TDD IN PYTHON USING PYCHARM

CERTIFICATE IN PYTHON PROGRAMMING - PY100

UNIVERSITY OF WASHINGTON

- Located in Paris, France (Seattle TZ + 9)
- Online student in the Computational Finance & Risk Management Master's program at UW
- Graduated from the Certificate in Computational Finance and the Certificate in C++ programming at UW

TEST DRIVEN DEVELOPMENT

- Learned C++ programming using TDD during the Certificate in C++ programming at UW
- Great fan of TDD although not a specialist
- PyCharm makes TDD really nice and easy

WHAT IS TEST DRIVEN DEVELOPMENT?

Wikipedia:

Test-driven development (TDD) is a software development process that relies on the repetition of a **very short** development cycle: requirements are turned into **very specific test cases**, then the software is **improved to pass the new tests, only**. This is opposed to software development that allows software to be added that is not proven to meet requirements […]

Test-driven development is related to the **test-first** programming concepts

https://en.wikipedia.org/wiki/Test-driven_development

WHAT IS TEST DRIVEN DEVELOPMENT?

WORKFLOW

Wikipedia:

1. Add a test
2. Run all tests and see if the new test fails
3. Write the code
4. Run tests
5. Refactor code

REPEAT

https://en.wikipedia.org/wiki/Test-driven_development

WHAT IS TEST DRIVEN DEVELOPMENT?

Wikipedia:

The size of the steps should always **be small**, with as few as **1 to 10 edits between each test run**. If new code does not **rapidly satisfy** a new test, or **other tests fail unexpectedly**, the programmer should **undo** or revert in preference to **excessive debugging**. Continuous integration helps by providing **revertible checkpoints**.

https://en.wikipedia.org/wiki/Test-driven_development

WHY USING TEST DRIVEN DEVELOPMENT?

Write your test before your code i.e. what before how

- Thinking carefully about **what** result should your code produce before thinking about **how** you're gonna get that result
- Thinking first about different results you might get gives you **hints about implementation**
- Forces you to write **testable code** (single responsibility)

WHY USING TEST DRIVEN DEVELOPMENT?

- Catching bugs **as soon as possible**
- As your code **grows** so does your **set of unit tests**
- Leads to « good » **code coverage**
- Extremely helpful when **refactoring** code to detect when code is **broken**

TDD USING PYCHARM IS SUPER EASY

```
Foo.py

class Foo:

    def bar(self):

        pass
```

- Create a new file named Foo.py
- class Foo()
- Declare at least one method (def __init__(self): pass or def bar(self): pass …)

TDD USING PYCHARM IS SUPER EASY

```
Foo.py

class Foo:

# right-click here <———————

   def bar(self):

      pass
```

- below class Foo:
  - right-click
  - select Go To –> Test –> Create New Test …
  - Update if needed the pop-up window and click ok

TDD USING PYCHARM IS SUPER EASY

```
test_Foo.py

from unittest import TestCase

class TestFoo(TestCase):

    pass
```

- PyCharm
  - creates automatically a new file named test_Foo.py
  - inserts the TestCase class from the unittest module
  - inserts the TestFoo(TestCase) class which inherits from the TestCase class

TDD USING PYCHARM IS SUPER EASY

```
test_Foo.py

from unittest import TestCase

class TestFoo(TestCase):

  pass
```

- Within class TestFoo(TestCase) you have access to the test methods inherited from the TestCase class:
  - self.assertEqual(…), self.assertNotEqual(…), self.assertAlmostEqual(…), self.assertFalse(…), self.assertIn(…), self.assertIs(…), self.assertDictEqual(…), self.assertListEqual(…) and many others …

TDD USING PYCHARM IS SUPER EASY

- Write your first test:
  - Each test name ie each method name should start with the keyword **test_**[chosen_test_name]
  - You won't get any error if you do not prefix your test name with **test_**
  - If you forget the **test_** keyword, the test is not registered and hence **not executed**

```
test_Foo.py

from unittest import TestCase

class TestFoo(TestCase):

   def test_bar(self):

       pass
```

TDD USING PYCHARM IS SUPER EASY

- write your first test:
  - Target result: the bar method from the Foo class should return the string ‘Foo::bar()’
  - Make sure your test can fail otherwise you can’t be sure that your test result is correct when your test passes E.g. if testing for equality use assertEqual and assertNotEqual
  - def test_bar(self): ….

```
test_Foo.py

from unittest import TestCase

class TestFoo(TestCase):

   def test_bar(self):

      from Foo import Foo

      foo_a = Foo()

      self.assertEqual(‘Foo::bar()’, foo_a.bar())

      self.assertNotEqual(‘Foo::not_bar()’, foo_a.bar())
```

TDD USING PYCHARM IS SUPER EASY

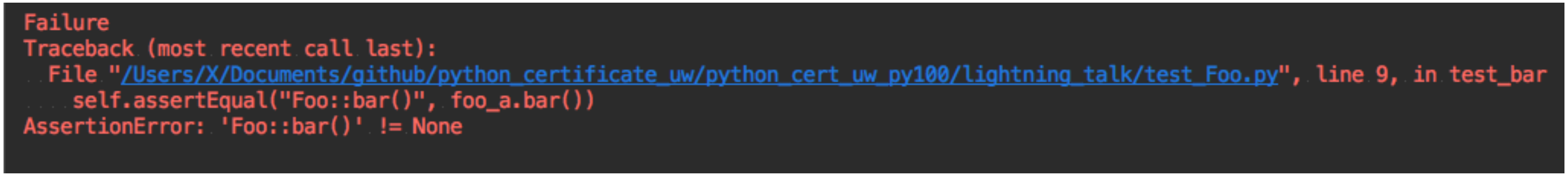
```
test_Foo.py
from unittest import TestCase
class TestFoo(TestCase):
   def test_bar(self):
      from Foo import Foo
      foo_a = Foo()
      self.assertEqual('Foo::bar()', foo_a.bar())
      self.assertNotEqual('Foo::not_bar()', foo_a.bar())
```

- Run the test -> **make the test fail**
  - In the project side bar right-click on the project directory
  - click on Run 'Unittests in [your_project_name]'

TDD USING PYCHARM IS SUPER EASY

```
Foo.py
class Foo:
    def bar(self):
        pass
```

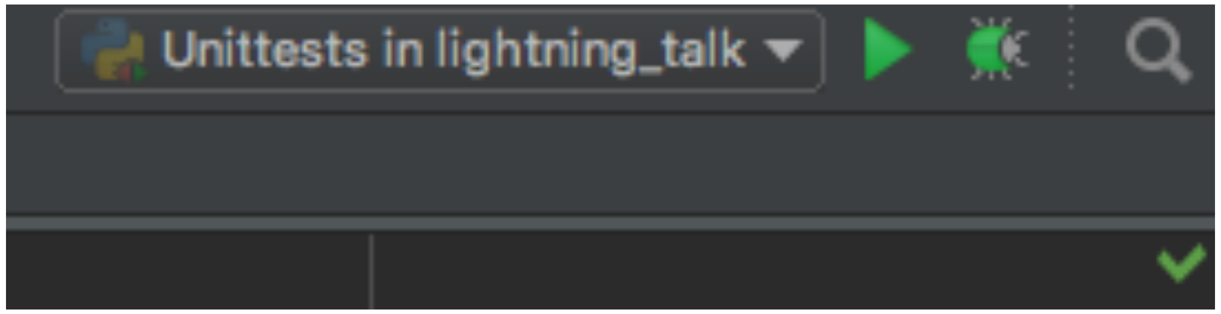
```
test_Foo.py
from unittest import TestCase
class TestFoo(TestCase):
    def test_bar(self):
        from Foo import Foo
        foo_a = Foo()
        self.assertEqual(‘Foo::bar()’, foo_a.bar())
        self.assertNotEqual(‘Foo::not_bar()’, foo_a.bar())
```

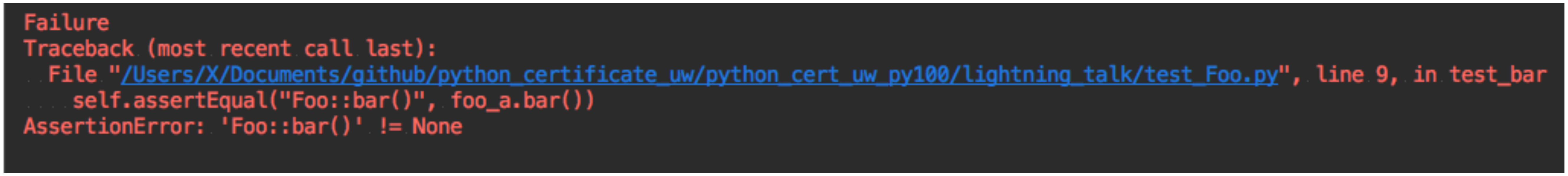


```
Failure
Traceback (most recent call last):
  File "/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/test_Foo.py", line 9, in test_bar
    self.assertEqual("Foo::bar()", foo_a.bar())
AssertionError: 'Foo::bar()' != None
```

TDD USING PYCHARM IS SUPER EASY

- Now:
    - you're coding using the Unittests framework
    - you're coding using TDD

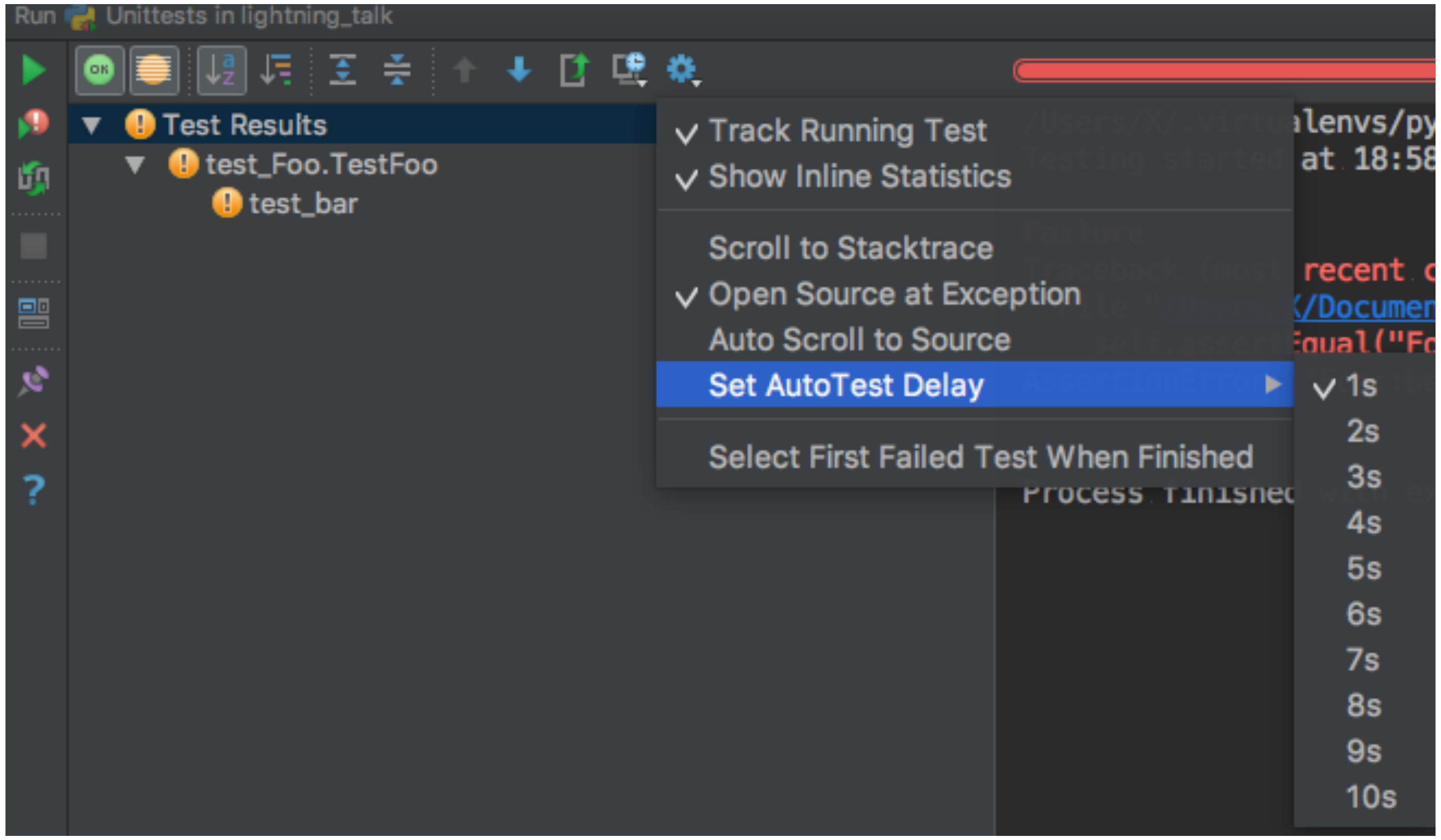




```
Failure
Traceback (most recent call last):
  File "/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/test_Foo.py", line 9, in test_bar
    self.assertEqual("Foo::bar()", foo_a.bar())
AssertionError: 'Foo::bar()' != None
```

TDD USING PYCHARM IS SUPER EASY

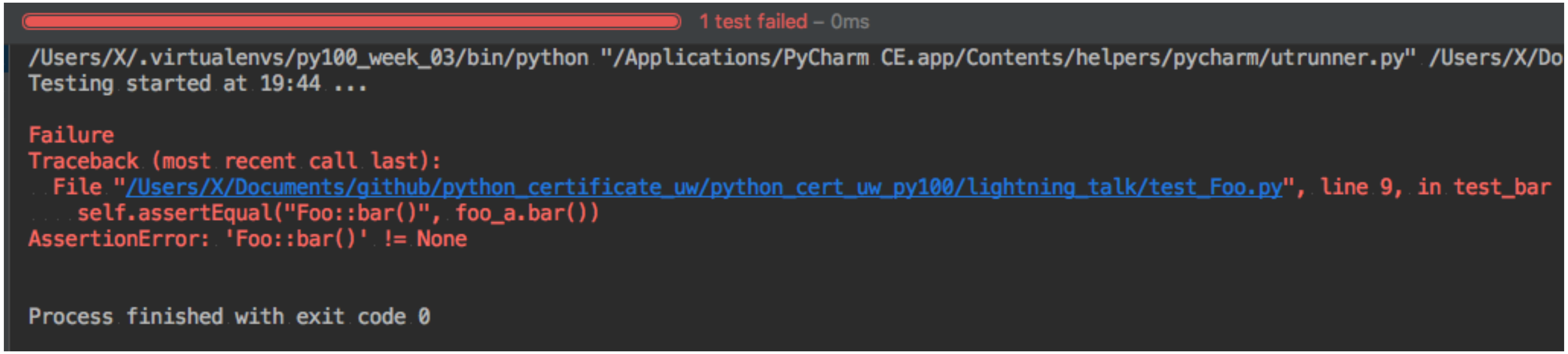
- Now:
  - you can use automatic execution of:
    - all of your tests written so far
    - some of your tests
  - you can set AutoTest Delay: from 1s to 10s



TDD USING PYCHARM IS SUPER EASY

- Implement the method bar()
  - work on the code until the test passes

```
Foo.py

class Foo:

    def bar(self):

        pass
```

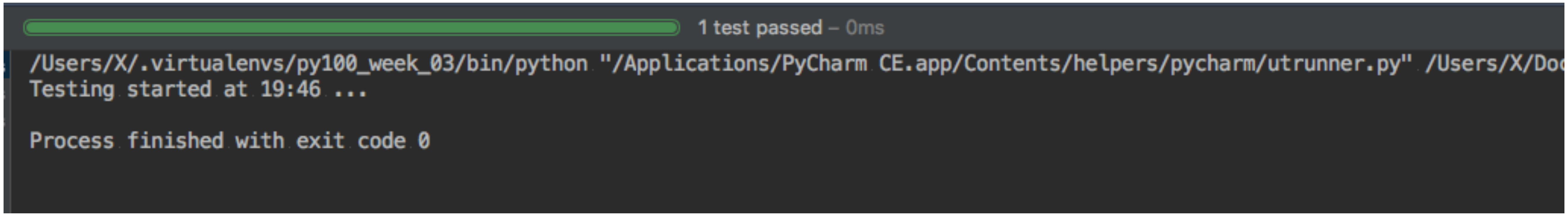


```
/Users/X/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/Contents/helpers/pycharm/utrunner.py" /Users/X/Do
Testing started at 19:44 ...

Failure
Traceback (most recent call last):
  File "/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/test_Foo.py", line 9, in test_bar
    self.assertEqual("Foo::bar()", foo_a.bar())
AssertionError: 'Foo::bar()' != None


Process finished with exit code 0
```

TDD USING PYCHARM IS SUPER EASY

- Implement the method bar()
    - work on the code until the test passes

```
Foo.py

class Foo:

  def bar(self):

    return ‘Foo::bar()’
```



1 test passed – 0ms

```
/Users/X/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/Contents/helpers/pycharm/utrunner.py" /Users/X/Doc
Testing started at 19:46 ...

Process finished with exit code 0
```

TDD - USEFUL THINGS

USING TDD TO WRITE CODE WHICH INTERACTS WITH THE USER

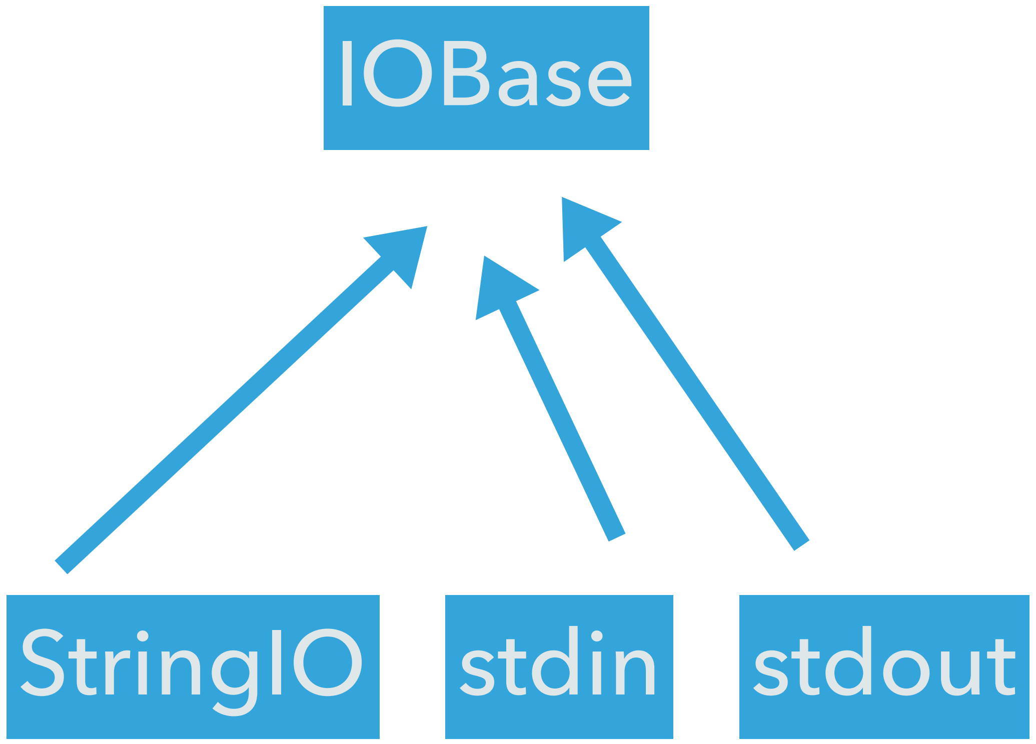
- How do you automatically test for messages which need to be printed to the standard output (console)?
- How do you automatically test user inputs?

TDD – USEFUL THINGS

USING TDD TO WRITE CODE WHICH INTERACTS WITH THE USER

- Implement your methods in terms of **IOBase** then use **StringIO (string stream)** to test the stream against **a string**

```
from io import IOBase

from io import StringIO

from sys import stdout

from sys import stdin
```



TDD – USEFUL THINGS

USING TDD TO WRITE CODE WHICH INTERACTS WITH THE USER

- Automatically test for messages which need to be printed to the **standard output** (console)
  - **# Build an empty string stream**
  - ostream=StringIO() # Output to the console: ostream=sys.stdout
  - **# Write into the stream from a string**
  - ostream.write(‘message’)
  - **# Test the string stream content against a string**
  - self.assertEqual(‘message’, ostream.getvalue())

TDD – USEFUL THINGS

USING TDD TO WRITE CODE WHICH INTERACTS WITH THE USER

- Automatically test for **user input (standard input)**
    - **# Build the string stream from the string**
    - istream=StringIO(`Charles Ives`) # User input istream=sys.stdin
    - **# Read from the stream into a string**
    - input=istream.readline().rstrip()
    - istream.flush()
    - **# Test for string equality**
    - self.assertEqual('Charles Ives', input)

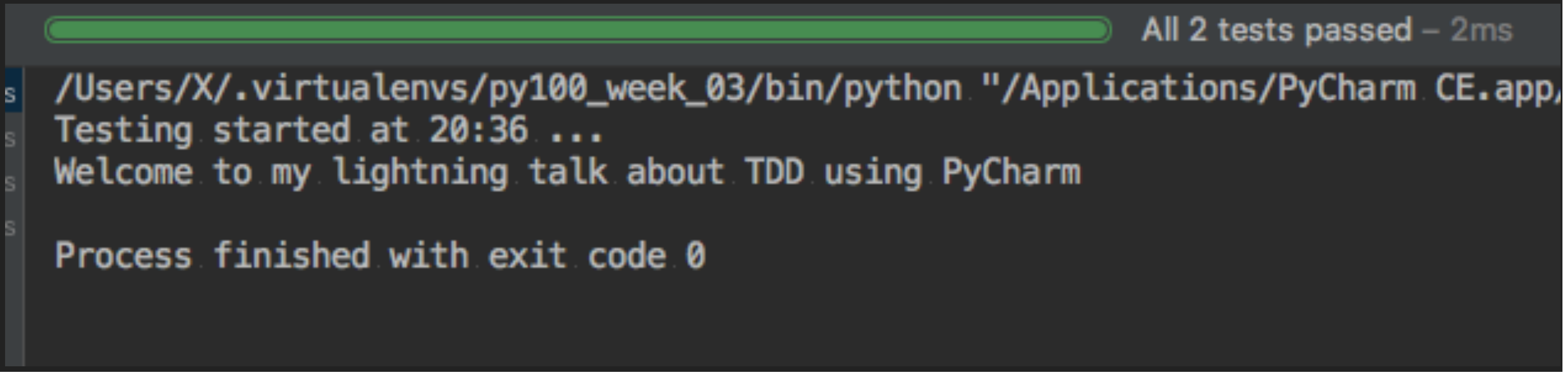
TDD – USEFUL THINGS

OUTPUT TO THE STANDARD OUTPUT

- Can’t test the following code automatically:

```
Foo.py
class Foo:
    def bar(self): return ‘Foo::bar()’
    def print_welcome_stdout(self) -> None:
        print("Welcome to my lightning talk about TDD using PyCharm")
```

```
test_Foo.py
from unitest import TestCase
class TestFoo(TestCase):
    def test_print_welcome_stdout(self):
        from Foo import Foo
        foo_a = Foo()
        foo_a.print_welcome_stdout()
        #self.assert… ?
```



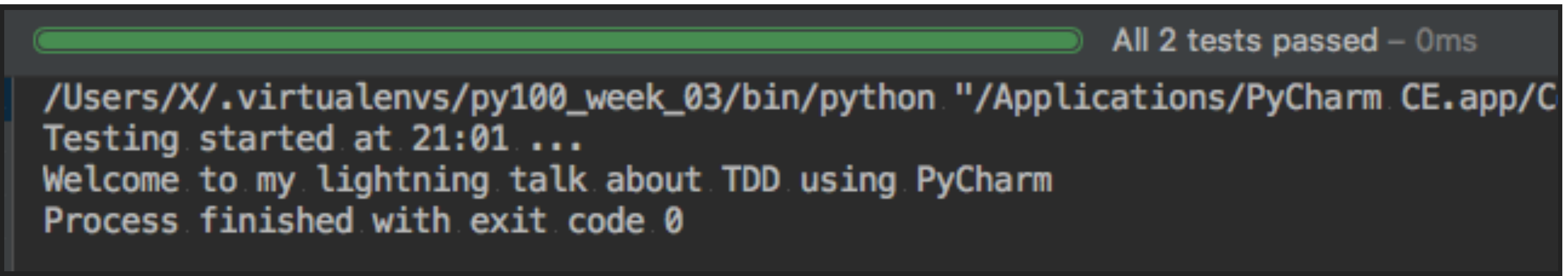
All 2 tests passed – 2ms

```
/Users/X/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/
Testing started at 20:36 ...
Welcome to my lightning talk about TDD using PyCharm

Process finished with exit code 0
```

TEST DRIVEN DEVELOPMENT (TDD)

```
Foo.py
class Foo:
    def bar(self): return ‘Foo::bar()’
    def print_welcome_stdout(self) -> None:
         print("Welcome to my lightning talk about TDD using PyCharm")
    from io import IOBase
    def print_welcome_stream(self, ostream: IOBase) -> None:
         ostream.write("Welcome to my lightning talk about TDD using PyCharm")
```

```
test_Foo.py
from unittest import TestCase
class TestFoo(TestCase):
    def test_print_welcome_stream(self):
          from Foo import Foo
          from io import StringIO
          foo_a = Foo()
          str_bench = "Welcome to my lightning talk about TDD using PyCharm »
          str_stream = StringIO()
          foo_a.print_welcome_stream(str_stream)
          self.assertEqual(str_bench, str_stream.getvalue())
          # To print to the console use sys.stdout
          from sys import stdout
          foo_a.print_welcome_stream(stdout)
```



```
All 2 tests passed – 0ms
/Users/X/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/C
Testing started at 21:01 ...
Welcome to my lightning talk about TDD using PyCharm
Process finished with exit code 0
```

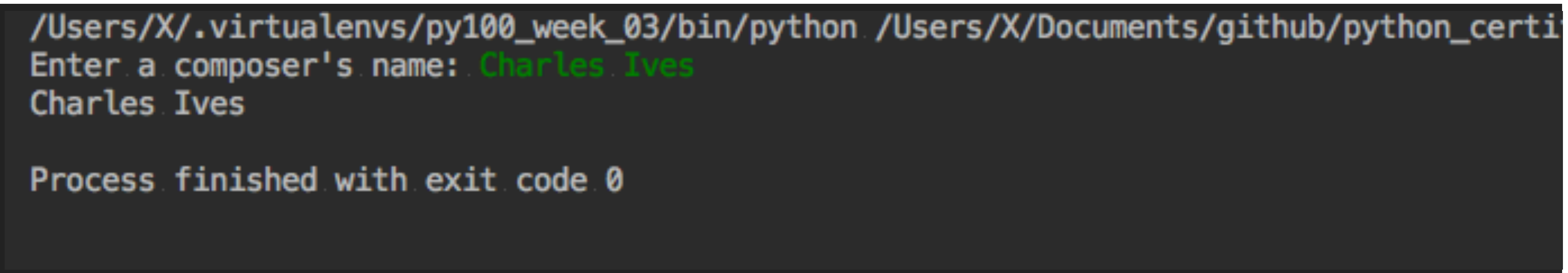
TDD – USEFUL THINGS

USER INPUT

- Can’t test automatically

```
Foo.py

class Foo:

    def get_user_input(self, message: str) -> str:

        print(message, end=‘ ‘)

        user_input=input()

        return user_input
```

```
main.py

from Foo import Foo

def main():

    foo_a = Foo()

    user_input=foo_a.get_user_input(‘Enter a composer’s name:’)

    print(user_input)

if __name__==‘__main__’: main()
```



```
/Users/X/.virtualenvs/py100_week_03/bin/python /Users/X/Documents/github/python_certi
Enter a composer's name: Charles Ives
Charles Ives

Process finished with exit code 0
```

Replace input() by IOBase derived class: StringIO

```
Foo.py
class Foo:
    def get_user_input_stream(self, message: str, ostream: IOBase, istream: IOBase) -> str:
        ostream.write(message)
        ostream.flush()
        input=istream.readline().rstrip()
        istream.flush()
        return input
```

```
test_Foo.py
from unittest import TestCase
class TestFoo(TestCase):
    def test_get_user_input_stream(self):
      from Foo import Foo
      from io import StringIO
      ostream=StringIO()
      message_bench="Please enter a composer's name:"
      user_input_bench='Maurice Durufle'
      istream=StringIO(user_input_bench) # The user input
      user_input=foo_a.get_user_input_stream(message_bench, ostream, istream)
      self.assertEqual(user_input_bench, user_input)
```

TDD - USEFUL THINGS

USER INPUT

- To use your function with the standard input use istream=sys.stdin

```
Foo.py
class Foo:
    def get_user_input_stream(self, message: str, ostream: IOBase, istream: IOBase) -> str:
        ostream.write(message)
        ostream.flush()
        input=istream.readline().rstrip()
        istream.flush()
        return input
```

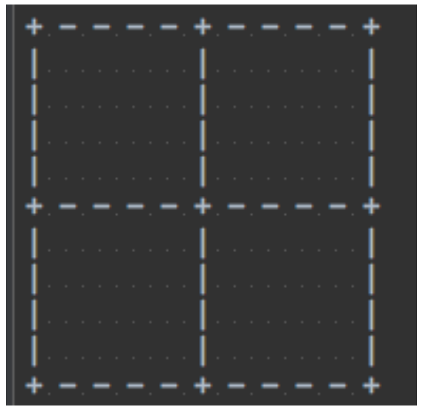
```
main.py
def main():
    from Foo import Foo
    from sys import stdout, stdin
    foo_a = Foo()
    user_input=foo_a.get_user_input_stream('Enter a composer's name:', stdout, stdin)
    print(user_input)
if __name__=='__main__': main()
```

TDD IN PRACTICE - EXAMPLE: GRID CLASS

GRID PRINTER ASSIGNMENT SIMPLIFIED VERSION

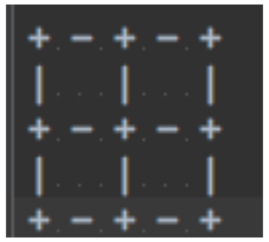
- Write some code which prints to the console the following squared grids given two arguments grid_size, num_rows_cols

9, 2



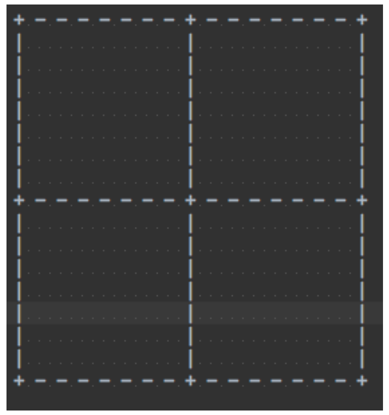
```
+ - - - - + - - - - +
|         |         |
|         |         |
|         |         |
|         |         |
+ - - - - + - - - - +
|         |         |
|         |         |
|         |         |
|         |         |
+ - - - - + - - - - +
```

3,2



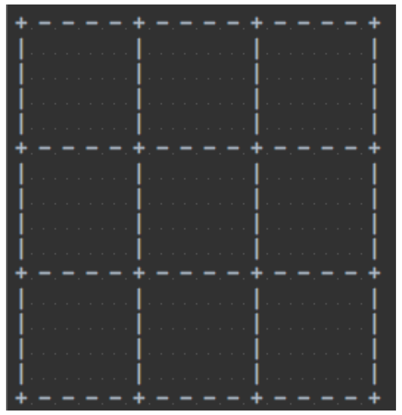
```
+ - + - +
|   |   |
+ - + - +
|   |   |
+ - + - +
```

15, 2



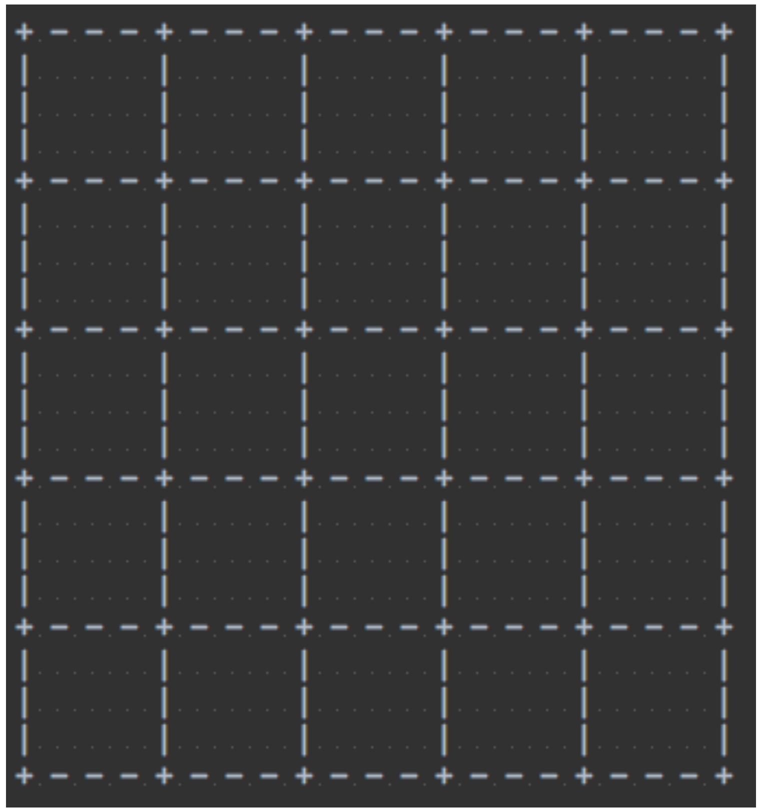
```
+ - - - - - - - + - - - - - - - +
|               |               |
|               |               |
|               |               |
|               |               |
|               |               |
|               |               |
|               |               |
+ - - - - - - - + - - - - - - - +
|               |               |
|               |               |
|               |               |
|               |               |
|               |               |
|               |               |
|               |               |
+ - - - - - - - + - - - - - - - +
```

14, 3



```
+ - - - - + - - - - + - - - - +
|         |         |         |
|         |         |         |
|         |         |         |
|         |         |         |
+ - - - - + - - - - + - - - - +
|         |         |         |
|         |         |         |
|         |         |         |
|         |         |         |
+ - - - - + - - - - + - - - - +
|         |         |         |
|         |         |         |
|         |         |         |
|         |         |         |
+ - - - - + - - - - + - - - - +
```

19, 5



```
+ - - - + - - - + - - - + - - - + - - - +
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
+ - - - + - - - + - - - + - - - + - - - +
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
+ - - - + - - - + - - - + - - - + - - - +
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
+ - - - + - - - + - - - + - - - + - - - +
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
+ - - - + - - - + - - - + - - - + - - - +
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
+ - - - + - - - + - - - + - - - + - - - +
```

TDD IN PRACTICE - EXAMPLE: GRID CLASS

GRID PRINTER ASSIGNMENT SIMPLIFIED VERSION

- Two classes:
  - Grid Class
  - GridPrinter class

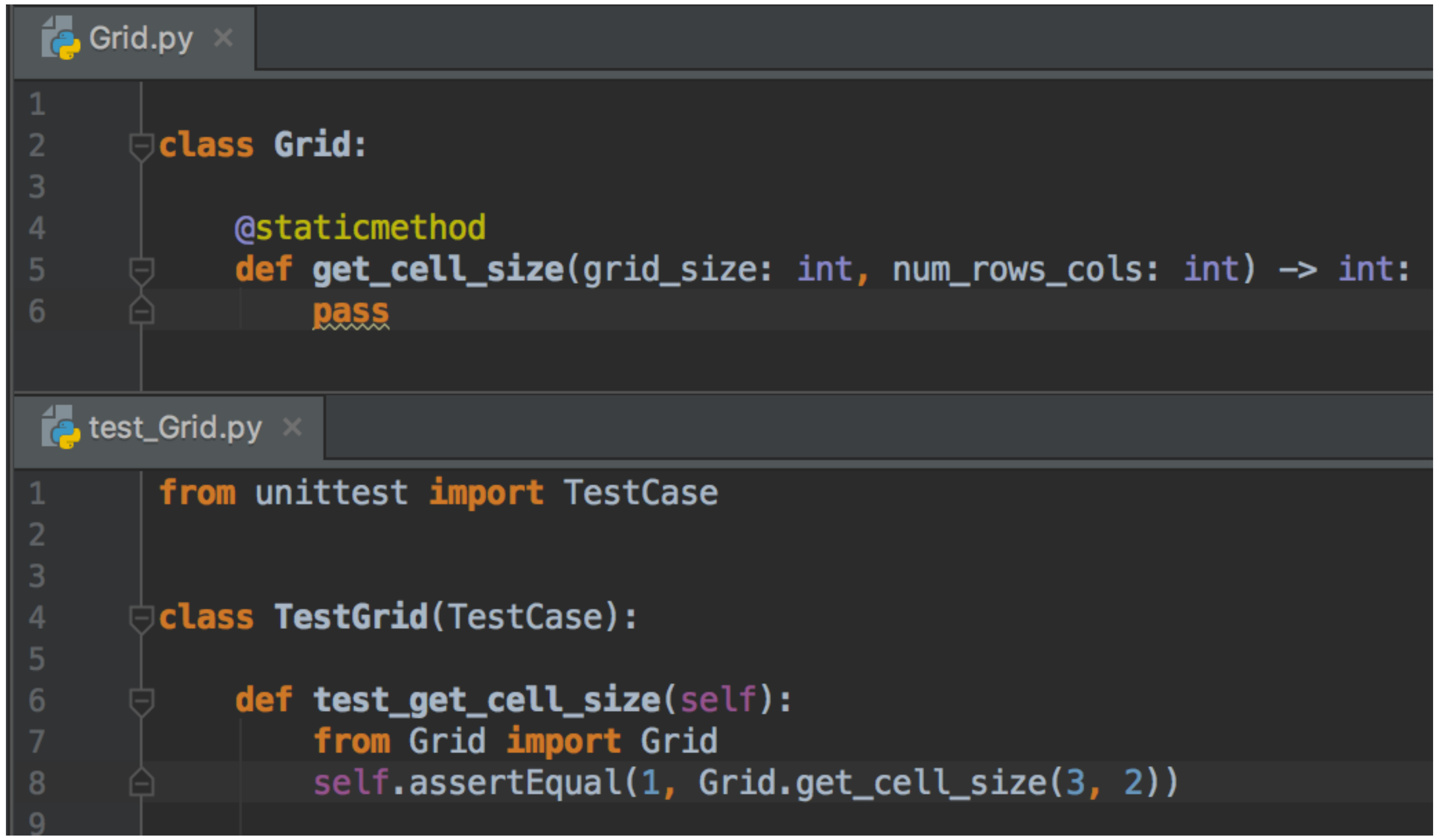
TDD IN PRACTICE - EXAMPLE: GRID CLASS

GRID PRINTER ASSIGNMENT SIMPLIFIED VERSION: GRID CLASS

- Can build the grid=Grid(grid_size, num_rows_cols)
- if:
  - computed cell_size given the arguments grid_size and num_rows_cols is an integer
  - and not(grid_size <= 0 or num_rows_cols <= 0)
  - and not (grid_size <= num_rows_cols)

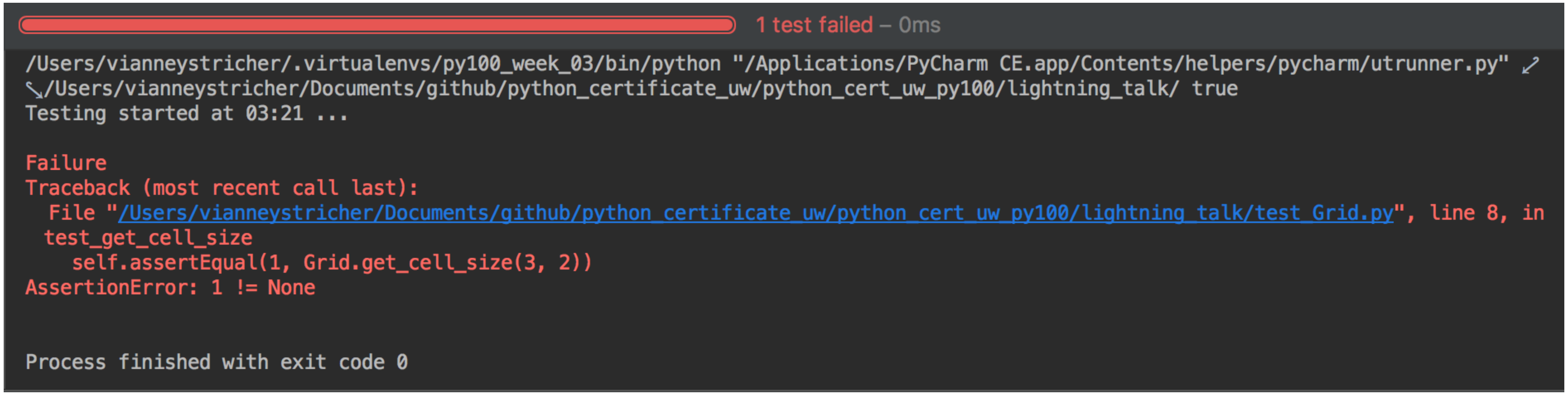
TDD IN PRACTICE - EXAMPLE: GRID CLASS

COMPUTED CELL_SIZE GIVEN ARGUMENTS IS AN INTEGER



```python

class Grid:

    @staticmethod
    def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
        pass
```

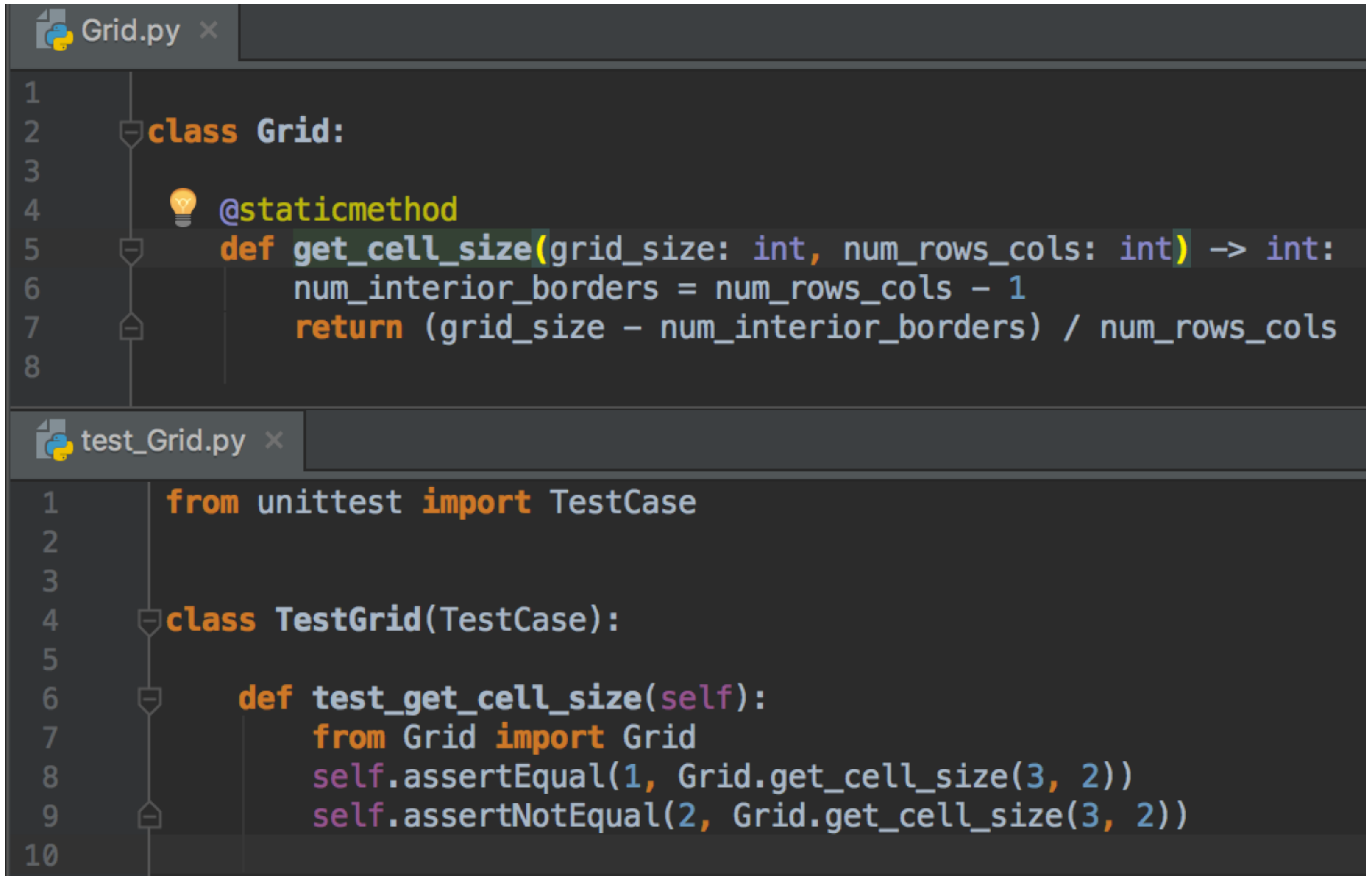
```python
from unittest import TestCase


class TestGrid(TestCase):

    def test_get_cell_size(self):
        from Grid import Grid
        self.assertEqual(1, Grid.get_cell_size(3, 2))
```



```
/Users/vianneystricher/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/Contents/helpers/pycharm/utrunner.py"
/Users/vianneystricher/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/ true
Testing started at 03:21 ...

Failure
Traceback (most recent call last):
  File "/Users/vianneystricher/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/test_Grid.py", line 8, in
  test_get_cell_size
    self.assertEqual(1, Grid.get_cell_size(3, 2))
AssertionError: 1 != None


Process finished with exit code 0
```

TDD IN PRACTICE – EXAMPLE: GRID CLASS

COMPUTED CELL_SIZE GIVEN ARGUMENTS IS AN INTEGER

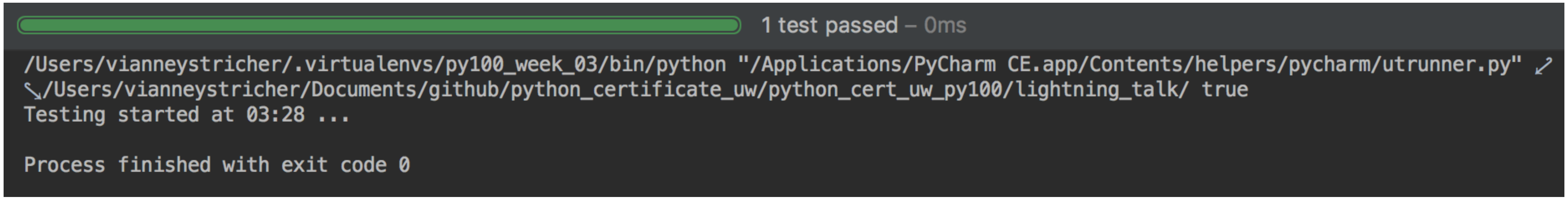


Grid.py

```python

class Grid:

    @staticmethod
    def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
        num_interior_borders = num_rows_cols - 1
        return (grid_size - num_interior_borders) / num_rows_cols

```

test_Grid.py

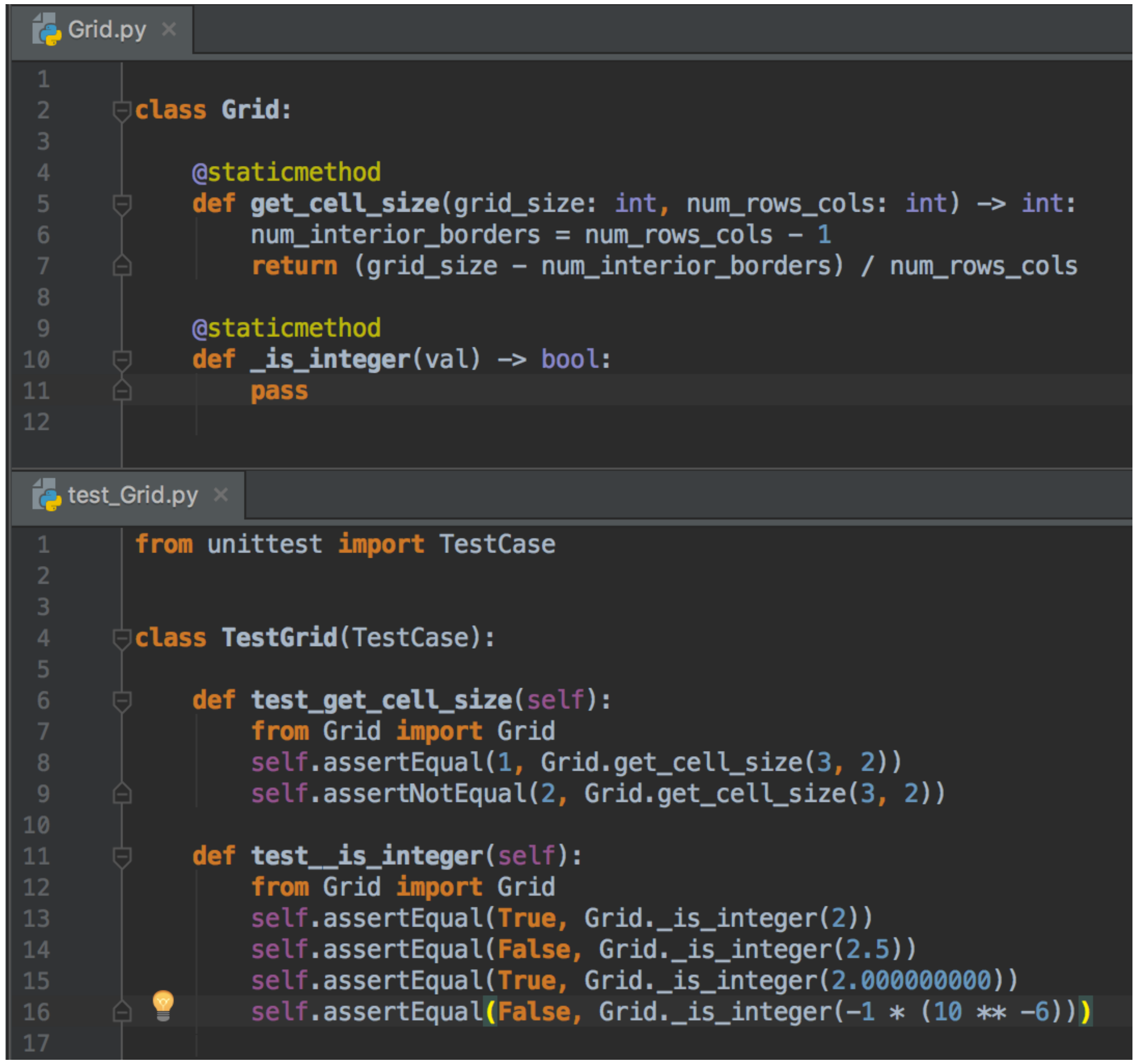
```python
from unittest import TestCase


class TestGrid(TestCase):

    def test_get_cell_size(self):
        from Grid import Grid
        self.assertEqual(1, Grid.get_cell_size(3, 2))
        self.assertNotEqual(2, Grid.get_cell_size(3, 2))

```



1 test passed – 0ms

```
/Users/vianneystricher/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/Contents/helpers/pycharm/utrunner.py"
/Users/vianneystricher/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/ true
Testing started at 03:28 ...

Process finished with exit code 0
```

TDD IN PRACTICE – EXAMPLE: GRID CLASS



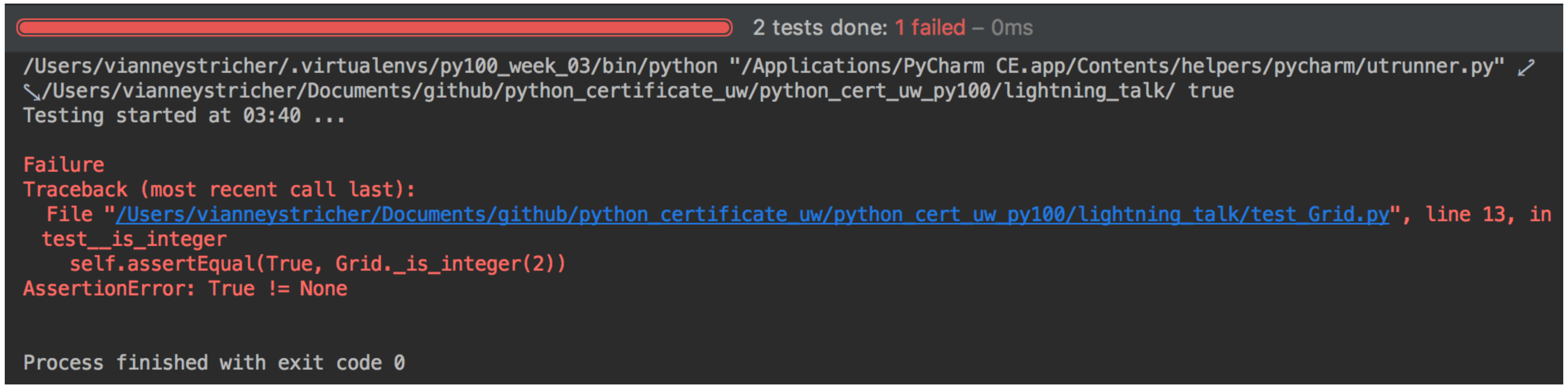
Grid.py

```python

class Grid:

    @staticmethod
    def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
        num_interior_borders = num_rows_cols - 1
        return (grid_size - num_interior_borders) / num_rows_cols

    @staticmethod
    def _is_integer(val) -> bool:
        pass

```

test_Grid.py

```python
from unittest import TestCase


class TestGrid(TestCase):

    def test_get_cell_size(self):
        from Grid import Grid
        self.assertEqual(1, Grid.get_cell_size(3, 2))
        self.assertNotEqual(2, Grid.get_cell_size(3, 2))

    def test__is_integer(self):
        from Grid import Grid
        self.assertEqual(True, Grid._is_integer(2))
        self.assertEqual(False, Grid._is_integer(2.5))
        self.assertEqual(True, Grid._is_integer(2.000000000))
        self.assertEqual(False, Grid._is_integer(-1 * (10 ** -6)))

```

COMPUTED CELL_SIZE GIVEN ARGUMENTS IS AN INTEGER

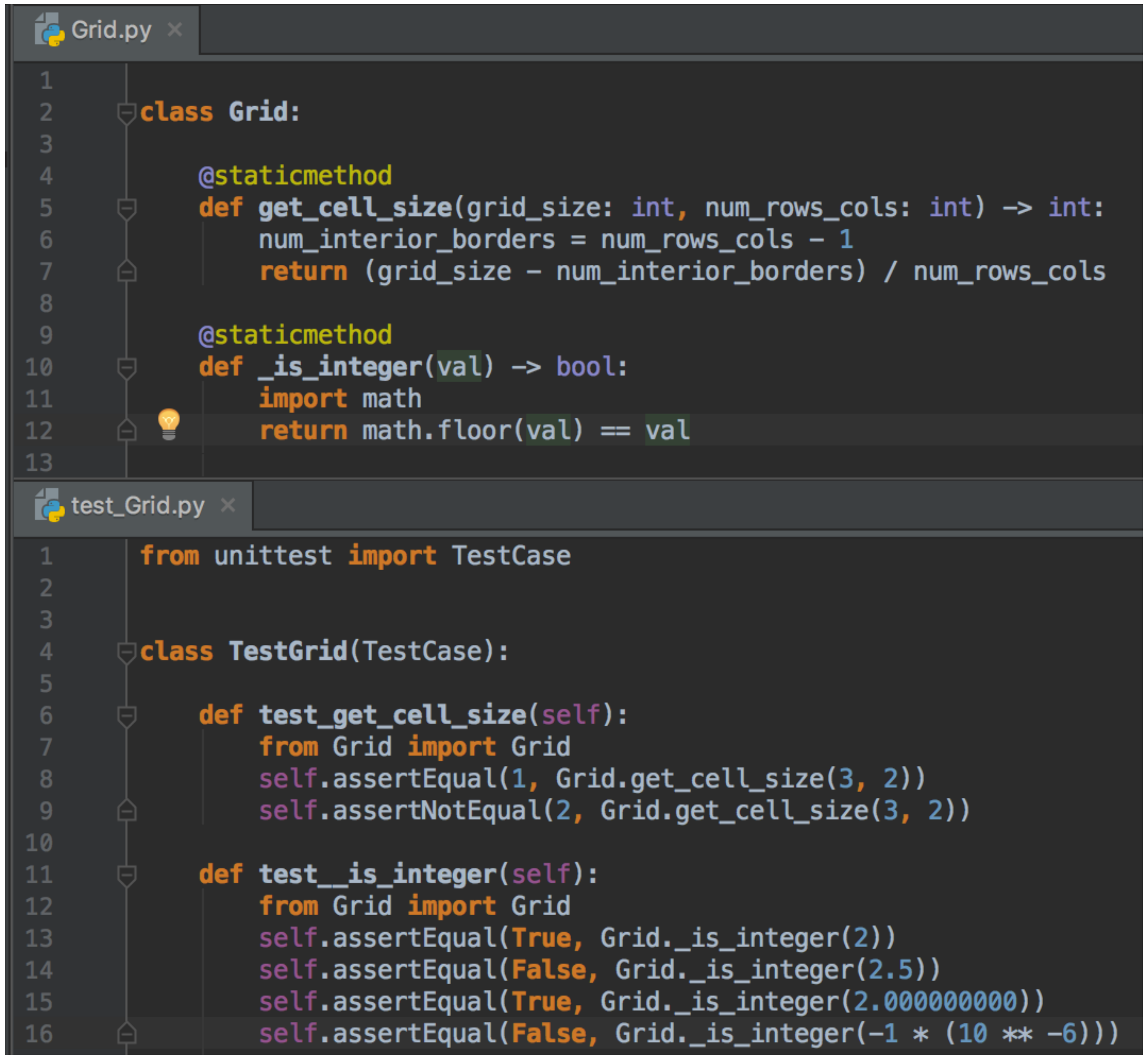


2 tests done: 1 failed – 0ms

```
/Users/vianneystricher/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/Contents/helpers/pycharm/utrunner.py"
/Users/vianneystricher/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/ true
Testing started at 03:40 ...

Failure
Traceback (most recent call last):
  File "/Users/vianneystricher/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/test_Grid.py", line 13, in
 test__is_integer
    self.assertEqual(True, Grid._is_integer(2))
AssertionError: True != None


Process finished with exit code 0
```

TDD IN PRACTICE - EXAMPLE: GRID CLASS

COMPUTED CELL_SIZE GIVEN ARGUMENTS IS AN INTEGER

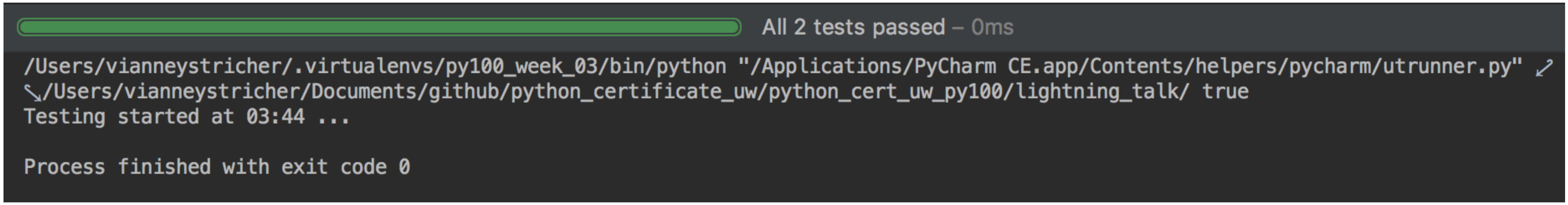


Grid.py

```python

class Grid:

    @staticmethod
    def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
        num_interior_borders = num_rows_cols - 1
        return (grid_size - num_interior_borders) / num_rows_cols

    @staticmethod
    def _is_integer(val) -> bool:
        import math
        return math.floor(val) == val

```

test_Grid.py

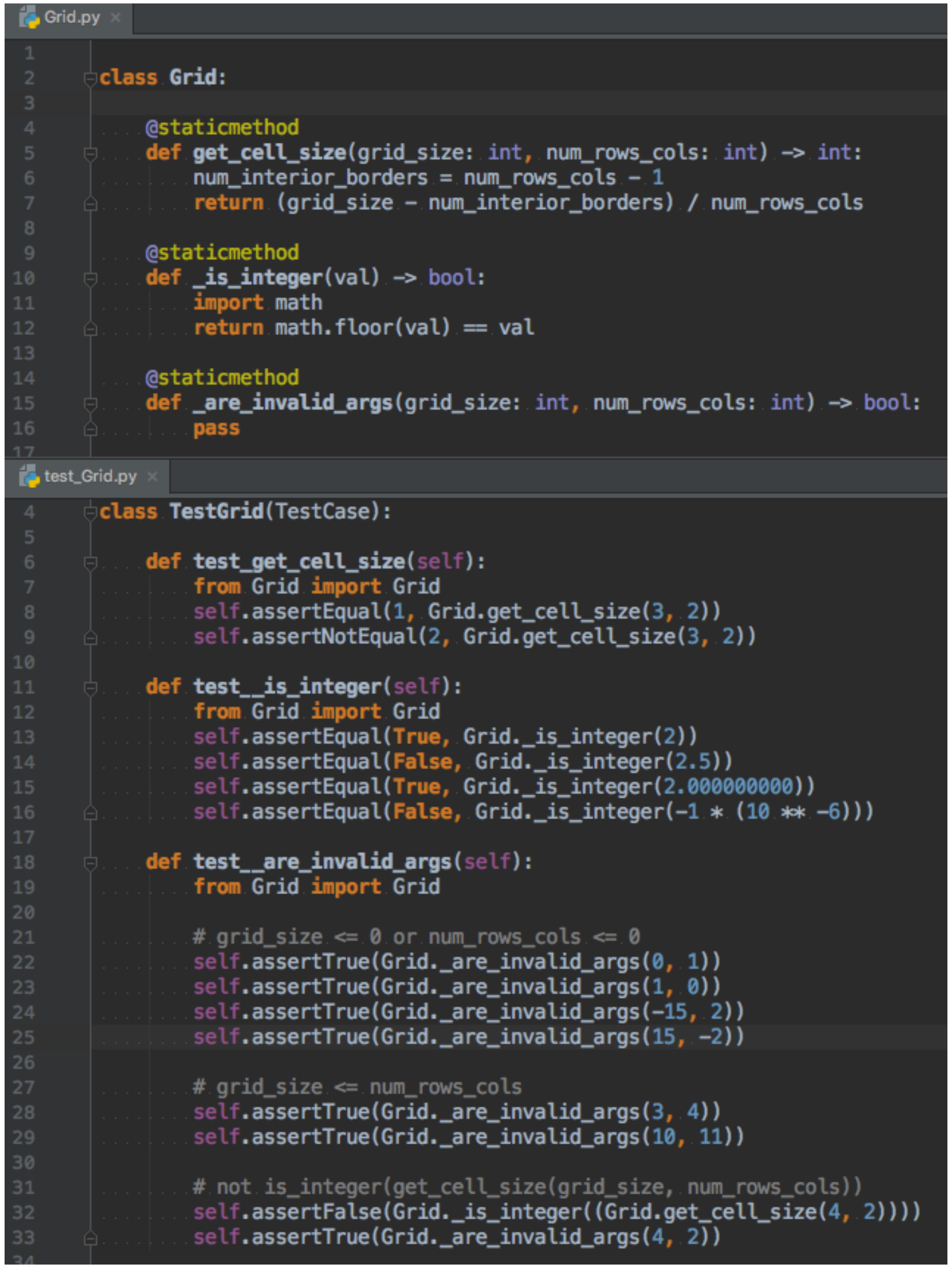
```python
from unittest import TestCase


class TestGrid(TestCase):

    def test_get_cell_size(self):
        from Grid import Grid
        self.assertEqual(1, Grid.get_cell_size(3, 2))
        self.assertNotEqual(2, Grid.get_cell_size(3, 2))

    def test__is_integer(self):
        from Grid import Grid
        self.assertEqual(True, Grid._is_integer(2))
        self.assertEqual(False, Grid._is_integer(2.5))
        self.assertEqual(True, Grid._is_integer(2.000000000))
        self.assertEqual(False, Grid._is_integer(-1 * (10 ** -6)))
```



All 2 tests passed – 0ms

```
/Users/vianneystricher/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/Contents/helpers/pycharm/utrunner.py" ⤦
⤥/Users/vianneystricher/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/ true
Testing started at 03:44 ...

Process finished with exit code 0
```

TDD IN PRACTICE – EXAMPLE: GRID CLASS



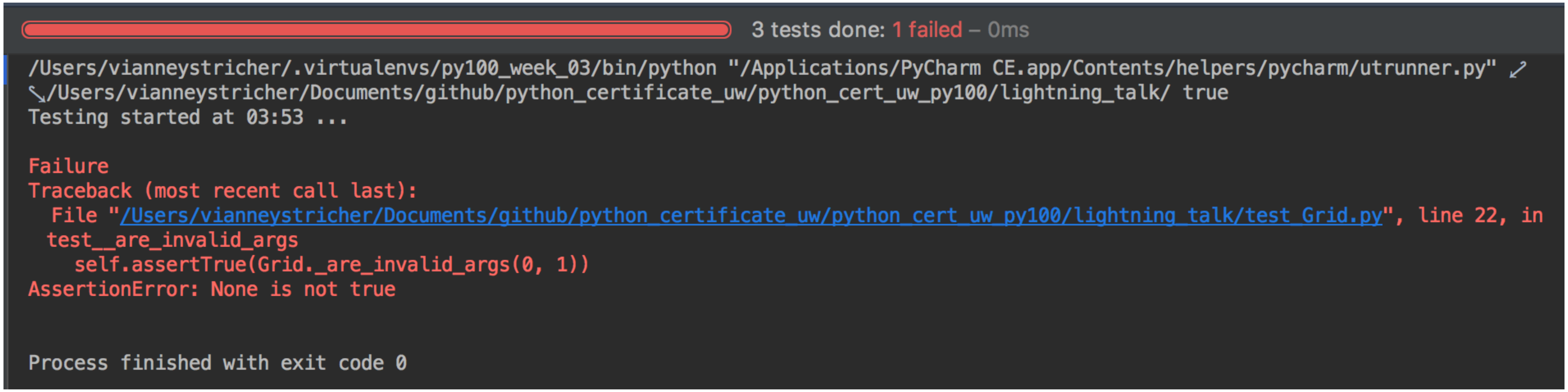
```python

class Grid:

    @staticmethod
    def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
        num_interior_borders = num_rows_cols - 1
        return (grid_size - num_interior_borders) / num_rows_cols

    @staticmethod
    def _is_integer(val) -> bool:
        import math
        return math.floor(val) == val

    @staticmethod
    def _are_invalid_args(grid_size: int, num_rows_cols: int) -> bool:
        pass
```

```python
class TestGrid(TestCase):

    def test_get_cell_size(self):
        from Grid import Grid
        self.assertEqual(1, Grid.get_cell_size(3, 2))
        self.assertNotEqual(2, Grid.get_cell_size(3, 2))

    def test__is_integer(self):
        from Grid import Grid
        self.assertEqual(True, Grid._is_integer(2))
        self.assertEqual(False, Grid._is_integer(2.5))
        self.assertEqual(True, Grid._is_integer(2.000000000))
        self.assertEqual(False, Grid._is_integer(-1 * (10 ** -6)))

    def test__are_invalid_args(self):
        from Grid import Grid

        # grid_size <= 0 or num_rows_cols <= 0
        self.assertTrue(Grid._are_invalid_args(0, 1))
        self.assertTrue(Grid._are_invalid_args(1, 0))
        self.assertTrue(Grid._are_invalid_args(-15, 2))
        self.assertTrue(Grid._are_invalid_args(15, -2))

        # grid_size <= num_rows_cols
        self.assertTrue(Grid._are_invalid_args(3, 4))
        self.assertTrue(Grid._are_invalid_args(10, 11))

        # not is_integer(get_cell_size(grid_size, num_rows_cols))
        self.assertFalse(Grid._is_integer((Grid.get_cell_size(4, 2))))
        self.assertTrue(Grid._are_invalid_args(4, 2))
```

INVALID ARGS PASSED TO THE CONSTRUCTOR

TDD IN PRACTICE – EXAMPLE: GRID CLASS

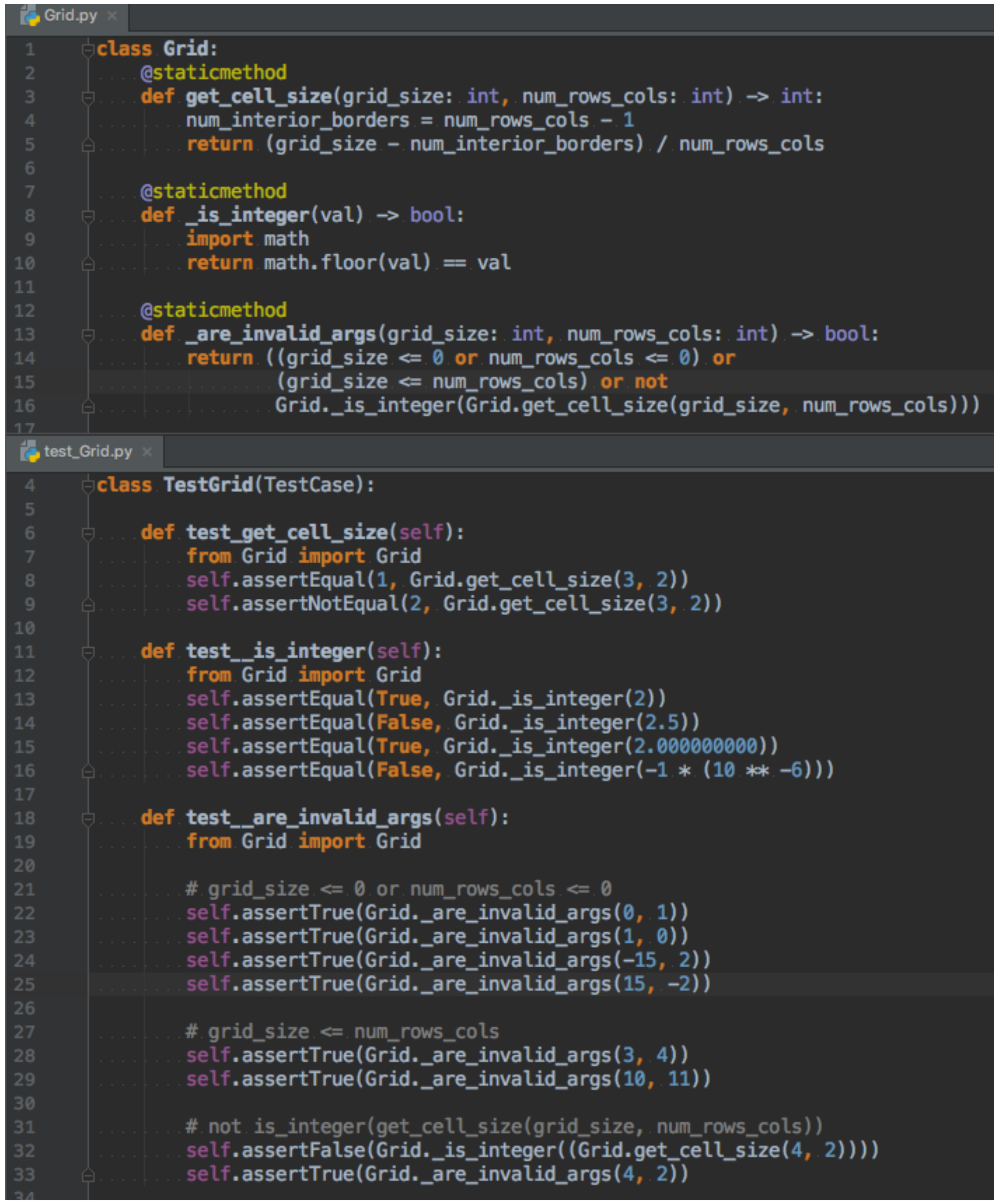
INVALID ARGS PASSED TO THE CONSTRUCTOR



3 tests done: 1 failed – 0ms

```
/Users/vianneystricher/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/Contents/helpers/pycharm/utrunner.py" 
/Users/vianneystricher/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/ true
Testing started at 03:53 ...

Failure
Traceback (most recent call last):
  File "/Users/vianneystricher/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/test_Grid.py", line 22, in
 test__are_invalid_args
    self.assertTrue(Grid._are_invalid_args(0, 1))
AssertionError: None is not true


Process finished with exit code 0
```

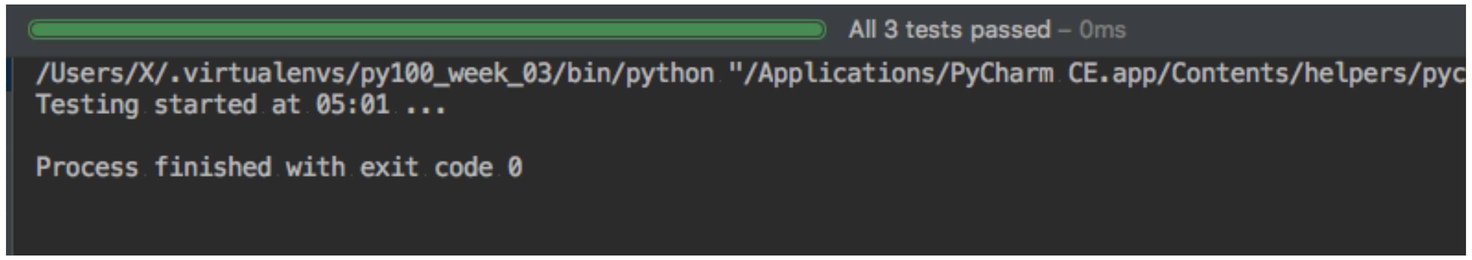
TDD IN PRACTICE – EXAMPLE: GRID CLASS



```python
class Grid:
    @staticmethod
    def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
        num_interior_borders = num_rows_cols - 1
        return (grid_size - num_interior_borders) / num_rows_cols

    @staticmethod
    def _is_integer(val) -> bool:
        import math
        return math.floor(val) == val

    @staticmethod
    def _are_invalid_args(grid_size: int, num_rows_cols: int) -> bool:
        return ((grid_size <= 0 or num_rows_cols <= 0) or
                (grid_size <= num_rows_cols) or not
                Grid._is_integer(Grid.get_cell_size(grid_size, num_rows_cols)))
```

```python
class TestGrid(TestCase):

    def test_get_cell_size(self):
        from Grid import Grid
        self.assertEqual(1, Grid.get_cell_size(3, 2))
        self.assertNotEqual(2, Grid.get_cell_size(3, 2))

    def test__is_integer(self):
        from Grid import Grid
        self.assertEqual(True, Grid._is_integer(2))
        self.assertEqual(False, Grid._is_integer(2.5))
        self.assertEqual(True, Grid._is_integer(2.000000000))
        self.assertEqual(False, Grid._is_integer(-1 * (10 ** -6)))

    def test__are_invalid_args(self):
        from Grid import Grid

        # grid_size <= 0 or num_rows_cols <= 0
        self.assertTrue(Grid._are_invalid_args(0, 1))
        self.assertTrue(Grid._are_invalid_args(1, 0))
        self.assertTrue(Grid._are_invalid_args(-15, 2))
        self.assertTrue(Grid._are_invalid_args(15, -2))

        # grid_size <= num_rows_cols
        self.assertTrue(Grid._are_invalid_args(3, 4))
        self.assertTrue(Grid._are_invalid_args(10, 11))

        # not is_integer(get_cell_size(grid_size, num_rows_cols))
        self.assertFalse(Grid._is_integer((Grid.get_cell_size(4, 2))))
        self.assertTrue(Grid._are_invalid_args(4, 2))
```

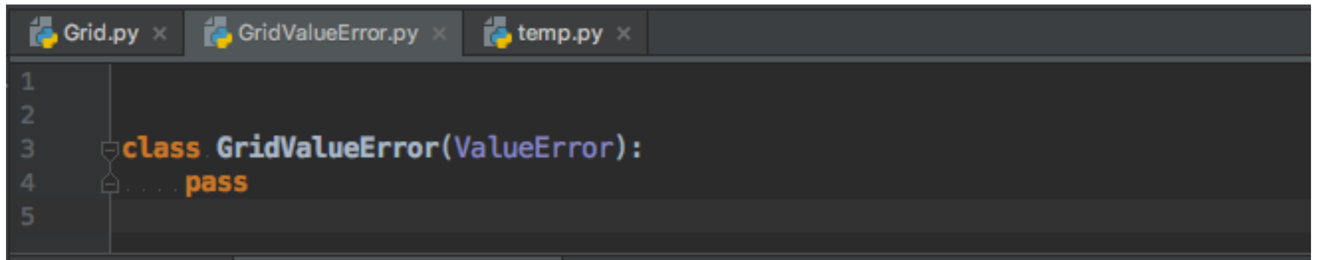
INVALID ARGS PASSED TO THE CONSTRUCTOR



```
/Users/X/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/Contents/helpers/pyc
Testing started at 05:01 ...

Process finished with exit code 0
```

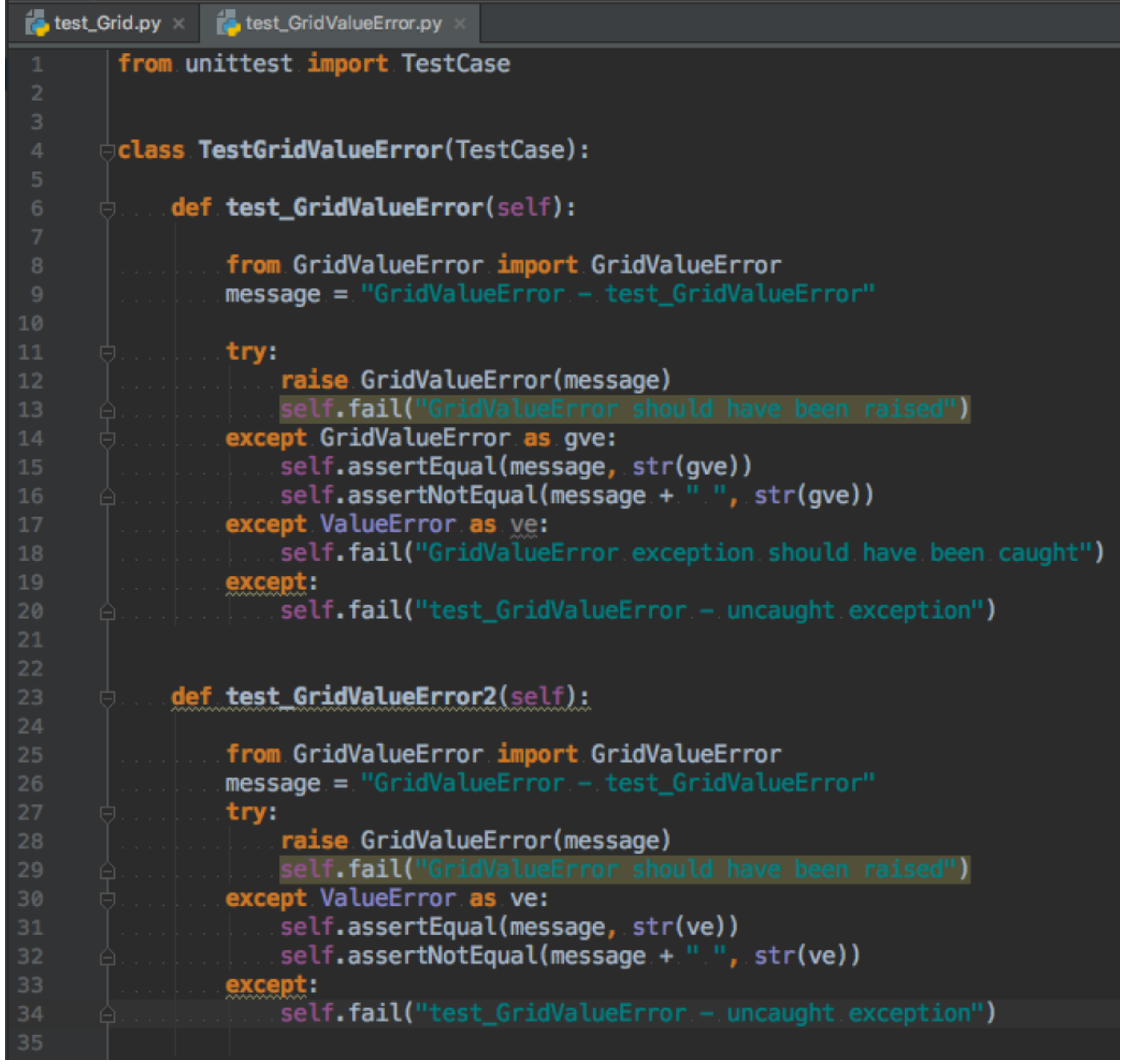
TDD IN PRACTICE – EXAMPLE: GRID CLASS

Define GridValueError exception which would be raised by Grid constructor if invalid arguments are passed



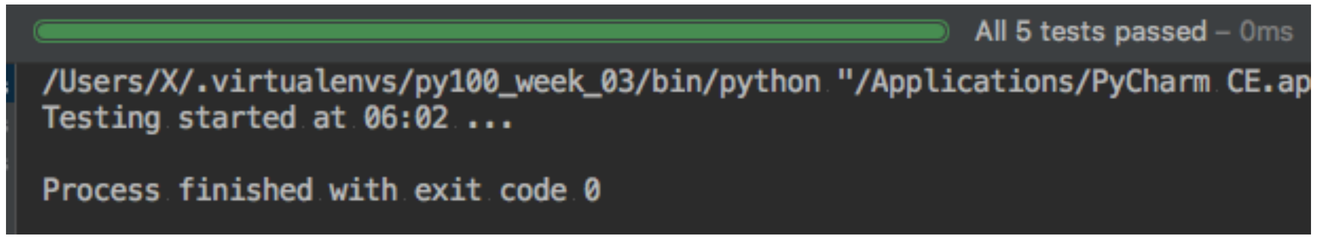
Grid.py × GridValueError.py × temp.py ×

```
class GridValueError(ValueError):
    pass
```



test_Grid.py × test_GridValueError.py ×

```
from unittest import TestCase


class TestGridValueError(TestCase):

    def test_GridValueError(self):

        from GridValueError import GridValueError
        message = "GridValueError - test_GridValueError"

        try:
            raise GridValueError(message)
            self.fail("GridValueError should have been raised")
        except GridValueError as gve:
            self.assertEqual(message, str(gve))
            self.assertNotEqual(message + " ", str(gve))
        except ValueError as ve:
            self.fail("GridValueError exception should have been caught")
        except:
            self.fail("test_GridValueError - uncaught exception")


    def test_GridValueError2(self):

        from GridValueError import GridValueError
        message = "GridValueError - test_GridValueError"
        try:
            raise GridValueError(message)
            self.fail("GridValueError should have been raised")
        except ValueError as ve:
            self.assertEqual(message, str(ve))
            self.assertNotEqual(message + " ", str(ve))
        except:
            self.fail("test_GridValueError - uncaught exception")
```



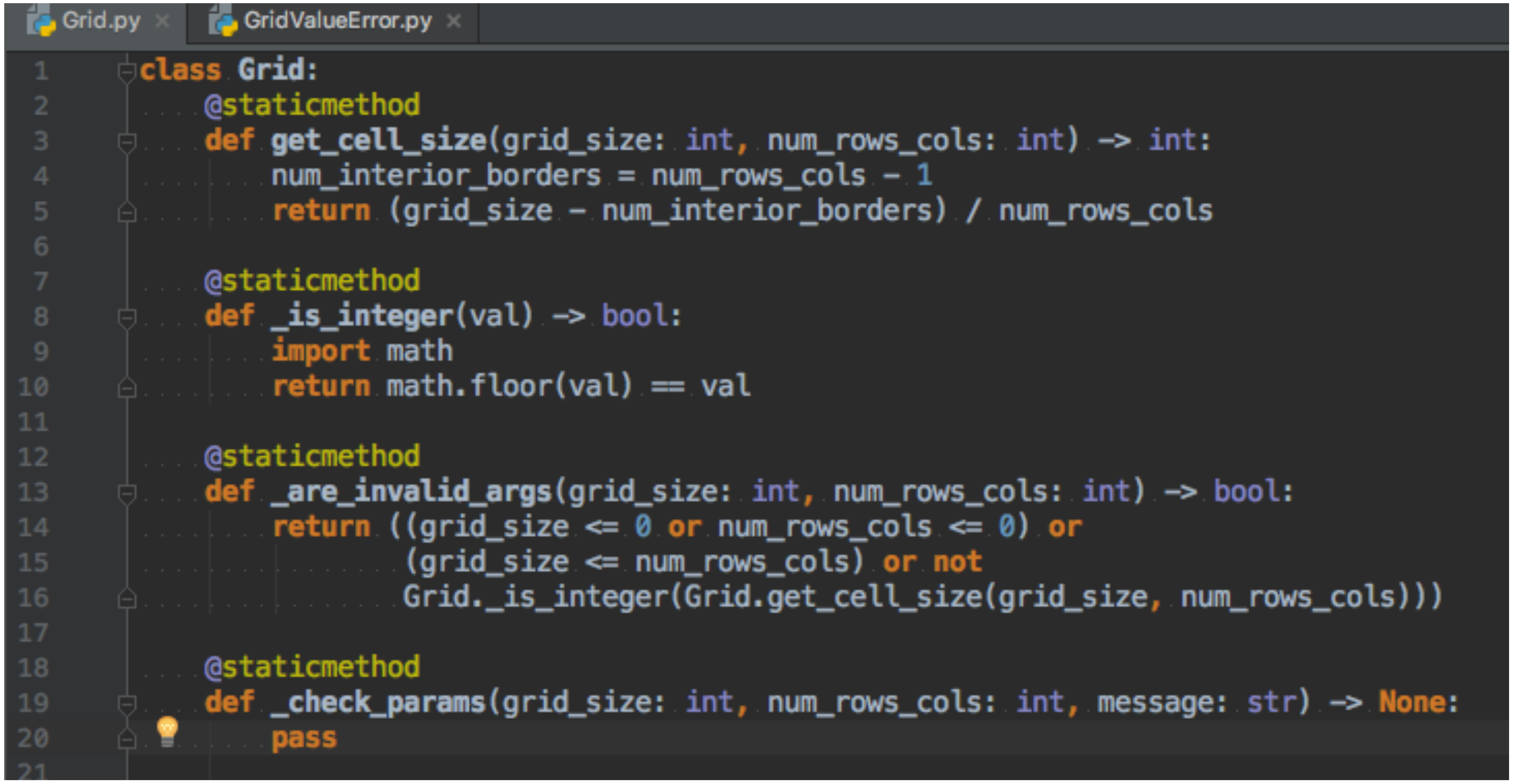
All 5 tests passed – 0ms

```
/Users/X/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.ap
Testing started at 06:02 ...

Process finished with exit code 0
```

TDD IN PRACTICE – EXAMPLE: GRID CLASS

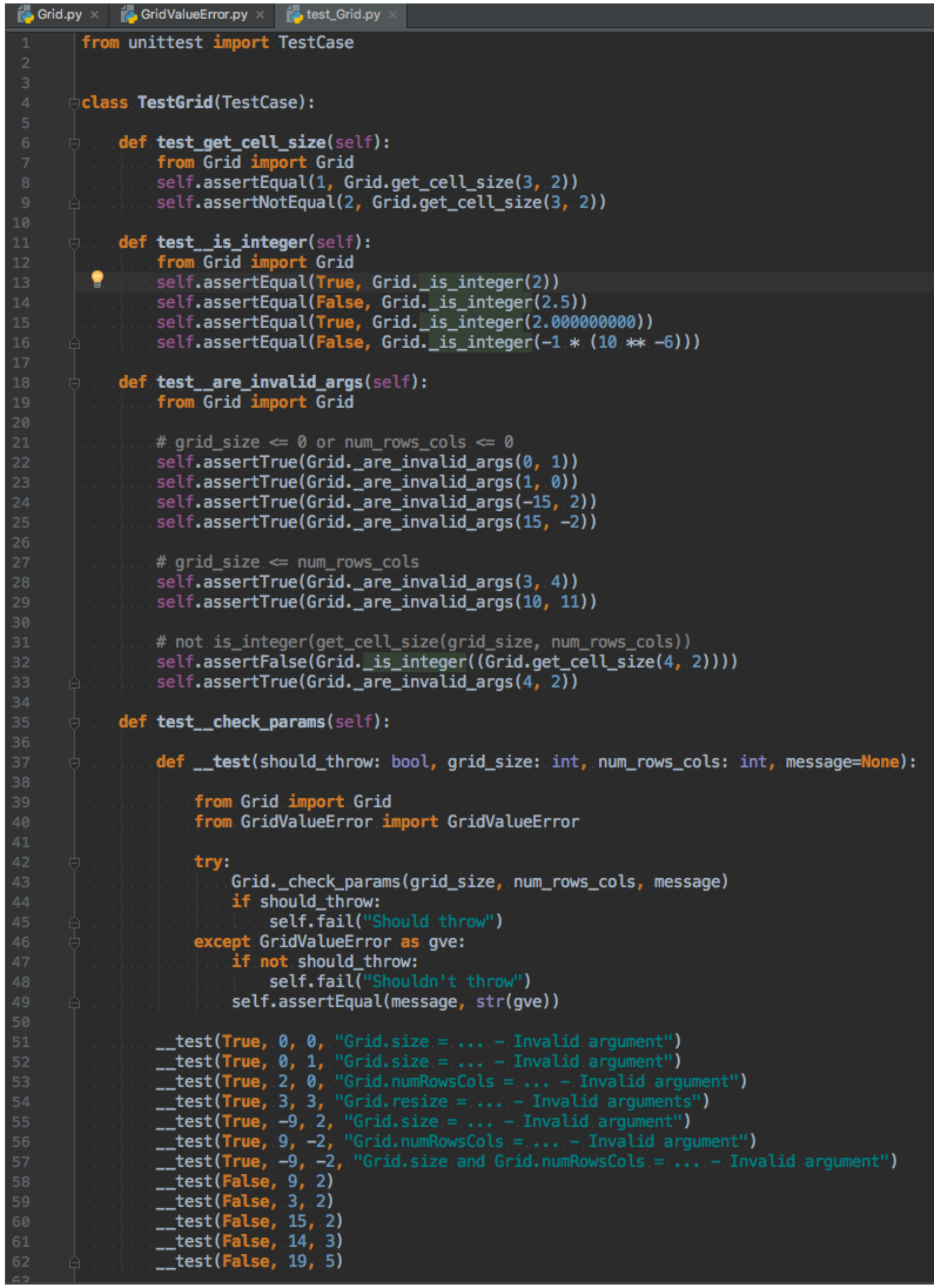
INVALID ARGS PASSED TO THE CONSTRUCTOR

Define _check_params which will raise GridValueError exception if invalid parameters are passed to the constructor



```python
class Grid:
    @staticmethod
    def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
        num_interior_borders = num_rows_cols - 1
        return (grid_size - num_interior_borders) / num_rows_cols

    @staticmethod
    def _is_integer(val) -> bool:
        import math
        return math.floor(val) == val

    @staticmethod
    def _are_invalid_args(grid_size: int, num_rows_cols: int) -> bool:
        return ((grid_size <= 0 or num_rows_cols <= 0) or
                (grid_size <= num_rows_cols) or not
                Grid._is_integer(Grid.get_cell_size(grid_size, num_rows_cols)))

    @staticmethod
    def _check_params(grid_size: int, num_rows_cols: int, message: str) -> None:
        pass
```

TDD IN PRACTICE – EXAMPLE: GRID CLASS



Grid.py × GridValueError.py × test_Grid.py ×

```python
from unittest import TestCase


class TestGrid(TestCase):

    def test_get_cell_size(self):
        from Grid import Grid
        self.assertEqual(1, Grid.get_cell_size(3, 2))
        self.assertNotEqual(2, Grid.get_cell_size(3, 2))

    def test__is_integer(self):
        from Grid import Grid
        self.assertEqual(True, Grid._is_integer(2))
        self.assertEqual(False, Grid._is_integer(2.5))
        self.assertEqual(True, Grid._is_integer(2.000000000))
        self.assertEqual(False, Grid._is_integer(-1 * (10 ** -6)))

    def test__are_invalid_args(self):
        from Grid import Grid

        # grid_size <= 0 or num_rows_cols <= 0
        self.assertTrue(Grid._are_invalid_args(0, 1))
        self.assertTrue(Grid._are_invalid_args(1, 0))
        self.assertTrue(Grid._are_invalid_args(-15, 2))
        self.assertTrue(Grid._are_invalid_args(15, -2))

        # grid_size <= num_rows_cols
        self.assertTrue(Grid._are_invalid_args(3, 4))
        self.assertTrue(Grid._are_invalid_args(10, 11))

        # not is_integer(get_cell_size(grid_size, num_rows_cols))
        self.assertFalse(Grid._is_integer((Grid.get_cell_size(4, 2))))
        self.assertTrue(Grid._are_invalid_args(4, 2))

    def test__check_params(self):

        def __test(should_throw: bool, grid_size: int, num_rows_cols: int, message=None):

            from Grid import Grid
            from GridValueError import GridValueError

            try:
                Grid._check_params(grid_size, num_rows_cols, message)
                if should_throw:
                    self.fail("Should throw")
            except GridValueError as gve:
                if not should_throw:
                    self.fail("Shouldn't throw")
                self.assertEqual(message, str(gve))

        __test(True, 0, 0, "Grid.size = ... - Invalid argument")
        __test(True, 0, 1, "Grid.size = ... - Invalid argument")
        __test(True, 2, 0, "Grid.numRowsCols = ... - Invalid argument")
        __test(True, 3, 3, "Grid.resize = ... - Invalid arguments")
        __test(True, -9, 2, "Grid.size = ... - Invalid argument")
        __test(True, 9, -2, "Grid.numRowsCols = ... - Invalid argument")
        __test(True, -9, -2, "Grid.size and Grid.numRowsCols = ... - Invalid argument")
        __test(False, 9, 2)
        __test(False, 3, 2)
        __test(False, 15, 2)
        __test(False, 14, 3)
        __test(False, 19, 5)
```

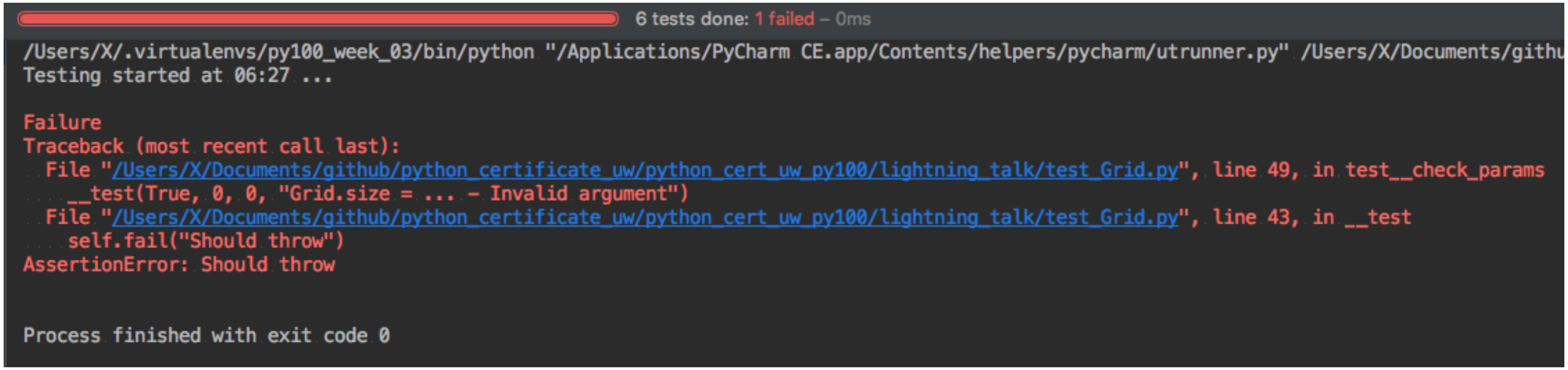
INVALID ARGS PASSED TO THE CONSTRUCTOR

Define _check_params which will raise GridValueError exception if invalid parameters are passed to the constructor

TDD IN PRACTICE – EXAMPLE: GRID CLASS

INVALID ARGS PASSED TO THE CONSTRUCTOR

Define _check_params which will raise GridValueError exception if invalid parameters are passed to the constructor



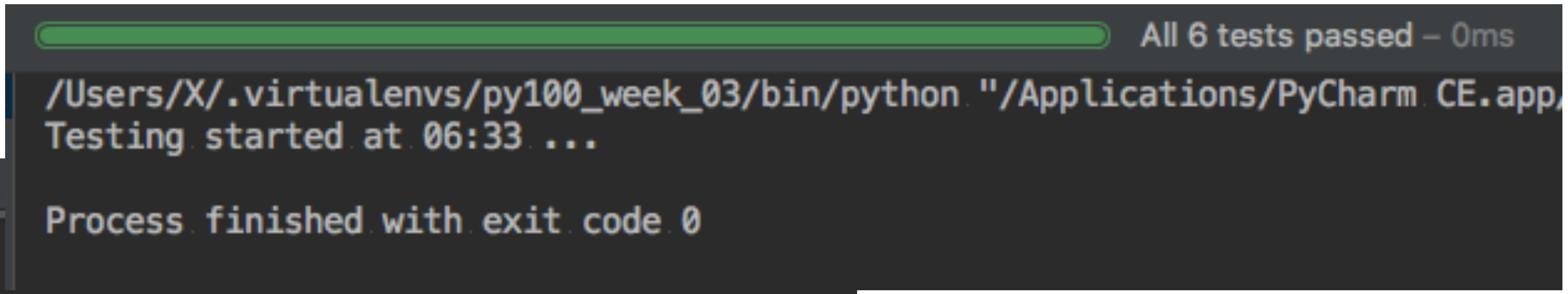
6 tests done: 1 failed – 0ms

```
/Users/X/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/Contents/helpers/pycharm/utrunner.py" /Users/X/Documents/githu
Testing started at 06:27 ...

Failure
Traceback (most recent call last):
  File "/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/test_Grid.py", line 49, in test__check_params
    __test(True, 0, 0, "Grid.size = ... - Invalid argument")
  File "/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/test_Grid.py", line 43, in __test
    self.fail("Should throw")
AssertionError: Should throw


Process finished with exit code 0
```

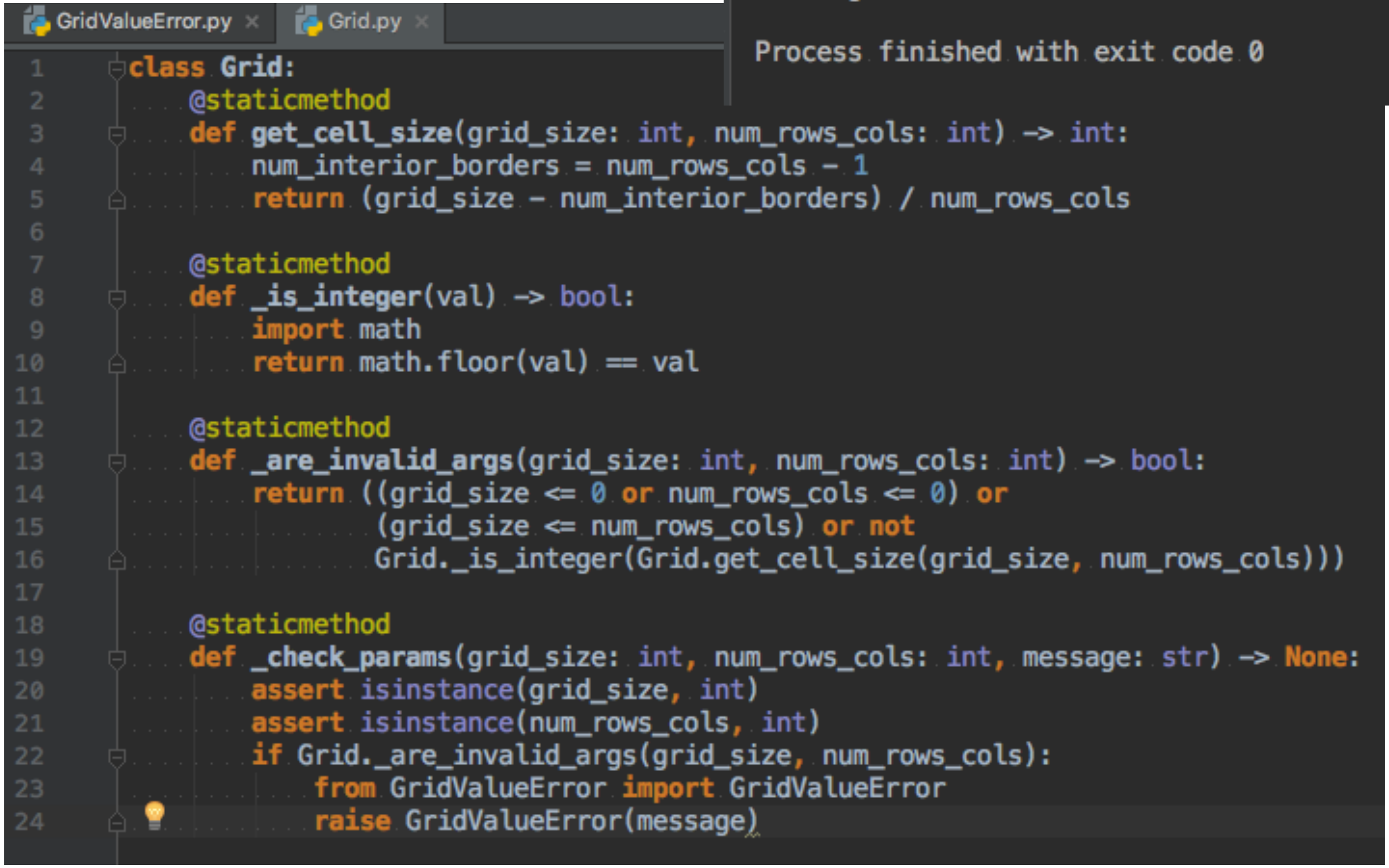
TDD IN PRACTICE – EXAMPLE: GRID CLASS

INVALID ARGS PASSED TO THE CONSTRUCTOR

Define _check_params which will raise GridValueError exception if invalid parameters are passed to the constructor



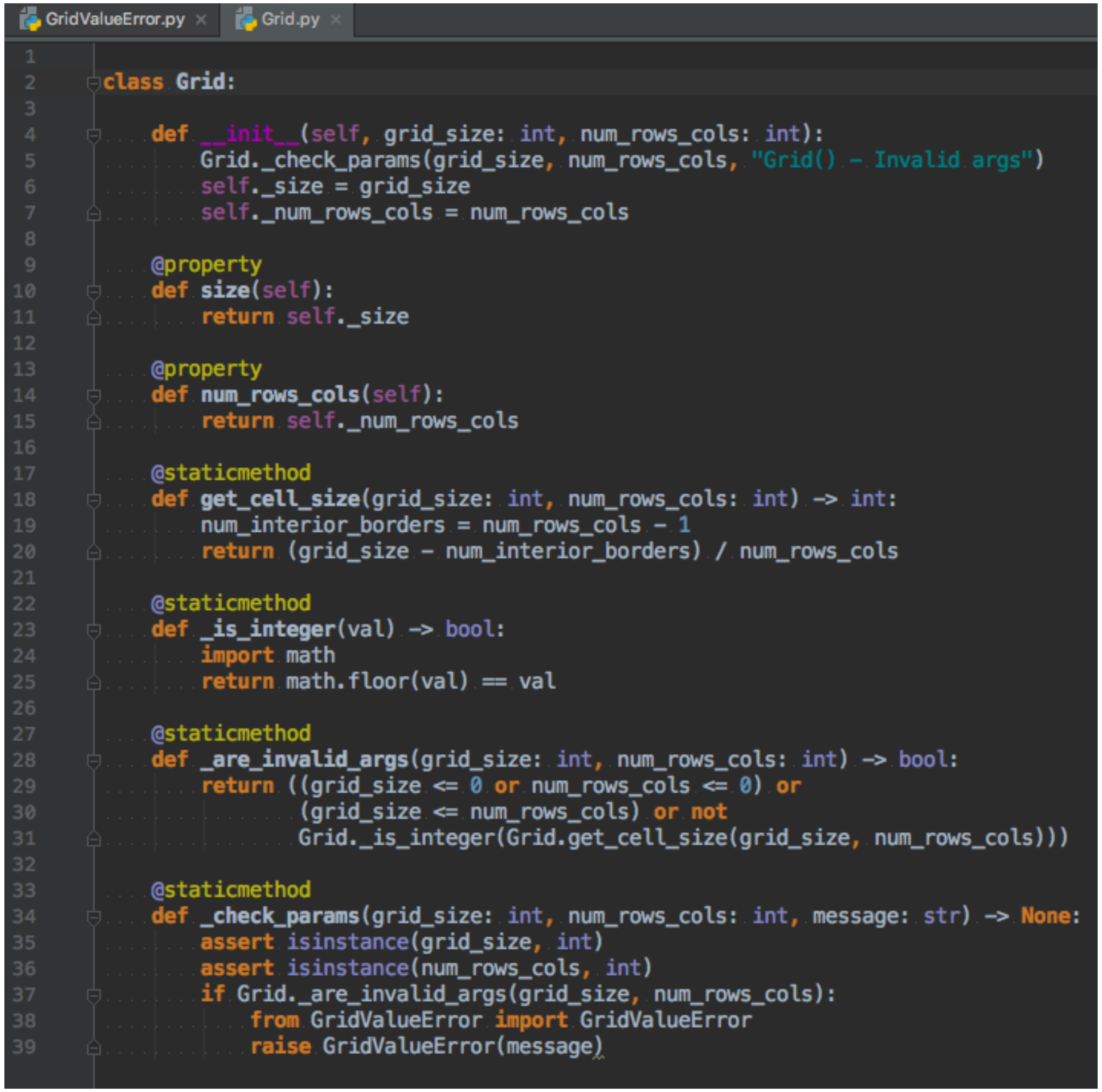
```
/Users/X/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/
Testing started at 06:33 ...

Process finished with exit code 0
```



```python
class Grid:
    @staticmethod
    def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
        num_interior_borders = num_rows_cols - 1
        return (grid_size - num_interior_borders) / num_rows_cols

    @staticmethod
    def _is_integer(val) -> bool:
        import math
        return math.floor(val) == val

    @staticmethod
    def _are_invalid_args(grid_size: int, num_rows_cols: int) -> bool:
        return ((grid_size <= 0 or num_rows_cols <= 0) or
                (grid_size <= num_rows_cols) or not
                Grid._is_integer(Grid.get_cell_size(grid_size, num_rows_cols)))

    @staticmethod
    def _check_params(grid_size: int, num_rows_cols: int, message: str) -> None:
        assert isinstance(grid_size, int)
        assert isinstance(num_rows_cols, int)
        if Grid._are_invalid_args(grid_size, num_rows_cols):
            from GridValueError import GridValueError
            raise GridValueError(message)
```

TDD IN PRACTICE – EXAMPLE: GRID CLASS

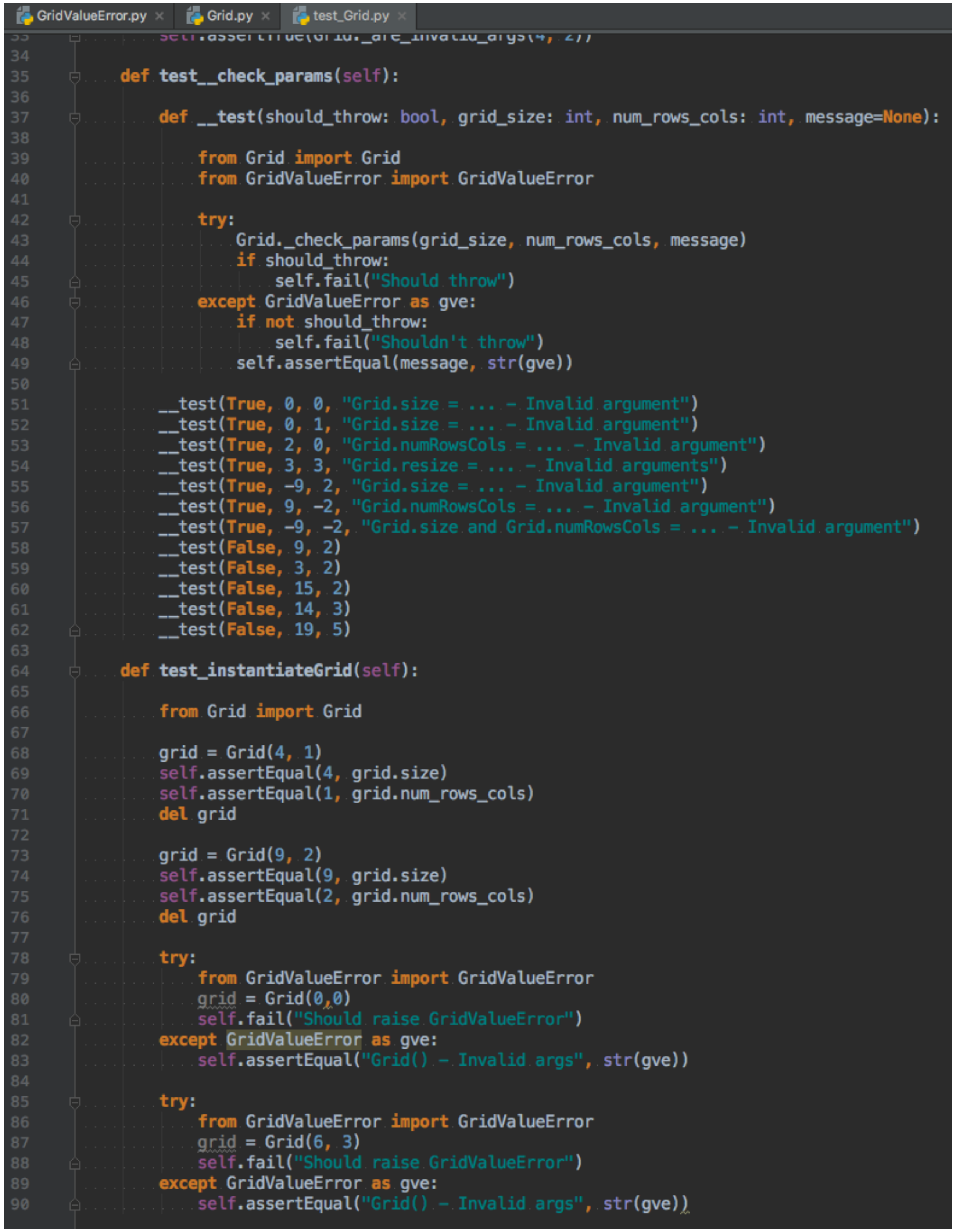
DEFINE THE CONSTRUCTOR



```python

class Grid:

    def __init__(self, grid_size: int, num_rows_cols: int):
        Grid._check_params(grid_size, num_rows_cols, "Grid() - Invalid args")
        self._size = grid_size
        self._num_rows_cols = num_rows_cols

    @property
    def size(self):
        return self._size

    @property
    def num_rows_cols(self):
        return self._num_rows_cols

    @staticmethod
    def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
        num_interior_borders = num_rows_cols - 1
        return (grid_size - num_interior_borders) / num_rows_cols

    @staticmethod
    def _is_integer(val) -> bool:
        import math
        return math.floor(val) == val

    @staticmethod
    def _are_invalid_args(grid_size: int, num_rows_cols: int) -> bool:
        return ((grid_size <= 0 or num_rows_cols <= 0) or
                (grid_size <= num_rows_cols) or not
                Grid._is_integer(Grid.get_cell_size(grid_size, num_rows_cols)))

    @staticmethod
    def _check_params(grid_size: int, num_rows_cols: int, message: str) -> None:
        assert isinstance(grid_size, int)
        assert isinstance(num_rows_cols, int)
        if Grid._are_invalid_args(grid_size, num_rows_cols):
            from GridValueError import GridValueError
            raise GridValueError(message)
```

TDD IN PRACTICE – EXAMPLE: GRID CLASS

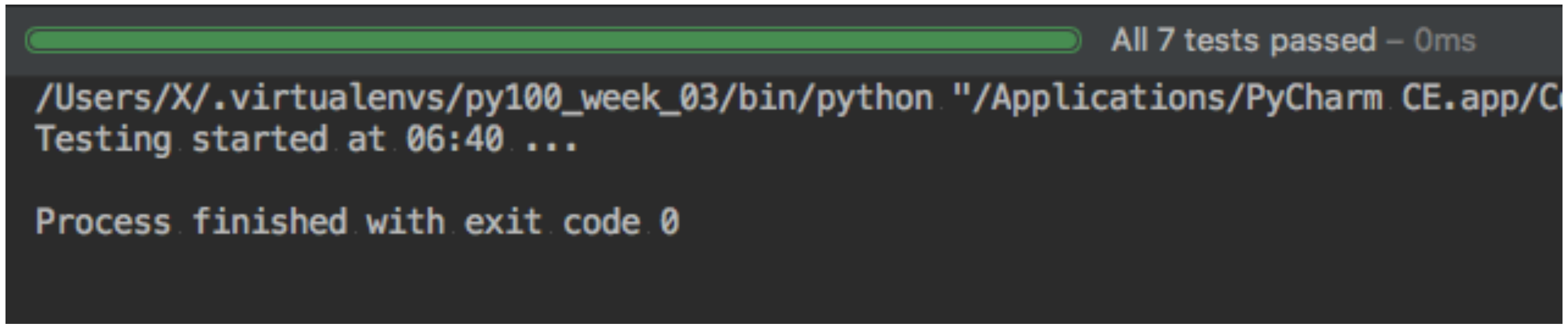
DEFINE THE CONSTRUCTOR


GridValueError.py ×   Grid.py ×   test_Grid.py ×

```python
        self.assertTrue(Grid._are_invalid_args(4, 2))

    def test__check_params(self):

        def __test(should_throw: bool, grid_size: int, num_rows_cols: int, message=None):

            from Grid import Grid
            from GridValueError import GridValueError

            try:
                Grid._check_params(grid_size, num_rows_cols, message)
                if should_throw:
                    self.fail("Should throw")
            except GridValueError as gve:
                if not should_throw:
                    self.fail("Shouldn't throw")
                self.assertEqual(message, str(gve))

        __test(True, 0, 0, "Grid.size = ... - Invalid argument")
        __test(True, 0, 1, "Grid.size = ... - Invalid argument")
        __test(True, 2, 0, "Grid.numRowsCols = ... - Invalid argument")
        __test(True, 3, 3, "Grid.resize = ... - Invalid arguments")
        __test(True, -9, 2, "Grid.size = ... - Invalid argument")
        __test(True, 9, -2, "Grid.numRowsCols = ... - Invalid argument")
        __test(True, -9, -2, "Grid.size and Grid.numRowsCols = ... - Invalid argument")
        __test(False, 9, 2)
        __test(False, 3, 2)
        __test(False, 15, 2)
        __test(False, 14, 3)
        __test(False, 19, 5)

    def test_instantiateGrid(self):

        from Grid import Grid

        grid = Grid(4, 1)
        self.assertEqual(4, grid.size)
        self.assertEqual(1, grid.num_rows_cols)
        del grid

        grid = Grid(9, 2)
        self.assertEqual(9, grid.size)
        self.assertEqual(2, grid.num_rows_cols)
        del grid

        try:
            from GridValueError import GridValueError
            grid = Grid(0,0)
            self.fail("Should raise GridValueError")
        except GridValueError as gve:
            self.assertEqual("Grid() - Invalid args", str(gve))

        try:
            from GridValueError import GridValueError
            grid = Grid(6, 3)
            self.fail("Should raise GridValueError")
        except GridValueError as gve:
            self.assertEqual("Grid() - Invalid args", str(gve))
```

TDD IN PRACTICE - EXAMPLE: GRID CLASS

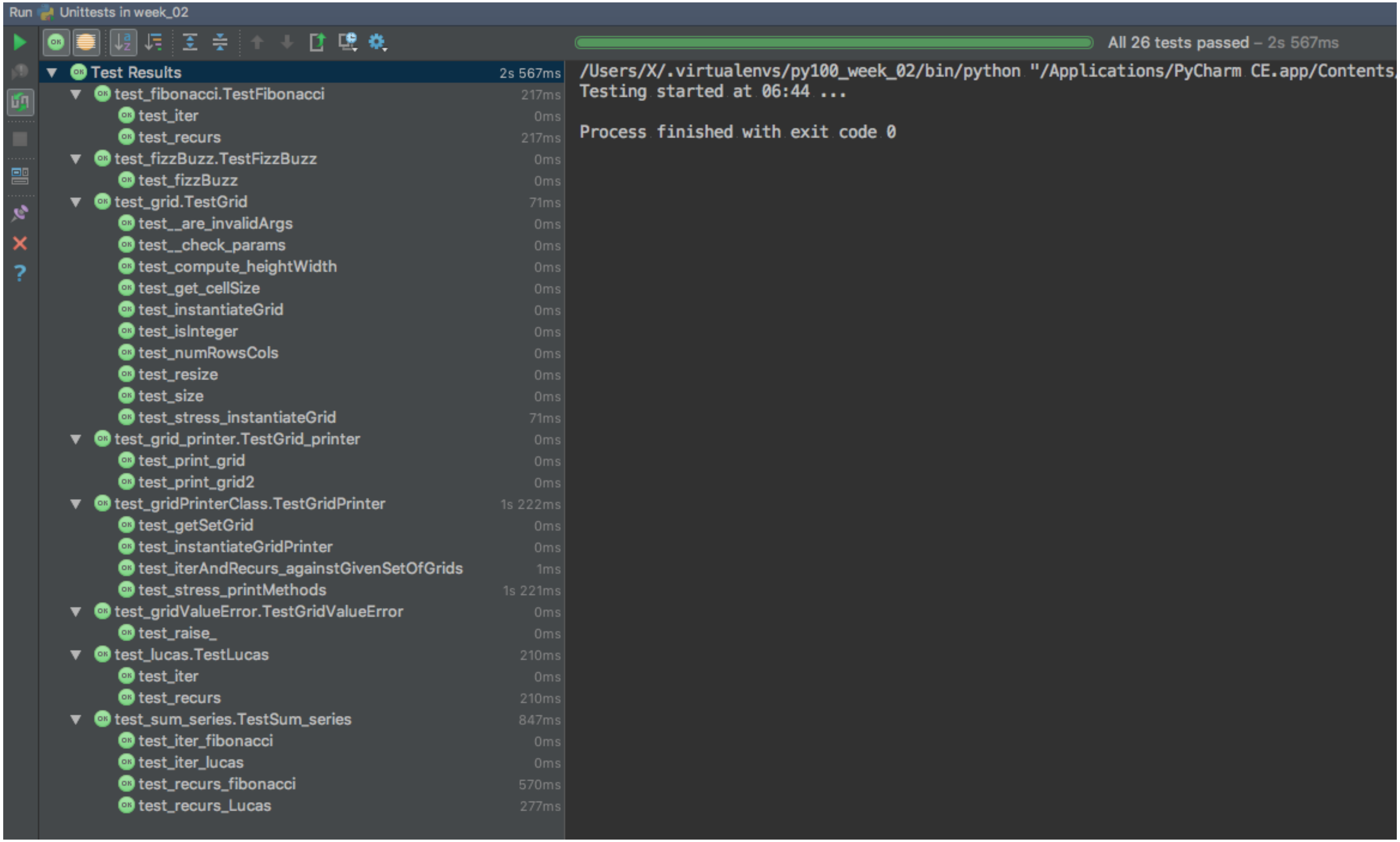
DEFINE THE CONSTRUCTOR



```
/Users/X/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/C
Testing started at 06:40 ...

Process finished with exit code 0
```

- As you build your program your set of unit tests grows
- Easy way to maintain « good coverage »

TDD IN PRACTICE – EXAMPLE: ASSIGNMENT 2

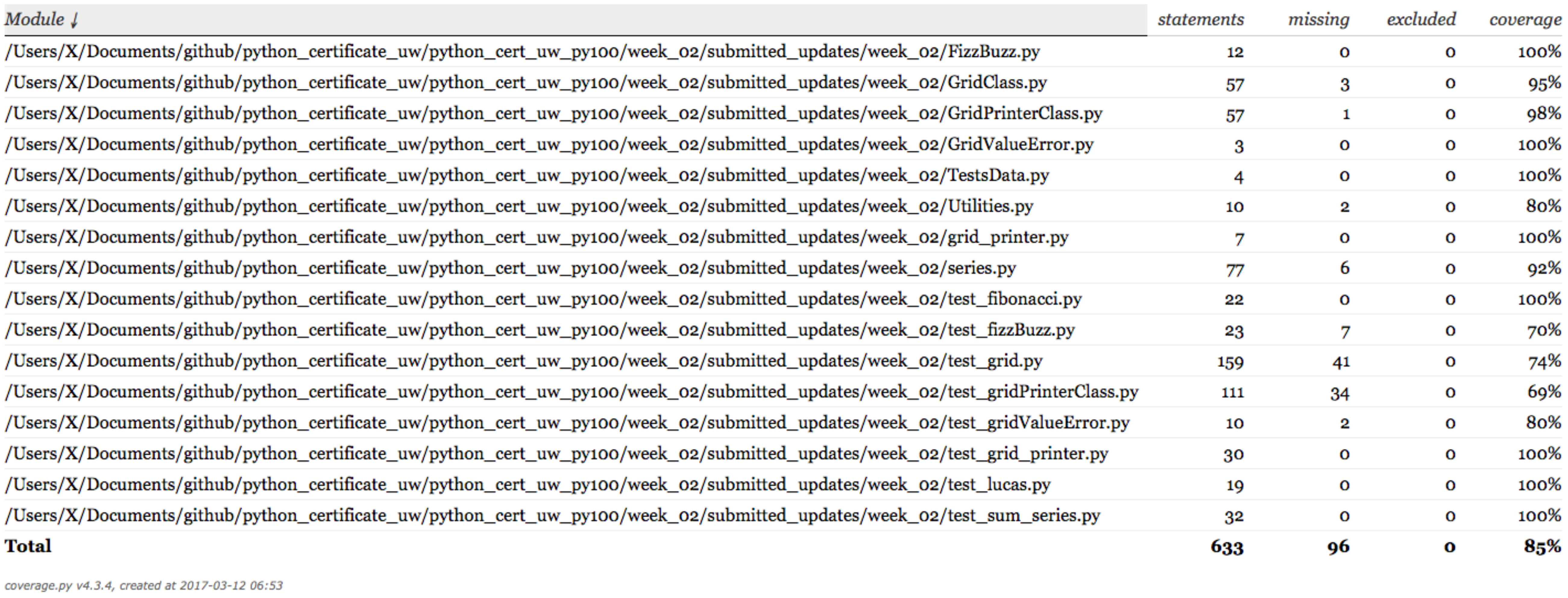


```
All 26 tests passed – 2s 567ms

Test Results                                          2s 567ms
  test_fibonacci.TestFibonacci                          217ms
    test_iter                                             0ms
    test_recurs                                         217ms
  test_fizzBuzz.TestFizzBuzz                              0ms
    test_fizzBuzz                                         0ms
  test_grid.TestGrid                                     71ms
    test__are_invalidArgs                                 0ms
    test__check_params                                    0ms
    test_compute_heightWidth                              0ms
    test_get_cellSize                                     0ms
    test_instantiateGrid                                  0ms
    test_isInteger                                        0ms
    test_numRowsCols                                      0ms
    test_resize                                           0ms
    test_size                                             0ms
    test_stress_instantiateGrid                          71ms
  test_grid_printer.TestGrid_printer                      0ms
    test_print_grid                                       0ms
    test_print_grid2                                      0ms
  test_gridPrinterClass.TestGridPrinter              1s 222ms
    test_getSetGrid                                       0ms
    test_instantiateGridPrinter                           0ms
    test_iterAndRecurs_againstGivenSetOfGrids             1ms
    test_stress_printMethods                         1s 221ms
  test_gridValueError.TestGridValueError                  0ms
    test_raise_                                           0ms
  test_lucas.TestLucas                                  210ms
    test_iter                                             0ms
    test_recurs                                         210ms
  test_sum_series.TestSum_series                        847ms
    test_iter_fibonacci                                   0ms
    test_iter_lucas                                       0ms
    test_recurs_fibonacci                               570ms
    test_recurs_Lucas                                   277ms
```

```
/Users/X/.virtualenvs/py100_week_02/bin/python "/Applications/PyCharm CE.app/Contents
Testing started at 06:44 ...

Process finished with exit code 0
```

TDD IN PRACTICE – EXAMPLE: ASSIGNMENT 2

COVERAGE REPORT – AVAILABLE IN PYCHARM PROFESSIONAL EDITION



Module ↓	*statements*	*missing*	*excluded*	*coverage*
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/FizzBuzz.py	12	0	0	100%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/GridClass.py	57	3	0	95%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/GridPrinterClass.py	57	1	0	98%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/GridValueError.py	3	0	0	100%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/TestsData.py	4	0	0	100%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/Utilities.py	10	2	0	80%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/grid_printer.py	7	0	0	100%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/series.py	77	6	0	92%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/test_fibonacci.py	22	0	0	100%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/test_fizzBuzz.py	23	7	0	70%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/test_grid.py	159	41	0	74%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/test_gridPrinterClass.py	111	34	0	69%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/test_gridValueError.py	10	2	0	80%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/test_grid_printer.py	30	0	0	100%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/test_lucas.py	19	0	0	100%
/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/week_02/submitted_updates/week_02/test_sum_series.py	32	0	0	100%
Total	**633**	**96**	**0**	**85%**

coverage.py v4.3.4, created at 2017-03-12 06:53

THANK YOU!