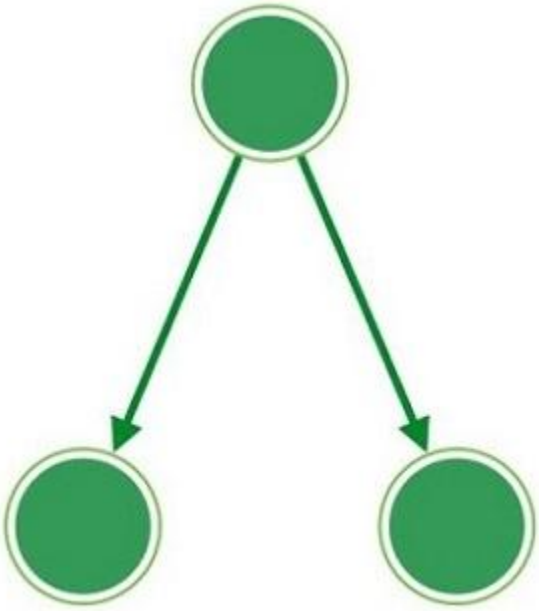
Python program control flow is regulated by various types of conditional statements, loops, and function calls. By default, the instructions in a computer program are executed in a sequential manner, from top to bottom, or from start to end. However, such sequentially executing programs can perform only simplistic tasks. We would like the program to have a decision-making ability, so that it performs different steps depending on different conditions.

Most programming languages including Python provide functionality to control the flow of execution of instructions. Normally, there are two type of control flow statements in any programming language and Python also supports them.

## Decision Making Statements

Decision making statements are used in the Python programs to make them able to decide which of the alternative group of instructions to be executed, depending on value of a certain Boolean expression.

The following diagram illustrates how decision-making statements work –



## The if Statements

Python provides if..elif..else control statements as a part of decision marking. It consists of three different blocks, which are **`if`** block, **`elif`** (short of else if) block and **`else`** block.

## Example

Following is a simple example which makes use of if..elif..else. You can try to run this program using different marks and verify the result.

Open Compiler

```
marks = 80
result = ""
if marks < 30:
   result = "Failed"
elif marks > 75:
   result = "Passed with distinction"
else:
   result = "Passed"

print(result)
```

This will produce following result:

```
Passed with distinction
```

## The match Statement

Python supports Match-Case statement, which can also be used as a part of decision making. If a pattern matches the expression, the code under that case will execute.

## Example

Following is a simple example which makes use of match statement.

Open Compiler

```
def checkVowel(n):
   match n:
      case 'a': return "Vowel alphabet"
      case 'e': return "Vowel alphabet"
      case 'i': return "Vowel alphabet"
```

```
        case 'o': return "Vowel alphabet"
        case 'u': return "Vowel alphabet"
        case _: return "Simple alphabet"
print (checkVowel('a'))
print (checkVowel('m'))
print (checkVowel('o'))
```

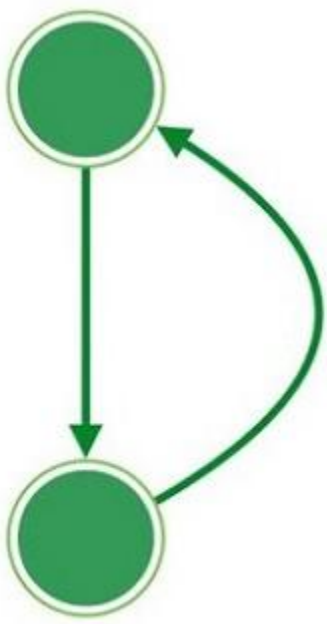
This will produce following result:

```
Vowel alphabet
Simple alphabet
Vowel alphabet
```

## Loops or Iteration Statements

Most of the processes require a group of instructions to be repeatedly executed. In programming terminology, it is called a **loop**. Instead of the next step, if the flow is redirected towards any earlier step, it constitutes a loop.

The following diagram illustrates how the looping works –



If the control goes back unconditionally, it forms an infinite loop which is not desired as the rest of the code would never get executed.

In a conditional loop, the repeated iteration of block of statements goes on till a certain condition is met. Python supports a number of loops like for loop, while loop which we will study in next chapters.

## The for Loop

The for loop iterates over the items of any sequence, such as a list, tuple or a string .

## Example

Following is an example which makes use of For Loop to iterate through an array in Python:

Open Compiler

```
words = ["one", "two", "three"]
for x in words:
  print(x)
```

This will produce following result:

```
one
two
three
```

## The while Loop

The while loop repeatedly executes a target statement as long as a given boolean expression is true.

## Example

Following is an example which makes use of While Loop to print first 5 numbers in Python:

Open Compiler

```
i = 1
while i < 6:
```

```
    print(i)
    i += 1
```

This will produce following result:

```
1
2
3
4
5
```

## Jump Statements

The jump statements are used to jump on a specific statement by breaking the current flow of the program. In Python, there are two jump statements break and continue.

### The break Statement

It terminates the current loop and resumes execution at the next statement.

### Example

The following example demonstrates the use of break statement –

Open Compiler

```
x = 0

while x < 10:
    print("x:", x)
    if x == 5:
        print("Breaking...")
        break
    x += 1

print("End")
```

This will produce following result:

```
x: 0
x: 1
x: 2
x: 3
x: 4
x: 5
Breaking...
End
```

## The continue Statement

It skips the execution of the program block and returns the control to the beginning of the current loop to start the next iteration.

## Example

The following example demonstrates the use of continue statement −

Open Compiler

```
for letter in "Python":
    # continue when letter is 'h'
    if letter == "h":
        continue
    print("Current Letter :", letter)
```

This will produce following result:

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : o
Current Letter : n
```