map()

map() is a function that takes in two arguments: a function and a sequence iterable. In the form: map(function, sequence)

The first argument is the name of a function and the second a sequence (e.g. a list). map() applies the function to all the elements of the sequence. It returns a new list with the elements changed by function.

When we went over list comprehension we created a small expression to convert Fahrenheit to Celsius. Let's do the same here but use map.

We'll start with two functions:

In [4]:
```
def fahrenheit(T):
    return ((float(9)/5)*T + 32)
def celsius(T):
    return (float(5)/9)*(T-32)

temp = [0, 22.5, 40,100]
```

Now lets see map() in action:

In [7]:
```
F_temps = map(fahrenheit, temp)

#Show
F_temps
```

Out[7]:
```
[32.0, 72.5, 104.0, 212.0]
```

In [8]:
```
# Convert back
map(celsius, F_temps)
```

Out[8]:
```
[0.0, 22.5, 40.0, 100.0]
```

In the example above we haven't used a lambda expression. By using lambda, we wouldn't have had to define and name the functions fahrenheit() and celsius().

In [10]:
```
map(lambda x: (5.0/9)*(x - 32), F_temps)
```

Out[10]:
```
[0.0, 22.5, 40.0, 100.0]
```

Great! We got the same result! Using map is much more commonly used with lambda expressions since the entire purpose of map() is to save effort on having to create manual for loops.

map() can be applied to more than one iterable. The iterables have to have the same length.

For instance, if we are working with two lists-map() will apply its lambda function to the elements of the argument lists, i.e. it first applies to the elements with the 0th index, then to the elements with the 1st index until the n-th index is reached.

For example lets map a lambda expression to two lists:

In [12]:
```
a = [1,2,3,4]
b = [5,6,7,8]
c = [9,10,11,12]

map(lambda x,y:x+y,a,b)
```

Out[12]:
```
[6, 8, 10, 12]
```

In [13]:
```
# Now all three lists
map(lambda x,y,z:x+y+z, a,b,c)
```

Out[13]:
```
[15, 18, 21, 24]
```

We can see in the example above that the parameter x gets its values from the list a, while y gets its values from b and z from list c. Go ahead and play with your own example to make sure you fully understand mapping to more than one iterable.

Great job! You should now have a basic understanding of the map() function.