1 Lambda functions (one line functions), and map/filter/reduce

Lambda functions are a Pythonic way of writing simple functions in a single line.

They may be applied to a list using the *map* statement.

The function *filter* offers a concise way to filter out all the elements of a list based on the results of a function.

The function *reduce* continually applies a function to a sequence until a single value is returned.

1.1 Defining one-line lambda functions

Instead of defining the function with def, we define the function with *lambda*.

Here is a 'normal' function:

```
# A named function that returns the power of a number:

def f(x, y):
    result = x ** y
    return result

print (f(4,3))

OUT:

64
```

Re-writing as a lambda function:

```
g = lambda x,y: x**y

print (g(4,3))

OUT:

64
```

1.2 Applying a lambda function to a list with map

In the example below we define a lambda function to convert Celsius to Fahrenheit, and then apply it to a list.

We will first define lambda separately, and then map, before looking at how they can be combined a single line.

The *map* command generates a 'map object'. To return a list we need to convert that to a list as below

```
celsius = [0, 10, 20, 30]
g = lambda x: (9/5)*x + 32
fahrenheit = list(map(g, celsius))

print (fahrenheit)

OUT:

[32.0, 50.0, 68.0, 86.0]
```

This may be compressed further into one line. But there is always a balance to be had between being as concise as possible, and being as clear for others as possible. It may, however, generally be assumed that other experienced Python coders will be familiar with a combined one-line lambda/map function.

```
celsius = [0, 10, 20, 30]
fahrenheit = list(map(lambda x: (9/5)*x + 32, celsius))

print (fahrenheit)

[32.0, 50.0, 68.0, 86.0]
```

1.3 Filtering a list with a lambda function

The function *filter* offers a concise way to filter out all the elements of a list, for which the function function returns True. The function filter(f,l) needs a function f as its first argument. f returns a Boolean value, i.e. either True or False.

This function will be applied to every element of the list l. Only if f returns True will the element of the list be included in the result list.

Here is an example where filter is used to return even numbers from a list. As with the map example above we will combine the filter and lambda function in a single line.

```
my_list = [0,1,2,3,4,5,6,7,8,9,10]
result = list(filter(lambda x: x % 2 == 0, my_list))

print (result)

OUT:

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

1.4 Reducing a list with a repeated lambda function call

The reduce function will continually apply a lambda function until only a single value is returned. Though it used to be in core python, it is now found in the functools module.

A simple example is to use reduce to multiply all values in a list. The reduce function replaces the first two elements of a list with the product of those two elements. It will continue until one value is left.

```
import functools
my_list = [10,30,34,56,89]
result = functools.reduce(lambda x,y: x*y, my_list)

print (result)

OUT:

50836800
```

1.5 Alternatives to map, filter and reduce

Most results from map/filter and reduce may also be obtained from using list comprehensions, as previously outlined. Some see list comprehensions as more Pythonic than map, filter and reduce.