Debugging Techniques
Exam 2 coming up, review: http://cs1110.cs.virginia.edu/know.html

POTD 11 (http://www.cs.virginia.edu/~up3f/cs1110/practice-of-the-day/practice_11.txt):
String

A	B	C	D	B	C	D	E	F	G

Substring (ss)

B	C	D

Can see the two times it occurs, and remove the second, as a human
Two "pointers" **i** and **j**
Check if the first two are the same

```
    if string[i] == ss[j]
```

If so, check for the whole substring (from **i** to the end of the substring)

```
        if string[i:len(ss) + 1] == ss:
```

If so, is this the first time or second? Use variable **count** initialized to zero
Only remove something if count is 1

```
            if count == 1:
                return string[:i] + string[i+1:]
```

Whether or not something was removed, increment count and move forward past this substring

```
            count += 1
            i += len(ss)
            count += 1
            i += len(ss)
```

If not, go back to the top of the loop incrementing **i** by 1

```
    elif string[i] != ss[j]
        i += 1
```

Run this loop **while** there is still space left in string

```
    while i <= len(string) ***at the top
```

Solution:

```python
def removeSecond(l, n):
   string = l
   substring = n
   count = 0

   for i in range(0, len(string)):
       if string[i:(i + len(substring))] == substring:
           count += 1
           if count == 2 :
               string = string[0:i] + string[i + len(substring): ]
   return string
```

Debugging by writing tests that include corner cases:
***a group of test cases is referred to as a “test suite”
Example 7 from http://www.cs.virginia.edu/~up3f/cs1110/examples/testing/:
Test case:

```
# best way to test it is to throw in a bunch of weird inputs
actualResults.append(count_pairs([4, 6, 4, -9, -1, 9, 9, 45, 22, 2, 5, (2**2)]))
# should recognize pairs (4, 6), (6, 4), (2, (2**2)), and ((2**2), 2)
expectedResults.append(4)
# because it doesn't, we know there is a bug
```

Adding print statements to the code to see where it went wrong:

```
def count_pairs(lst):
   count = 0
   print("001:", lst)
   pairs = []
   for i in lst:
       print("002: i =", i)
       for j in lst:
           print("003: j =", j)
           if (i + j == 10):
               count += 1
               print("004: (", i, j, ") is a pair")
               pairs.append("(" + str(i) + str(j) + ")")
   print(pairs)
   return count
# on our first test case, returns pairs as ['(46)', '(64)', '(64)', '(64)', '(46)',
'(55)', '(46)']
# clear that the problem is it's counting the same pair twice
# clear that it shouldn't have recognize (55) since there's only one 5
# now we know how to fix it
```

Official solution online along with more problems and their solutions at link above