List Comprehensions

In addition to sequence operations and list methods, Python includes a more advanced operation called a list comprehension.

List comprehensions allow us to build out lists using a different notation. You can think of it as essentially a one line `for` loop built inside of brackets. For a simple example:

Example 1

In [1]:

```
# Grab every letter in string
lst = [x for x in 'word']
```

In [2]:

```
# Check
lst
```

Out[2]:

```
['w', 'o', 'r', 'd']
```

This is the basic idea of a list comprehension. If you're familiar with mathematical notation this format should feel familiar for example: x^2 : x in { 0,1,2...10 }

Let's see a few more examples of list comprehensions in Python:

Example 2

In [3]:

```
# Square numbers in range and turn into list
lst = [x**2 for x in range(0,11)]
```

In [4]:

```
lst
```

Out[4]:

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Example 3

Let's see how to add in `if` statements:

In [5]:

```python
# Check for even numbers in a range
lst = [x for x in range(11) if x % 2 == 0]
```

In [6]:

```python
lst
```

Out[6]:

```
[0, 2, 4, 6, 8, 10]
```

Example 4

Can also do more complicated arithmetic:

In [7]:

```python
# Convert Celsius to Fahrenheit
celsius = [0,10,20.1,34.5]

fahrenheit = [((9/5)*temp + 32) for temp in celsius ]

fahrenheit
```

Out[7]:

```
[32.0, 50.0, 68.18, 94.1]
```

Example 5

We can also perform nested list comprehensions, for example:

In [8]:

```python
lst = [ x**2 for x in [x**2 for x in range(11)]]
lst
```

Out[8]:

```
[0, 1, 16, 81, 256, 625, 1296, 2401, 4096, 6561, 10000]
```

Later on in the course we will learn about generator comprehensions. After this lecture you should feel comfortable reading and writing basic list comprehensions.