Validating Your Code

Shankar Kulumani

Flight Dynamics & Control Lab

THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

What do I validate that a program is working?

- When your program runs without a syntax error, you might be so relieved, that you won't try and consider if the results are actually correct
- There must always be a method to provide confidence that the output is what you expect
- A wrong answer might be worse than no answer - Many, Many examples of software errors causing mission failures!

Determining test cases

- Come up with a test case that you have independently determined answer by another method
  - Hand calculations
  - Examples from a textbook (make sure it's actually correct first!)
  - Another well-trusted/validated program (its not about matching someone else's answer but actually the correct answer!)
- Compare the output of your program with the test case and make sure it matches
  - One good approach is to come up with all of the test cases first, before even writing your program.
  - Write your program to pass all of the test cases
- You will usually need many test cases for your code
  - Every branch/line of code should be evaluated through all of your test cases
  - For `if` statements you'll need a case for each branch

Unit Testing

- The process of testing small components of code is called `unit testing`
  - We test each `unit` independently
  - Easier to test small parts of code rather than something huge and complicated
  - After all units are tested we can then move onto building larger programs
- Testing should be fast and automatic
  - Writing software is a continual process - never ends
  - Testing can make sure that future modifications don't break old code
  - If something does break, we can easily find the error and correct it

Testing Example

- Now we'll practice unit testing in Python

```
# content of test_sample.py
import numpy as np

def inc(x):
    return x + 1

def test_answer():
    np.testing.assert_equal(inc(3), 5)
```