##### **STUDENT NOTES**

**INTODUCTION:**

* Python is an interpreted, object-oriented, high-level programming language that can be used for a wide variety of applications.
* Python is a powerful general-purpose programming language.
* First developed in the late 1980s by Guido van Rossum.
* Python is open source programming language.
* Guido van Rossum named it after the BBC Comedy TV series **Monty Python’s Flying Circus**

**Advantages of Python Programming**

* Python is easy to learn and use.
* Large number of **libraries** available that can be used in your projects today.
* High Level Programming Language.
* Huge Community
* Object-oriented language
* Portable across Operating systems

#### Organizations using Python

* Google
* microsoft
* facebook
* mozilla
* cisco
* Quora
* YouTube etc...

#### Python Based Jobs

* Data Analysis \_#py #excel #sql
* Artificial Intelligence
* Automation
* Web Application
* Desktop Application
* Hacking
* School Students

**LIBRARY**

A library in programming is a collection of pre-written code that helps programmers solve common problems or perform specific tasks. It provides ready-made functions, classes, and modules that can be easily integrated into programs, saving time and effort. Libraries are like toolboxes that programmers can use to access and reuse existing code, making development faster and more efficient.

#### High-level Languages

High-level languages, like C, C++, JAVA,Python etc., are very near to English. It makes programming process easy. However, it must be translated into machine language before execution. This translation process is either conducted by either a compiler or an interpreter. Also known as source code.

#### Machine Code

Machine languages are very close to the hardware. Every computer has its machine language. A machine language programs are made up of series of binary pattern. (Eg. 110110) It represents the simple operations which should be performed by the computer. Machine language programs are executable so that they can be run directly.

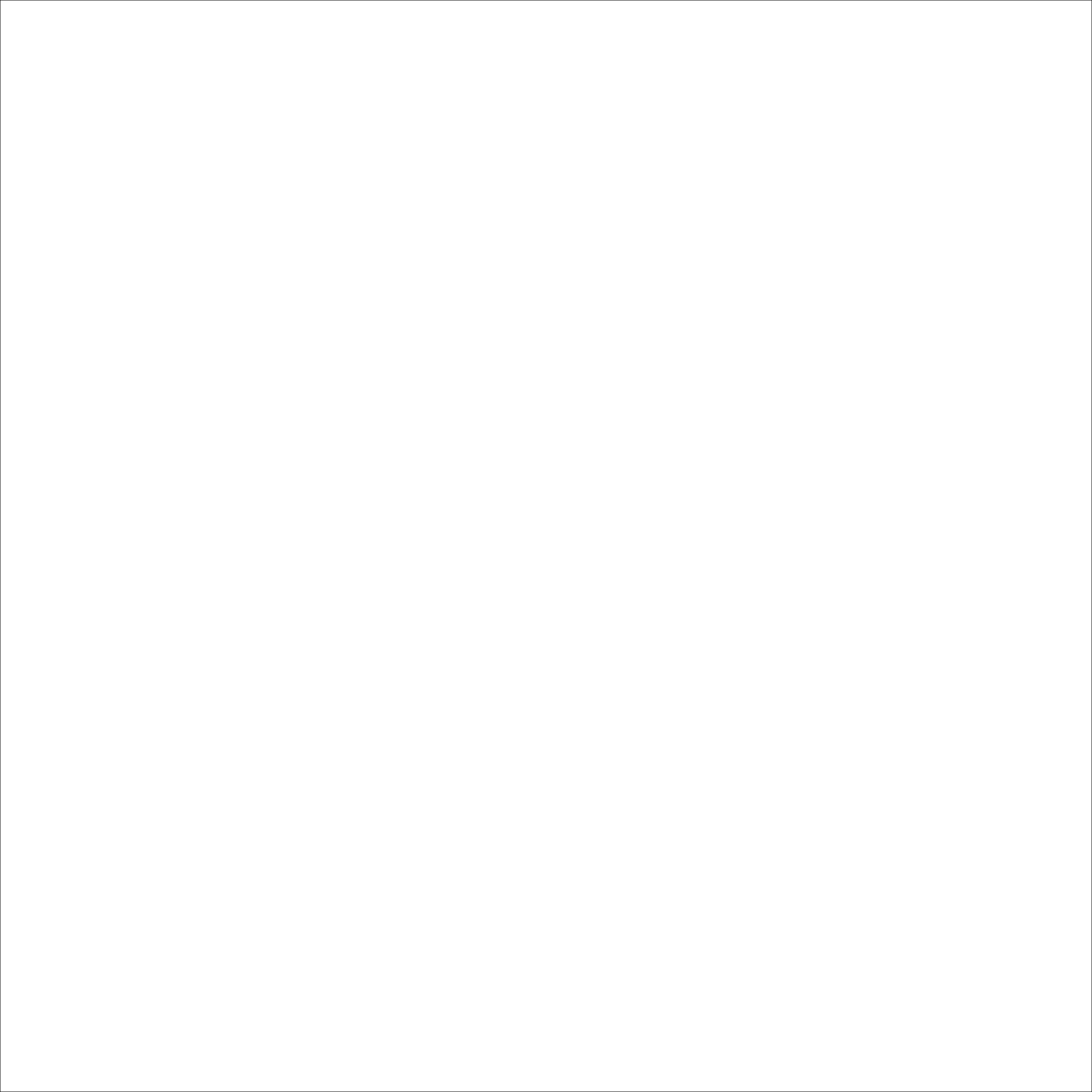
#### Compiler

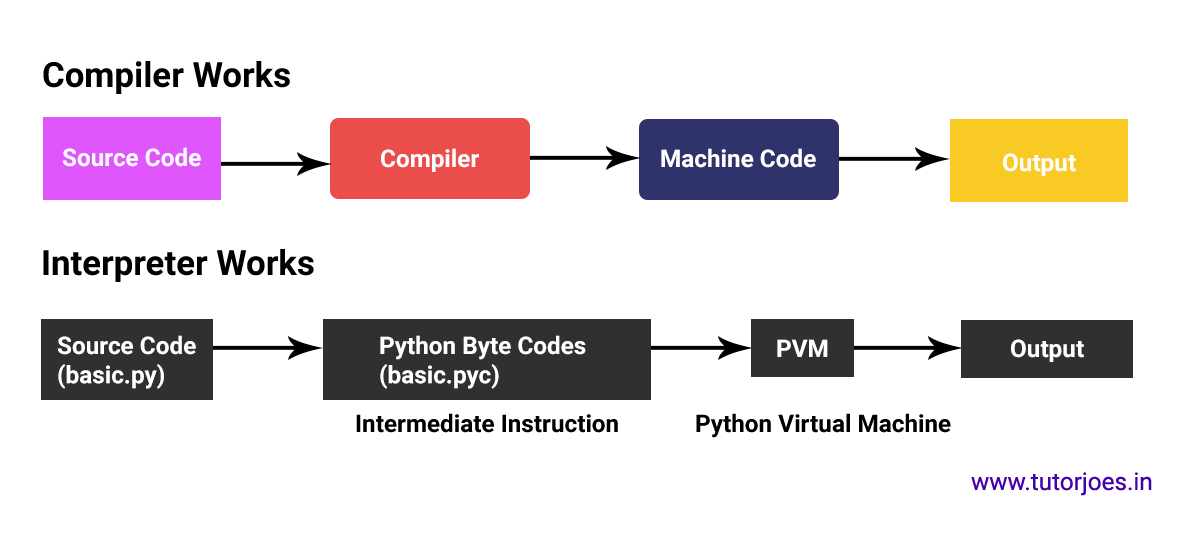
A Compiler transforms code written in a high-level programming language into a machine code or any other Intermediate code.It converts the text that a programmer writes into a format the CPU can understand.The conversion manner is diffrent.It convers the Entire source code in one go and reports all the errors of the program along with the line numbers.After all the error are removed,the program is recompiled and after that the Compiler is no needed in memory as the object program is available.

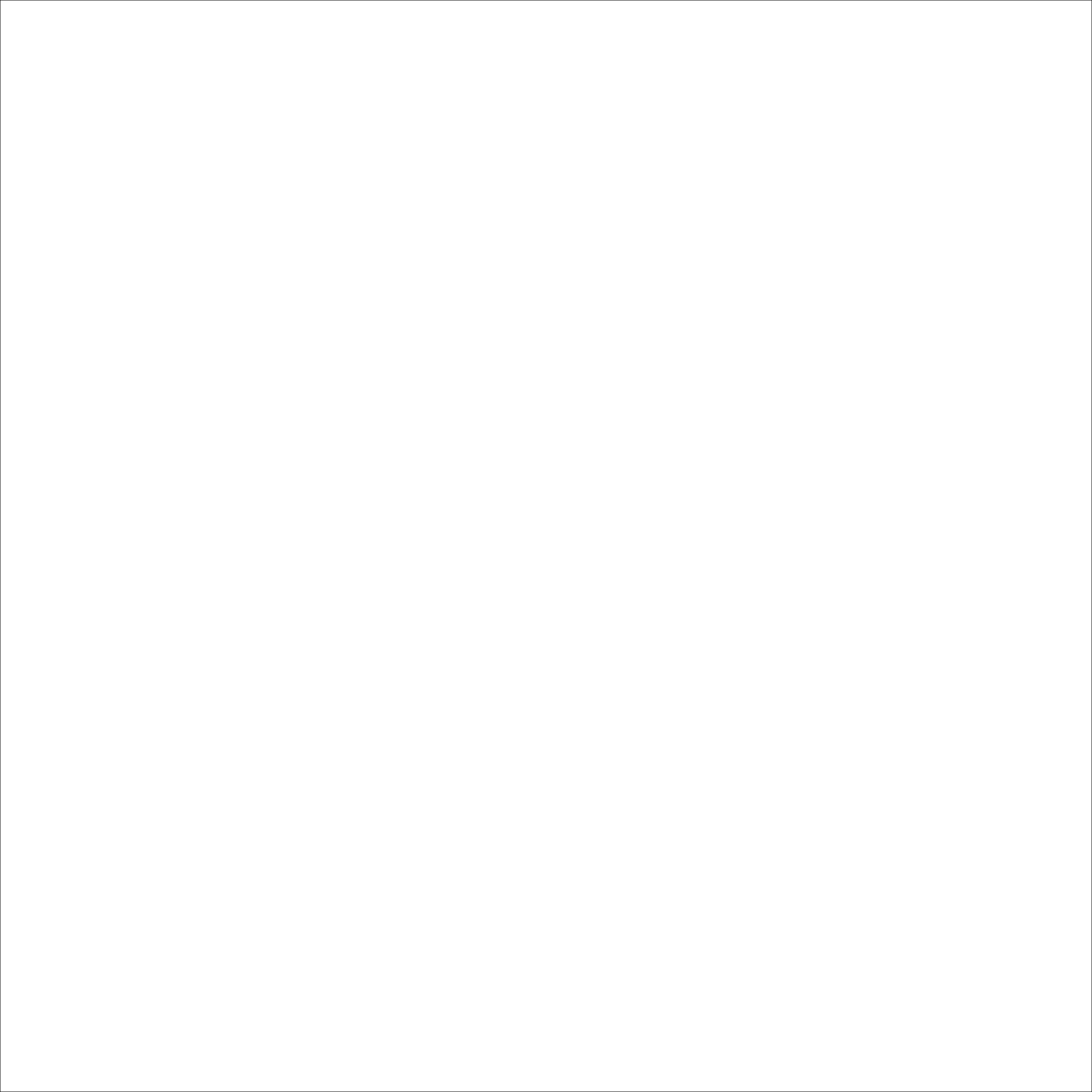
#### interpreter

The interpreter converts the source code line-by-line during Run Time,translates it into machine code or virtual machine code. Interpreter allows evaluation and modification of the program while it is executing.if there is error in any line,it reports it at the same time and program executio cannot resume untill the errror is rectified.Interprerer must always be present in the memory every time the program is run.it is first interpreted and then executed

\_







**TWO TYPES MODE:**

**Interactive Mode**

**Script Mode**

### **Keywords in Python Programming**

Keywords are the words that convey a special meaning to the language compiler/interpreter. These are reserved for special purpose and must not be used as normal identifier names. **Eg: for,break,continue,etc.**

* A keyword is a word having special meaning reserved by programming language
* We cannot use a keyword as a variable name, function name or any other identifier.
* True and False are truth values in Python. They are the results of comparison operations or logical (Boolean) operations in Python.
* None is a special constant in Python that represents the absence of a value or a null value.
* and, or, not are the logical operators in Python. and will result into True only if both the operands are True.
* The async and await keywords are provided by the asyncio library in Python. They are used to write concurrent code in Python.

#### Source Code

**import** keyword

**print**(keyword.kwlist)

#### Output

['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else',   
'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return',  
 'try', 'while', 'with', 'yield']

### **Variables in Python Programming**

In programming, a "variable" is a container in which a data value can be stored within the computer’s memory. The stored value can then be referenced using the variable’s name. Data to be stored in a variable is assigned in a Python program declaration statement with the = assignment operator.

**Syntax:**  
    variable\_name = variable\_value;

**Example:**  
       a = 10;  
       name = "Joes";

**Here are some rules for creating and using variables in Python:**

* Variable names can only contain letters, numbers, and underscores. They cannot start with a number.
* Variable names should be descriptive and meaningful, and should not contain spaces. It's common to separate words in a variable name with an underscore.
* Variable names should not conflict with Python keywords, such as print, if, or for.
* Python is case sensitive, so name and Name are considered two different variables.
* A variable must be assigned a value before it can be used.
* The data type of a variable is determined by the type of value assigned to it. In Python, the data type of a variable can change dynamically.
* Variables can be assigned a new value at any time, and their data type can change as well.
* In Python, it's a good practice to initialize a variable with a meaningful value when it's created. This helps avoid potential bugs caused by using an uninitialized variable.

**EX:**

name= "Ram"  
User\_name= "Ram"  
name2= "Ram"

**INPUT FUNCTION IN PYTHON PROGRAMING:**

In Python programming, the input() function is used to get input from the user. It displays a prompt on the screen and waits for the user to enter some text. Once the user enters the input and presses Enter, the input() function captures that value as a string and returns it to the program.

Example:

name = input("Enter your name: ")

print("Hello, " + name + "!")

### **Pycharm Installation in Python----**

### **Single and Multiline Comment in Python**

In Python, there are two types of comments: single-line and multi-line.

* Single-line comments start with a hash symbol (#) and extend to the end of the line:
* Multi-line comments start and end with three quotation marks (""").

#### Source Code

# Basic Program in Python

# Basic Program in Python

# Basic Program in Python

'''

Basic Program in Python

Basic Program in Python

Basic Program in Python

Basic Program in Python

Basic Program in Python

'''

a = 10

b = 20

c = a + b

**print**(c)

#### Output

30

### **Type Casting in Python**

In Python, type casting is the process of converting one data type to another. Python is a dynamically-typed language, which means that the data type of a variable can change based on the value assigned to it. However, sometimes you may need to convert a variable from one data type to another.

There are several built-in functions in Python that can be used for type casting:

* int(): Converts a value to an integer.
* float(): Converts a value to a floating-point number.
* str(): Converts a value to a string.
* bool(): Converts a value to a Boolean (True or False).

#### Source Code

"""

a = 10.0

print(a)

print(type(a))

b = int(a)

print(b)

print(type(b))

int()

float()

str()

"""

a = int(input("Enter The Value of A : "))

b = int(input("Enter The Value of B : "))

c = a + b

**print**("Total : " + str(c))

#### Output

Enter The Value of A : 20

Enter The Value of B : 20

Total : 40

This is a simple program in Python that performs the following operations:

* Accepts two integer inputs from the user, a and b.
* Adds a and b and stores the result in a variable c.
* Converts the result stored in c to a string using the str() function.
* Prints the message "Total : " followed by the result stored in c.

In this program, the user inputs are obtained using the input() function and then cast to integers using int(). The result of the addition of a and b is then stored in c, and to display it, the c is first converted to a string using str() before it's concatenated with the string "Total : " using the + operator. Finally, the result is displayed on the screen using the print() function.

### **String and String Functions in Python**

Python has several built-in functions associated with the string data type. These functions let us easily modify and manipulate strings. Creating Strings is the simplest and easy to use in Python. To create a string in Python, we simply enclose a text in single as well as double-quotes.

* type() => The returns the type of the object.
* upper() => All the characters in a given string are uppers case.
* lower() => All the characters in a given string are lower case.
* capitalize() => The first character is the upper case
* The title() => The first character in every word of the string is an upper case.
* count() => Finds the number of times a specified value in the given string.
* find() => Finds the first occurrence of the specified value.
* replace() => Replaces a specified phrase with another specified phrase.
* isalnum() => Checks whether all the characters in a given string is alphanumeric or not
* isalpha() => returns True if all the characters in the string are alphabets
* islower() => Checks if all characters in the string are lowercase
* isupper() => Checks if all characters in the string are uppercase
* strip() => The used to trim whitespaces from the string object

#### Source Code

# String And String Function

s = "tutor Joes"

**print**(s)

**print**(type(s))

**print**(s.upper())

**print**(s.lower())

**print**(s.capitalize())

**print**(s.title())

**print**(s.count("t"))

**print**(s.endswith("ED"))

**print**(s.find("o"))

**print**(s.find("o", 5))

**print**(s.replace("o", '0'))

a = "joes1234"

**print**("Is Upper : ", a.isupper())

**print**("Is Lower : ", a.islower())

**print**("Is Alpha Numeric : ", a.isalnum())

**print**("Is Alpha : ", a.isalpha())

s = "he\nis\ngood"

**print**(s)

**print**(s.splitlines())

**print**(s.splitlines(True))

a = "Tutor Joes Computer Education"

**print**(a.split(" "))

a = "Tutor,Joes,Computer,Education"

**print**(a.split(","))

s=" Joes "

**print**(len(s))

**print**(len(s.strip()))

**print**(len(s.lstrip()))

**print**(len(s.rstrip()))

s='12-03-2020'

**print**(s.partition('-'))

#### Output

tutor Joes

TUTOR JOES

tutor joes

Tutor joes

Tutor Joes

2

False

3

7

tut0r J0es

Is Upper : False

Is Lower : True

Is Alpha Numeric : True

Is Alpha : False

he

is

good

['he', 'is', 'good']

['he\n', 'is\n', 'good']

['Tutor', 'Joes', 'Computer', 'Education']

['Tutor', 'Joes', 'Computer', 'Education']

13

4

9

8

('12', '-', '03-2020')

### **String Manipulation in Python**

String manipulation is a process of manipulating the string, such as slicing, parsing, analyzing, etc. String slicing in python programming is all about fetching a substring from a given string by slicing it from a **start**to **end** index.

**Syntax :**  
   s [ start : end ]  
   s [ start : ]  
   s [ : end ]  
   s [ : : ]

This program is using the concept of slicing in python, which allows you to extract a portion of a string by specifying a range of indices.

The following operations are performed in the above code:

* print(s): This line will print the original string "sample".
* print(s[0:2]): This line uses slicing to extract a substring of the original string "sa", starting at index 0 and ending at index 2 (not included).
* print(s[:5]): This line uses slicing to extract a substring of the original string "sampl", starting at index 0 and ending at index 5 (not included).
* print(s[1:]): This line uses slicing to extract a substring of the original string "ample", starting at index 1 and going till the end of the string.
* print(s[-1]): This line uses slicing to extract the last character of the string "e" by specifying the index as -1, as in python, negative index is used to access elements from the end of the list.
* print(s[-2:-1]): This line uses slicing to extract the second last character of the string "l" by specifying the range as -2 to -1, as in python

#### Source Code

# String Manipulation

'''

 S a m p l e

 0 1 2 3 4 5

-6 -5 -4 -3 -2 -1

'''

s = "sample"

**print**(s)

**print**(s[0:2])

**print**(s[:5])

**print**(s[1:])

**print**(s[-1])

**print**(s[-2:-1])

**print**(s[:-1])

**print**(s[::-1])

#### Output

sample

sa

sampl

ample

e

l

sampl

elpmas

### **Arithmetic Operators in Python**

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication ,division and also python have floor division,exponentiation.

* Addition ( + ) => This operator is a binary operator and is used to add two operands.
* Subtraction ( - ) => This operator is a binary operator and is used to subtract two operands.
* Multiplication ( \* ) => This operator is a binary operator and is used to multiply two operands.
* Division ( / ) => This is a binary operator that is used to divide the first operand(dividend) by the second operand(divisor) and give the quotient as result.
* Modulus ( % ) => This is a binary operator that is used to return the remainder when the first operand(dividend) is divided by the second operand(divisor).
* Exponentiation ( \* \* ) => The performs exponential (power) calculation on operators
* Floor division ( / / ) => The division of operands where the result is the quotient in which the digits after the decimal point are removed.

#### Source Code

# Arithmetic operators

"""

+ Addition

- Subtraction

\* Multiplication

/ Division

% Modulus

\*\* Exponentiation

// Floor division

"""

a = 123

b = 10

**print**(a + b)

**print**(a - b)

**print**(a \* b)

**print**(a / b)

**print**(a // b)

**print**(a % b)

**print**(2\*\*3)

#### Output

133

113

1230

12.3

12

3

8

### **Assignment Operators in Python**

Assignment operators are used to assigning value to a variable. The left side operand of the assignment operator is a variable and right side operand of the assignment operator is a value. This operator is used to assign the value on the right to the variable on the left

|  |  |  |
| --- | --- | --- |
| **Compound Operator** | **Sample Expression** | **Expanded Form** |
|  |  |  |
| += | a+=2 | a=a+2 |
|  |  |  |
| -= | a-=6 | a=a-6 |
|  |  |  |
| \*= | a\*=7 | a=a\*7 |
|  |  |  |
| /= | a/=4 | a=a/4 |
|  |  |  |
| %= | a%=9 | a=a%9 |
|  |  |  |
| \*\*= | a\*\*=3 | a=a\*\*3 |
|  |  |  |
| //= | a//=2 | a=a//2 |
|  |  |  |

This program is using the assignment operators along with the basic arithmetic operations in python

* a = 125: This line assigns the value 125 to the variable a.
* print(a): This line prints the value of a which is 125.
* a += 5: This line uses the shorthand assignment operator += to add 5 to the current value of a and assigns the result back to a. The value of a becomes 130.
* print(a): This line prints the updated value of a which is 130.
* a -= 10: This line uses the shorthand assignment operator -= to subtract 10 from the current value of a and assigns the result back to a. The value of a becomes 120.
* print(a): This line prints the updated value of a which is 120.
* a \*= 10: This line uses the shorthand assignment operator \*= to multiply the current value of a by 10 and assigns the result back to a. The value of a becomes 1200.
* print(a): This line prints the updated value of a which is 1200.
* a /= 10: This line uses the shorthand assignment operator /= to divide the current value of a by 10 and assigns the result back to a. The value of a becomes 120.
* print(a): This line prints the updated value of a which is 120.
* a %= 10: This line uses the shorthand assignment operator %= to find the remainder when dividing the current value of a by 10 and assigns the result back to a. The value of a becomes 0.0.
* print(a): This line prints the updated value of a which is 0.0.
* a \*\*=10: This line uses the shorthand assignment operator \*\*= to raise the current value of a to the power of 10 and assigns the result back to a. The value of a becomes 0.0.
* print(a): This line prints the updated value of a which is 0.0.
* a //= 10: This line uses the shorthand assignment operator //= to divide the current value of a by 10 and assigns the result rounded down to the nearest integer back to a. The value of a becomes 0.0.
* print(a): This line prints the updated value of a which is 0.0.

#### Source Code

# Assignment Operators

"""

= Assignment

+= Addition

-= Subtraction

\*= Multiplication

/= Division

%= Modulus

\*\*= Exponentiation

//= Floor division

"""

a = 125

**print**(a)

a += 5 # a=a+5

**print**(a)

a -= 10 # a=a-10

**print**(a)

a \*= 10 # a=a\*10

**print**(a)

a /= 10

**print**(a)

a %= 10

**print**(a)

a \*\*=10

**print**(a)

a //= 10

**print**(a)

#### Output

125

130

120

1200

120.0

0.0

0.0

0.0

### **Comparison Operators or Relational Operators in Python**

A comparison operator in python, also called python relational operators are used to establish some sort of relationship between the two operands. Some of the relevant examples could be less than, greater than or equal to operators. Relational operators compares the values of two operands and returns **TRUE**or **FALSE** based on whether the condition is met.

|  |  |
| --- | --- |
| **Operator** | **uses** |
| == | Equal operator |
|  |  |
| != | Not Equal operator |
|  |  |
| < | Less than operator |
|  |  |
| > | Greater than operator |
|  |  |
| <= | Less than or equal to operator |
|  |  |
| >= | Greater than or equal to operator |
|  |  |

The above program is using comparison operators in Python to compare the values of two variables a and b.

* a = 20: This line assigns the value 20 to the variable a.
* b = 20: This line assigns the value 20 to the variable b.
* print(a == b): This line uses the comparison operator == to check if the value of a is equal to the value of b. The output will be True because 20 is equal to 20.
* print(a != b): This line uses the comparison operator != to check if the value of a is not equal to the value of b. The output will be False because 20 is equal to 20.
* print(a > b): This line uses the comparison operator > to check if the value of a is greater than the value of b. The output will be False because 20 is not greater than 20.
* print(a < b): This line uses the comparison operator < to check if the value of a is less than the value of b. The output will be False because 20 is not less than 20.
* print(a >= b): This line uses the comparison operator >= to check if the value of a is greater than or equal to the value of b. The output will be True because 20 is equal to 20.
* print(a <= b): This line uses the comparison operator <= to check if the value of a is less than or equal to the value of b. The output will be True because 20 is equal to 20.

#### Source Code

# Comparison Operators or Relational Operators

"""

== Equal

!= Not equal

> Greater than

< Less than

>= Greater than or equal to

<= Less than or equal to

"""

a = 20

b = 20

**print**(a == b)

**print**(a != b)

**print**(a > b)

**print**(a < b)

**print**(a >= b)

**print**(a <= b)

#### Output

**True**

**False**

**False**

**False**

**True**

**True**

### **Logical Operators in Python**

Logical operators are used to combine multiple conditions in a single expression in Python. The three logical operators in Python are and, or, and not.

* and: This operator returns True if both the conditions on either side of the operator are True, otherwise it returns False.
* or: This operator returns True if either of the conditions on either side of the operator is True, otherwise it returns False.
* not: This operator inverts the truth value of a single condition. If a condition is True, the not operator will make it Fals
* e and vice versa.

This program is using logical operators in Python to check if the value of a variable a falls within a certain range.

* a = 25: This line assigns the value 25 to the variable a.
* print(a >= 10 and a <= 20): This line uses the logical operator and to check if the value of a is greater than or equal to 10 AND less than or equal to 20. Since 25 is not in the range 10 to 20, the output will be False.
* print(a >= 10 or a <= 20): This line uses the logical operator or to check if the value of a is greater than or equal to 10 OR less than or equal to 20. Since 25 is greater than 10 the output will be true
* print(not(a >= 10 and a <= 20)): This line uses the logical operator not to check if the value of a is not in the range of 10 to 20. The output will be True because 25 is not in the range 10 to 20.

#### Source Code

# Logical Operators in Python

"""

and

or

not

"""

a = 25

**print**(a >= 10 **and** a <= 20)

**print**(a >= 10 **or** a <= 20)

**print**(**not**(a >= 10 **and** a <= 20))

#### Output

False

True

True

### **Bitwise Operators in Python**

In Python, bitwise operators are used to perform bitwise calculations on integers. The integers are first converted into binary and then operations are performed on bit by bit, hence the name bitwise operators. Then the result is returned in decimal format. Bitwise AND operator: Returns 1 if both the bits are 1 else 0

|  |  |
| --- | --- |
| **Operator** | **Description** |
|  |  |
| & | Bitwise AND |
|  |  |
| | | Bitwise OR |
|  |  |
| ^ | Bitwise XOR |
|  |  |
| ~ | Bitwise NOT |
|  |  |
| << | Left shift |
|  |  |
| >> | Right shift |
|  |  |

This program uses bitwise operations in Python.

* a & b: The "&" operator performs a bitwise AND operation, resulting in the value 9 (binary representation of 25 is 11001 and 45 is 101101, so the AND operation is 100001 which is 9 in decimal).
* a | b: The "|" operator performs a bitwise OR operation, resulting in the value 61 (binary representation of 25 is 11001 and 45 is 101101, so the OR operation is 111101 which is 61 in decimal).
* a ^ b: The "^" operator performs a bitwise XOR operation, resulting in the value 52 (binary representation of 25 is 11001 and 45 is 101101, so the XOR operation is 010100 which is 52 in decimal).
* ~a: The "~" operator performs a bitwise NOT operation, resulting in the value -26 (in binary, the NOT of 11001 is 00110, which is -26 in two's complement).
* a << 2: The "<<" operator performs a bitwise left shift operation, resulting in 100 (the binary representation of 25 is 11001, so the left shift operation is 100100 which is 100 in decimal).
* a >> 2: The ">>" operator performs a bitwise right shift operation, resulting in 6 (the binary representation of 25 is 11001, so the right shift operation is 00110 which is 6 in decimal).

#### Source Code

# Bitwise Operators

"""

& AND

| OR

^ XOR

~ NOT

<< Zero fill left shift

>> Signed right shift

"""

a = 25

b = 45

**print**(a & b)

**print**(a | b)

**print**(a ^ b)

**print**(~a)

**print**(a << 2)

**print**(a >> 2)

#### Output

9

61

52

-26

100

6

### **IF Statement in Python**

The if statement is the most basic of all the control flow statements. It tells your program to execute a certain section of code only if a particular test evaluates to true. The if statement is written with the if keyword.

**Syntax :**  
        if ( condition ) :  
               // body of the statements will execute if the condition is true

This program is a simple implementation of an "if-else" statement in Python to check whether a given number is even or odd.

* n = int(input("Enter The Number : ")) - This line takes the input from the user and converts it to an integer.
* if n % 2 == 0: - The "if" statement checks if the value of n divided by 2 has a remainder of 0. If it's true, the following indented block of code is executed.
* print(n, " is Even Number") - If the condition in the if statement is true, this line will print the message "n is Even Number".

The program checks if the given number is even or odd by checking if it's divisible by 2 (i.e. n % 2 == 0), and if it is, the program outputs the message "n is Even Number".

#### Source Code

# IF Statement in Python

n = int(input("Enter The Number : "))

**if** n % 2 == 0:

**print**(n, " is Even Number")

#### Output

Enter The Number : 34

34 is Even Number

### **IF - Else Statement in Python**

The if-else statement is used to execute both the true part and the false part of a given condition. If the condition is true, the if block code is executed and if the condition is false, the else block code is executed.

**Syntax :**  
        if ( condition ) :  
               // body of the statements will execute if the condition is true  
        else :  
               // body of the statements will execute if the condition is false

This program is a simple Python script that prompts the user to enter their name and age, and then checks if the entered age is greater than or equal to 18. If the age is greater than or equal to 18, the program prints a message stating that the person is eligible to vote, along with their name and age. If the age is less than 18, the program prints a message stating that the person is not eligible to vote, along with their name and age.

#### Source Code

# IF Else Statement in Python

name = input("Enter Your Name : ")

age = int(input("Enter Your Age : "))

**if** age >= 18:

**print**(name, " age is ", age, " Eligible for Vote.")

**else**:

**print**(name, " age is ", age, " Not Eligible for Vote.")

#### Output

Enter Your Name : Ram

Enter Your Age : 23

Ram age is 23 Eligible for Vote.

### **Elif Statement in Python**

The elif condition is used to multiple conditional expressions after the if condition or between the if and else conditions. The elif block is executed if the specified condition evaluates to True.

**Syntax :**  
        if ( condition 1 ) :  
               // body of the statements will execute if the condition1 is true  
        elif ( condition 2 ) :  
               // body of the statements will execute if the condition2 is true  
        .  
        .  
        else :  
               // body of the statements will execute if the condition1 is false condition2 is False

This program is a Python script that prompts the user to enter a number of days and then calculates a fine based on that number.

* It starts by using the input() function to ask the user to enter a number of days, which is stored in the "days" variable.
* Then, the program uses an if-elif block to check the value of the "days" variable against a series of conditions.
* If the value of "days" is equal to 0, the program prints "Good No Fine"
* If the value of "days" is greater than or equal to 1 and less than or equal to 5, the program calculates the fine as 0.5 \* days and prints the fine amount
* If the value of "days" is greater than 5 and less than or equal to 10, the program calculates the fine as 1 \* days and prints the fine amount
* If the value of "days" is greater than 10 and less than or equal to 30, the program calculates the fine as 5 \* days and prints the fine amount
* If none of the above conditions are met, the program will print "Membership Cancel"

So the program is checking the number of days and based on the number of days it is calculating the fine.

#### Source Code

# elif Statement in Python

"""

0 No Fine

1-5 0.5

5-10 1

10-30 5

>30 Membership Cancel

"""

days = int(input("Enter The Days : "))

**if** days == 0:

**print**("Good No Fine")

**elif** days >= 1 **and** days <= 5:

**print**("Fine Amount : ", days \* 0.5)

**elif** days > 5 **and** days <= 10:

**print**("Fine Amount : ", days \* 1)

**elif** days > 10 **and** days <= 30:

**print**("Fine Amount : ", days \* 5)

**else**:

**print**("Membership Cancel")

#### Output

Enter The Days : 5

Fine Amount : 2.5

### **Nested If Statement in Python**

Nested If Statement means to place one If inside another If Statement. Nested ifs are very common in programming. when you nest ifs, the main thing to remember is that an else statement always refers to the nearest if statement that is within the same block as the else and that is not already associated with an else.

**Syntax:**  
    if ( Expression 1 ) :  
        // Executes when the Expression 1 is true  
           if ( Expression 2 ) :  
             // Executes when the Expression 2 is true

This program is a Python script that prompts the user to enter three marks, then calculates the total and average of those marks, and then uses if-else statements to determine the result and grade based on the marks.

* It starts by using the input() function to ask the user to enter three marks, m1, m2, and m3, which are then stored in variables. The program then calculates the total of the marks by adding the three marks and stores it in the variable 'total' and also calculates the average of the marks by dividing total with 3.0 and stores it in the variable 'average'
* Then, the program uses an if-elif block to check the value of the three marks against a series of conditions.
* If all three marks are greater than or equal to 35, the program will print "Result : Pass" and then again check the average of the marks and calculate the grade If average is greater than or equal to 90 and less than or equal to 100, the program will print "Grade : A" If average is greater than or equal to 80 and less than or equal to 89, the program will print "Grade : B" If average is greater than or equal to 70 and less than or equal to 79, the program will print "Grade : C" If none of the above conditions are met, the program will print "Grade : D"
* If any of the three marks is less than 35, the program will print "Result : Fail" and "Grade : No Grade"

So the program is checking the student's three marks, calculating the total and average of the marks, and then determining the result and grade based on the marks and average.

#### Source Code

# Nested If Statement in Python

"""

3 Marks as Input

Total

Average

Result

If Pass Grade

    90-100 A

    80-89 B

    70-79 C

    Else D

"""

m1 = int(input("Enter Mark-1 : "))

m2 = int(input("Enter Mark-2 : "))

m3 = int(input("Enter Mark-3 : "))

total = m1 + m2 + m3

average = total / 3.0

**print**("Total : ", total)

**print**("Average : ", average)

**if** m1 >= 35 **and** m2 >= 35 **and** m3 >= 35:

**print**("Result : Pass")

**if** average >= 90 **and** average <= 100:

**print**("Grade : A")

**elif** average >= 80 **and** average <= 89:

**print**("Grade : B")

**elif** average >= 70 **and** average <= 79:

**print**("Grade : C")

**else**:

**print**("Grade : D")

**else**:

**print**("Result : Fail")

**print**("Grade : No Grade")

#### Output

Enter Mark-1 : 90

Enter Mark-2 : 90

Enter Mark-3 : 90

Total : 270

Average : 90.0

Result : Pass

Grade : A

### **While Loop in Python**

The while loop is repeats a statement or block while its controlling expression is true.The condition can be any Boolean expression. The body of the loop will be executed as long as the conditional expression is true. When condition becomes false, control passes to the next line of code immediately following the loop.

* If the condition to true, the code inside the while loop is executed.
* The condition is evaluated again.
* This process continues until the condition is false.
* When the condition to false, the loop stops.

**Syntax:**  
      while ( Condition ) :  
          // body of statement

The first while loop in the code you provided will continue to execute and print the numbers from 1 to 10. It initializes the variable i to 1 and then uses the while loop to check if i is less than or equal to 10. If the condition is true, it prints the current value of i and then increments the value of i by 1. This process will repeat until i is no longer less than or equal to 10, at which point the while loop will exit.

The second while loop in the code is designed to print the even numbers from 2 to 20. It initializes the variable i to 2, and the variable n to 20. Then, it uses the while loop to check if i is less than or equal to 20. If the condition is true, it prints the current value of i and then increments the value of i by 2. This process will repeat until i is no longer less than or equal to 20, at which point the while loop will exit.

#### Source Code

# While Loop

"""

1.While Loop

2.For Loop

"""

i = 1

**while** i <= 10:

**print**(i)

i += 1

**print**("--------------------")

**print**("Even No : ")

n = 20

i = 2

**while** i <= 20:

**print**(i)

i += 2

#### Output

1

2

3

4

5

6

7

8

9

10

--------------------

Even No :

2

4

6

8

10

12

14

16

18

20