Funky Loop Stuff

Here are two Python statements that you can use inside loops. You can get by without these, but sometimes they might make your code easier. They work with both for-loops and while-loops

- **break** This causes you to immediately leave the innermost loop that you are currently executing.
- **continue** This causes you to go back to the top of the current loop. With a for-loop the loop variable is automatically incremented.

For example consider the following code:

```
for i in range(1, 3):
        for j in range(1, 4):
            if j == 2:
                break
            print( "(%d, %d)"%(i, j) )
```

This prints

(1,1)

(2, 1)

because we leave the inner loop each time j becomes 2.

```
for i in range(1, 3):
        for j in range(1, 4):
                if j == 2:
                        continue
                print( "(%d, %d)"%(i, j) )
```

This prints

(1, 1)
(1, 3)
(2, 1)
(2, 3)

because only the j===2 steps are skipped

On the other hand, the following

```
while True:
    x = input( "blah: " )
    if x == "":
        break
    elif x == "bob":
        continue
    print(x)
```

will read and echo lines from the user. If a line is just "bob" it won't be echoed. The loop continues until the user enters an empty string.

You can use break and continue statements anywhere you find them convenient. The most common uses are to allow you to exit early from a for-loop. For example, the following two blocks of code that test whether variable x is prime are equivalent:

```
isPrime = True
for d in range(2, x):
        if x%d== 0:
                isPrime = False
                break
```

```
isPrime = True
d = 2
while d < x and isPrime:
        if x%d==0:
                isPrime = False
        d = d+1
```