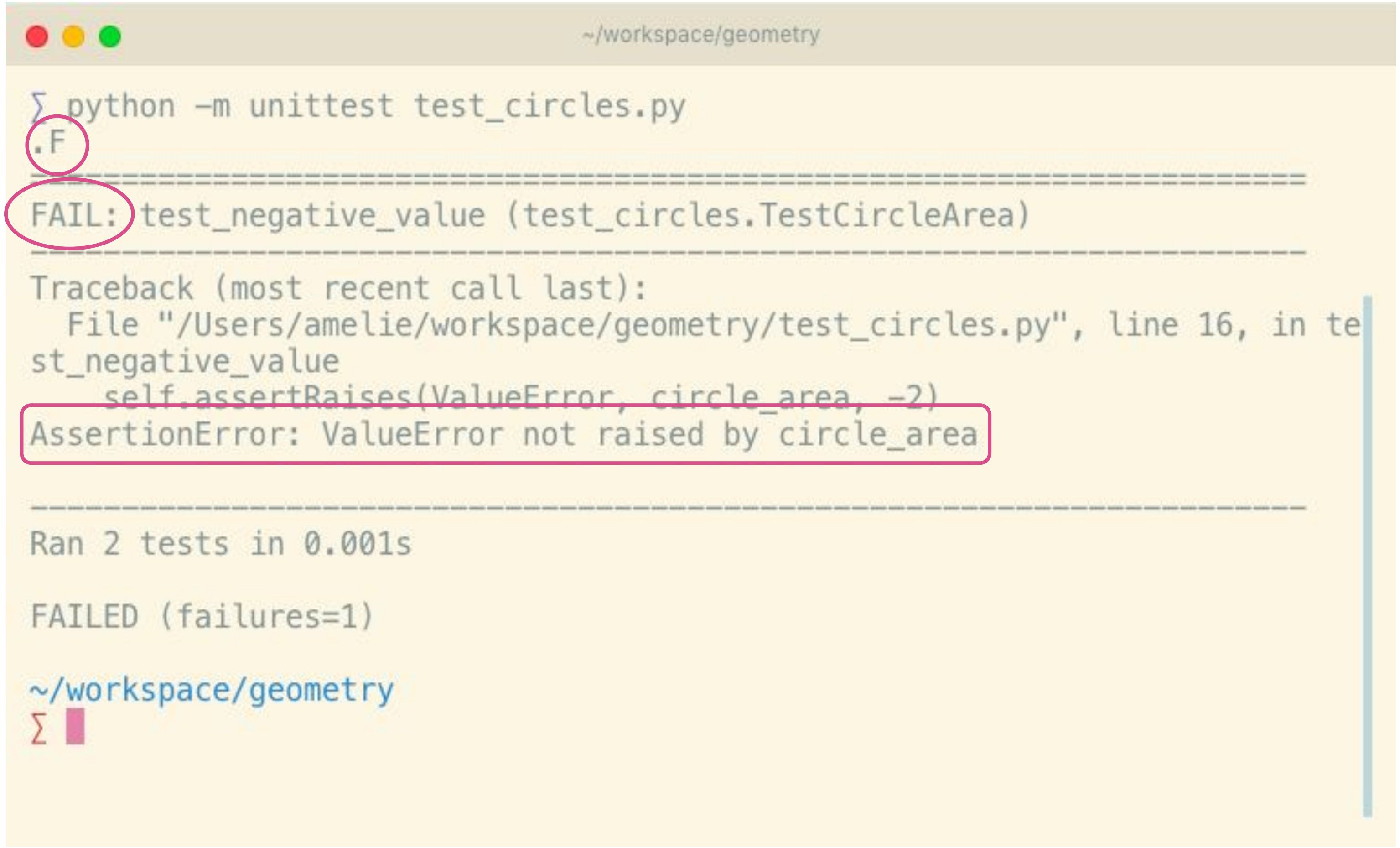
```
∑ python -m unittest test_circles.py
.F
======================================================================
FAIL: test_negative_value (test_circles.TestCircleArea)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/amelie/workspace/geometry/test_circles.py", line 16, in te
st_negative_value
    self.assertRaises(ValueError, circle_area, -2)
AssertionError: ValueError not raised by circle_area


----------------------------------------------------------------------
Ran 2 tests in 0.001s

FAILED (failures=1)

~/workspace/geometry
∑
```



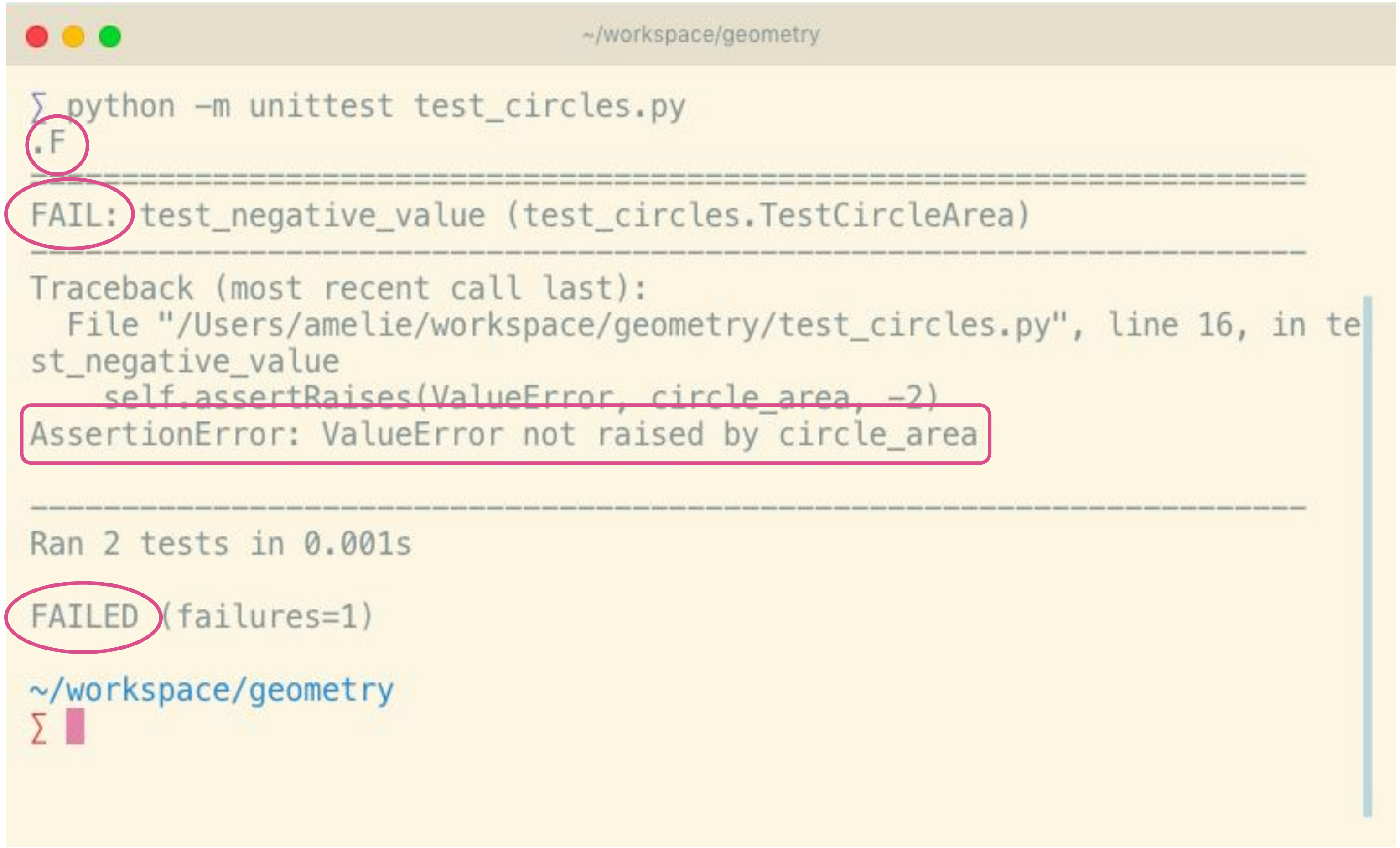
```
∑ python -m unittest test_circles.py
.F
======================================================================
FAIL: test_negative_value (test_circles.TestCircleArea)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/amelie/workspace/geometry/test_circles.py", line 16, in te
st_negative_value
    self.assertRaises(ValueError, circle_area, -2)
AssertionError: ValueError not raised by circle_area


----------------------------------------------------------------------
Ran 2 tests in 0.001s

FAILED (failures=1)

~/workspace/geometry
∑
```



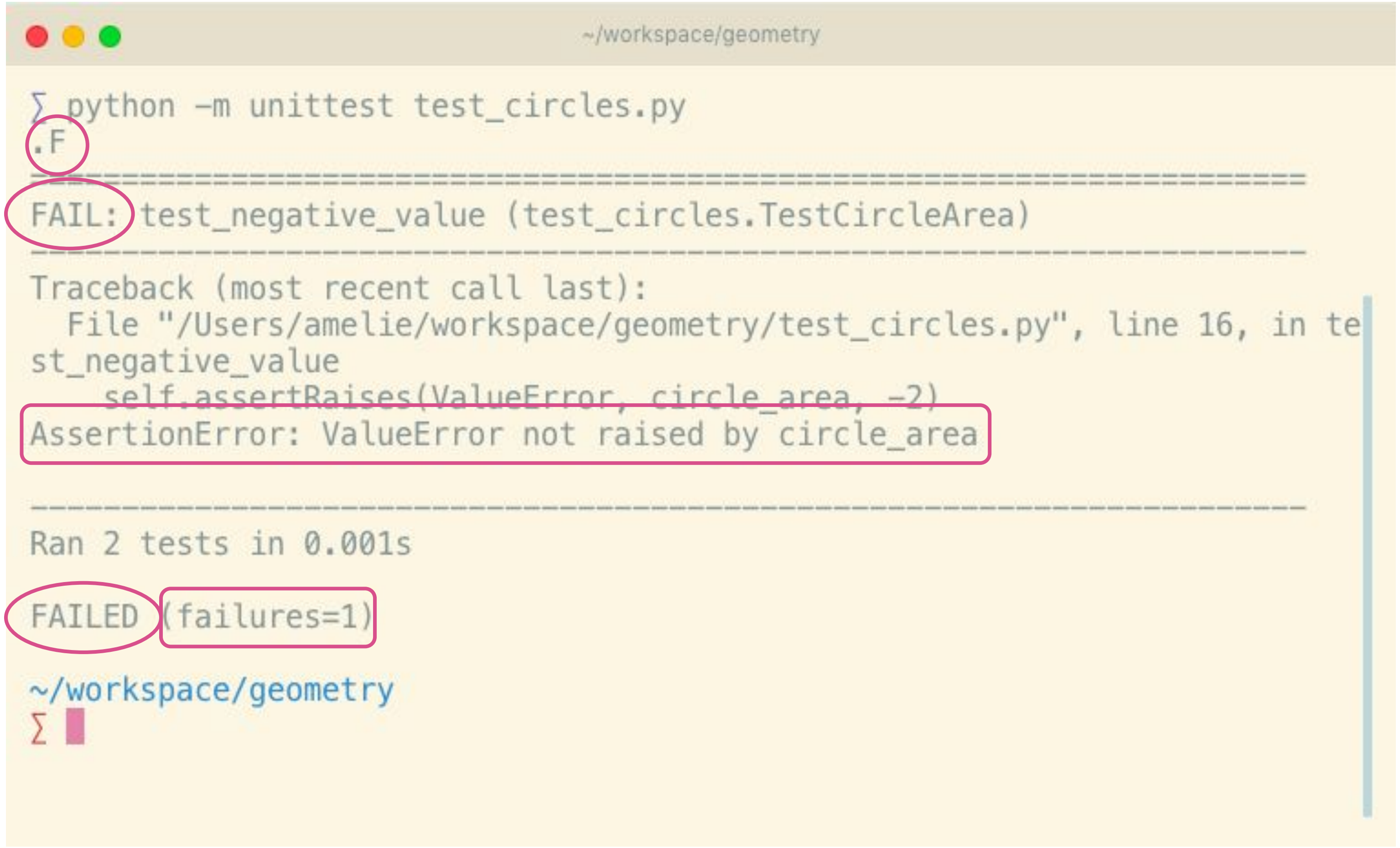
```
∑ python -m unittest test_circles.py
.F
======================================================================
FAIL: test_negative_value (test_circles.TestCircleArea)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/amelie/workspace/geometry/test_circles.py", line 16, in te
st_negative_value
    self.assertRaises(ValueError, circle_area, -2)
AssertionError: ValueError not raised by circle_area


----------------------------------------------------------------------
Ran 2 tests in 0.001s

FAILED (failures=1)

~/workspace/geometry
∑
```



```
∑ python -m unittest test_circles.py
.F
======================================================================
FAIL: test_negative_value (test_circles.TestCircleArea)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/amelie/workspace/geometry/test_circles.py", line 16, in te
st_negative_value
    self.assertRaises(ValueError, circle_area, -2)
AssertionError: ValueError not raised by circle_area

----------------------------------------------------------------------
Ran 2 tests in 0.001s

FAILED (failures=1)

~/workspace/geometry
∑
```

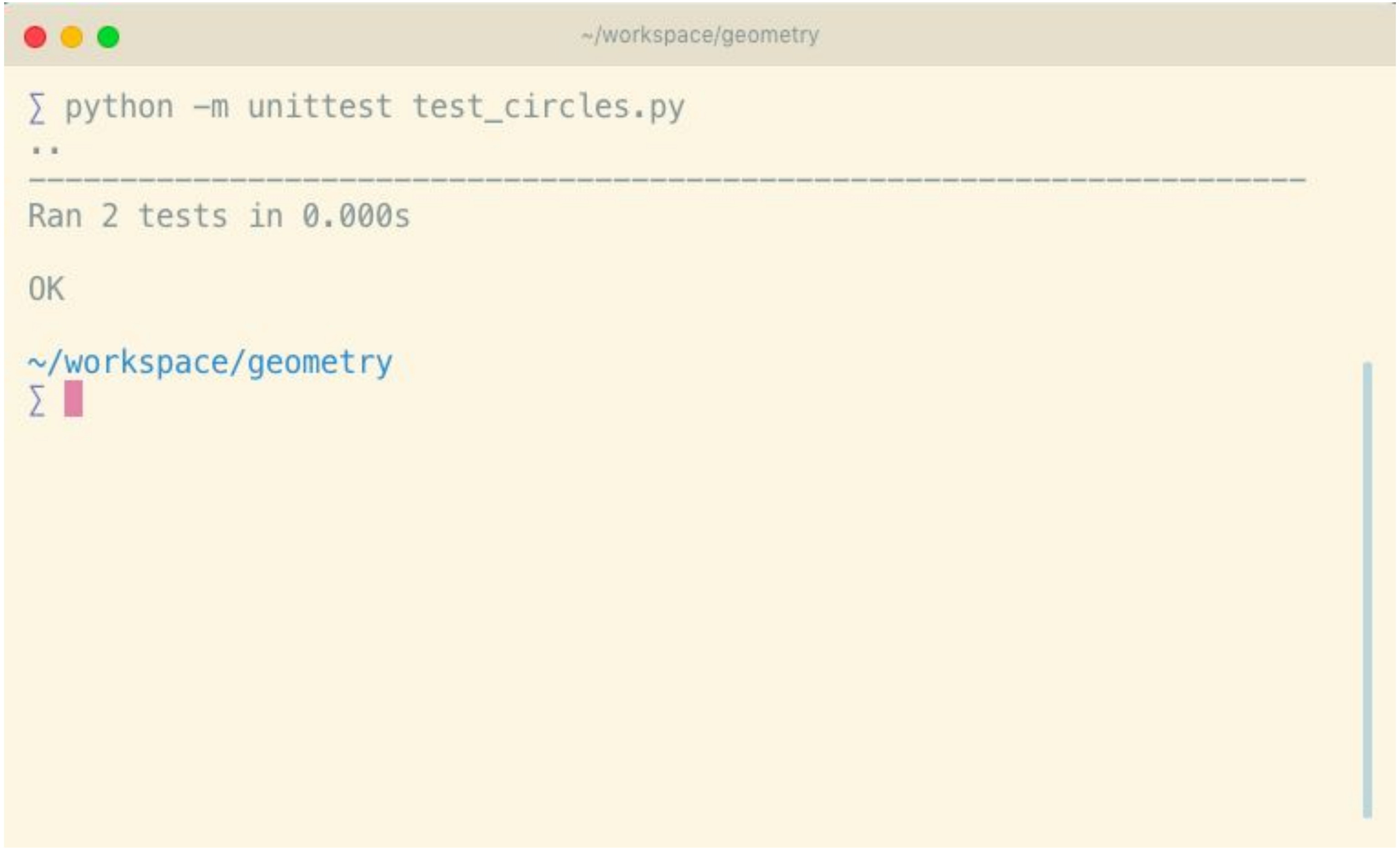


```
∑ python -m unittest test_circles.py
.F
======================================================================
FAIL: test_negative_value (test_circles.TestCircleArea)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/amelie/workspace/geometry/test_circles.py", line 16, in te
st_negative_value
    self.assertRaises(ValueError, circle_area, -2)
AssertionError: ValueError not raised by circle_area

----------------------------------------------------------------------
Ran 2 tests in 0.001s

FAILED (failures=1)

~/workspace/geometry
∑
```

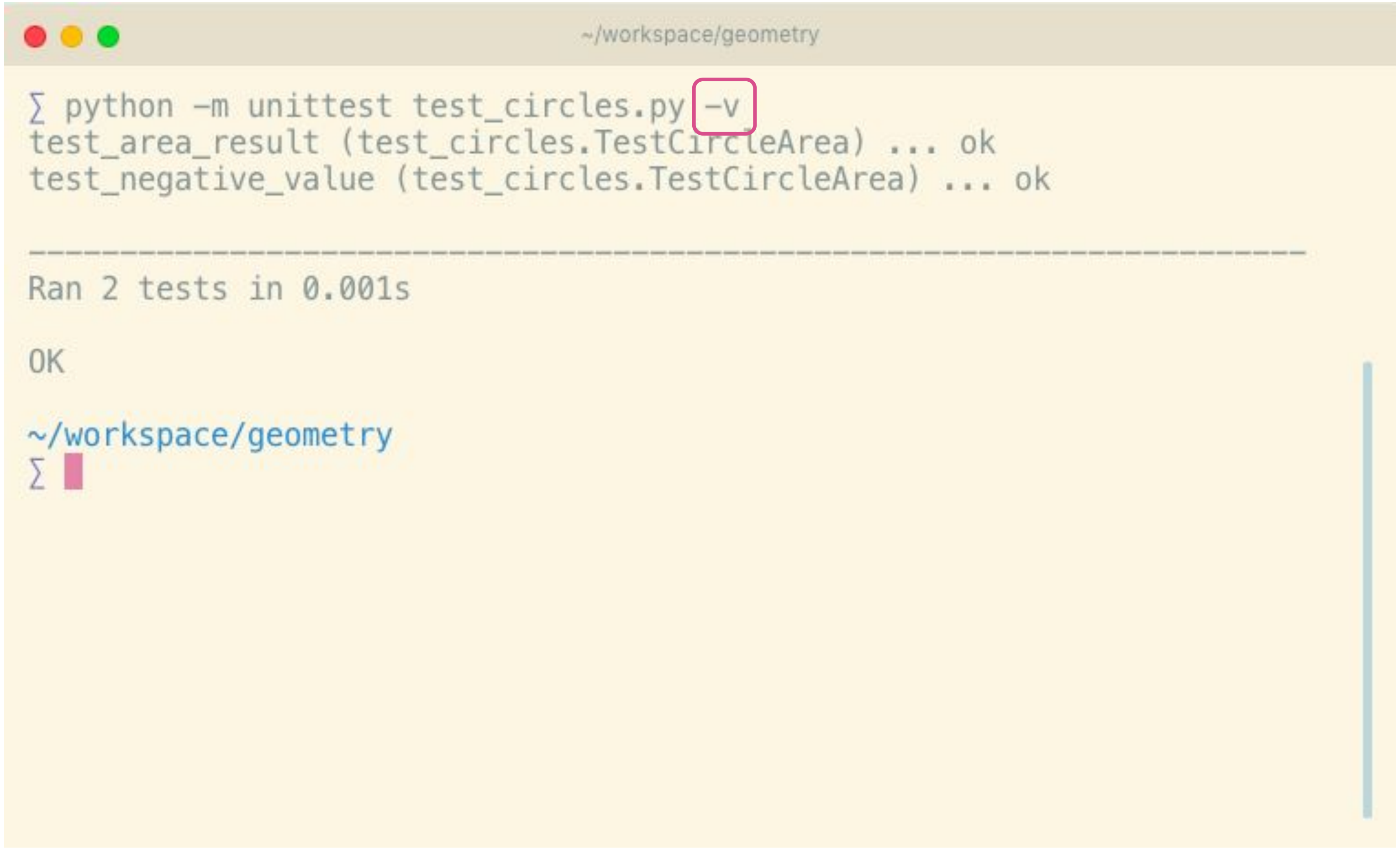


```
~/workspace/geometry

∑ python -m unittest test_circles.py
.F
======================================================================
FAIL: test_negative_value (test_circles.TestCircleArea)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/amelie/workspace/geometry/test_circles.py", line 16, in te
st_negative_value
    self.assertRaises(ValueError, circle_area, -2)
AssertionError: ValueError not raised by circle_area

----------------------------------------------------------------------
Ran 2 tests in 0.001s

FAILED (failures=1)

~/workspace/geometry
∑
```

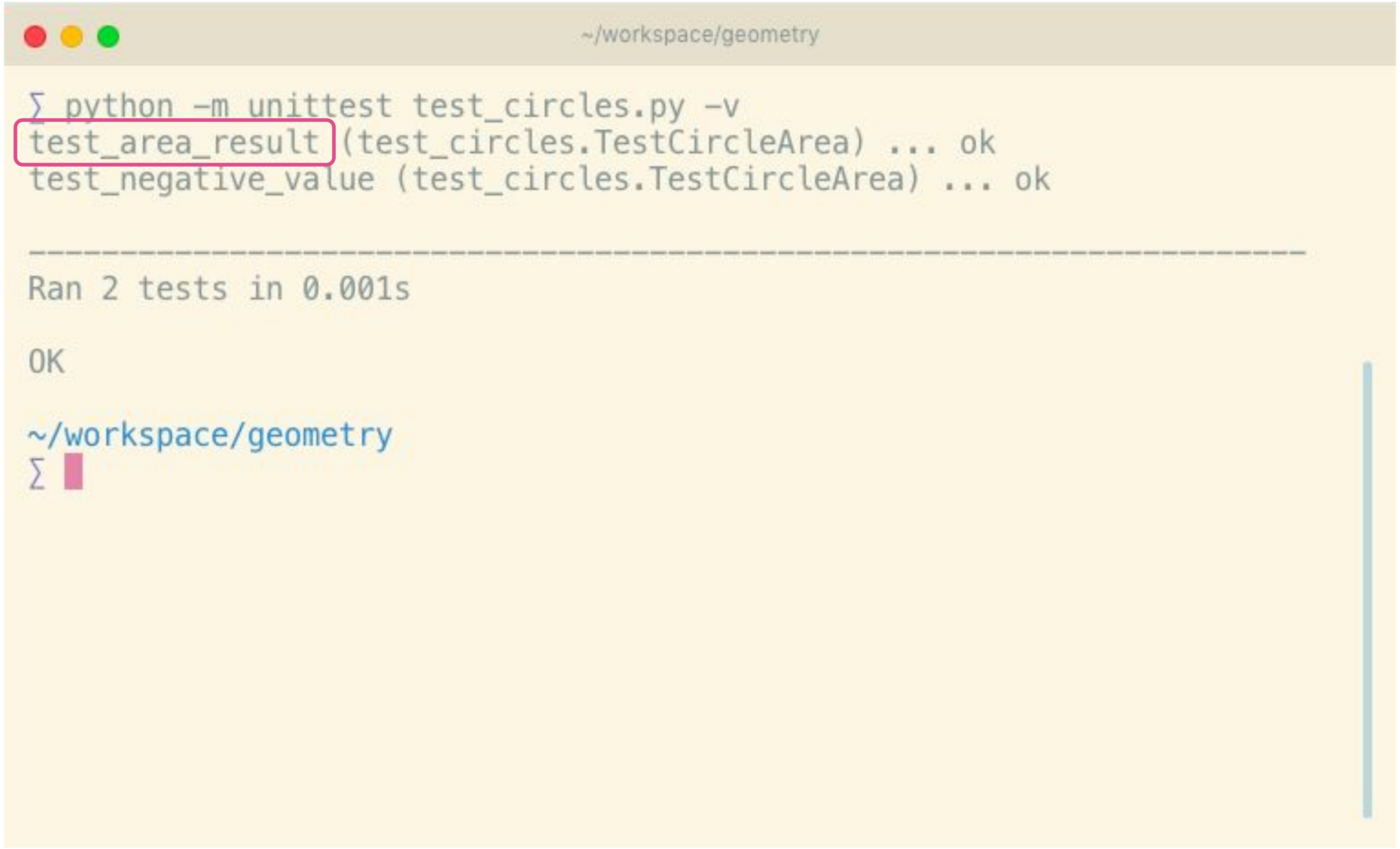
```python
from math import pi

def circle_area(r):
  if r < 0:
    raise ValueError("The radius cannot be negative.")
  return pi*(r**2)
```



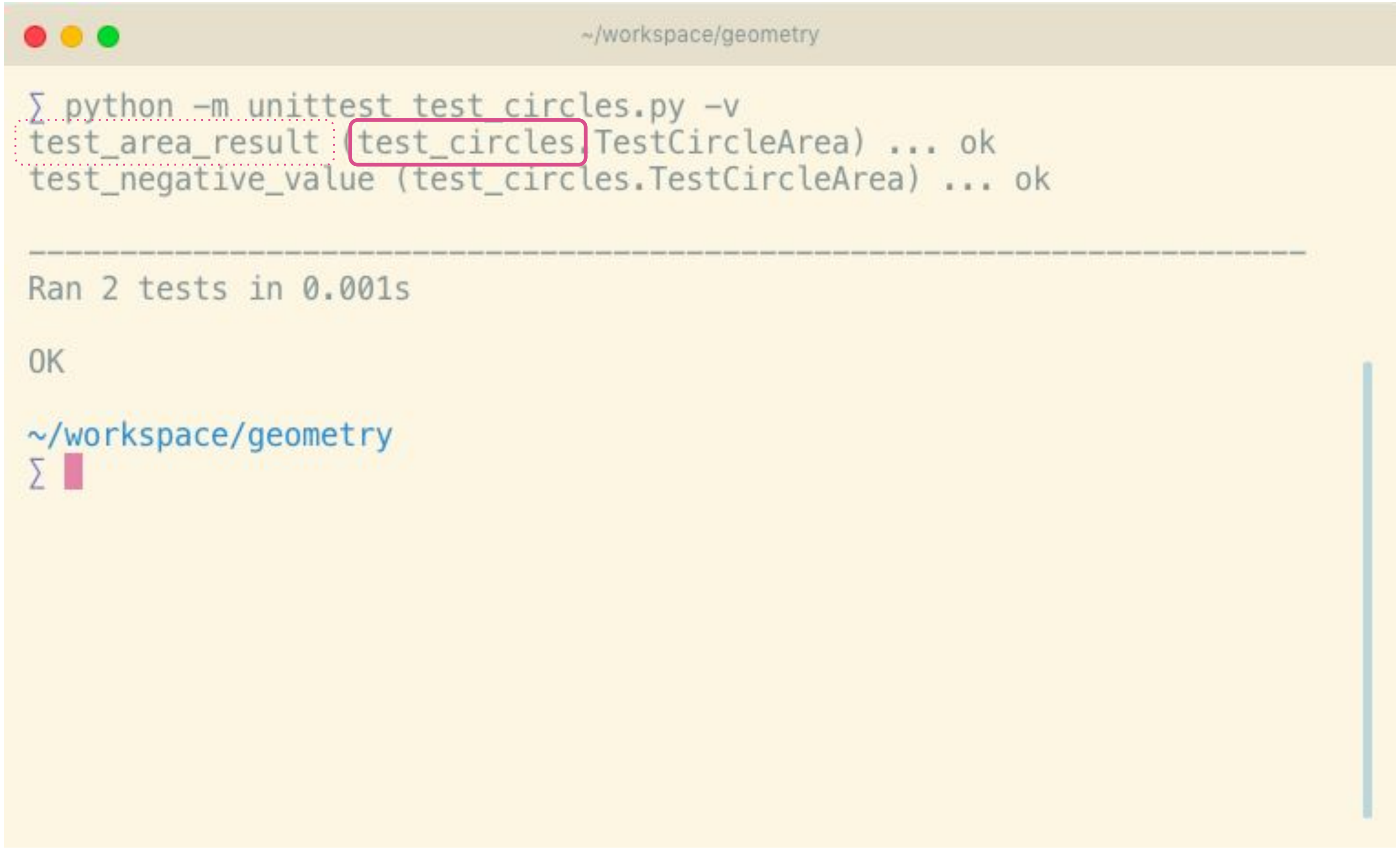
```
∑ python -m unittest test_circles.py
..
----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK

~/workspace/geometry
∑ 
```



```
∑ python -m unittest test_circles.py -v
test_area_result (test_circles.TestCircleArea) ... ok
test_negative_value (test_circles.TestCircleArea) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK

~/workspace/geometry
∑ 
```

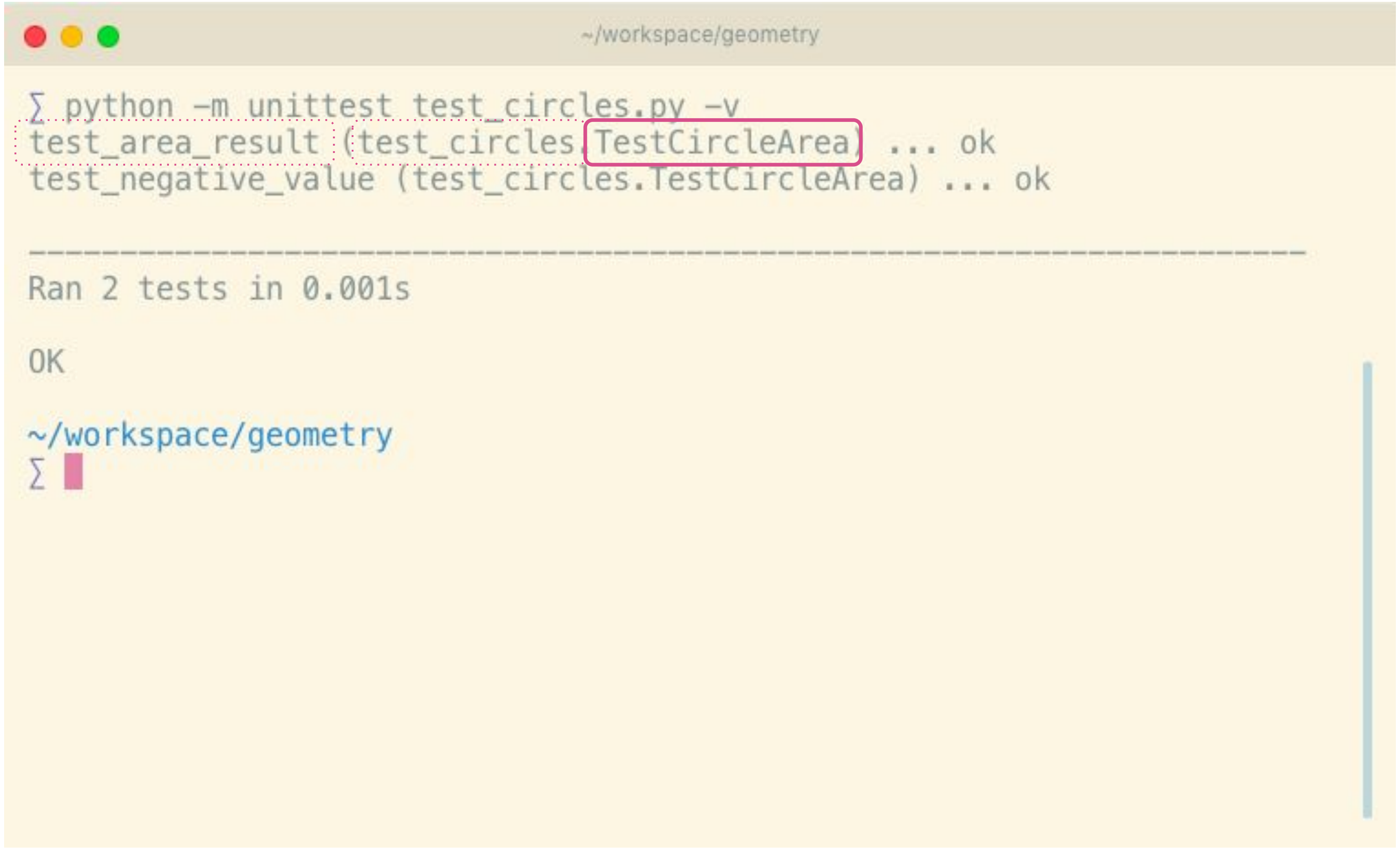

~/workspace/geometry

```
∑ python -m unittest test_circles.py -v
test_area_result (test_circles.TestCircleArea) ... ok
test_negative_value (test_circles.TestCircleArea) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK

~/workspace/geometry
∑ 
```



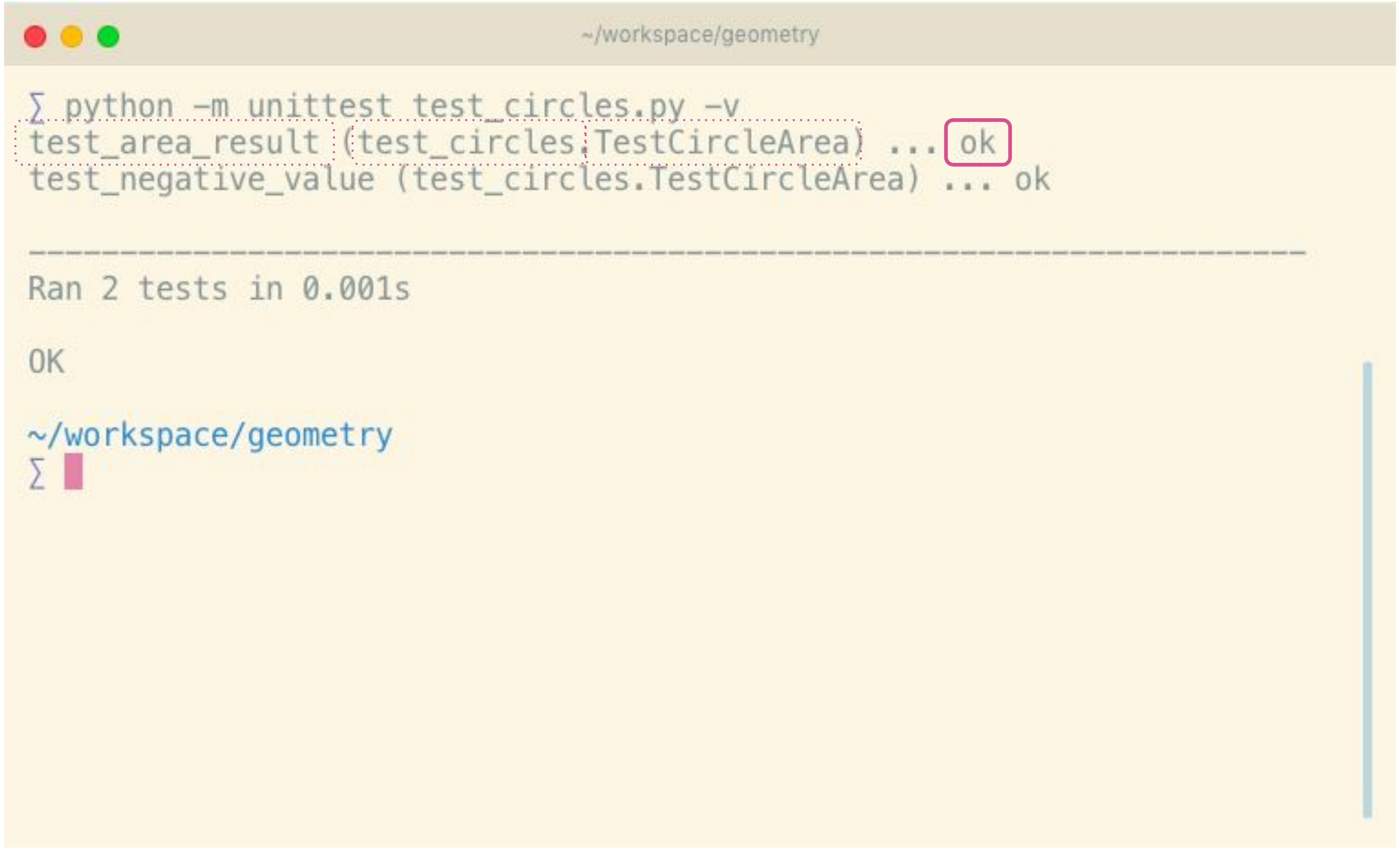
~/workspace/geometry

```
∑ python -m unittest test_circles.py -v
test_area_result (test_circles.TestCircleArea) ... ok
test_negative_value (test_circles.TestCircleArea) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK

~/workspace/geometry
∑ 
```



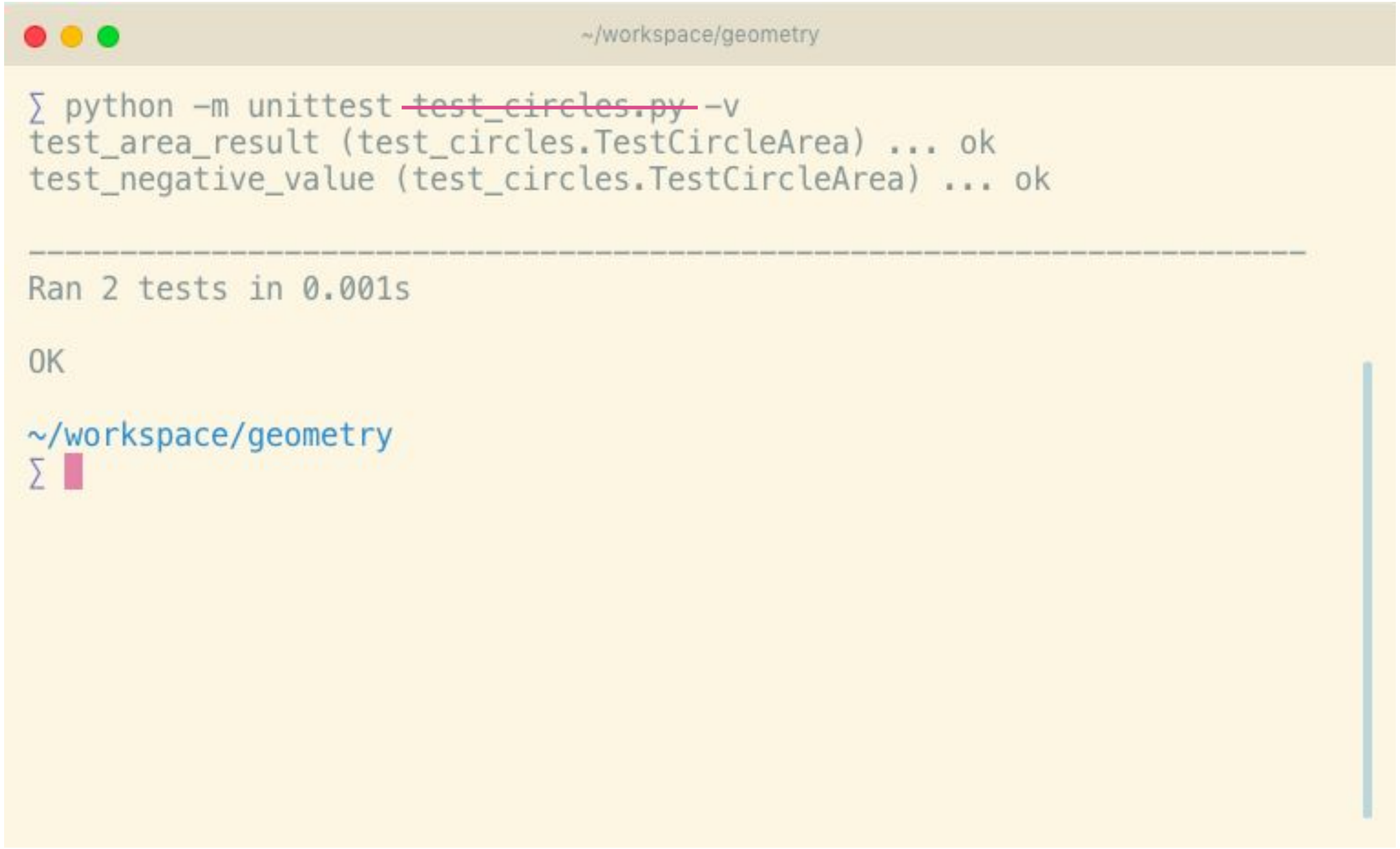
~/workspace/geometry

```
∑ python -m unittest test_circles.py -v
test_area_result (test_circles.TestCircleArea) ... ok
test_negative_value (test_circles.TestCircleArea) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK

~/workspace/geometry
∑ 
```

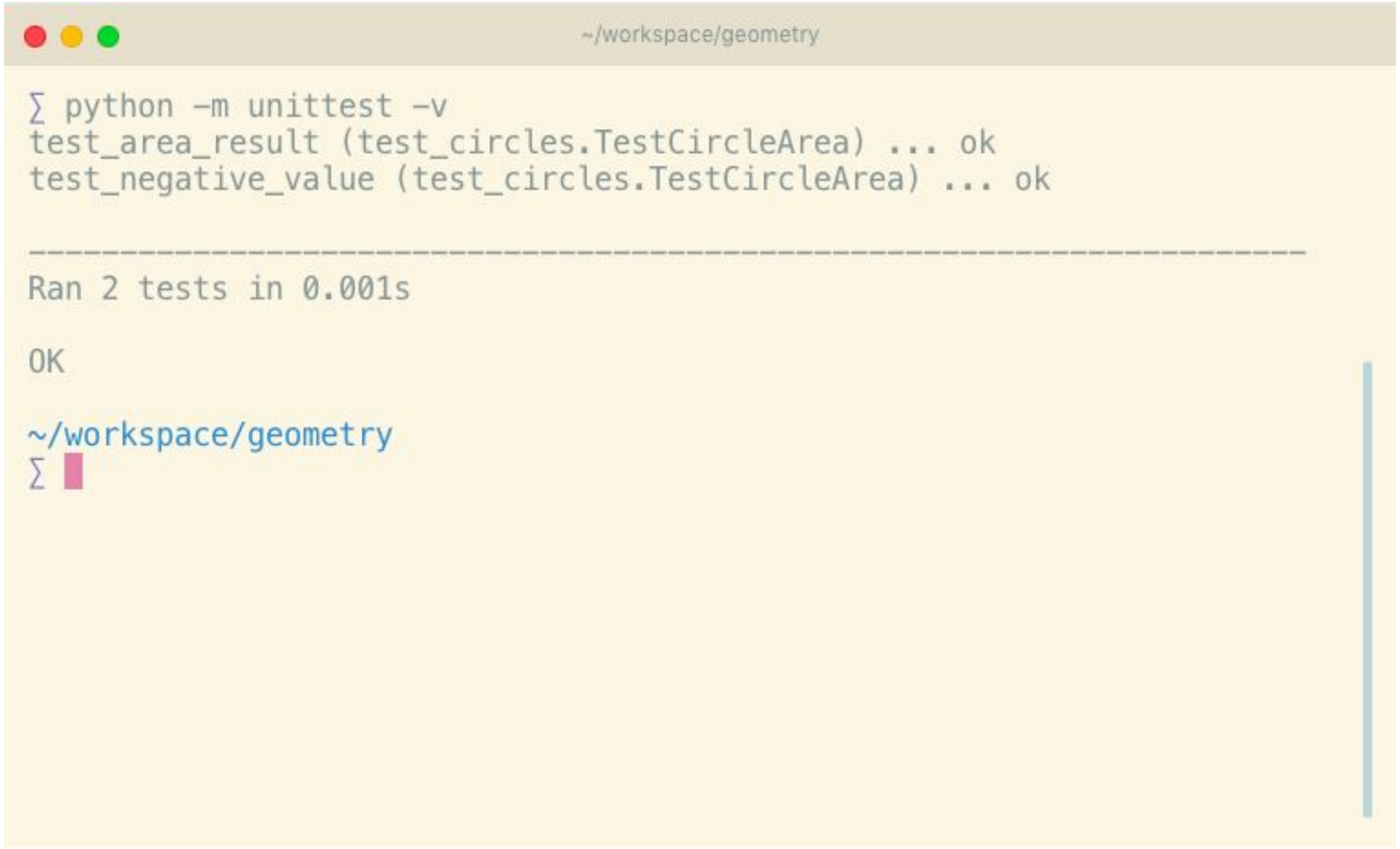


~/workspace/geometry

```
∑ python -m unittest test_circles.py -v
test_area_result (test_circles.TestCircleArea) ... ok
test_negative_value (test_circles.TestCircleArea) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK

~/workspace/geometry
∑
```

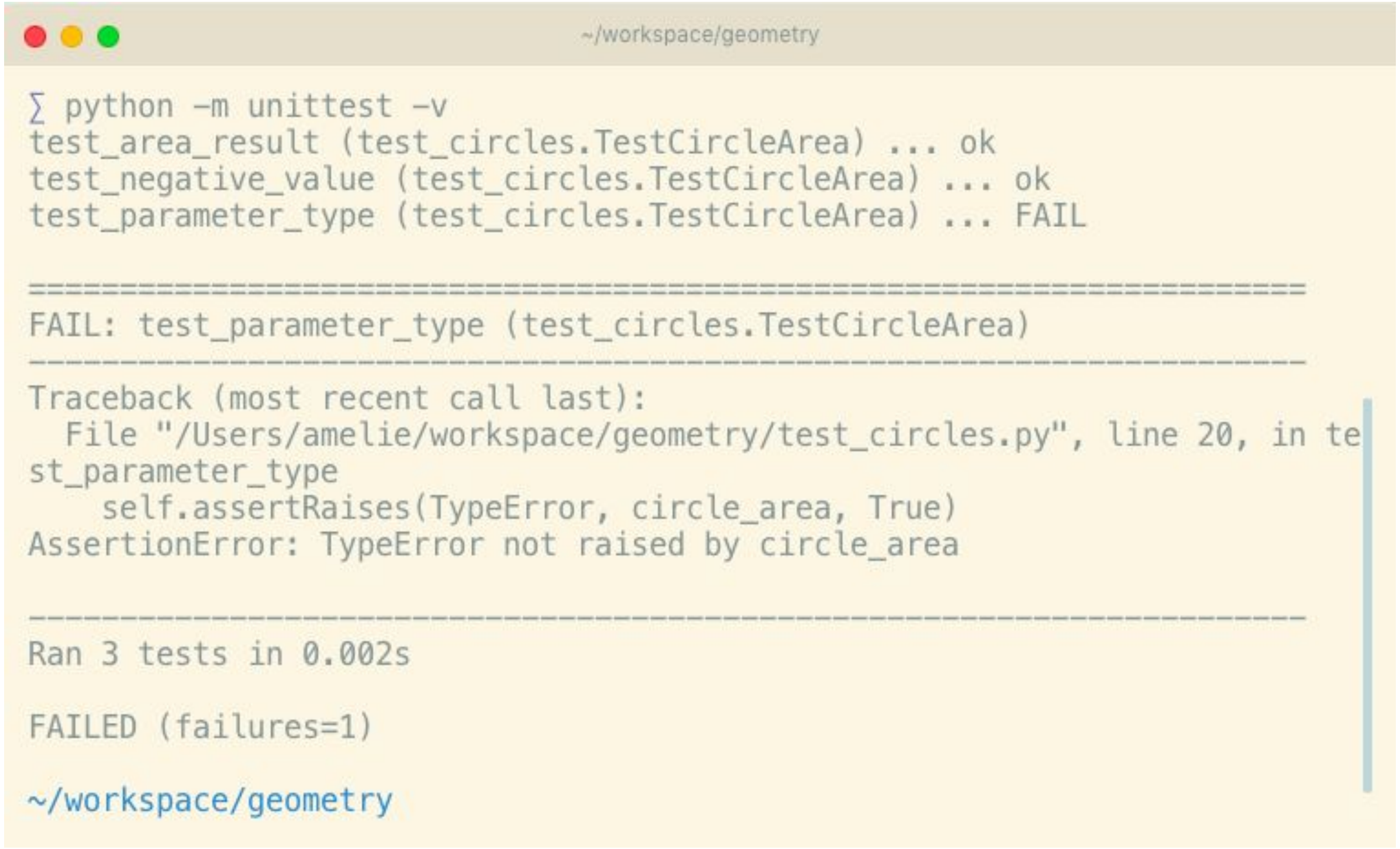

~/workspace/geometry

```
∑ python -m unittest ~~test_circles.py~~ -v
test_area_result (test_circles.TestCircleArea) ... ok
test_negative_value (test_circles.TestCircleArea) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK

~/workspace/geometry
∑ 
```


~/workspace/geometry

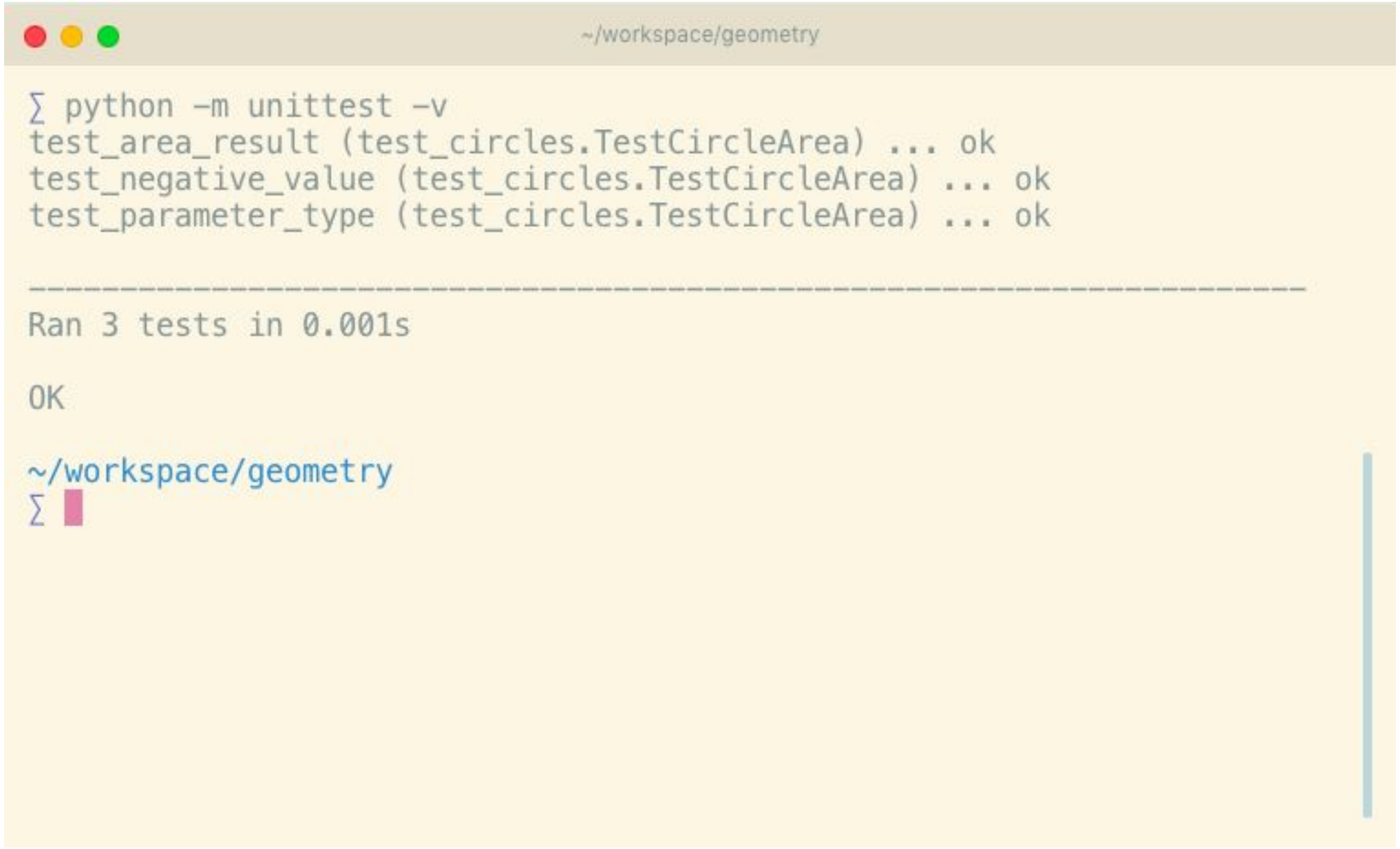
```
∑ python -m unittest -v
test_area_result (test_circles.TestCircleArea) ... ok
test_negative_value (test_circles.TestCircleArea) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK

~/workspace/geometry
∑ 
```

```python
    def test_parameter_type(self):
      self.assertRaises(TypeError, circle_area, 3+5j)
      self.assertRaises(TypeError, circle_area, True)
      self.assertRaises(TypeError, circle_area, "radius")
```



```
∑ python -m unittest -v
test_area_result (test_circles.TestCircleArea) ... ok
test_negative_value (test_circles.TestCircleArea) ... ok
test_parameter_type (test_circles.TestCircleArea) ... FAIL

======================================================================
FAIL: test_parameter_type (test_circles.TestCircleArea)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/amelie/workspace/geometry/test_circles.py", line 20, in te
st_parameter_type
    self.assertRaises(TypeError, circle_area, True)
AssertionError: TypeError not raised by circle_area

----------------------------------------------------------------------
Ran 3 tests in 0.002s

FAILED (failures=1)

~/workspace/geometry
```

```python
from math import pi

def circle_area(r):
  if type(r) not in [int, float]:
    raise TypeError("The radius must be a real number.")
  if r < 0:
    raise ValueError("The radius cannot be negative.")
  return pi*(r**2)
```



~/workspace/geometry

```
∑ python -m unittest -v
test_area_result (test_circles.TestCircleArea) ... ok
test_negative_value (test_circles.TestCircleArea) ... ok
test_parameter_type (test_circles.TestCircleArea) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK

~/workspace/geometry
∑ 
```

```python
import unittest
from math import pi
from circles import circle_area

class TestCircleArea(unittest.TestCase):
  def test_area_result(self):
    self.assertAlmostEqual(circle_area(1), pi)
    self.assertAlmostEqual(circle_area(0), 0)
    self.assertAlmostEqual(circle_area(2.1), pi * 2.1**2)

  def test_negative_value(self):
    self.assertRaises(ValueError, circle_area, -2)

  def test_parameter_type(self):
    self.assertRaises(TypeError, circle_area, 3+5j)
    self.assertRaises(TypeError, circle_area, True)
    self.assertRaises(TypeError, circle_area, "radius")
```

why should I test?

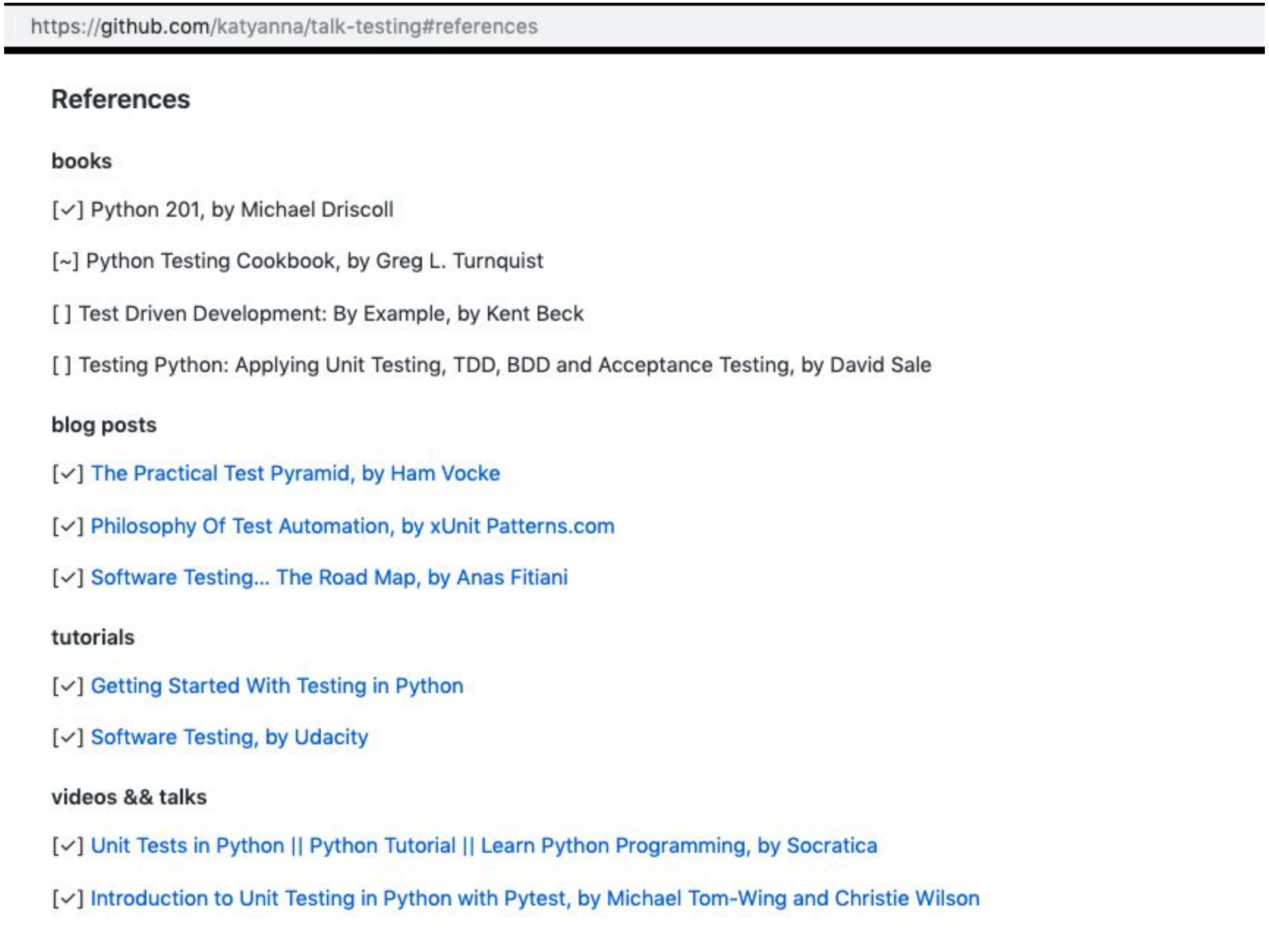
identify bugs early

confidence on the code

what's next?

Integration	PyTest	Selenium	nose
TDD	BDD	DDT	ATDD
Black-box	Hypothesis	Property-based	Mock
Maintainability	Test Pyramid	Continuous Integration	?

references



https://github.com/katyanna/talk-testing#references

References

books

[✓] Python 201, by Michael Driscoll

[~] Python Testing Cookbook, by Greg L. Turnquist

[ ] Test Driven Development: By Example, by Kent Beck

[ ] Testing Python: Applying Unit Testing, TDD, BDD and Acceptance Testing, by David Sale

blog posts

[✓] The Practical Test Pyramid, by Ham Vocke

[✓] Philosophy Of Test Automation, by xUnit Patterns.com

[✓] Software Testing... The Road Map, by Anas Fitiani

tutorials

[✓] Getting Started With Testing in Python

[✓] Software Testing, by Udacity

videos && talks

[✓] Unit Tests in Python || Python Tutorial || Learn Python Programming, by Socratica

[✓] Introduction to Unit Testing in Python with Pytest, by Michael Tom-Wing and Christie Wilson https://github.com/katyanna/talk-testing/README.md

Thank you! <3

@amelie_kn
amelie.kn@gmail.com