For Loops in Python

Iteration
For loops allow us to iterate over each element of a set or sequence

Syntax:

```
for <var> in <set>:
do . . .
do . . .
```

Example

```
words = ['a', 'rose', 'is', 'a', 'rose', 'is', 'a', 'rose']
for w in words:
    print w
```

The Range Function

Iteration
Python has a built-in function, range, which allows us to generate a list containing the numbers specified in the range. What do these do?

```
range (0, 10)
range (10)
range (1, 11, 2)
range (0, -10, -1)
```

Range in Loops

```
for number in range(0, 10):
    print number
```

```
a = 'congratulations'
for number in range(len(a)):
    print number, a[number]
```

Breaking out of Loops

- `break`: stops the execution of the loop, independent of the test
  ```
  for x in range(0,1000):
      if (x % 7) == 0:
          print 'first number divisible by 7:', x
          break
  ```

- `continue`: skip the rest of the loop body, but continue with the loop
  ```
  for x in range(0,1000):
      if (x % 7) == 0:
          print x
      else:
          continue
          print str(x) "is divisible by 7"
Loop Else

Iteration
else can follow a loop, then it is executed in case the loop is NOT exited by break.

```python
for x in range(100):
    if x > 101:
        print(x)
        break
    else:
        print('not found')
```