Unit Testing - how do I make sure my code is actually correct

Shankar Kulumani

Flight Dynamics & Control Lab

THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Testing your code

Testing your code is very important
Testing and running code in parallel

- Testing unit should focus on one tiny bit of functionality and prove it correct
- Each test should be fully independent
- Tests should run fast
- Run your tests before, during and after your development
- Write long, descriptive names for test functions

Testing your code

Testing your code is very important
Testing and running code in parallel

- Testing unit should focus on one tiny bit of functionality and prove it correct
- Each test should be fully independent
- Tests should run fast
- Run your tests before, during and after your development
- Write long, descriptive names for test functions

Testing framework

- Python has many frameworks available - handle all the complicated stuff
- `pytest` - very simple and easy to use

```
import numpy

def func(a, b):
    return a+b

def test_func():
    numpy.testing.assert_allclose(func(1, 2), 3)
```

Now just run `pytest` in terminal

Testing framework

- Python has many frameworks available - handle all the complicated stuff
- `pytest` - very simple and easy to use

```
import numpy

def func(a, b):
    return a+b

def test_func():
    numpy.testing.assert_allclose(func(1, 2), 3)
```

Now just run `pytest` in terminal

Guidelines

- Every function/branch requires atleast one test case
- Verify the tests using hand calculations/textbook examples
- Test while coding
- Test discovery
  - `pytest` will look for specific naming structure
  - Test should be in a module called `test_module.py`
  - Many tests can be organized into a directory called `tests`