More on Lambda

Take this function:

```
def Double(n):
    return 2*n
```

You can call this in a few ways:

```
Double(3)       =>  6
Double(1 + 2)   =>  6

x = 3
Double(x)       =>  6
```

The first thing that happens when you call a function and give it a parameter is that the parameter is evaluated by the interpreter, before the function is called. E.g. 1 + 2 is calculated in the second line before the function is called, or x is replaced by 3 in the fourth line.

Note that this also happens with the first call, for uniformity. Python checks what the value of 3 is before it calls the function, because it'd be more awkward to not check whenever the value is a number than it would be to always check.

Evaluation

This process of figuring out the value of things is known as **evaluation.** It's the process of taking any expression and simplifying it down to a basic value.

Relevance to Lambda

Here's an equivalent of Double() written using lambda:

```
lambda n : 2*n
```

If you wanted to give this a name, you could do this:

```
Double = lambda n : 2*n
```

Functionally, it now looks very similar to the Double() function above.

When testing the Map() function in Assignment 19, you could write either of the following lines:

```
Map(Double, [1, 2, 3])              =>  [2, 4, 6]

Map(lambda n : 2*n, [1, 2, 3])      =>  [2, 4, 6]
```

With the first call, the interpreter evaluates Double, replacing it with the lambda function represented by Double, *even if you defined Double using a def*.