DECISION CONTROL STATEMENTS

It is mandatory to properly indent the statements that are dependent on the previous statement .

SELECTION/CONDITIONAL BRANCHING STATEMENTS

These decision control statements usually jumps from one part of the code to another part of code depending on whether a particular condition is satisfied or not.

NOTE: Python does not support switch case

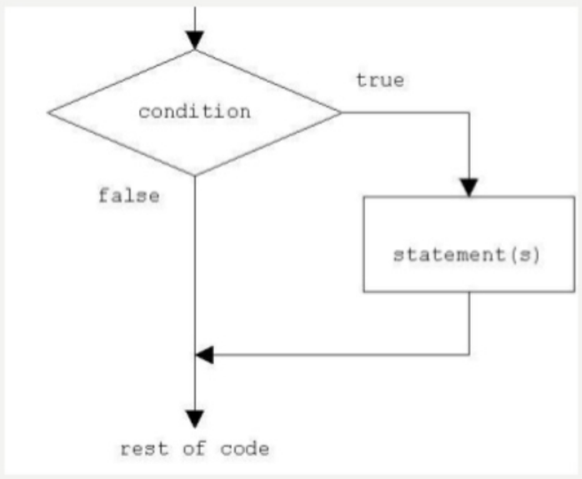
IF STATEMENT

A **header** in Python is a specific keyword followed by a colon.The group of statmenets following a header is called a **suite.** Header and its suite are together known as **clause.**

SYNTAX

```
if expression:
      statement(s)
         or
if (expression):
     statement(s)
```

FLOWCHART

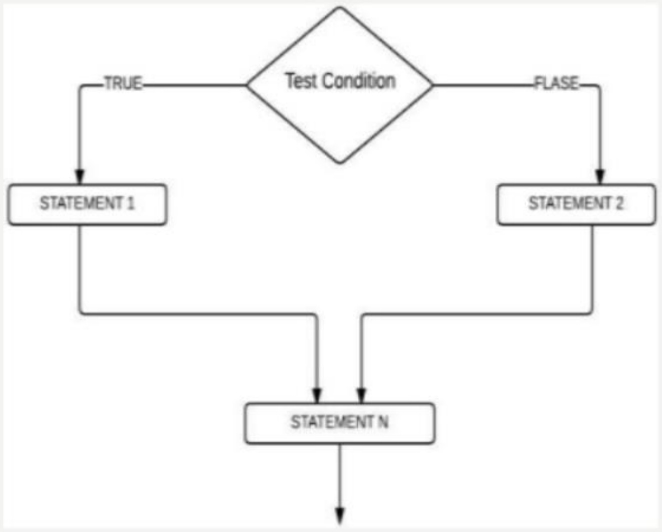


IF-ELSE

SYNTAX

```
if expression:
    statement(s)
else :
    statement(s)
```

FLOWCHART



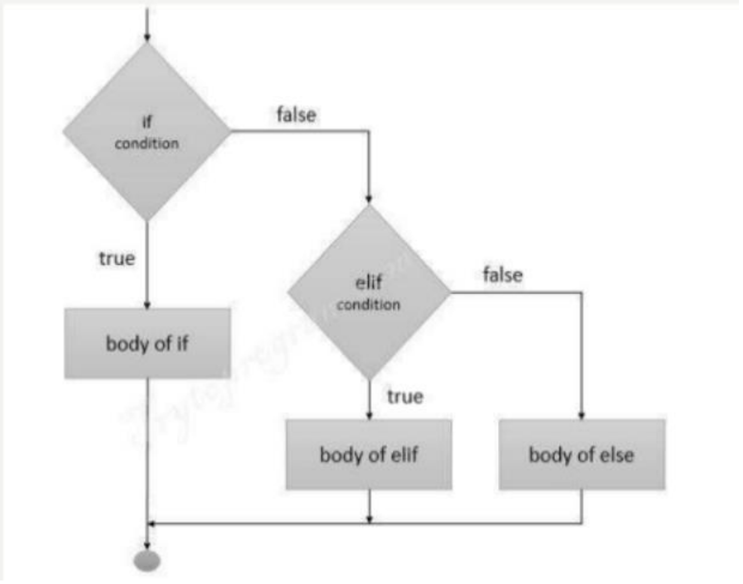
If and else statements are used to determine which option in a series of possibilities is True.

IF-ELIF-ELSE

SYNTAX

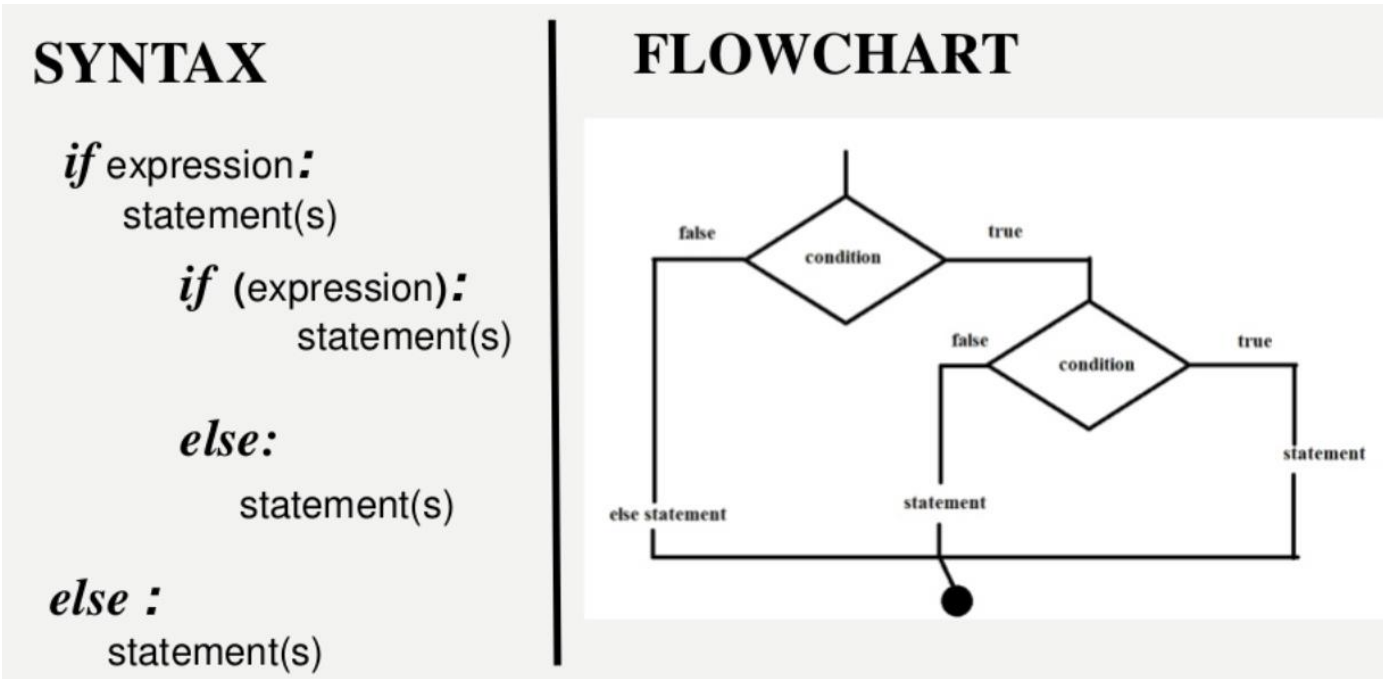
```
if expression:
    statement(s)

elif (expression):
    statement(s)

else :
    statement(s)
```

FLOWCHART



You can have multiple elif statements in within the if-else clause

NESTED IF

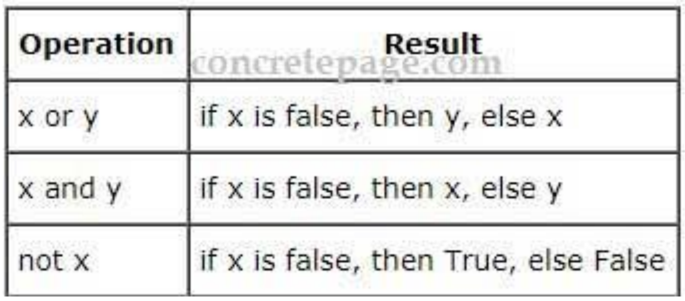


SYNTAX

```
if expression:
    statement(s)
        if (expression):
            statement(s)

        else:
            statement(s)

else :
    statement(s)
```

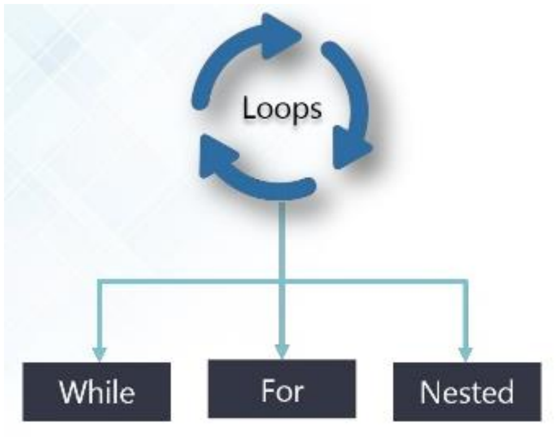
LOGICAL OPERATORS

Symbols links &&, ||, !, are not allowed in conditional statements, instead keywords like and, or, not are used



Operation	Result
x or y	if x is false, then y, else x
x and y	if x is false, then x, else y
not x	if x is false, then True, else False

LOOPS STRUCURES/ ITERATIVE STATMENETS



BREAK STATEMENT

With the break statement we can stop the loop before it has looped through all the items:

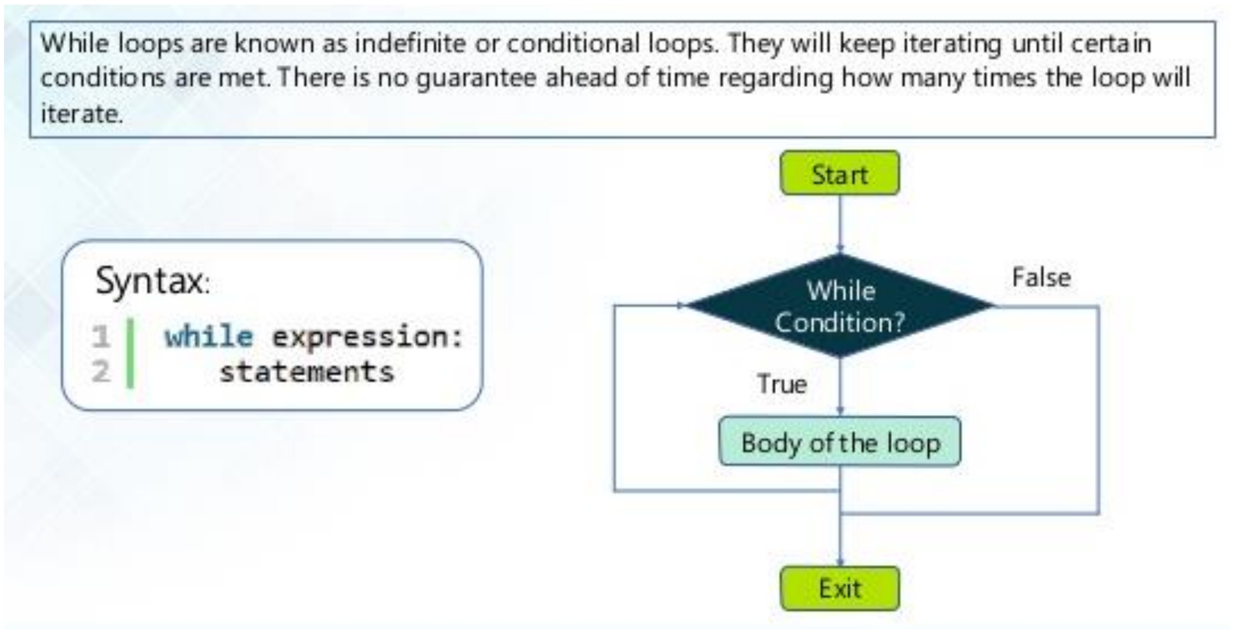
CONTINUE STATEMENT

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

PASS STATEMENT

The pass statement is used when a statement is required syntactically but no command or code has to be executed. It specifies a null operation or No Operation (NOP) statement. Nothing happens when the pass statement is executed.

WHILE LOOP



While loops are known as indefinite or conditional loops. They will keep iterating until certain conditions are met. There is no guarantee ahead of time regarding how many times the loop will iterate.

Syntax:

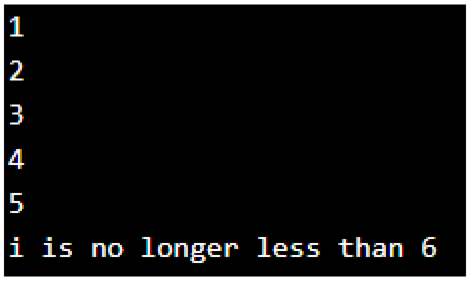
```
while expression:
    statements
```

A while loop is also referred to as a top-checking loop since control condition is placed as the first line of the code.

ELSE IN WHILE LOOP

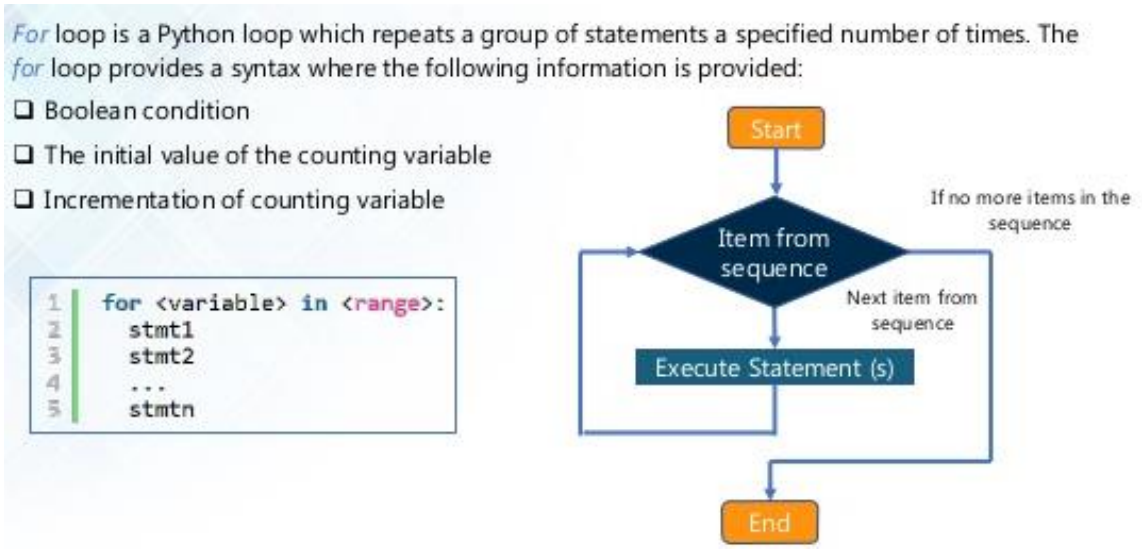
If the else statement is used with the while loop, the else statement is executed when the condition becomes False.

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```



```
1
2
3
4
5
i is no longer less than 6
```

FOR LOOP



For loop is a Python loop which repeats a group of statements a specified number of times. The *for* loop provides a syntax where the following information is provided:

- Boolean condition
- The initial value of the counting variable
- Incrementation of counting variable

```
for <variable> in <range>:
  stmt1
  stmt2
  ...
  stmtn
```

The For loop is usually known as a determinate or definite loop because the programmer knows exactly how many times the loop will repeat. The for loop does not require an indexing variable to set beforehand.

The for … in statement is a looping statement used in Python to iterate over a **sequence** of objects i.e., go through each item in a sequence.

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)
```

LOOPING THROUGH A STRING

Even strings are iterable objects, they contain a sequence of characters:

Loop through the letters in the word "banana":

```
for x in "banana":
  print(x)
```

RANGE FUNCTION

The range () is used to iterate over a sequence if numbers. The syntax of range( ) is

range(beg, end, [step])

The range () produces a sequence of numbers starting with bed (inclusive) and ending with end (exclusive). The step argument is optional and by default every number in the range is incremented by 1 but we cn specify a different increment using step. It can be both positive, negative but not zero.

```
for x in range(2, 30, 3):
  print(x)
```

KEY POINT TO REMEMBER:

- If range function is given a single argument, it produces an object with values ranging from 0 to argument-1.
  range(6) is equal to writing range(0,6).

ELSE IN FOR LOOP

The else keyword in a for loop specifies a block of code to be executed when the loop is finished:

Print all numbers from 0 to 5, and print a message when the loop has ended:

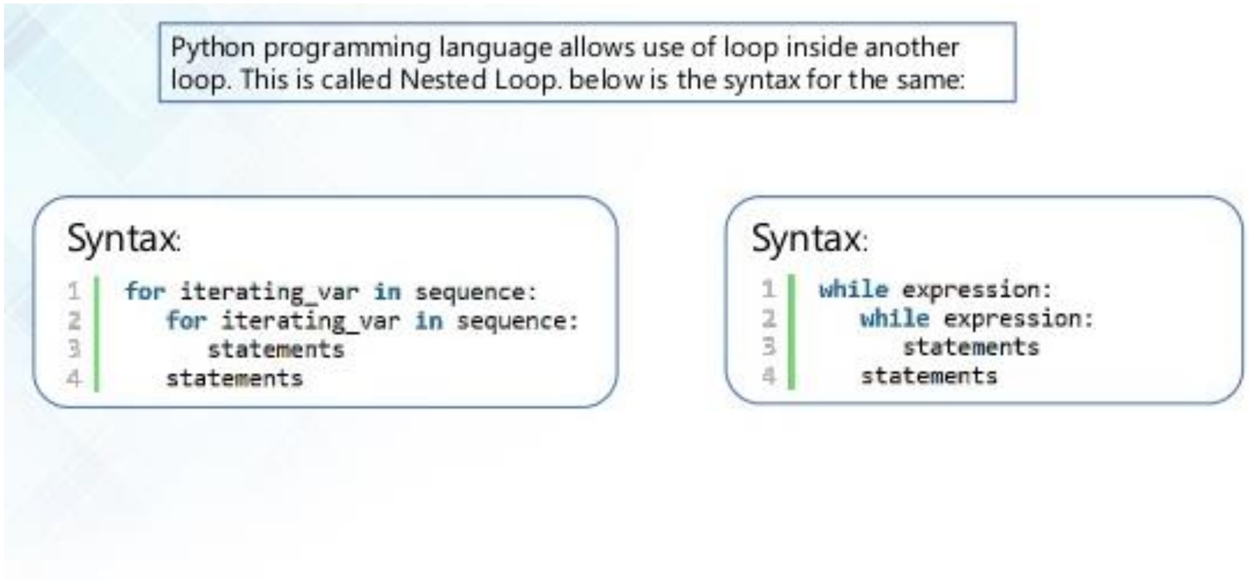
```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

PASS IN FOR LOOP

for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error.

```
for x in [0, 1, 2]:
  pass
```

NESTED LOOPS



Python programming language allows use of loop inside another loop. This is called Nested Loop. below is the syntax for the same:

Syntax:

```
for iterating_var in sequence:
   for iterating_var in sequence:
      statements
   statements
```

Syntax:

```
while expression:
   while expression:
      statements
   statements
```