# Lesson 3 - Iteration

|  |  |
| --- | --- |
| **Lesson Outcomes**  In this lesson you will learn:   * how to repeat parts of the same program; * to use fixed and conditional loops; * to mix and match IF commands with looping statements. | **C:\Users\Graham\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\X6CHINOH\MC900441498[1].png** |

## Introduction to looping – conditional loops

In programming one of the most powerful concepts is **Iteration;** that is to repeat several functions or statements within a program a fixed or conditional amount of times. You can represent iteration in a simple program flow chart:

Count =1

Count = Count +1

Does Count = 10?

No

Yes

Output Count

The above program sets the variable **Count** to 1, adds one and loops until Count is equal to 10. Notice how the Count variable is used to keep a track of the number of loops.

We can write this program in Python using the following code:



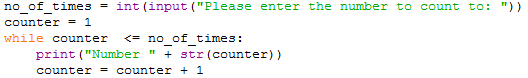
**Condition** e.g. keep looping while count is less than 10.

Add one to count variable. Notice it is indented as part of the while statement

In the above program a while loop is used. The while loop will repeat all the indented block of code underneath until a **CONDITION** is met, in this case count < 10. This is called a **CONDTIONAL** loop i.e. the loop will continue until a specified condition is met.

Let’s look at another more complex example:

|  |
| --- |
| **1** |
| **2** |
| **3** |
| **4** |
| **5** |



The first line no\_of\_times = int(input(“Please enter the number to count to”)), takes a text input from the user converts it to a number using the int command and assigns the value to the variable no\_of\_times.

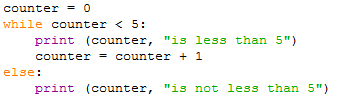
**Line 2,** the variable counter is assigned to one.

**Line 3** sets up a while loop which will continue until counter is **less than or equal** to the variable no\_of\_times.

**Line 4** will print out the “Number” and add to it (concatenate) the contents of counter (using the **str** command to convert a number to TEXT)

**Line 5** Increment the counter by one.

You can also use a while loop with an **else** statement which will execute if the WHILE condition hasn’t been met. An example:



### Tasks

3.1 Write a program that displays the word “hello” on the screen 4 times using the while loop.

3.2 Write a program that prompts the user to enter a short message and the number of times it is to be displayed and then displays the message the required number of times.

3.3 Write a program to display the squares of all the integers from 1 to 12.

3.4 Write a program that reads in a series of numbers and adds them up until the user enters zero.

3.5 Write a program that asks the user for a number between 10 and 30 inclusive and will validate, that is test, the input. It should repeatedly ask the user for this number until the input is within the valid range.

**HINT** you will need to keep looping until a valid number is entered.

3.6 Expand your program from Exercise 3.4 to display the average as well as the sum of the numbers entered. Make sure you do not count the “0” value as entry.

3.7 Write a program that displays a conversion table for pounds to kilograms, ranging from 1 pound to 20 pounds (1 kg = 2.2 pounds).

3.8 Write a program that asks the user to enter 8 integers and displays the largest.

## Fixed Loops – iterating by sequence

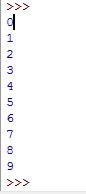
In the previous exercises you looked at **CONDITIONAL LOOPS** i.e. keep looping while a condition is true**.** The alternative to **conditional** loops is **fixed or unconditional loops.** Unconditional loops will iterate through a set number of times based on either a range or sequence of values. In Python you can create fixed loops in a variety of different ways. The most basic is looping through a sequence of values:

Let’s look at the following example:



The for command tells Python to create a **FIXED UNCONDITIONAL LOOP**. The range command tells Python to go through a sequence of numbers starting from the default 0 to the specified number (notice the program ends at number 9)

Running this program you should get the output:



Alternatively you can specify the start and end range:



The basic makeup of the range function is:

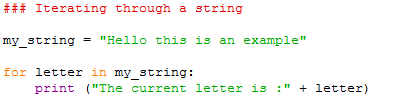
**range**([*start*], *stop*[, *step*])

Start is the staring value, stop is then end, and step is the incremental values which is 1 by default.

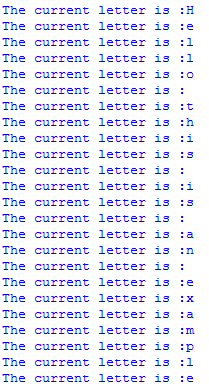
Try this:



You can also use loop on lists and strings. For example, you may wish to iterate through a string in the following manner:



In the above program a loop is created to go through each character in the string. You should get the output:



Looping through lists is more or less the same. We will cover lists later on the course.

### Tasks

3.9 Write a program that asks the user to enter a number. The program should output all the numbers starting from the user number to 0 e.g. count down from the user number entered.

3.10 Write a program to display an “n times table” for a given integer n. For n = 4, the output should be:

1 \* 4 = 4

2 \* 4 = 8

…

12 \* 4 = 48

3.11 Write a program that asks for a number, and displays the squares of all the integers between 1 and this number inclusive.

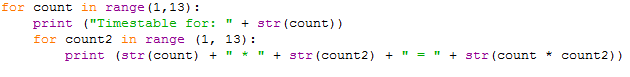
3.12 n factorial, usually written n!, is defined to be the product (multiplication) of all the integers in the range 1 to n:

N! = 1 \* 2 \* 3 \* 4 …………..\* n

Write a program that calculates n! for a given positive n.

## Loops within in loops

In programming it is often useful to have loops within loops, that is repeating a section of code x number of times by y number of times e.g.:



The above program will output all the times tables from one to twelve. A loop has been created at the beginning from one to 12, and then within that another loop from one to 12. Notice the indentation is used so Python knows which blocks of code are associated with which loops.

### Tasks

3.13 Write a program that asks the user to enter the number of stars per row and the number of rows to be displayed. For example, entering 5 and 3 should display:

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

**HINT:** to print one character next to another without a new line is print(“\*”, end=””)

### Extension

3.14 Write a program to display all the prime numbers up to a user entered number. Remember a prime number is a number which can be divided evenly only by 1 or itself.