

# More On Python

## Learning Outcomes

- Types of Operators in Python
- Assignment Operator
- Rational Operator
- Logical Operator
- Operator Preferences
- Algorithm
- Flowchart
- Conditional Statements in Python
- Types of Control Structures
- If,if...else, and if...elif...else

In the previous lesson, you have learnt about introduction to Python and Variables. Now you will learn how to use operators to perform arithmetic and logical operations in programming.

## Introduction

An **operator** is a symbol that will perform mathematical operations on variables or on values. Operators operate on **operands** (values) and **return** a result.

An operator needs one and more operands to perform any operation. The valid combination of both operands and operators makes an expression that returns a computed result.

## Types Of Operators In Python

Operators can be different types based on the kind of operation they perform:

### Arithmetic Operators

The **arithmetic operators** are used to perform basic mathematical calculations i.e. Addition (+), Subtraction (-), Multiplication (\*), Division (/), etc. Let's take an example: 2+3

Here, + is an operator for addition. It adds 2 and 3 and prints 5 in the interpreter.

This is an arithmetic operator.

The arithmetic operator can also be classified as:

 **Unary Operator**

 **Binary Operator**

**Unary Operator:** These operators operate on only one operand e.g.,  $x = +200$  (the value assigned to 'x' is +200),  $y = -40$  (the value assigned to 'y' is -40).

### Program 1

```
>>> x=+200      # +Unary
>>> y=-40       # -Unary
>>> print(x)
200
>>> print(y)
-40
>>> print(y)
-40
...
>>>
>>>
>>>
```

**Binary Operators:** These operators operate on two operands, e.g.,  $a + b$ . Here '+' is a binary operator working on 'a' and 'b'. Binary operators can further be classified as:

Operator	Symbol	Usage	Application
Addition	+	To obtain the sum of the values	Value of $13+3.5$ is 16.5
Subtraction	-	To subtract the values	Value of $45-25$ is 20
Multiplication	*	To find the product of the data	Values of $6.2*6$ is 37.2
Division	/	To divide the numbers and give an output in decimal form	Value of $5/2$ is 2.5 Value of $-5/2$ is -2.5 Value of $10.0/3$ is 3.333
Floor Division	//	To divide the numbers and give an output in the integer form	Value of $5/2$ is 2 Value of $-5/2$ is -3
Remainder	%	To find the remainder when one value is divided by other.	Value of $3\%2$ is 1 Value of $10\%6$ is 4 Value of $6\%10$ is 6
Exponential	**	To calculate power of numbers	Value of $2**3$ is 8

## Program 2

```

sample.py - C:/Users/user/AppData/Local/Programs/Python/Python312/sa
File Edit Format Run Options Window Help
x=int(input("Enter a value: "))
y=int(input("Enter a value: "))
print("Addition of x & y = ", x+y)
print("Subtraction of x & y = ", x-y)
print("Multiplication of x & y = ", x*y)
print("Division of x & y = ", x/y)
print("Floor Division Result of x 7 Y is = ", x//y)
print("Square of y = ", y**2)
>>>

=== RESTART: C:/Users/user/AppData/Loca
Enter a value: 40
Enter a value: 10
Addition of x & y = 50
Subtraction of x & y = 30
Multiplication of x & y = 400
Division of x & y = 4.0
Floor Division Result of x 7 Y is = 4
Square of y = 100

```

**String Operators:** String operators represent the different types of operations that can be employed on the program's string type of variables. Python allows several string operators that can be applied on the python string. Few of them are mentioned below:

1. Assignment operator: "="
2. Concatenate operator: "+"
3. String repetition operator: "\*."
4. String slicing operator: "[ ]"
5. String comparison operator: "==" & "!="
6. Membership operator: "in" & "not in"
7. Escape sequence operator: "\"
8. String formatting operator: "%" & "{}"

1. **Assignment Operator “=”:** A string can be assigned to any variable with an assignment operator “=”. Python string can be defined with either single quotes [‘ ’], double quotes[“ ”] or triple quotes[“””]. `var_name = “string”` assigns “string” to variable `var_name`

#### Program 3

```
File Edit Format Run Options Win
string1 = "hello"
string2 = "Varanasi"
string3 = "Bharat"
print(string1)
print(string2)
print(string3)
```

```
=== RESTART: C:/Users/
hello
Varanasi
Bharat
.>>>
```

2. **Concatenate Operator “+”:** Two strings can be concatenated or join using the “+” operator in python, as explained in the below example code:

#### Program 4

```
File Edit Format Run Options Window Help
string1 = "hello"
string2 = "Varanasi"
string3 = "Bharat"

string_combined = string1 + string2

print(string_combined)
```

```
>>>
=== RESTART: C:
helloVaranasi
>>>
```

3. **String Repetition Operator “\*”:** The same string can be repeated in python by n times using `string*n`, as explained in the below example.

#### Program 5

```
File Edit Format Run Options Wind
string1 = "hello"
string2 = (string1*2)
string3 = (string1*3)

print(string1)
print(string2)
print(string3)
```

```
=== RESTART: C:/Users/use
hello
hellohello
hellohellohello
>>>
```

**4. String slicing operators “[ ]”:** Characters from a specific index of the string can be accessed with string [index] operator. The index is interpreted as a positive index starting from 0 from the left side and a negative index starting from -1 from the right side.

String	H	E	L	L	O	W	O	R	L	D
Positive Index	0	1	2	3	4	5	6	7	8	9
Negative Index	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- string[a]: Returns a character from a positive index a of the string from the left side as displayed in the index graph above.
- string[-a]: Returns a character from a negative index a of the string from the right side as displayed in the index graph above.
- string[a:b]: Returns characters from positive index a to positive index b of the as displayed in index graph above.
- string[a:-b]: Returns characters from positive index a to the negative index b of the string as displayed in the index graph above.
- string[a:]: Returns characters from positive index a to the end of the string.
- string[:b] Returns characters from the start of the string to the positive index b.
- string[-a:]: Returns characters from negative index a to the end of the string.
- string[:-b]: Returns characters from the start of the string to the negative index b.
- string[::-1]: Returns a string with reverse order.

**Program 6.**

```
File Edit Format Run Options Window
string1 = "helloworld"

print(string1[1])
print(string1[-3])
print(string1[1:5])
print(string1[1:-3])
print(string1[2:])
print(string1[:5])
print(string1[:-2])
print(string1[-2:])
print(string1[::-1])
```

```
=== RESTART: C:/Us
e
r
ello
ellowo
lloworld
hello
hellowor
ld
dlrowolleh
>>>
```

## 5. String Comparison Operator “==” & “!=”:

The string comparison operator in python is used to compare two strings.

- “==” operator returns Boolean True if two strings are the same and return Boolean False if two strings are not the same.
- “!=” operator returns Boolean True if two strings are not the same and return Boolean False if two strings are the same.

These operators are mainly used along with if condition to compare two strings where the decision is to be taken based on string comparison.

### Program 7.

```
sample.py - C:/Users/user/AppData/Local/Programs/Python/Python38-32
File Edit Format Run Options Window Help
string1 = "hello"
string2 = "helloworld"
string3 = "helloworld"
string4 = "world"
print(string1==string4)
print(string2==string3)
print(string1!=string4)
print(string2!=string3)
```

```
>>>
=== RESTART: C:/U
False
True
True
False
>>>
```

## 6. Membership Operator “in” & “not in”:

Membership operator is used to search whether the specific character is part/member of a given input python string.

**“a” in the string:** Returns boolean **True** if “a” is in the string and returns **False** if “a” is not in the string.

**“a” not in the string:** Returns boolean **True** if “a” is not in the string and returns **False** if “a” is in the string.

A membership operator is also useful to find whether a specific substring is part of a given string.

### Program 8.

```
sample.py - C:/Users/user/AppData/Local/Programs/Python/Python38-32
File Edit Format Run Options Window Help
string1 = "helloworld"
print("w" in string1)
print("W" in string1)
print("t" in string1)
print("t" not in string1)
print("hello" in string1)
print("Hello" in string1)
print("hello" not in string1)
```

```
>>>
= RESTART:
True
False
False
True
True
False
False
>>>
```

## 7. Escape Sequence Operator “\.”:

To insert a non-allowed character in the given input string, an escape character is used. An escape character is a “\” or “backslash” operator followed by a non-allowed character. An example of a non-allowed character in python string is inserting double quotes in the string surrounded by double-quotes.

### 1. Example of non-allowed double quotes in python string:

```
sample.py - C:/Users/user/AppData/Local/Programs/Python/Python38-32
File Edit Format Run Options Window Help
string = "Hello world I am from "India""
print(string)
```

```
>>> string = "Hello world I am from "India""
SyntaxError: invalid syntax
>>>
```



**8. String Formatting Operator “%.”:** String formatting operator is used to format a string as per requirement. To insert another type of variable along with string, the “%” operator is used along with python string. “%” is prefixed to another character indicating the type of value we want to insert along with the python string. Please refer to the below table for some of the commonly used different string formatting specifiers :

Operator	Description
%d	Signed decimal integer
%u	unsigned decimal integer
%c	Character
%s	String
%f	Floating-point real number

#### Program 10.

```
sample.py - C:/Users/user/AppData/Local/Programs/Python/Python312/sample.py
File Edit Format Run Options Window Help
name = "india"
age = 19
marks = 20.56
string1 = 'Hey %s' % (name)
print(string1)
string2 = 'my age is %d' % (age)
print(string2)
string3= 'Hey %s, my age is %d' % (name, age)
print(string3)
string3= 'Hey %s, my subject mark is %f' % (name, marks)
print(string3)
```

```
--- RESTART: C:/Users/user/AppData/Local/Fl
Hey india
my age is 19
Hey india, my age is 19
Hey india, my subject mark is 20.560000
>>>
```

## Operator Precedence

An expression in python may have more than one operator involved in it. When, more than one operator is to be evaluated in an expression, the interpreter decides at run time which operator should be evacuated first. The decision based on the precedence and associativity of the operations as explained below:

Operator	Description
()	Parenthesis
**	Exponentiation
+a,-a	+unary,-unary
*,./,/,%	Multiplication, division, floor division, modulus (Remainder)
+,-	Binary addition and subtraction
<,<=,>,>=,==,!=	Relational operators
Not, and, or	Boolean/logical operators

Precedence is the priority order of operator according to which it is evaluated. Each operator has precedence associated with it. The precedence is used to determine the order of evaluation of an expression involving more than one operator. There are different levels of precedence from high to low. An operator belonging to a higher level is evaluated first

**Associativity:** If two operators have the same precedence (priority), then they are either evaluated from 'left' to 'right' or from 'right' to 'left' based on their level. It is termed as associativity, which tells the direction of execution of operators ("left to Right or Right to left") when operator in an expression have the same precedence.

## Algorithm

An algorithm is well defined step by step procedure or a way to write a program. It helps us to understand the problem and its solution in a better way.

***“Algorithm is defined as: “A sequence of activities to be processed for getting desired output from a given input”.***

Donald Ervin Knuth has given a list of five properties for an algorithm, these properties are:

- i. **Finiteness:** An algorithm must always terminate after a finite number of steps. It means after every step one reach closer to solution of the problem and after a finite number of steps algorithm reaches to an end point.
- ii. **Definiteness:** Each step of an algorithm must be precisely defined. It is done by well thought actions to be performed at each step of the algorithm. Also the actions are defined unambiguously for each activity in the algorithm.
- iii. **Input:**Any operation you perform need some beginning value/quantities associated with different activities in the operation. So the value/quantities are given to the algorithm before it begins.
- iv. **Output:** One always expects output/result (expected value/quantities) in terms of output from an algorithm. The result may be obtained at different stages of the algorithm. If some result is from the intermediate stage of the operation then it is known as intermediate result and result obtained at the end of algorithm is known as end result. The output is expected value/quantities always have a specified relation to the inputs.
- v. **Effectiveness:**Algorithms to be developed/written using basic operations. Actually operations should be basic, so that even they can in principle be done exactly and in a finite amount of time by a person, by using paper and pencil only.



The algorithm and flowchart include following three types of control structures.



1. **Sequence:** In the sequence structure, statements are placed one after the other and the execution takes place starting from up to down.
2. **Branching (Selection):** In branch control, there is a condition and according to a condition, a decision of either TRUE or FALSE is achieved. In the case of TRUE, one of the two branches is explored; but in the case of FALSE condition, the other alternative is taken. Generally, the 'IF-THEN' is used to represent branch control.
3. **Loop (Repetition):** The Loop or Repetition allows a statement(s) to be executed repeatedly based on certain loop condition e.g. WHILE, FOR loops.

### Advantages of algorithm

- It is a step-wise representation of a solution to a given problem, which makes it easy to understand
- An algorithm uses a definite procedure.
- It is not dependent on any programming language, so it is easy to understand for anyone even without programming knowledge.
- Every step in an algorithm has its own logical sequence so it is easy to debug.

## How To Write Algorithms

- Step 1:** Define your algorithms input: Many algorithms take in data to be processed, e.g. to calculate the area of rectangle input may be the rectangle height and rectangle width.
- Step 2:** Define the variables: Algorithm's variables allow you to use it for more than one place. We can define two variables for rectangle height and rectangle width as HEIGHT and WIDTH (or H & W). We should use meaningful variable name e.g. instead of using H & W use HEIGHT and WIDTH as variable name.
- Step 3:** Outline the algorithm's operations: Use input variable for computation purpose, e.g. to find area of rectangle multiply the HEIGHT and WIDTH variable and store the value in new variable (say) AREA. An algorithm's operations can take the form of multiple steps and even branch, depending on the value of the input variables.
- Step 4:** Output the results of your algorithm's operations: In case of area of rectangle output will be the value stored in variable AREA. if the input variables described a rectangle with a HEIGHT of 2 and a WIDTH of 3, the algorithm would output the value of 6.

## Flowchart

The first design of flowchart goes back to 1945 which was designed by John Von Neumann. Unlike an algorithm, Flowchart uses different symbols to design a solution to a problem.



It is another commonly used programming tool. By looking at a flowchart one can understand the operations and sequence of operations performed in a system. Flowchart is often considered as a blueprint of a design used for solving a specific problem.





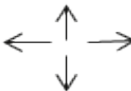




*“The flowchart is a diagram which visually presents the flow of data through processing systems. It is diagrammatic / Graphical representation of sequence of steps to solve a problem.”*

**Algorithms** are nothing but sequence of steps for solving problems. So a flowchart can be used for representing an algorithm. A flowchart, will describe the operations (and in what sequence) are required to solve a given problem.”

### Advantages of Flowchart:

- Flowchart is an excellent way of communicating the logic of a program.
- Easy and efficient to analyze problem using flowchart.
- During program development cycle, the flowchart plays the role of a blueprint, which makes program development process easier.
- After successful development of a program, it needs continuous timely maintenance during the course of its operation. The flowchart makes program or system maintenance easier.
- It is easy to convert the flowchart into any programming language code.

### To draw a flowchart following standard symbols are use:

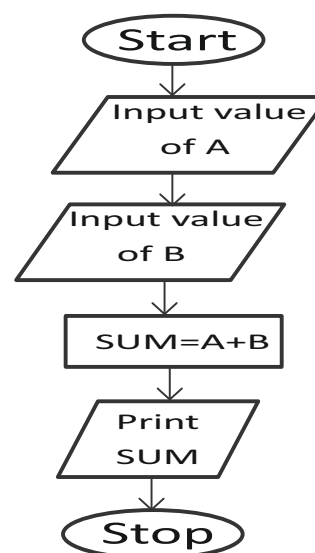
Symbol Name	Symbol	Function
Oval		Used to represent start and end of flow chart.
Parallelogram		Used to input & output Operations
Rectangle		Processing: Used for arithmetic operations and data-manipulations
Diamond		Decision making. Used to represent the operation in which there are two/three alternatives, true and false etc
Arrows		Flow line Used to indicate the flow of logic by connecting symbols
Circle		Page Connector
		Off Page Connector
		Predefined Process /Function Used to represent a group of statements performing one processing task.
		Pre-processor

## Algorithm & Flowchart to find the sum of two numbers:

### Algorithm

- Step-1:** Start
- Step-2:** Input first numbers say A
- Step-3:** Input second number says B
- Step-4:**  $SUM = A + B$
- Step-5:** Display SUM
- Step-6:** Stop

### Flowchart:



## Types of Control Structure

### Sequential Statement:

In sequential construct, the statements in the program are executed in a sequential manner, where one statement is followed by the other, with no possibility of branching off to other action.

**Program:** To calculate the area and perimeter of a square, first, enter the side of square, then calculate its area and perimeter, and print the result.

```
sample.py - C:/Users/user/AppData/Local/Programs/Python/Pyth...  
File Edit Format Run Options Window Help  
side=int (input ("Enter the side of a square: n"))  
area=side*side  
perimeter= 4*side  
print("Area= ", area)  
print("Perimeter= ", perimeter)
```

### Conditional Statements:

In programing languages, conditional statements cause the program control to transfer to a specific location depending on the outcome of the conditional expression. Every decision involves a choice between the two alternatives 'Yes' and 'No' result. If a conditional statement is true, then one set of statements are executed, otherwise, the other set of statements are executed.

### Iterative Statements:

These statements enable the execution of a set of statements to repeat till the condition is true. As soon as the condition becomes false, the control comes out of the loop and the repetition stops.

# Conditional Statements in Python

There come situations in real life when we need to make some decisions and based on these decisions, we decide what we should do next. Similar situations arise in programming also where we need to make some decisions and based on these decisions we will execute the next block of code. Decision-making statements in programming languages decide the direction of the flow of program execution.

Conditional statements are also called decision-making statements. We use those statements while we want to execute a block of code when the given condition is true or false.

Type of condition statement in Python:

**If statement.**

**Nested if statement.**

**If Else statement.**

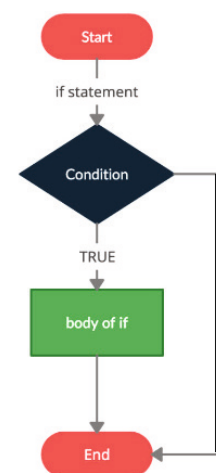
**Nested if else statement.**

**Elif statement.**

## If Statement:

Python if Statement is used for decision-making operations. It contains a body of code which runs only when the condition given in the if statement is true. If the condition is false, then the optional else statement runs which contains some code for the else condition.

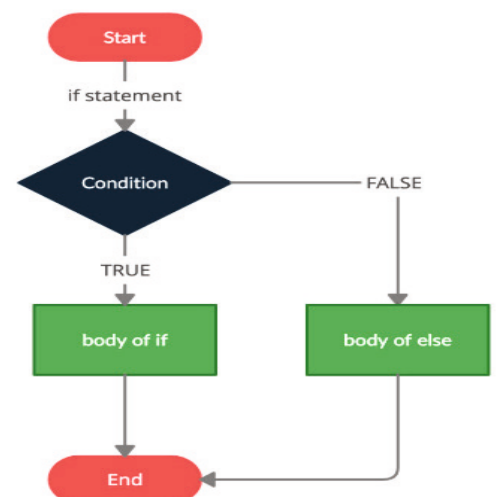
**Syntax :** `if(condition):`  
`{`  
`#if statement`  
`}`



## If else statement:

The statement itself says that if a given condition is true or false. True means executing the “if” statement to the output. False means executing the “else” statement to the output.

**Syntax :** `if(condition):`  
`{`  
`# if statement`  
`}`  
  
`else:`  
`{`  
`# else statement`  
`}`





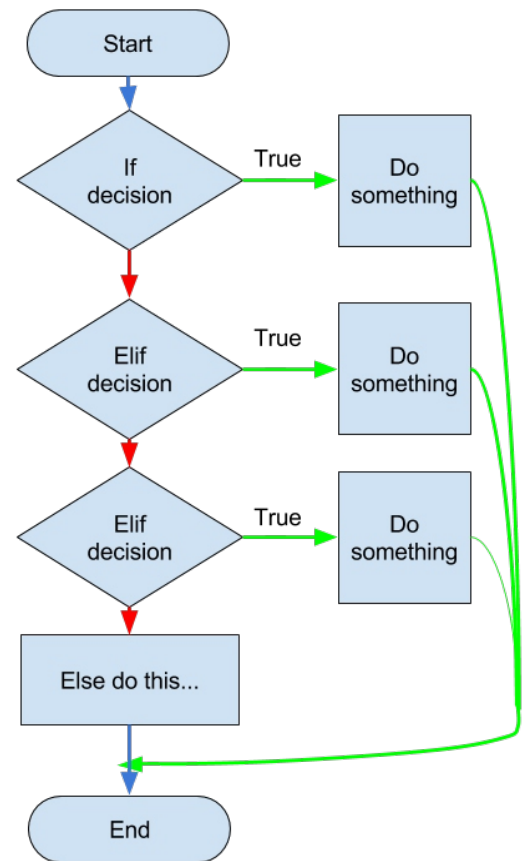
## Elif Statement:

Elif is a shortcut of else if condition statements.

In Python one or more conditions are used in the elif statement.

**Syntax:**

```
if(condition):  
    {  
        # if statement  
    }  
elif(condition):  
    {  
        # elif statement  
    }  
else:  
    {  
        # else statement  
    }
```

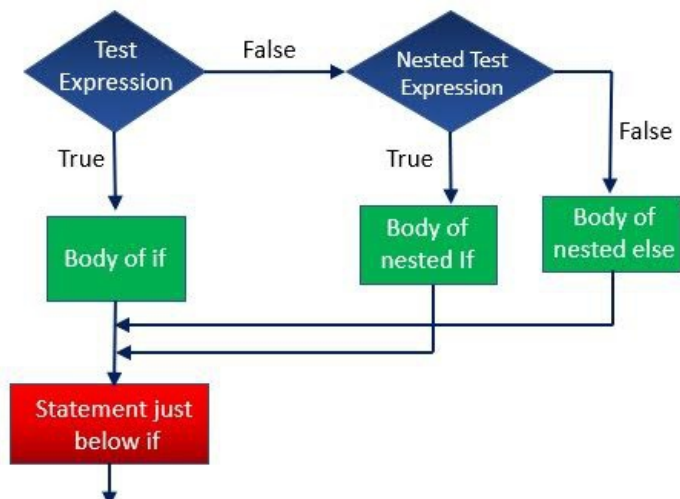


## Nested if statement:

In python is using “if” statements inside other “if” statements it is called nested if statement.

**Syntax:**

```
if(condition):  
    {  
        if(condition):  
            {  
                # if statement  
            }  
    }  
}
```

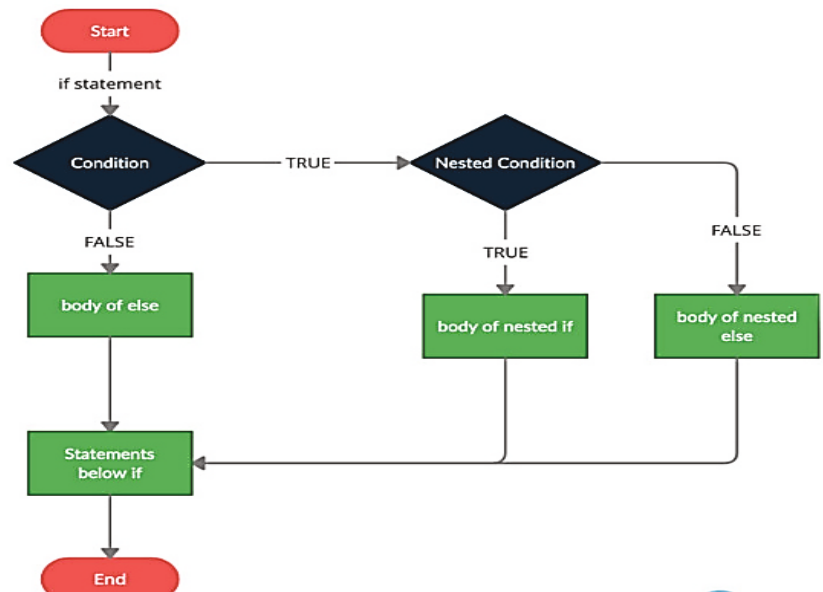


## Nested if else statement:

In Python is using one “if else” statement inside other if else statements it is called nested if else statement.

### Syntax:

```
if(condition) :  
{  
    if(condition):  
    {  
        #if statement  
    }  
    else:  
    {  
        #else statement  
    }  
}  
else:  
{  
    #else statement  
}
```



### Quick Look



- 💡 Operators are symbols that perform arithmetic and logical operations on operand and provide a meaningful result.
- 💡 Unary Operators: Operate on only one operand, e.g.,  $x = +200$
- 💡 Binary Operators: Operate on two operands, e.g.,  $a + b$ .
- 💡 Assignment operators(=) is used to assign the value of an expression to a variable.
- 💡 Rational operators are used to show relationship between operand.
- 💡 Precedence is the priority order of an operator according to which it is evaluated.
- 💡 Conditional statements cause the program control to transfer to a specific location depending on the out come of the conditional expression.
- 💡 Iterative Statement enable the execution of a set of statement to repeat till the condition is true.





## Section - I

### ► Objective Type Questions

*Pass Statement      for loop      Loops      Basic formula      Range*

#### A. Fill in the blanks with the correct option:

1. \_\_\_\_\_ help us to remove the redundancy of code when a task has to be repeated several times.
2. In Python \_\_\_\_\_ when a statement is required syntactically but you do not want any command or code to execute.
3. A \_\_\_\_\_ is a programming concept that, when it's implemented, executes a piece of code over and over again "for" a certain number of times, based on a sequence.
4. \_\_\_\_\_ function only works with integers i.e. whole numbers.
5. The first part of the while loop is called \_\_\_\_\_.

#### B. Write T for the true statement and F for the false one:

1. Break statement causes the loop to skip the reminder of its body and immediately retest its condition prior to reiterating . ☐
2. The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute. ☐
3. Initialization refers to first part of a while loop. Before entering the condition section some initialization is required . ☐
4. Python allow you to use else block with **for** but not with **while** loops. ☐
5. There is the utility of a while loop in a gaming application or an application where we enter some sort of main event loop, which continues to run until the user selects an action to break that infinite loop. ☐

### C. Choose the right option::

1. What will be the output of the following Python code?

```
x = ['ab', 'cd']
for i in range(len(x)):
    x[i] = x[i].upper()

print(x)
```

- a) ['ab','cd']      b) ['AB','CD']  
c) [None, None]      d) none of the above

2. Which type of loop can be used to perform the following iteration: You choose a positive integer at random and then print the numbers from 1 up to and including the selected integer.

- a) a for-loop or a while-loop      b) only a for-loop  
c) only a while-loop      d) None of the above

3. What is the output of the following?

```
i = 2
while True:
    if i % 3 == 0:
        break
    print(i)
    i += 2
```

- a) 2 4 6 8 10 " |      b) 2 4  
c) 23      d) error

4. What is the output of the following?

```
x = "abcdef"
i = "a"
while i in x:
    x = x[:-1]
    print(i, end=" ")
```

- a) i i i i i I      b) a a a a a a  
c) a a a a a      d) none of the above

5. What is the output of the following?

```
x = 'abcd'
for i in x:
    print(i.upper())
```

- a) a b c d      b) A B C D  
c) a B c D      d) none of the above



## D. Application Based Questions:-



### A. Give the outputs of the following codes:

1. `p=10`

`q=20`

`print(p>10 and b<50)`

2. `a=6`

`if a%2==0:`

`print(a,'is an even number')`

`else:`

`print(a,'is an odd number')`

### B. Correct and Rewrite the following statements:

1. `x=20, y=30`

`Sum= x+y`

`Print(Sum of number is + sum))`

2. `a=6`

`if A%2=0`

`Print"a, is odd number"`

## Section - II

### ► Descriptive Type Questions

#### E. Answer The Following questions:

1. Explain Iterative statement in Python. Explain its type

---

---

2. Write a Python program to print even numbers by using while loop.

---

---

3. Define range() function and its operators.

---

---

4. What is infinite loop? Explain its importance in loops.

---

---

5. Write a Python program print Fibonacci series as given numbers

---

---

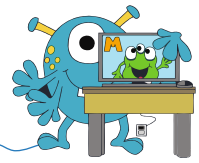
# Activity

TIME








## LAB ACTIVITY

*Practice Makes Perfect*



A. Write the code for the following programs using script mode:

-  Use a string replication operator to display a number 15 times.
-  **Ask user to input two numbers and design a program that can perform the following tasks:**
  - Multiplication
  - Addition
  - Subtraction
-  Finding the remainder when one number is divided by another.
-  Finding the quotient when integer division is performed.
-  Finding the quotient when normal division is performed.

B. Write a program to display a message, i.e “Eligible to Vote”, if the following condition is true: Age of the person should be  $\geq 18$




C. A leap year is a leap year if it is divisible by 4. Write a program in Python to find if the input year by the user is a leap year or not.

## GROUP DISCUSSION

*For Better Concept Clarity*






Divide the class into three parts and discuss:

-  Types of Control Structure
-  How to write Algorithms
-  Types of Operators in Python

## ONLINE LINKS

*For Searching More*



-  <https://www.geeksforgeeks.org/python-language-introduction/>
-  [https://www.w3schools.com/python/python\\_intro.asp](https://www.w3schools.com/python/python_intro.asp)
-  <https://realpython.com/python-introduction/>