Debugging With PyCharm

Disclaimer

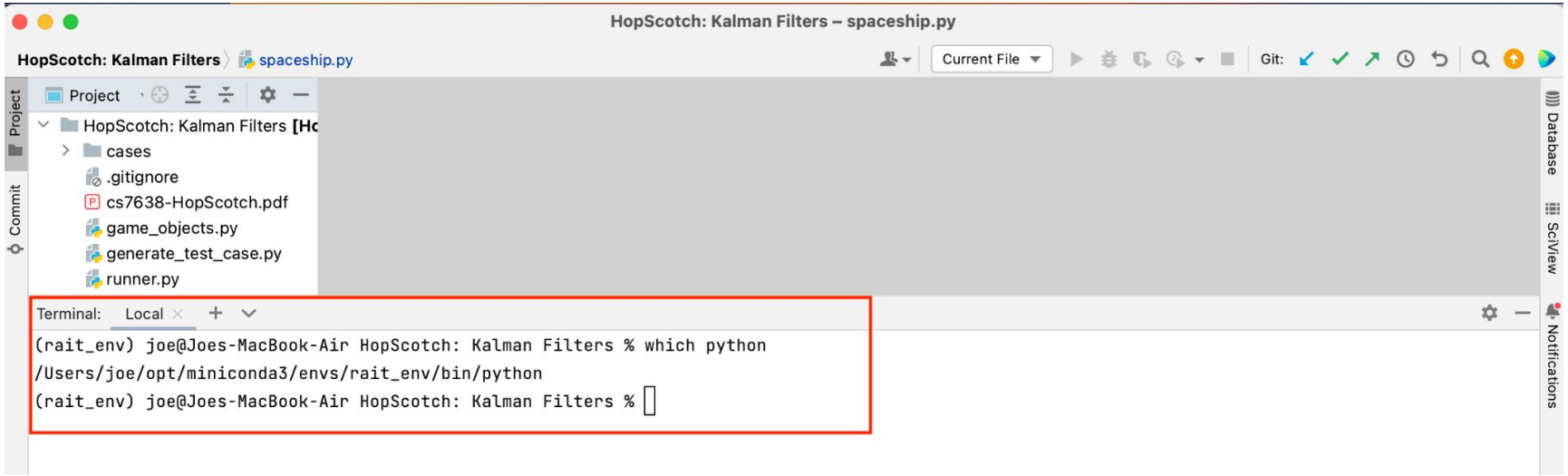
- No right/wrong way to debug
- Ultimately a skill that takes practice
- Very difficult to debug a fundamental misunderstanding

Why PyCharm?

- Community edition is free to use
  - Professional edition is available for GaTech Students
  - Can download from here
- High quality debugging and code inspection tools
- Integrates nicely with testing/grading code

PyCharm Project Setup

- **Make sure you're using the course Python environment**



General Debugging Tips

- **Theoretical understanding is critical!**
- Use public threads to rubber-duck
- Use the project visualizations!
- Tests can be run individually
  - Start with the simplest case to debug (e.g. no noise) if you’re failing multiple cases
- Use unit tests to verify basic functionality

Unit Tests?

- A unit test is just a function that verifies your code works
- The more testable your code is (and the more tests you have), the less debugging you have to do!

How to Add a Test

- General process
  - Write code, ideally in small snippets that can run in isolation
  - Write tests that exercise this code
  - Don’t need to worry about this code working/not working **ever again**

Adding Code

- Say we need to write code that computes the Euclidean distance
- Better to write a separate function or embed this in a project function like **estimate_next_position**?

Sample Code

HopScotch: Kalman Filters – spaceship.py

HopScotch: Kalman Filters 〉 spaceship.py

Current File

Project

Commit

- HopScotch: Kalman Filters [Hc
  - cases
  - .gitignore
  - cs7638-HopScotch.pdf
  - game_objects.py
  - generate_test_case.py
  - runner.py
  - settings.py
  - spaceship.py
  - test.py
  - utilities.py
  - visualizer.py
- External Libraries
- Scratches and Consoles

spaceship.py

```python


import math


def euclidean_distance(p1, p2):
    x1, y1 = p1
    x2, y2 = p2

    return math.sqrt((x2 - y1)**2 + (y2 - y1)**2)


2 usages
class Spaceship():
    """The Spaceship to guide across the galaxy."""
```

Adding Tests

- Tests are usually written in separate files
  - Follow the naming convention file_we_want_to_test_**tests.py**
  - Convention is to either prefix/suffix with “tests”. Naming this way allows PyCharm to automatically find tests
- Every test has three parts
  - **Setup**: create the data your code will need
  - **Execute**: run your code with the setup data
  - **Assert**: verify your code’s return with the expected output

Sample Test

HopScotch: Kalman Filters – spaceship_tests.py

HopScotch: Kalman Filters 〉 spaceship_tests.py

Current File

Project

Commit

- HopScotch: Kalman Filters [Hc
  - cases
  - .gitignore
  - cs7638-HopScotch.pdf
  - game_objects.py
  - generate_test_case.py
  - runner.py
  - settings.py
  - spaceship.py
  - spaceship_tests.py
  - test.py
  - utilities.py
  - visualizer.py
- External Libraries
- Scratches and Consoles

spaceship.py | spaceship_tests.py

```python
from unittest import TestCase
from spaceship import euclidean_distance


class SpaceshipTests(TestCase):
    def test_euclidean_distance(self):
        # setup
        p1 = (0, 3)
        p2 = (4, 0)

        # execute
        actual = euclidean_distance(p1, p2)

        # assert
        expected = 5
        self.assertEqual(expected, actual)
```

Running Tests

- PyCharm should automatically detect and provide methods to run your test code
- From the test file, you should see the option to create a run or debug configuration

Running Tests

HopScotch: Kalman Filters – spaceship_tests.py

HopScotch: Kalman Filters 〉 spaceship_tests.py

Current File

Project

Commit

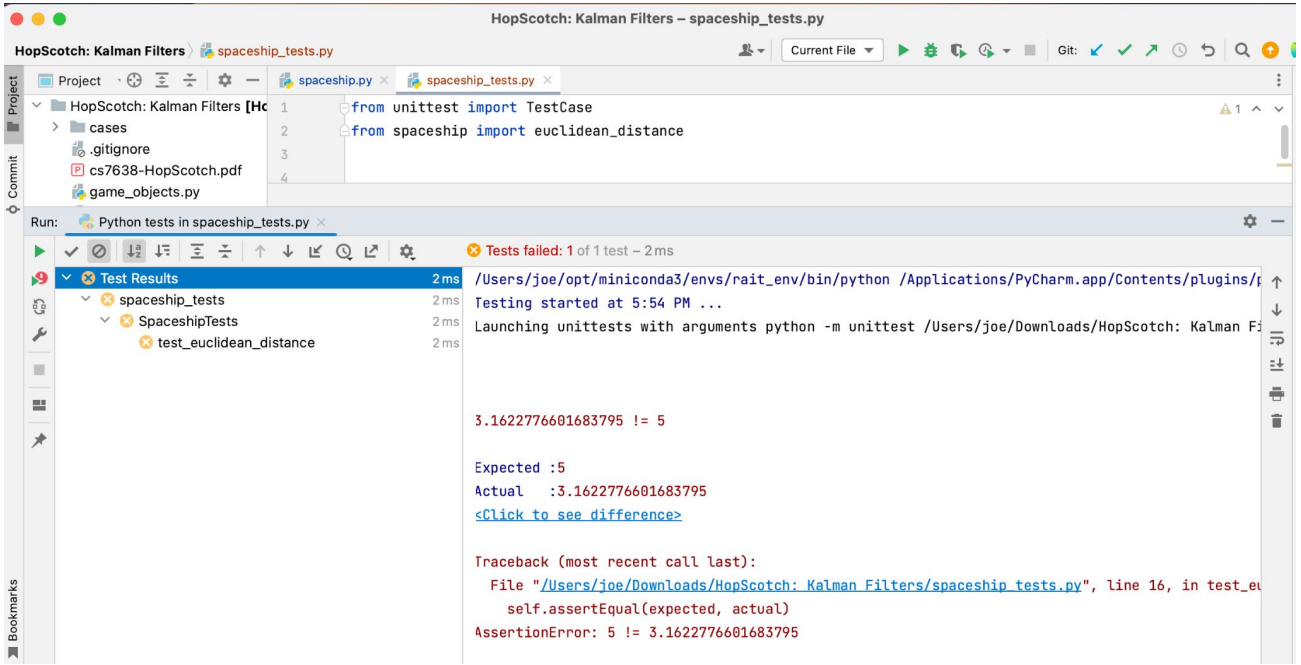
- HopScotch: Kalman Filters [Hc
  - cases
  - .gitignore
  - cs7638-HopScotch.pdf
  - game_objects.py
  - generate_test_case.py
  - runner.py
  - settings.py
  - spaceship.py
  - spaceship_tests.py
  - test.py
  - utilities.py
  - visualizer.py
- External Libraries
- Scratches and Consoles

spaceship.py | spaceship_tests.py

```python
from unittest import TestCase
from spaceship import euclidean_distance


class SpaceshipTests(TestCase):
    def test_euclidean_distance(self):
        # setup
        p1 = (0, 3)
        p2 = (4, 0)

        # execute
        actual = euclidean_distance(p1, p2)

        # assert
        expected = 5
        self.assertEqual(expected, actual)
```

Debugging

- Our first test failed - how can we figure out what went wrong?



Debugging

- Use the “Bug” icon in PyCharm to activate Debugging
- Use breakpoints to halt the execution
  - Without a breakpoint, the execution will continue to completion
- Use “Step Over” and “Step Into” to either advance over the current line
- Use the “Play” button to continue to the next breakpoint

Debugging Controls

HopScotch: Kalman Filters – spaceship_tests.py

HopScotch: Kalman Filters 〉 spaceship_tests.py

Current File

Git:

Project

HopScotch: Kalman Filters [Hc
- cases
- .gitignore
- cs7638-HopScotch.pdf
- game_objects.py
- generate_test_case.py
- runner.py
- settings.py
- spaceship.py
- spaceship_tests.py
- test.py
- utilities.py
- visualizer.py

External Libraries

Scratches and Consoles

spaceship.py | spaceship_tests.py

```
1  from unittest import TestCase
2  from spaceship import euclidean_distance
3
4
5  class SpaceshipTests(TestCase):
6      def test_euclidean_distance(self):   self: test_euclidean_distance (spaceship_tests.SpaceshipTests)
7          # setup
8          p1 = (0, 3)
9          p2 = (4, 0)
10
11         # execute
12         actual = euclidean_distance(p1, p2)
```

SpaceshipTests 〉 test_euclidean_distance()

Debug: Python tests in spaceship_tests.py

Debugger | Console

MainThread

- test_euclidean_distance, spaceship_tests.py:8
- _callTestMethod, case.py:549
- run, case.py:591
- __call__, case.py:650
- run, suite.py:122

Evaluate expression (⏎) or add a watch (⇧⌘⏎)

self = {SpaceshipTests} test_euclidean_distance (spaceship_tests.SpaceshipTests)

Project | Commit | Database | SciView | Notifications

Finding the Bug

HopScotch: Kalman Filters – spaceship.py

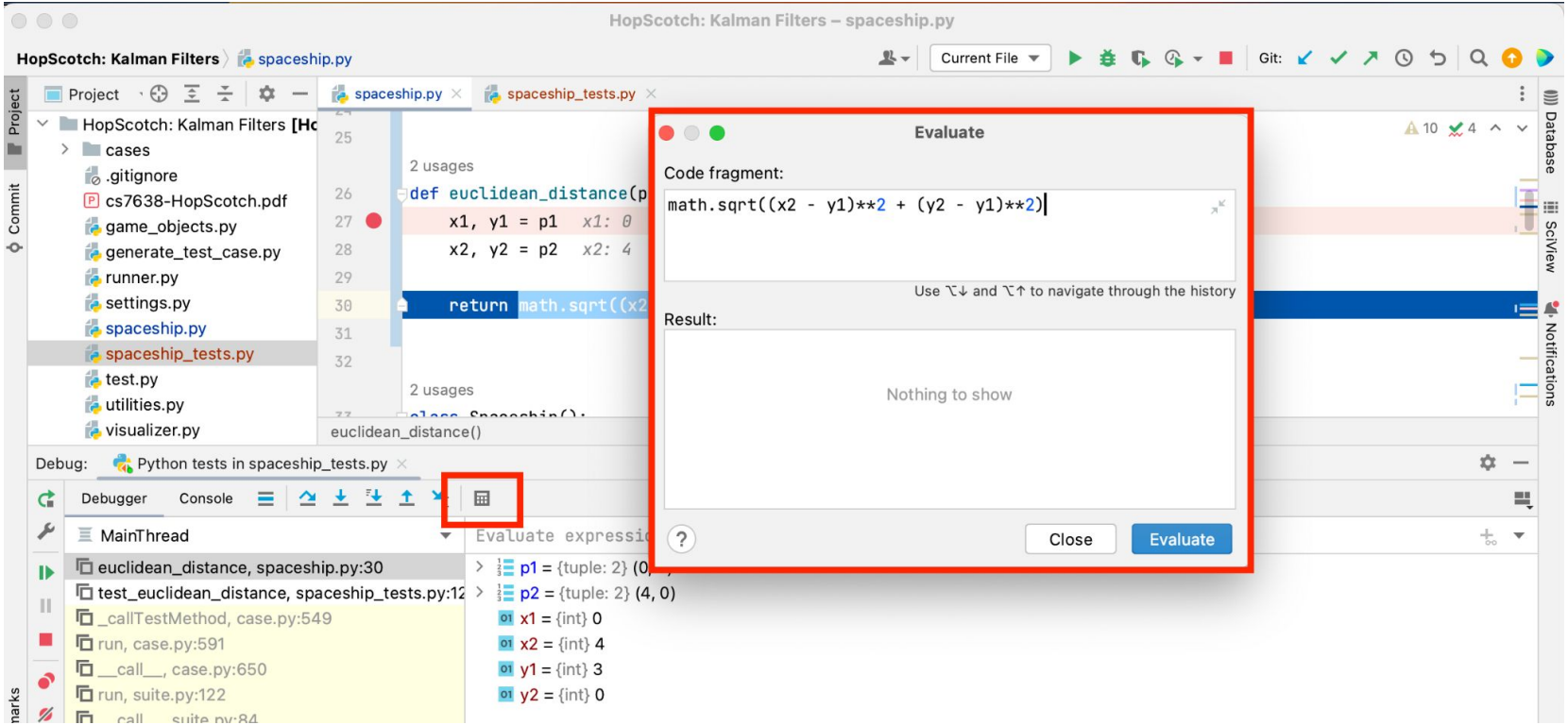
```
2 usages
def euclidean_distance(p1, p2):   p1: (0, 3)   p2: (4, 0)
    x1, y1 = p1   x1: 0   y1: 3
    x2, y2 = p2   x2: 4   y2: 0

    return math.sqrt((x2 - y1)**2 + (y2 - y1)**2)

2 usages
```

Debug: Python tests in spaceship_tests.py

Evaluate expression (⏎) or add a watch (⇧⌘⏎)

- p1 = {tuple: 2} (0, 3)
- p2 = {tuple: 2} (4, 0)
- x1 = {int} 0
- x2 = {int} 4
- y1 = {int} 3
- y2 = {int} 0

Finding the Bug



Fixing the Bug

HopScotch: Kalman Filters – spaceship.py

HopScotch: Kalman Filters 〉 spaceship.py

Current File

Project

HopScotch: Kalman Filters [Hc
- cases
- .gitignore
- cs7638-HopScotch.pdf
- game_objects.py
- generate_test_case.py
- runner.py
- settings.py
- spaceship.py
- spaceship_tests.py
- test.py
- utilities.py
- visualizer.py

External Libraries

Scratches and Consoles

spaceship.py | spaceship_tests.py

```
2 usages
def euclidean_distance(p1, p2):
    x1, y1 = p1
    x2, y2 = p2

    return math.sqrt((x2 - x1)**2 + (y2 - y1)**2)


2 usages
class Spaceship():
    """The Spaceship to guide across the galaxy."""

    def __init__(self, bounds):
```

Debug: Python tests in spaceship_tests.py

Debugger | Console

Tests passed: 1 of 1 test – 3 min 23 sec

Test Results 3 min 23 sec

```
/Users/joe/opt/miniconda3/envs/rait_env/bin/python /Applications/PyCharm.app/Contents/plugin
Testing started at 6:07 PM ...
```

Verifying with Tests

HopScotch: Kalman Filters – spaceship_tests.py

HopScotch: Kalman Filters 〉 spaceship_tests.py

Python tests for spaceship_tests.SpaceshipTests.test_euclidean_distance

Project

- HopScotch: Kalman Filters [Hc
  - cases
  - .gitignore
  - cs7638-HopScotch.pdf
  - game_objects.py
  - generate_test_case.py
  - runner.py
  - settings.py
  - spaceship.py
  - spaceship_tests.py
  - test.py
  - utilities.py
  - visualizer.py
- External Libraries
- Scratches and Consoles

spaceship.py | spaceship_tests.py

```python
from unittest import TestCase
from spaceship import euclidean_distance


class SpaceshipTests(TestCase):
    def test_euclidean_distance(self):
        # setup
        p1 = (0, 3)
        p2 = (4, 0)

        # execute
        actual = euclidean_distance(p1, p2)

        # assert
```

SpaceshipTests 〉 test_euclidean_distance()

Run: Python tests for spaceship_tests.SpaceshipTests.test_euclide...

Tests passed: 1 of 1 test – 0 ms

Test Results 0 ms

```
/Users/joe/opt/miniconda3/envs/rait_env/bin/pythor
Testing started at 6:12 PM ...
Launching unittests with arguments python -m unitt



Ran 1 test in 0.001s

OK
```

Key Things to Remember

- Small functions are easier to test/debug
- Unit tests allow you to keep your code easy to verify
- Debugging can still be difficult

Debugging Project Tests

- Prefer to debug a single test case rather than the whole suite
- Disable multiprocessing, if applicable

HopScotch: Kalman Filters – test.py

HopScotch: Kalman Filters 〉 test.py

Python tests for spaceship_tests.SpaceshipTests.test_euclidean_distance

Project

spaceship.py | test.py | spaceship_tests.py

HopScotch: Kalman Filters [Hc
- cases
- .gitignore
- cs7638-HopScotch.pdf
- game_objects.py
- generate_test_case.py
- runner.py
- settings.py
- spaceship.py
- spaceship_tests.py
- test.py
- utilities.py
- visualizer.py

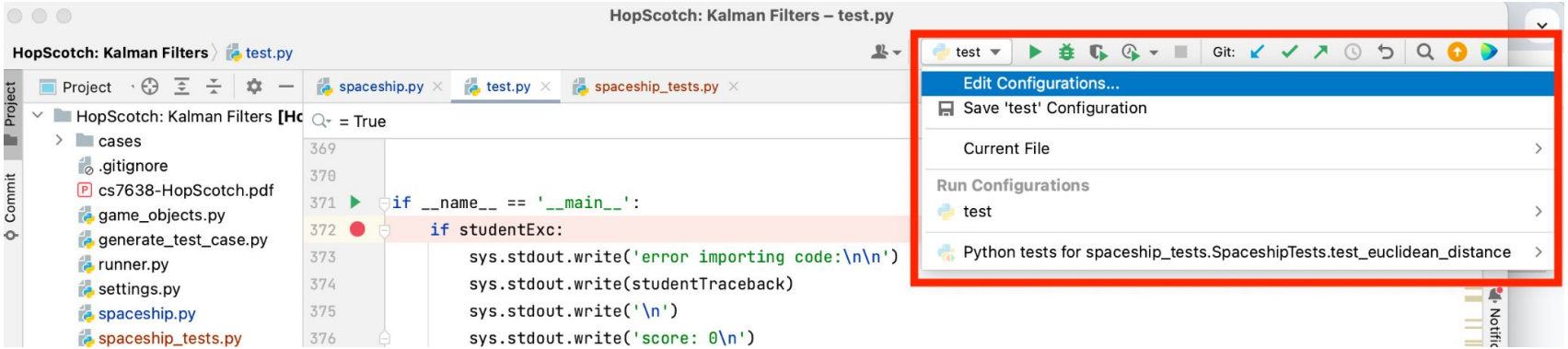
External Libraries

= True

```
226            fout_msg = '' if not display else display.fout.getvalu
227        self.result_queue.put((retcode, t, fout_msg + '\n' + msg))
228
229    multiprocessing_flag = False
230    # multiprocessing_flag = True
231
       1 usage
232    class MultiCaseGrader():
233
234        def __init__(self,
235                     fout,
```

Debugging Project Tests

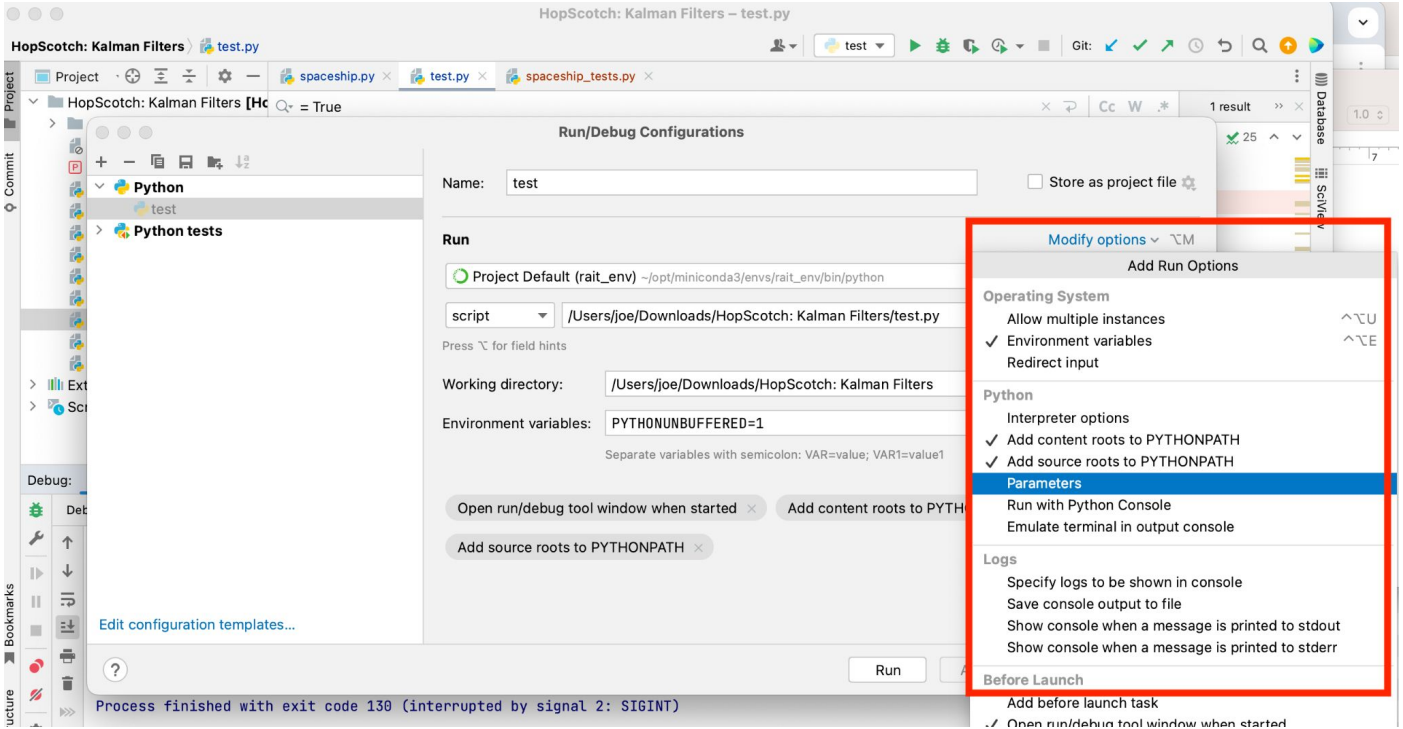
- Project entrypoint run from command line
- Have to pass these arguments to the debugger



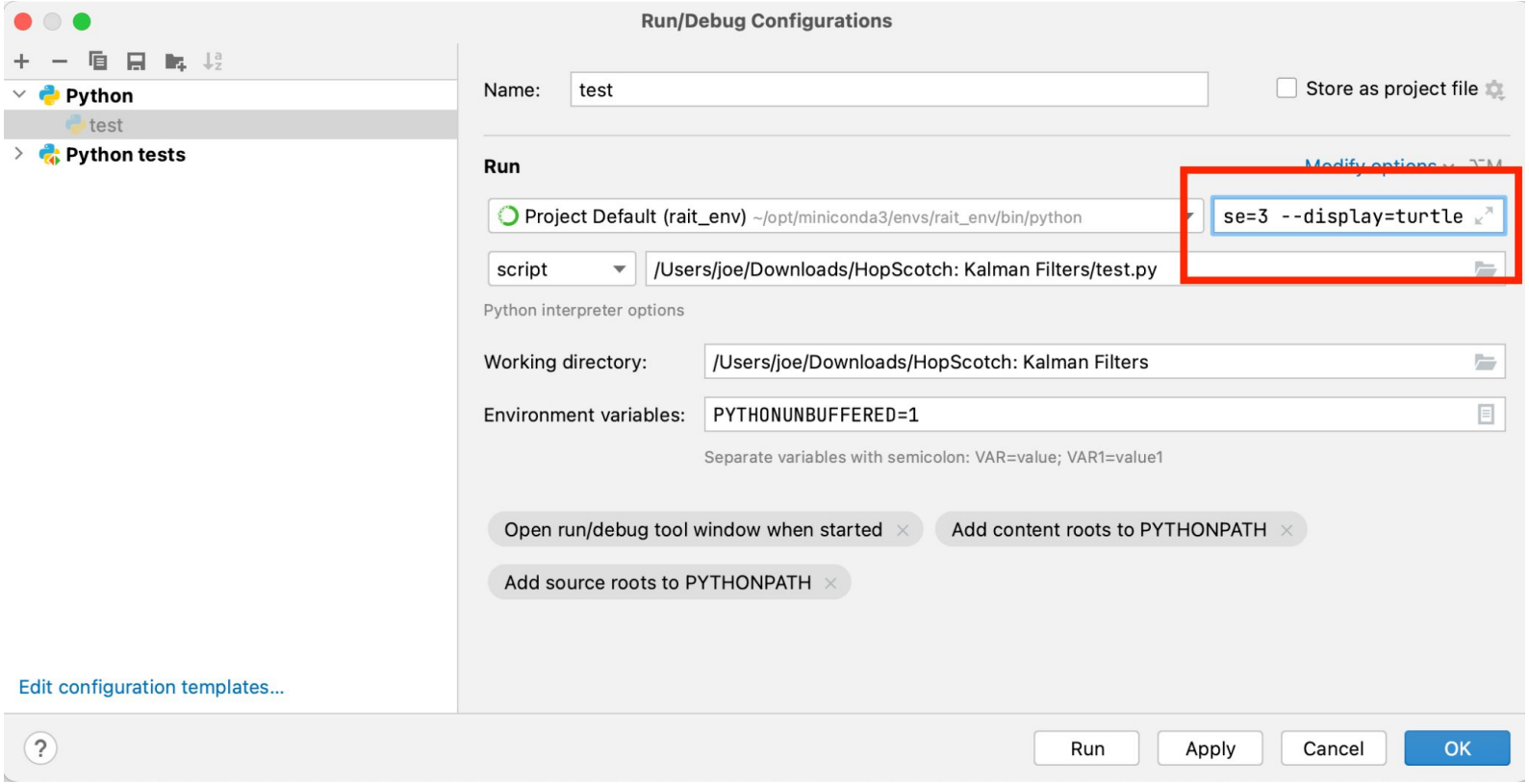
Debugging Project Tests

- Look at the entrypoint to the tests
- Figure out what options are parsed from the command line
  - E.g. KF **--method=estimate --case=3 --display=turtle**

Debugging Project Tests



Debugging Project Tests



Debugging Project Tests

- Set a breakpoint and verify that arguments set for the run configuration are passed/parsed as expected

Debugging Project Tests

HopScotch: Kalman Filters – test.py

HopScotch: Kalman Filters 〉 test.py

Project

HopScotch: Kalman Filters [Hc
- cases
- .gitignore
- cs7638-HopScotch.pdf
- game_objects.py
- generate_test_case.py
- runner.py
- settings.py
- spaceship.py
- spaceship_tests.py
- test.py
- utilities.py
- visualizer.py

External Libraries

Scratches and Consoles

spaceship.py | test.py | spaceship_tests.py

= True

1 result

```
382            mcg = MultiCaseGrader(sys.stdout)
383            mcg.run()
385            args = parser().parse_args()   args: Namespace(method='estimate', case='3', display='turtle')
386            if  args = {Namespace} Namespace(method='estimate', case='3', display='turtle')
                       case = {str} '3'
                       display = {str} 'turtle'
                       method = {str} 'estimate'
                       Protected Attributes
                                                                  t\'.')
                                                                  un the command\n')
                       Set value  F2
                        args.display = 'text'
394            main(method_name=args.method,
```

if __name__ == '__main__' 〉 else 〉 if student_id 〉 try 〉 else

Debug: test

Debugger | Console

MainThread

<module>, test.py:386

Evaluate expression (⏎) or add a watch (⇧⌘⏎)

- ESTIMATE_TIMEOUT = {int} 200
- FAILURE_EXCEPTION = {str} 'exception_raised'
- FAILURE_TIMEOUT = {str} 'execution_time_exceeded'
- JUMP_TIMEOUT = {int} 50
- TASKS = {tuple: 35} (GradingTask(case_num=1, method_name='estimate', weight=1, timeout=200), GradingTask(case_num=:... View
- args = {Namespace} Namespace(method='estimate', case='3', display='turtle')

Troubleshooting Project Debugging

- Delete the configuration, start over
- Can also modify our code
  - Comment out the command line parsing
  - Run our tests as Python unit tests