Strings can do operations on themselves
`.lower()`, `.upper()`,`.capitalize()`

```
In [1]: "funKY tOwn".capitalize()

Out[1]: 'Funky town'

In [2]: "funky tOwn".lower()

Out[2]: 'funky town'
```

`.split([sep [,maxsplit]])`

```
In [3]: "funKY tOwn".split()

Out[3]: ['funKY', 'tOwn']

In [4]: "funKY tOwn".capitalize().split()

Out[4]: ['Funky', 'town']

In [5]: [x.capitalize() for x in "funKY tOwn".split()]

Out[5]: ['Funky', 'Town']

In [6]: "I want to take you to, funKY tOwn".split("u")

Out[6]: ['I want to take yo', ' to, f', 'nKY tOwn']

In [7]: "I want to take you to, funKY tOwn".split("you")

Out[7]: ['I want to take ', ' to, funKY tOwn']
```

`.strip()`, `.join()`, `.replace()`

```
In [8]: csv_string = 'Dog,Cat,Spam,Defenestrate,1, 3.1415   \n\t'
        csv_string.strip()

Out[8]: 'Dog,Cat,Spam,Defenestrate,1, 3.1415'

In [9]: clean_list = [x.strip() for x in csv_string.split(",")]
        print clean_list

        ['Dog', 'Cat', 'Spam', 'Defenestrate', '1', '3.1415']
```

`.join()` allows you to glue a list of strings together with a certain string

```
In [10]: print ",".join(clean_list)

         Dog,Cat,Spam,Defenestrate,1,3.1415

In [11]: print "\t".join(clean_list)

         Dog     Cat     Spam    Defenestrate    1       3.1415
```

`.replace()` strings in strings

In [12]:
```
csv_string = 'Dog,Cat,Spam,Defenestrate,1, 3.1415   \n\t'
alt_csv = csv_string.strip().replace(' ','')
print alt_csv
```

```
Dog,Cat,Spam,Defenestrate,1,3.1415
```

In [13]:
```
print csv_string.strip().replace(' ','').replace(',','\t')
```

```
Dog     Cat     Spam    Defenestrate    1       3.1415
```

`.find()`

incredibly useful searching, returning the index of the search

In [14]:
```
s = 'My Funny Valentine'
s.find("y")
```

Out[14]:
```
1
```

In [15]:
```
s.find("y",2)
```

Out[15]:
```
7
```

In [16]:
```
s[s.find("Funny"):]
```

Out[16]:
```
'Funny Valentine'
```

In [17]:
```
s.find("z")
```

Out[17]:
```
-1
```

In [18]:
```
ss = [s,"Argentine","American","Quarentine"]
for thestring in ss:
   if thestring.find("tine") != -1:
      print "'" + str(thestring) + "' contains 'tine'."
```

```
'My Funny Valentine' contains 'tine'.
'Argentine' contains 'tine'.
'Quarentine' contains 'tine'.
```

`string` module
exposes useful variables and functions

In [19]:
```
import string
string.swapcase("fUNKY tOWN")
```

Out[19]:
```
'Funky Town'
```

In [21]:
```
print string.ascii_letters
print string.digits
```

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
```

[back]

String Formatting

the (new) preferred way

is `string.format(value0,value1,....)`

```
In [22]: 'on {0}, I feel {1}'.format("saturday","groovy")

Out[22]: 'on saturday, I feel groovy'

In [23]: 'on {}, I feel {}'.format("saturday","groovy")

Out[23]: 'on saturday, I feel groovy'

In [24]: 'on {0}, I feel {1}'.format(["saturday","groovy"])

         ---------------------------------------------------------------------------
         IndexError                                Traceback (most recent call last)
         /Users/jbloom/Classes/python-
         bootcamp/DataFiles_and_Notebooks/06_AdvancedStrings/<ipython-input-24-37beb7743cdb>
         in <module>()
         ----> 1 'on {0}, I feel {1}'.format(["saturday","groovy"])

         IndexError: tuple index out of range


In [25]: 'on {0}, I feel {0}'.format(["saturday","groovy"])

Out[25]: "on ['saturday', 'groovy'], I feel ['saturday', 'groovy']"

In [26]: 'on {0}, I feel {0}'.format("saturday","groovy")

Out[26]: 'on saturday, I feel saturday'
```

you can assign by argument position or by name

```
In [28]: '{desire} to {place}'.format(desire='Fly me',\
                                      place='The Moon')

Out[28]: 'Fly me to The Moon'

In [29]: '{desire} to {place} or else I wont visit {place}.'.format( \
                           desire='Fly me',place='The Moon')

Out[29]: 'Fly me to The Moon or else I wont visit The Moon.'

In [30]: f = {"desire": "I want to take you", "place": "funky town"}

In [31]: '{desire} to {place}'.format(**f)

Out[31]: 'I want to take you to funky town'

 In [ ]:
```