Loops

Sometime you will need to loop over some code many times. This makes performing the same task on data much easier to program.

```
In [ ]:
#count from 1 to 5
for i in range(1,6):
    print(i)
```

```
1
2
3
4
5
```

In the above example we count from 1 to 5 (it stop at 6 and does not run). The 'i' variable is short for itterator, and it will count up. We then print this value.

Times tables

We will print out a times table for 1 to 5

```
In [ ]:
table = 8
for i in range(1,6):
    print(str(i) + " x " + str(table) + " = " + str(i * table))
```

```
1 x 8 = 8
2 x 8 = 16
3 x 8 = 24
4 x 8 = 32
5 x 8 = 40
```

We calculate the answer with i * table, so if i = 2 and the table is 8, the answer will be 16. We use str() to conver a number into a string so that we can join it with the other parts.

We can also loop over data in a list

We can add all the values in a list

```
In [ ]:
data = [2.1, 4.3, 5.1, 2.4,8.5]
total = 0                              #Keep track of the lotal

for i in range(0,len(data)):
    total = total + data[i]

print(total)
```

```
22.4
```

Counting down in a loop

Be carful doing this, you could crash your programme or computer in some languages

```
In [ ]: data = ["a","b","c","d"]

        for i in range(len(data)-1,-1,-1):
            print(data[i])
```

```
d
c
b
a
```

In the above code, we need to start at the end len(data)-1 because if there are 4 elements, the last index will be 3. We need to count to -1, as otherwise if we count to 0 it will not print the 0 index letter. Lastly the 3rd number -1 is the step, we are counting in -1, ie going down 1. You can count in -4, or +6 or any other step.