Small Basic

“For” loops
The simplest kind of repetition is a counted loop – otherwise known as a ‘for’ loop because of the syntax:

```
For n = 1 To 10
    TextWindow.WriteLine("this is loop iteration no "+ n)
EndFor
```

The counter is n, it starts with a value of 1 and ends when the value is 10. This kind of loop is used when something has to be repeated a given number of times.

Things to note:

- In this example, n is the ‘loop control variable’. You can use this variable inside the loop but you MUST NOT change it, for example – this next loop has a problem (what will happen?)

```
For n = 1 To 10
    n=5
EndFor
```

- You can use any increment by adding ‘step’ and the number you need to step by:

```
For n = 1 To 10 Step 5
    TextWindow.WriteLine(n)
EndFor
```

- You can count down by using step -1 (the start value must be higher than the end value):

```
For n = 10 To 1 Step -1
    TextWindow.WriteLine("this is loop iteration no "+ n)
EndFor
```

- Any one of the numbers can be a variable:

```
start = 10
finish = 200
For n = start To finish
    TextWindow.WriteLine( n)
Endfor
```

Exercise 1

1. Write separate programs that display
   - the numbers 1 to 10
   - the even numbers between 0 and 100 (hint – use step 2)
   - all the square numbers between 1 and 100 (HINT: 100 is the square of 10)
   - the seven times table – the output should be something like:

```
  2 x 7 is 14
  3 x 7 is 21        ...etc
```
Adding up numbers
Computers can add numbers quickly and accurately. For example, the following program will display the sum of 345 and 576:

```
number1 = 345
number2 = 567
total = number1 + number2
TextWindow.WriteLine(total)
```

You could also do this in just one line of code. Compare the following code with the previous code sample – which is better? (Why?)

```
TextWindow.WriteLine(345 + 567)
```

Using a ‘for’ loop to add numbers
What if you had to keep adding another number? Using the first code sample, we would have to add the next number to the total we already have.

```
total = total + number3
```

How does this work? Remember that the computer carries out instructions from right to left.

- The computer FIRST adds the variables `total` and `number3`
- It then stores the value (the sum) in the variable ‘total’.

You can use this line of code inside a ‘for’ loop to add a very long list of numbers, for example:

```
total = 0
for n = 1 To 50
    total = total + n
endfor
'display the total
```

Exercise 2
1. Use a ‘for’ loop to add the even numbers between 0 and 100. Display the total only.

2. Move the output statement (the ‘writeline’) from the previous code, so that the total is displayed after each number is added.

3. Use a ‘for’ loop to calculate the amount you would have if you invested $1000 for 5 years at 5% (where the interest is paid every year). (Hint – every year you need to:
   - Calculate the interest on the amount
   - Add that interest to the amount

4. **Find out how much you would earn if you were paid $1 on the first day, $2 on the second, $4 on the third, etc, for 10 days. Would this be better than being paid $100 per day?**

The ‘jargon’
The line of code `total = total + number` is sometimes called the ‘accumulator’. You can use a similar line of code to ‘increment’ (add one to) a variable, as in:

```
    n = n + 1
```

Both these statements are commonly used in all languages. In fact, they are so common that some languages shorten the statement to

```
    n++  (equivalent to n = n+1)
```

and

```
    total+=number  (equivalent to total = total +number)
```