4. Map , Filter and Reduce

These are three functions which facilitate a functional approach to programming. We will discuss them one by one and understand their use cases.

4.1. Map

`Map` applies a function to all the items in an input_list. Here is the blueprint:

Blueprint

```
map (function_to_apply, list_of_inputs)
```

Most of the times we want to pass all the list elements to a function one-by-one and then collect the output. For instance:

```
items = [1, 2, 3, 4, 5]
squared = []
for i in items:
    squared.append(i**2)
```

`Map` allows us to implement this in a much simpler and nicer way. Here you go:

```
items = [1, 2, 3, 4, 5]
squared = list( map (lambda x: x**2, items))
```

Most of the times we use lambdas with `map` so I did the same. Instead of a list of inputs we can even have a list of functions!

```
def multiply(x):
    return (x*x)
def add(x):
    return (x+x)

funcs = [multiply, add]
for i in range(5):
    value = list( map (lambda x: x(i), funcs))
    print(value)

# Output:
# [0, 0]
# [1, 2]
# [4, 4]
# [9, 6]
# [16, 8]
```

4.2. Filter

As the name suggests, `filter` creates a list of elements for which a function returns true. Here is a short and concise example:

```
number_list = range(-5, 5)
less_than_zero = list(filter(lambda x: x < 0, number_list))
print(less_than_zero)

# Output: [-5, -4, -3, -2, -1]
```

The filter resembles a for loop but it is a builtin function and faster.

Note: If **map** & filter do not appear beautiful to you then you can read about `list/dict/tuple` comprehensions.

4.3. Reduce

`Reduce` is a really useful function for performing some computation on a list and returning the result. It applies a rolling computation to sequential pairs of values in a list. For example, if you wanted to compute the product of a list of integers.

So the normal way you might go about doing this task in python is using a basic for loop:

```
product = 1
list = [1, 2, 3, 4]
for num in list:
    product = product * num

# product = 24
```

Now let's try it with reduce:

```
from functools import reduce
product = reduce((lambda x, y: x * y), [1, 2, 3, 4])

# Output: 24
```