

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

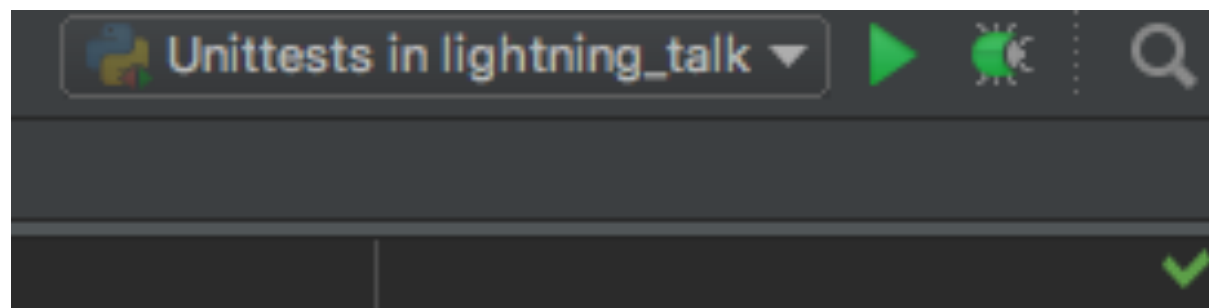
class TestFoo(TestCase):
    def test_bar(self):
        from Foo import Foo
        foo_a = Foo()
        self.assertEqual('Foo::bar()', foo_a.bar())
        self.assertEqual('Foo::not_bar()', foo_a.bar())
```

```
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

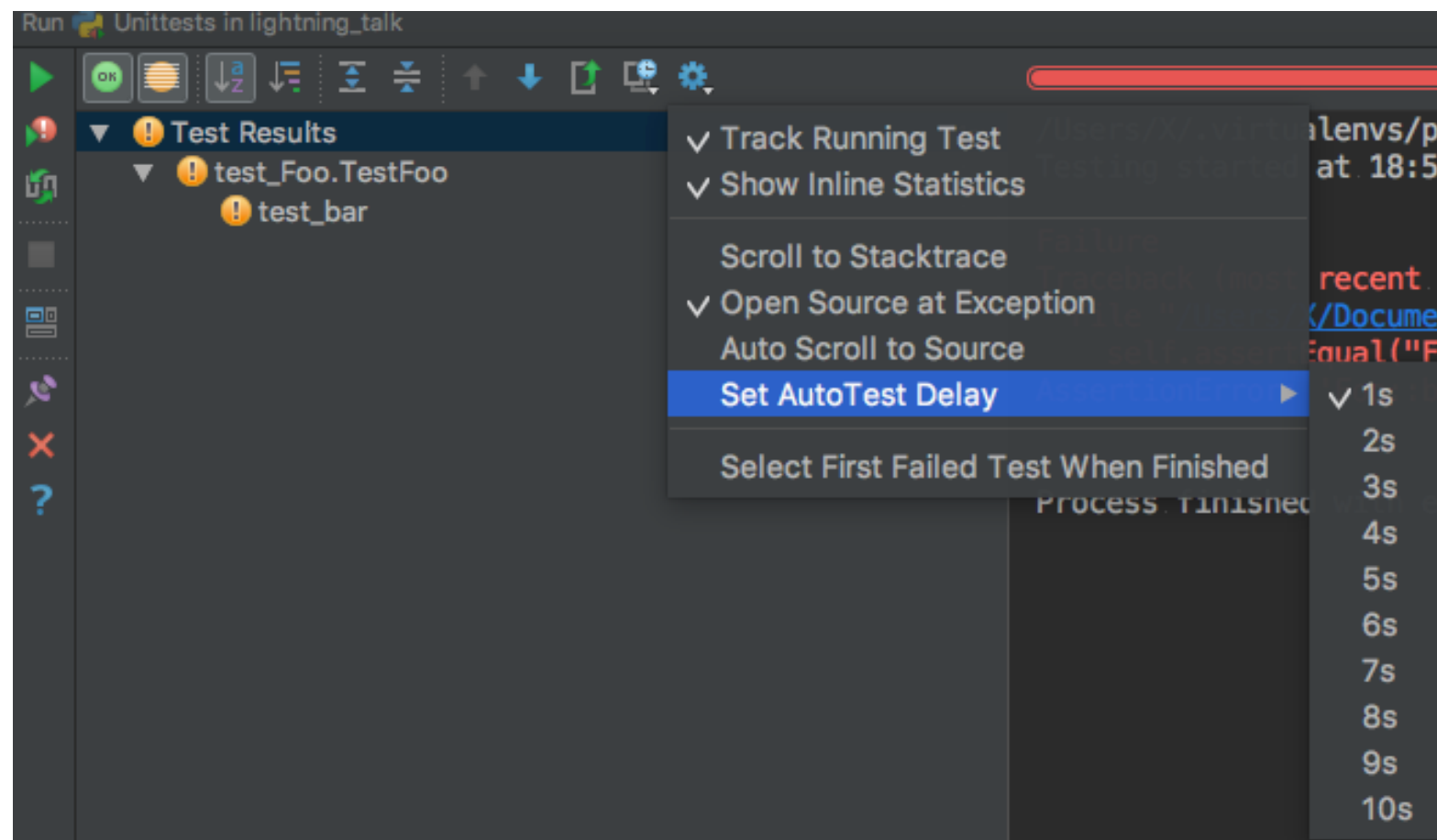


```
Failure
Traceback (most recent call last):
  File "/Users/X/Documents/github/python\_certificate\_uw/python\_cert\_uw\_py100/lightning\_talk/test\_Foo.py", line 9, in test_bar
    self.assertEqual("Foo::bar()", foo_a.bar())
AssertionError: 'Foo::bar()' != None
```

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

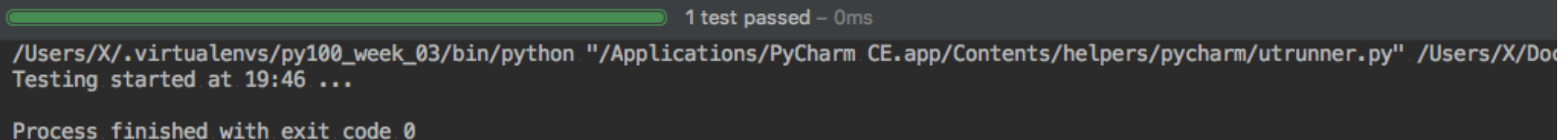
```
Foo.py  
  
class Foo:  
    def bar(self):  
        pass
```

```
1 test failed - 0ms  
/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()'
```



A screenshot of the PyCharm test runner interface. At the top, a green progress bar is followed by the text "1 test passed - 0ms". Below this, the command path is shown: `/Users/X/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/Contents/helpers/pycharm/utrunner.py" /Users/X/Doc`. The next line says "Testing started at 19:46 ...". At the bottom, it states "Process finished with exit code 0".

```
/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

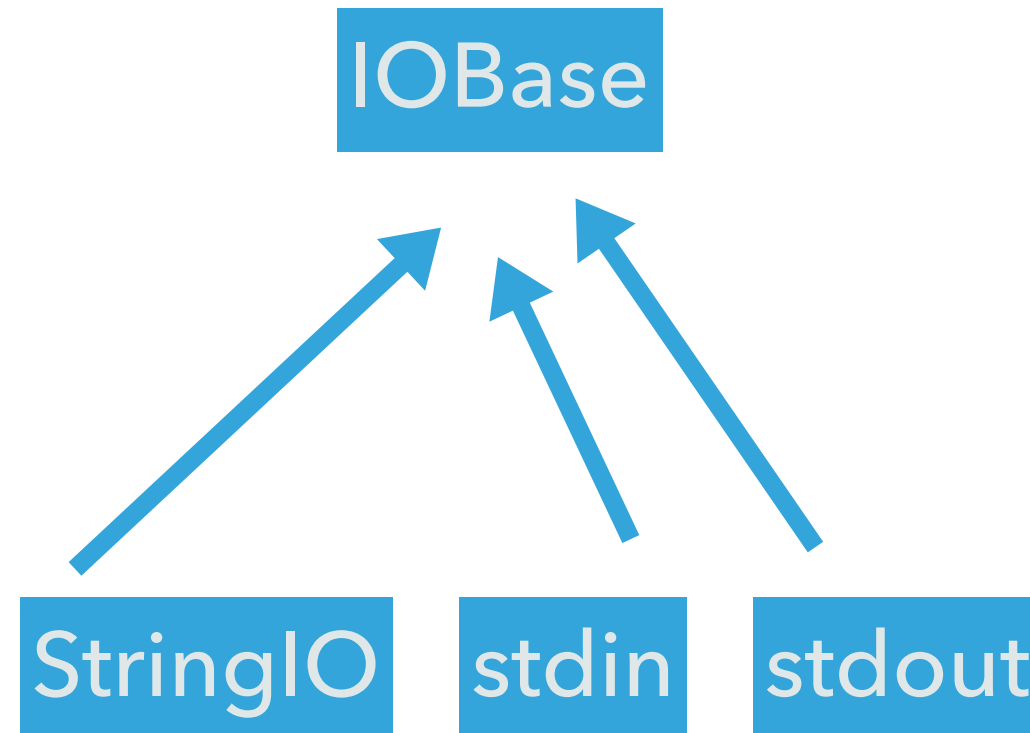
- ▶ How do you automatically test for messages which need to be printed to the standard output (console)?
- ▶ How do you automatically test user inputs?

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 unittest import TestCase

class TestFoo(TestCase):

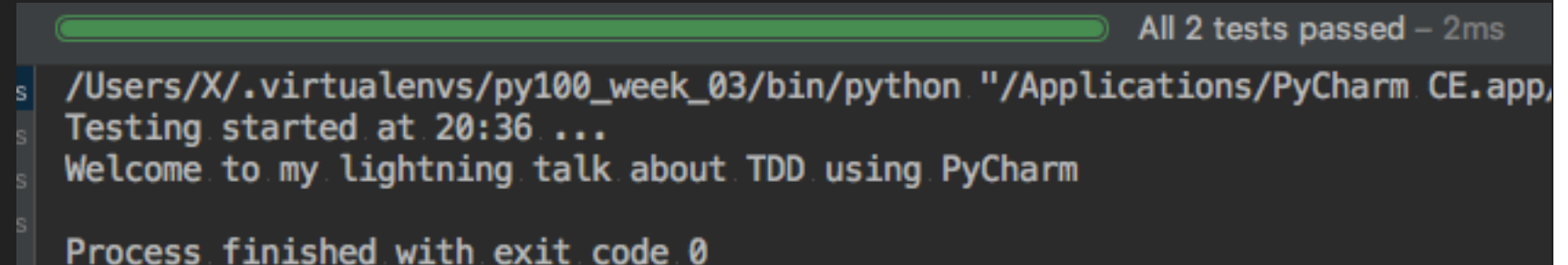
    def test_print_welcome_stdout(self):

        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

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)
```

```

/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

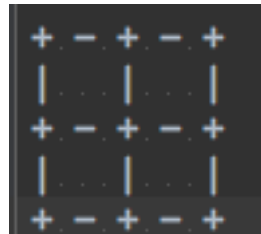
GRID PRINTER ASSIGNMENT SIMPLIFIED VERSION

- 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

```
Grid.py x
1
2 class Grid:
3
4     @staticmethod
5     def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
6         pass

test_Grid.py x
1 from unittest import TestCase
2
3
4 class TestGrid(TestCase):
5
6     def test_get_cell_size(self):
7         from Grid import Grid
8         self.assertEqual(1, Grid.get_cell_size(3, 2))
9
```

1 test 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 03:21 ...
```

```
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 x
1
2 class Grid:
3
4     @staticmethod
5     def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
6         num_interior_borders = num_rows_cols - 1
7         return (grid_size - num_interior_borders) / num_rows_cols
8

test_Grid.py x
1 from unittest import TestCase
2
3
4 class TestGrid(TestCase):
5
6     def test_get_cell_size(self):
7         from Grid import Grid
8         self.assertEqual(1, Grid.get_cell_size(3, 2))
9         self.assertNotEqual(2, Grid.get_cell_size(3, 2))
10
```

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

COMPUTED CELL_SIZE GIVEN ARGUMENTS IS AN INTEGER

```
Grid.py x
1
2 class Grid:
3
4     @staticmethod
5     def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
6         num_interior_borders = num_rows_cols - 1
7         return (grid_size - num_interior_borders) / num_rows_cols
8
9     @staticmethod
10    def _is_integer(val) -> bool:
11        pass
12

test_Grid.py x
1 from unittest import TestCase
2
3
4 class TestGrid(TestCase):
5
6     def test_get_cell_size(self):
7         from Grid import Grid
8         self.assertEqual(1, Grid.get_cell_size(3, 2))
9         self.assertNotEqual(2, Grid.get_cell_size(3, 2))
10
11    def test__is_integer(self):
12        from Grid import Grid
13        self.assertEqual(True, Grid._is_integer(2))
14        self.assertEqual(False, Grid._is_integer(2.5))
15        self.assertEqual(True, Grid._is_integer(2.000000000))
16        self.assertEqual(False, Grid._is_integer(-1 * (10 ** -6)))
17
```

```
2 tests done: 1 failed – 0ms

/Users/vianneystreicher/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/Contents/helpers/pycharm/utrunner.py" ↗
/Users/vianneystreicher/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/vianneystreicher/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 x
1
2 class Grid:
3
4     @staticmethod
5     def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
6         num_interior_borders = num_rows_cols - 1
7         return (grid_size - num_interior_borders) / num_rows_cols
8
9     @staticmethod
10    def _is_integer(val) -> bool:
11        import math
12        return math.floor(val) == val
13
test_Grid.py x
1 from unittest import TestCase
2
3
4 class TestGrid(TestCase):
5
6     def test_get_cell_size(self):
7         from Grid import Grid
8         self.assertEqual(1, Grid.get_cell_size(3, 2))
9         self.assertNotEqual(2, Grid.get_cell_size(3, 2))
10
11    def test__is_integer(self):
12        from Grid import Grid
13        self.assertEqual(True, Grid._is_integer(2))
14        self.assertEqual(False, Grid._is_integer(2.5))
15        self.assertEqual(True, Grid._is_integer(2.000000000))
16        self.assertEqual(False, Grid._is_integer(-1 * (10 ** -6)))
```

All 2 tests passed – 0ms

```
/Users/vianneystreicher/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app/Contents/helpers/pycharm/utrunner.py" ↗
↘/Users/vianneystreicher/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

INVALID ARGS PASSED TO THE CONSTRUCTOR

```
Grid.py x
1
2 class Grid:
3
4     @staticmethod
5     def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
6         num_interior_borders = num_rows_cols - 1
7         return (grid_size - num_interior_borders) / num_rows_cols
8
9     @staticmethod
10    def _is_integer(val) -> bool:
11        import math
12        return math.floor(val) == val
13
14    @staticmethod
15    def _are_invalid_args(grid_size: int, num_rows_cols: int) -> bool:
16        pass
17
```

```
test_Grid.py x
4 class TestGrid(TestCase):
5
6     def test_get_cell_size(self):
7         from Grid import Grid
8         self.assertEqual(1, Grid.get_cell_size(3, 2))
9         self.assertNotEqual(2, Grid.get_cell_size(3, 2))
10
11    def test__is_integer(self):
12        from Grid import Grid
13        self.assertEqual(True, Grid._is_integer(2))
14        self.assertEqual(False, Grid._is_integer(2.5))
15        self.assertEqual(True, Grid._is_integer(2.000000000))
16        self.assertEqual(False, Grid._is_integer(-1 * (10 ** -6)))
17
18    def test__are_invalid_args(self):
19        from Grid import Grid
20
21        # grid_size == 0 or num_rows_cols == 0
22        self.assertTrue(Grid._are_invalid_args(0, 1))
23        self.assertTrue(Grid._are_invalid_args(1, 0))
24
25        # grid_size <= num_rows_cols
26        self.assertTrue(Grid._are_invalid_args(3, 4))
27        self.assertTrue(Grid._are_invalid_args(10, 11))
28
29        # not is_integer(get_cell_size(grid_size, num_rows_cols))
30        self.assertFalse(Grid._is_integer((Grid.get_cell_size(4, 2))))
31        self.assertTrue(Grid._are_invalid_args(4, 2))
```

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

```
Grid.py x
1
2 class Grid:
3
4     @staticmethod
5     def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
6         num_interior_borders = num_rows_cols - 1
7         return (grid_size - num_interior_borders) / num_rows_cols
8
9     @staticmethod
10    def _is_integer(val) -> bool:
11        import math
12        return math.floor(val) == val
13
14    @staticmethod
15    def _are_invalid_args(grid_size: int, num_rows_cols: int) -> bool:
16        return ((grid_size == 0 or num_rows_cols == 0) or
17                (grid_size <= num_rows_cols) or not
18                Grid._is_integer(Grid.get_cell_size(grid_size, num_rows_c
19
20
21 test_Grid.py x
22
23 from unittest import TestCase
24
25 class TestGrid(TestCase):
26
27     def test_get_cell_size(self):
28         from Grid import Grid
29         self.assertEqual(1, Grid.get_cell_size(3, 2))
30         self.assertNotEqual(2, Grid.get_cell_size(3, 2))
31
32     def test__is_integer(self):
33         from Grid import Grid
34         self.assertEqual(True, Grid._is_integer(2))
35         self.assertEqual(False, Grid._is_integer(2.5))
36         self.assertEqual(True, Grid._is_integer(2.000000000))
37         self.assertEqual(False, Grid._is_integer(-1 * (10 ** -6)))
38
39     def test__are_invalid_args(self):
40         from Grid import Grid
41
42         # grid_size == 0 or num_rows_cols == 0
43         self.assertTrue(Grid._are_invalid_args(0, 1))
44         self.assertTrue(Grid._are_invalid_args(1, 0))
45
46         # grid_size <= num_rows_cols
47         self.assertTrue(Grid._are_invalid_args(3, 4))
48         self.assertTrue(Grid._are_invalid_args(10, 11))
49
50         # not is_integer(get_cell_size(grid_size, num_rows_cols))
51         self.assertFalse(Grid._is_integer((Grid.get_cell_size(4, 2))))
52         self.assertTrue(Grid._are_invalid_args(4, 2))
```

INVALID ARGS PASSED TO THE CONSTRUCTOR

```
All 3 tests passed – 0ms
/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 x GridValueError.py x temp.py x
1
2
3 class GridValueError(ValueError):
4     pass
5

test_Grid.py x test_GridValueError.py x
1 from unittest import TestCase
2
3
4 class TestGridValueError(TestCase):
5
6     def test_GridValueError(self):
7
8         from GridValueError import GridValueError
9         message = "GridValueError - test_GridValueError"
10
11         try:
12             raise GridValueError(message)
13             self.fail("GridValueError should have been raised")
14         except GridValueError as gve:
15             self.assertEqual(message, str(gve))
16             self.assertNotEqual(message + " ", str(gve))
17         except ValueError as ve:
18             self.fail("GridValueError exception should have been caught")
19         except:
20             self.fail("test_GridValueError - uncaught exception")
21
22
23     def test_GridValueError2(self):
24
25         from GridValueError import GridValueError
26         message = "GridValueError - test_GridValueError"
27
28         try:
29             raise GridValueError(message)
30             self.fail("GridValueError should have been raised")
31         except ValueError as ve:
32             self.assertEqual(message, str(ve))
33             self.assertNotEqual(message + " ", str(ve))
34         except:
35             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

```
Grid.py x GridValueError.py x temp.py x
2 class Grid:
3
4     @staticmethod
5     def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
6         num_interior_borders = num_rows_cols - 1
7         return (grid_size - num_interior_borders) / num_rows_cols
8
9     @staticmethod
10    def _is_integer(val) -> bool:
11        import math
12        return math.floor(val) == val
13
14    @staticmethod
15    def _are_invalid_args(grid_size: int, num_rows_cols: int) -> bool:
16        return ((grid_size == 0 or num_rows_cols == 0) or
17                (grid_size <= num_rows_cols) or not
18                Grid._is_integer(Grid.get_cell_size(grid_size, num_rows_cols)))
19
20    @staticmethod
21    def _check_params(grid_size: int, num_rows_cols: int, message: str) -> None:
22        pass
23
```

TDD IN PRACTICE – EXAMPLE: GRID CLASS

```

1  from unittest import TestCase
2
3
4  class TestGrid(TestCase):
5
6      def test_get_cell_size(self):
7          from Grid import Grid
8          self.assertEqual(1, Grid.get_cell_size(3, 2))
9          self.assertNotEqual(2, Grid.get_cell_size(3, 2))
10
11     def test__is_integer(self):
12         from Grid import Grid
13         self.assertEqual(True, Grid._is_integer(2))
14         self.assertEqual(False, Grid._is_integer(2.5))
15         self.assertEqual(True, Grid._is_integer(2.000000000))
16         self.assertEqual(False, Grid._is_integer(-1 * (10 ** -6)))
17
18     def test__are_invalid_args(self):
19         from Grid import Grid
20
21         # grid_size == 0 or num_rows_cols == 0
22         self.assertTrue(Grid._are_invalid_args(0, 1))
23         self.assertTrue(Grid._are_invalid_args(1, 0))
24
25         # grid_size <= num_rows_cols
26         self.assertTrue(Grid._are_invalid_args(3, 4))
27         self.assertTrue(Grid._are_invalid_args(10, 11))
28
29         # not is_integer(get_cell_size(grid_size, num_rows_cols))
30         self.assertFalse(Grid._is_integer((Grid.get_cell_size(4, 2))))
31         self.assertTrue(Grid._are_invalid_args(4, 2))
32
33     def test__check_params(self):
34
35         def __test(should_throw: bool, grid_size: int, num_rows_cols: int, message=None):
36
37             from Grid import Grid
38             from GridValueError import GridValueError
39
40             try:
41                 Grid._check_params(grid_size, num_rows_cols, message)
42                 if should_throw:
43                     self.fail("Should throw")
44             except GridValueError as gve:
45                 if not should_throw:
46                     self.fail("Shouldn't throw")
47             self.assertEqual(message, str(gve))
48
49         __test(True, 0, 0, "Grid.size = ... - Invalid argument")
50         __test(True, 0, 1, "Grid.size = ... - Invalid argument")
51         __test(True, 2, 0, "Grid.numRowsCols = ... - Invalid argument")
52         __test(True, 3, 3, "Grid.resize = ... - Invalid arguments")
53         __test(False, 9, 2)
54         __test(False, 3, 2)
55         __test(False, 15, 2)
56         __test(False, 14, 3)
57         __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/github
Testing started at 06:27 ...

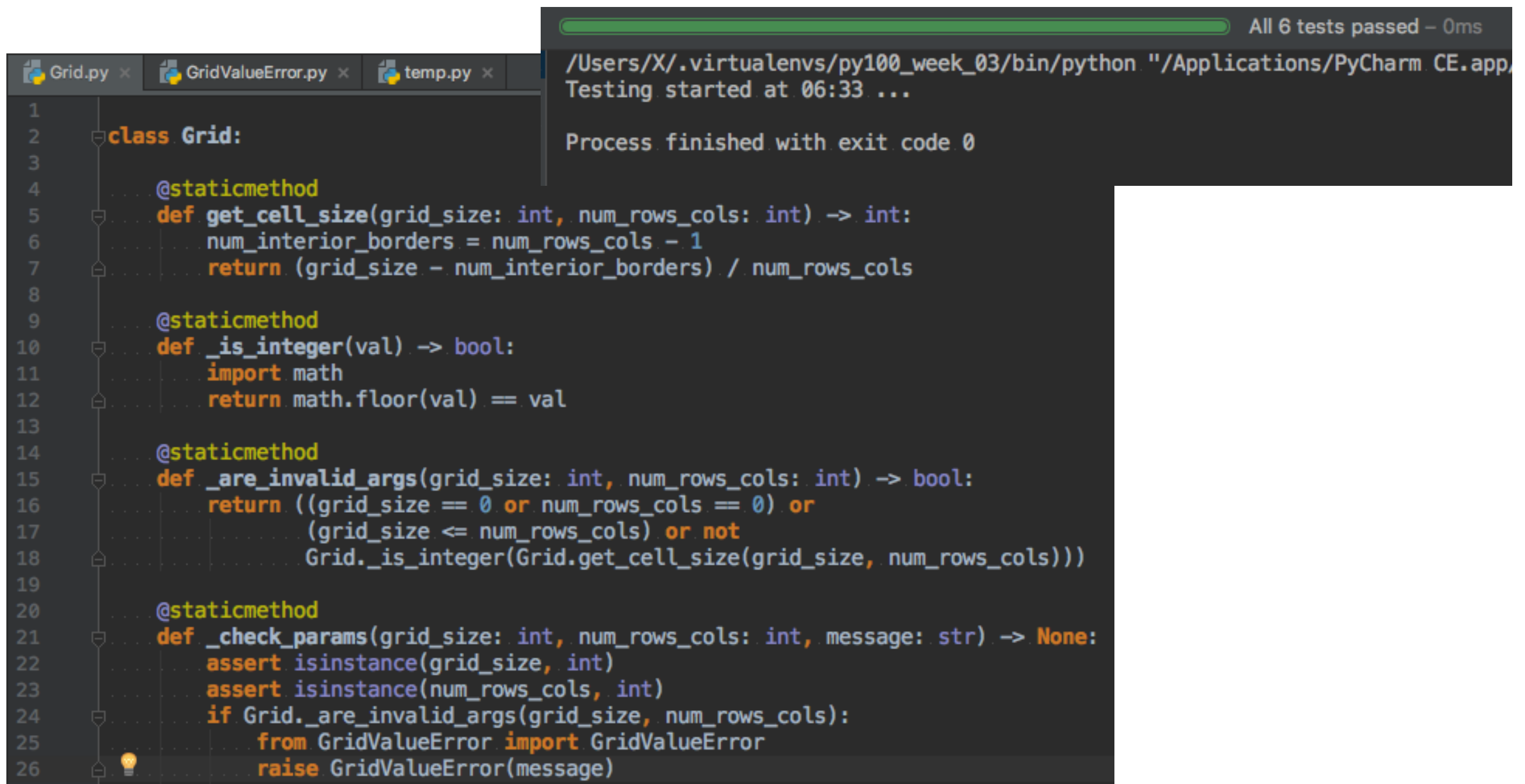
Failure
Traceback (most recent call last):
  File "/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/test_Grid.py", line 49, in test__check_params
    __test(True, 0, 0, "Grid.size = ... - Invalid argument")
  File "/Users/X/Documents/github/python_certificate_uw/python_cert_uw_py100/lightning_talk/test_Grid.py",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



```
Grid.py x GridValueError.py x temp.py x
1
2 class Grid:
3
4     @staticmethod
5     def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
6         num_interior_borders = num_rows_cols - 1
7         return (grid_size - num_interior_borders) / num_rows_cols
8
9     @staticmethod
10    def _is_integer(val) -> bool:
11        import math
12        return math.floor(val) == val
13
14    @staticmethod
15    def _are_invalid_args(grid_size: int, num_rows_cols: int) -> bool:
16        return ((grid_size == 0 or num_rows_cols == 0) or
17                (grid_size <= num_rows_cols) or not
18                Grid._is_integer(Grid.get_cell_size(grid_size, num_rows_cols)))
19
20    @staticmethod
21    def _check_params(grid_size: int, num_rows_cols: int, message: str) -> None:
22        assert isinstance(grid_size, int)
23        assert isinstance(num_rows_cols, int)
24        if Grid._are_invalid_args(grid_size, num_rows_cols):
25            from GridValueError import GridValueError
26            raise GridValueError(message)
```

All 6 tests passed – 0ms

/Users/X/.virtualenvs/py100_week_03/bin/python "/Applications/PyCharm CE.app" ...

Testing started at 06:33 ...

Process finished with exit code 0

TDD IN PRACTICE – EXAMPLE: GRID CLASS

DEFINE THE CONSTRUCTOR

```
Grid.py x
1
2 class Grid:
3
4     def __init__(self, grid_size: int, num_rows_cols: int):
5         Grid._check_params(grid_size, num_rows_cols, "Grid() - Invalid args")
6         self._size = grid_size
7         self._num_rows_cols = num_rows_cols
8
9     @property
10    def size(self):
11        return self._size
12
13    @property
14    def num_rows_cols(self):
15        return self._num_rows_cols
16
17    @staticmethod
18    def get_cell_size(grid_size: int, num_rows_cols: int) -> int:
19        num_interior_borders = num_rows_cols - 1
20        return (grid_size - num_interior_borders) / num_rows_cols
21
22    @staticmethod
23    def _is_integer(val) -> bool:
24        import math
25        return math.floor(val) == val
26
27    @staticmethod
28    def _are_invalid_args(grid_size: int, num_rows_cols: int) -> bool:
29        return ((grid_size == 0 or num_rows_cols == 0) or
30                (grid_size <= num_rows_cols) or not
31                Grid._is_integer(Grid.get_cell_size(grid_size, num_rows_cols)))
32
33    @staticmethod
34    def _check_params(grid_size: int, num_rows_cols: int, message: str) -> None:
35        assert isinstance(grid_size, int)
36        assert isinstance(num_rows_cols, int)
37        if Grid._are_invalid_args(grid_size, num_rows_cols):
38            from GridValueError import GridValueError
39            raise GridValueError(message)
40
```

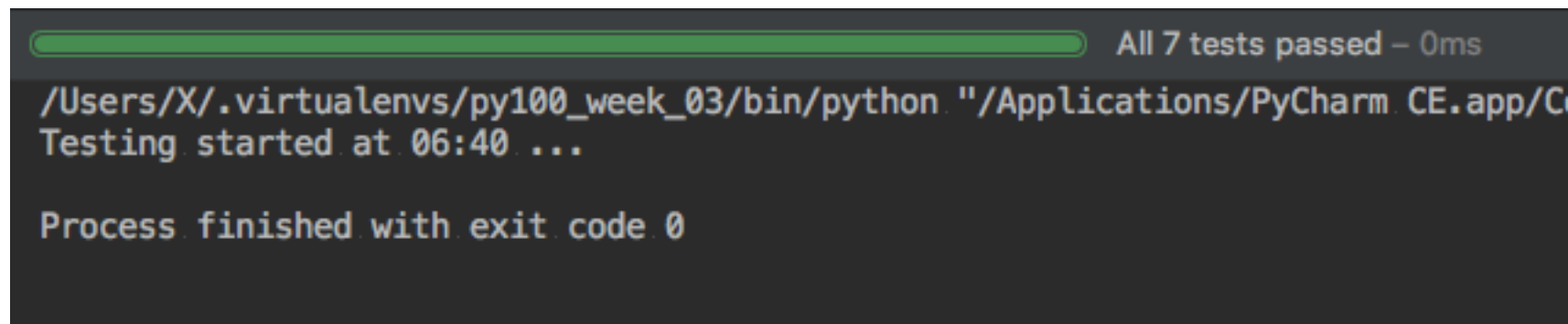

TDD IN PRACTICE – EXAMPLE: GRID CLASS

DEFINE THE CONSTRUCTOR

```
test_Grid.py x
35 def __test_should_throw(self, should_throw: bool, grid_size: int, num_rows_cols: int, message=None):
36
37     from Grid import Grid
38     from GridValueError import GridValueError
39
40     try:
41         Grid._check_params(grid_size, num_rows_cols, message)
42         if should_throw:
43             self.fail("Should throw")
44     except GridValueError as gve:
45         if not should_throw:
46             self.fail("Shouldn't throw")
47         self.assertEqual(message, str(gve))
48
49     __test(True, 0, 0, "Grid.size = ... - Invalid argument")
50     __test(True, 0, 1, "Grid.size = ... - Invalid argument")
51     __test(True, 2, 0, "Grid.numRowsCols = ... - Invalid argument")
52     __test(True, 3, 3, "Grid.resize = ... - Invalid arguments")
53     __test(False, 9, 2)
54     __test(False, 3, 2)
55     __test(False, 15, 2)
56     __test(False, 14, 3)
57     __test(False, 19, 5)
58
59 def test_instantiateGrid(self):
60
61     from Grid import Grid
62
63     grid = Grid(4, 1)
64     self.assertEqual(4, grid.size)
65     self.assertEqual(1, grid.num_rows_cols)
66     del grid
67
68     grid = Grid(9, 2)
69     self.assertEqual(9, grid.size)
70     self.assertEqual(2, grid.num_rows_cols)
71     del grid
72
73     try:
74         from GridValueError import GridValueError
75         grid = Grid(0, 0)
76         self.fail("Should raise GridValueError")
77     except GridValueError as gve:
78         self.assertEqual("Grid() - Invalid args", str(gve))
79
80     try:
81         from GridValueError import GridValueError
82         grid = Grid(6, 3)
83         self.fail("Should raise GridValueError")
84     except GridValueError as gve:
85         self.assertEqual("Grid() - Invalid args", str(gve))
```

TDD IN PRACTICE – EXAMPLE: GRID CLASS


DEFINE THE CONSTRUCTOR




























A screenshot of a terminal window with a dark background. At the top, a green progress bar is followed by the text "All 7 tests passed – 0ms". Below this, the command path "/Users/X/.virtualenvs/py100_week_03/bin/python. "/Applications/PyCharm CE.app/C" is visible. The next line says "Testing started at 06:40 ...". The final line indicates "Process finished with exit code 0".




























































































































































































































































































































































































































```
All 7 tests passed – 0ms
/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

Run  Unittests in week_02

                           All 26 tests passed – 2s 567ms

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!