Basic debugging tips

- If your functions aren't working as expected, use the following debugging techniques:
  - Print each of the inputs and their types to make sure your inputs are correct. Use the type() function to get a variables type.
  - Come up with test cases whose answers you can calculate manually and then check if your function outputs the same answers.
  - Make sure you are returning the output of your function. To debug loops, print the variables that are changing in your loop and do so by formatting them in well written print statements.
  - Use good variable names. Long variable names are any day better than short and uninformative ones.
  - When you google, make sure you are searching for Python 3 stuff, not Python 2!
  - At the end, Restart and Run all cells to make sure everything is working as expected.

printing things out...

- easiest way to debug, and super useful for simple applications
- e.g. here we want to square a list of numbers, but we make a mistake. Instead of for i in nums (which is what we want), we loop over the length of nums (for i in range(len(nums))). So instead of squaring the actual list of numbers, we are squaring 0,1,2...N, where N is the length of the list of numbers
- inserting a simple print statement to show us what 'i' is each time we go through the loop is super helpful...
  - allows us to see that i is 0,1 instead of 10,12...

In [ ]:
```
# write a function to square a variable length list of numbers
def sqr_list(*nums):
  out = []
  for i in range(len(nums)):
    print("i is:", i)
    out.append(i**2)

  return out
```

In [ ]:
```
sqr_list(10,12)
```

use the %pdb "line magic" [more on line magics next week, but they basically add functionality in the ipython notebook environment]

- good for when you're getting an error and not sure why
- %pdb 1 will turn the debugger on
- %pdb 0 will turn the debugger off
- will stop execution at the error and enter "debug" mode
  - this will launch a 'command line', and at the command line you can type in variable names and do other operations to test your code
- type 'q' at the command line to exit debugging mode

In [ ]:
```
%pdb 1
```

In [ ]:
```
# write a function to square a variable length list of numbers
def sqr_list(*nums):
  out = []
  for i in range(len(nums)):
    # name error!
    outl.append(i**2)

  return out
```

In [ ]:
```
sqr_list(10,12)
```

One built in debugger using: pdb.set_trace()

- will set a breakpoint at a fixed location
- when breakpoint reached, you will enter debug mode and a command line will pop up similar to the one you get with %pdb 1 line magic

In [ ]:
```
import pdb
```

In [ ]:
```
# write a function to square a variable length list of numbers
def sqr_list(*nums):
  out = []
  for i in range(len(nums)):
    pdb.set_trace()
    out.append(i**2)

  return out
```

In [ ]:
```
sqr_list(10,20)
```