Table of Contents

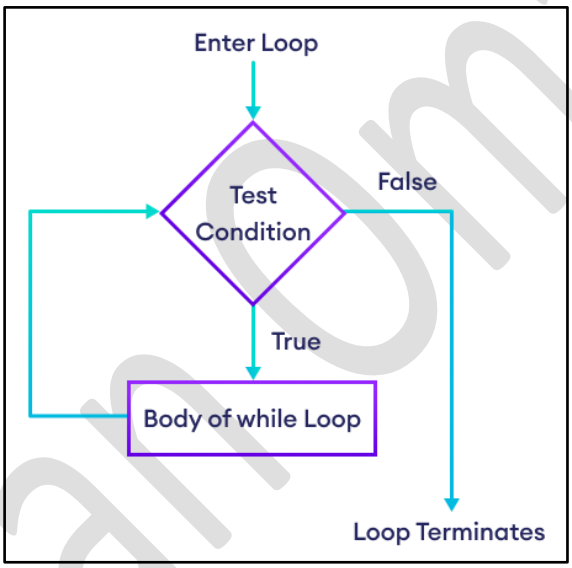
Chapter 9

Loops

Loops are important in Python or in any other programming language as they help you to execute a block of code repeatedly. You will often come face to face with situations where you would need to use a piece of code over and over but you don't want to write the same line of code multiple times.

Python has two primitive loop commands:
- while loops
- for loops

The while Loop



With the while loop we can execute a set of statements as long as a condition is true.

Ex.

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Remember to increment i, or else the loop will continue forever.
The while loop requires relevant variables to be ready, in this example we need to define an indexing variable, i, which we set to 1.

The break Statement

With the break statement we can stop the loop even if the while condition is true:

Exit the loop when i is 3:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

The continue Statement

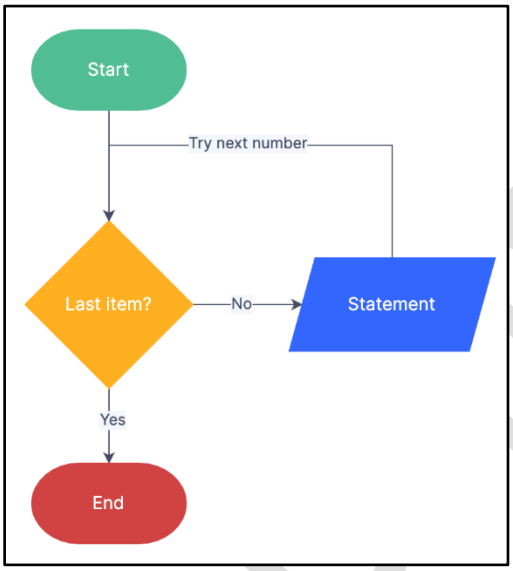
With the continue statement we can stop the current iteration, and continue with the next:

Example
Continue to the next iteration if i is 3:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

The For loops:

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.
With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.




```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
print(x)
# The for loop does not require an indexing variable to set beforehand.

for i in "Madi":
   print(i)
# M a d i
```

The range() Function

To loop through a set of code a specified number of times, we can use the range() function,
The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

```
for x in range(6):
  print(x)
# range(6) is not the values of 0 to 6, but the values 0 to 5.
```

The range() function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: range(2, 30, 3):

Increment the sequence with 3 (default is 1):

```
for x in range(2, 30, 3):
  print(x)
# 2 5 8 11 14 17 20 23 26 29
```