# Debugging and Testing

# Overview

- Tracebacks
  - tell us where an error occurred
- Assert statements
  - prevent unintentional user error
- Unit tests
  - make sure the code works the way you expect
  - prevent making changes which break something
  - easy to tell where code broke
- pdb
  - get into code during execution and find out what is going on at the exact moment of the error

# Traceback

- If python can't read what you wrote, it sends a trail of error messages which "trace" the error through your code.
- The traceback is in the order that the program runs in, with the first thing being at the top, and the exact line where the error occurred at the bottom.
- Start at the bottom and work your way backwards

# Assert Statements

- Require a statement to be true
- Format: assert x relation y, comment
  - e.g. assert x == y, 'error: x is not equal to y'

# Exercise: tracebacks and assert

Find the error in **traceback_example.py** from the traceback generated from typing

$python traceback_example.py 3

# Unit Testing

- Treat each function in your code as a "unit"
- Create at least one test for each unit

- Unit testing allows you to:
  - test that everything is working as expected
  - think of cases you may not have considered
  - immediately know if changes you made in one function affected another function

- You should balance number of tests with usefulness of tests. Each unit test should test something different about a function

# Exercise: unit testing

What test(s) would you create to make sure that **sum_list** is works correctly. *You don't need to write the test, just come up with test cases*.

# pdb - python debugger

- python module
- pdb.set_trace()
  - pauses execution of code at a single line and returns you to interactive mode
- Helpful if:
  - Your code produces an error before it produces an output
  - Your code produces something unexpected
  - Your traceback is unintelligable
  - You want to put in a bunch of print statements
- Lots more availble in the module. Explore when ready

# Exercise: pdb

Run the program pdb_example.py by typing *python pdb_example.py* followed by 4 positive numbers (without commas) into your command line.

$python pdb_example.py 3 4 5 6

Look at the code. Did it do what you expected? If not, use pdb.set_trace() to determine why not. If it worked the way you expected, the try writting some assertion statements to catch possible problem input cases.