DSC 190

DATA STRUCTURES & ALGORITHMS

Lecture 13 | Part 1

Today's Lecture

Catch-Up Lecture

- This lecture is **optional**.

- But it might help you with your homework.

- A chance to catch up.

Testing Your Code

▶ Testing code is **essential** (for homework and real life).

▶ Consider it to be part of the problem.

▶ How do we test Python code?

Approach #1: Run it by hand

► Write your code.

► Open up a Python interpreter.

► Type in a few examples, see if code works.

► It doesn't work. Repeat.

Downsides

- ► You often run the same test over and over again.

- ► You have to type it in every time.

- ► This is **annoying**.

Main Idea

If something is annoying, you'll avoid doing it. Spend the time to make things less annoying.

Approach #2: Unit Testing Frameworks

► Create a file that only includes tests.

► Write test for each way that code will be used.
  ► Example: for a stack, write test for push, pop, peek.

► Try to anticipate "corner cases".

► Write the tests **before** you write the code.

Unit Testing in Python

▶ `unittest`: built-in module for unit testing

▶ `pytest`: nicer to use, more "modern"

```python
import stack
import pytest

def test_push_then_peek():
    s = stack.Stack(10)
    s.push(1)
    s.push(5)
    s.push(3)
    assert s.peek() == 3

def test_push_then_pop():
    s = stack.Stack(10)
    s.push(1)
    s.push(5)
    s.push(3)
    assert s.pop() == 3
```

Debugging

▶ Testing and debugging go hand-in-hand.

▶ Should know how to use the Python debugger.

Unit Testing Guidelines

▶ Should test "public" interface, not "private" implementation details.

▶ Should "exercise" all of the code (coverage).

▶ Write the tests before the code.