Python Programming

Features of Python: -

Simple:- Python is a simple and small language.

Easy to Learn:- A python program is clearly defined and easily readable. The structure of the program is very simple.

Versatile:- Python supports development of a wide range of applications ranging from simple text processing , world wide web, to games.

Free and Open source:-

Python is an example of an open source software. Therefore any one can freely distribute it and modify it.

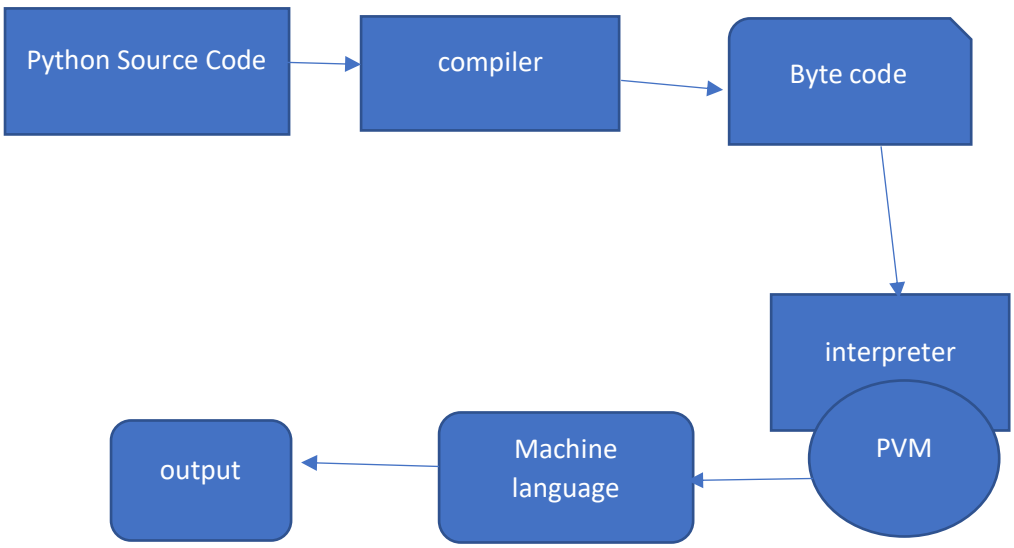
High level language:- Python is a high level language.

Interactive :- Programs in python work in interactive mode which allows interactive testing and debugging piece of code. Programmers can easily interact with the interpreter directly at the python prompt to write their programs.

Portable:- Python is a portable language and hence the programs behave the same on a wide variety of hardware platforms and has the same interface on all platforms. The programs work on all platforms on any operating system.

Object oriented:- Python supports object oriented as well as procedure oriented style of programming. While object oriented technique encapsulates data and functionalities within objects. Procedure oriented technique on the other hand builds the program around procedure or functions which are nothing but reusable pieces of programs.

Interpreted:- python is processed at run time by the interpreter. Basically python converts the source code into an intermediate form called bytecode, which is then translated into native language of your computer so that it can be executed. Byte code makes the python code portable since users just have to copy the code and run it without worrying about compiling , linking and loading processes.



Byte Code:-

Whenever the python scripts compiles , it automatically generated a compiled code called as byte code. The byte code is not actually interpreted to machine code.

The byte code is loaded into python run-time and interpreted by virtual machine , which is a piece of code that reads each instruction in the byte-code and executes whatever the operation is indicated.

Byte code is automatically created in the same directory as .py file. We can find byte code instruction in .pyc file.

Python compiler:- python compiler converts source code into byte code. This means a compiler converts the .py source code into a .pyc byte code for the python virtual machine.

Python Virtual Machine(PVM):- PVM is a program which provides programming environment. The role of PVM is to convert the byte code instructions into machine code so the cpu can execute machine instructions and display the output.

Python virtual machine written in c and when ran compiles into machine language.

Executed bytecode similar to the way a cpu executes machine instructions.

The standard version is called Cpython which is the python when we download from the official site. Cpython has a compile phase. Compiles into byte code. Cpu cant read bytecode. We need an interpreter that is served by python virtual machine.

How to create .pyc file explicitely:-

Python -m prog1.py

Prog1.pyc will form

To run

Python prog1.pyc

Writing python program:-

We can write the program in any editor or notepad and save with .py extension. Or write the program on IDLE to run the programs.

To execute the program from command prompt , type ***python program_name.py***

Literal constants:- For example 1,2.3,'A' "hello" all are literal constants. It is a constant because its value cannot be changed. The 1 always represents itself and nothing else.

Numbers:-

Number like 4 or other whole numbers are referred to as integers. Bigger whole numbers are called Long integers. **There is no limit to the size of an integer that can be represented in python**.

But **floating point** numbers do have a limited range and a limited precision. In python floating point numbers in a range **1.8 X 10^{-308} to 1.8 X 10^{308}** , with 16 to 17 digits of precision. Any number beyond this will return **inf ,** which means infinity.

Numbers of a+bi form (like 3+8i) are complex numbers.

Commas are not allowed in numbers in python.

Floating point numbers are very efficient at handling large numbers, there are some issues while dealing with them like below:-

1. **Arithmetic overflow problem:-**
   When we multiply two very large floating numbers , we may get an arithmetic overflow , means the result may be outside the floating point number range(inf) , which means infinity. The result infinity denotes an arithmetic overflow has occurred.
2. **Arithmetic underflow:-** we can get an arithmetic underflow while doing division of two floating point numbers when a calculated result is too small in magnitude to represent. For example dividing 3.0e-400/5.0e200 , result will be 0.0 which indicates underflow.
3. Loss of precision problem:- When we divide 1/3 we know the results is 0.3333333….. , where 3 is repeated infinitely. Since any floating point has a limited precision and range, the result is an approximation of the true value.
4. Python automatically displays a rounded result to keep the number of digits manageable. For most applications, this slight loss in accuracy is a concern in scientific computing.

Built in format() function:-

Any floating point value may contain an arbitrary number of decimal places. So we can use format() function to produce a number with specific number of decimal places. But note that it will be String format only.

Format() function can also be used to insert a comma in the number as shown below.

format(123456,',')

Operation on Numbers:-

>>> 10+7

17

>>> 50 +40 -55

55

>>> (-30*4)+500

380

Division by zero:-

Dividing a number by zero generates an error and no output is produced

15/0

Traceback (most recent call last):

File "<pyshell#14>", line 1, in <module>

15/0

ZeroDivisionError: division by zero

Dividing two integers:- Quotient and Remainder

When we dividing two numbers and if we want the quotient and remainder then for quotient we have to use // (floor division operator) and to get remainder we have to use modulo (%) operator.

But if we use only / (division operator) , then we get floating point quotient value.

Exponentiation:-

Python also supports ** operator which is used for exponentiation, that is raising of one number to the power of another.

2**3

8

Strings:-

A string is a group of characters.

Using Single Quotes(‘): for example ‘HELLO’.

Using Double Quotes(“): for example “Hello”

Using Triple Quotes(‘’’ ‘’’): We can specify multi line strings using triple quotes.

‘’’ Good morning everyone

Welcome to the python class’’’

Strings are immutable. This means once we create a string , we cannot change it.

String Concatenation:-

Python concatenates two string placed side by side.

Print(“Hello” “…….” “good morning”)

Unicode Strings:-

If we want to write some text other than English, like in hindi, we should prefix the string with ‘U’

For example:-

U” भारत”

Escape Sequences:-

Some characters like “ ‘ cannot be directly included in a string. For that we must place a backslash before them , which is called escape character.

print('what's is your name?')

SyntaxError: invalid syntax

print('what\'s is your name?')

what's is your name?

We can use \n for new line and \t for inserting tab in a string.

Raw Strings:-

A raw string is specified by prefixing r or R to the string. It will consider the string as it is written.

print(R"fsfffsf/fssf'sfsffsf*")

fsfffsf/fssf'sfsffsf*

String Formatting:-

The same format() function can also be used to control display of strings.

Format(value,format_specifier)

Where value is the value or the string to be displayed, and format_specifier can contain a combination of formatting options.

>>>format(‘hello’,’<30’)

‘ hello’

>>>format(‘hello’,’>30’)

‘ hello’

>>>format(‘hello’,’^30’)

‘Hello ‘

Variables and Identifiers:-

Variables are reserved memory locations that stores values. Identifiers are names given to identify something. This can be a variabkle,functions,class,module.

Valid identifier example:

_my_var, num1,r,First etc.

Invalid identifier :-

1num, my-var, %var,Basic sal,H#R&A

Rules:-

1. First character must be an underscore or a letter(upper or lower)
2. The rest of the identifier name can be underscores,letters, or digits.
3. Identifiers names are case-sensitive. Myvar and myvar are not same.
4. Characters such as @,$ and % are not allowed within identifiers.

Data Types:-

The five standard data types supported by python includes- numbers, string,list,tuple,dictionary.

Programs to display data of different types using variables:-

```
a=6
b=123.5
c=a+b
s="hello"


print(a)
print(b)
print(c)
print(s)
print(type(a))
print(type(b))
print(type(c))
print(type(s))
d=complex(3,6)
print(d)
```

Multiple Assignment:-
Python allows programmers to assign a single value to more than one variables simultaneously. For example
a=b=c=1
we can also assign different values to multiple variables simulatanously
sum, a,b,msg= 0,3,5, “hello”
this is same as
sum=0
a=3
b=5
msg=”hello”

Multiple statements on a single line:-
We can write multiple statements on a single line by giving ; operator between statements

```
msg="hello"; hi="hi"
print(msg);print(hi)
print(msg);
print(hi)
```

Boolean:-

Boolean is another data type in python. A variable of Boolean type can have one of the two values- True or False. Boolean variables are also created when we assign a value to them or when we use relational operator on them.

20==30 "python"=="python"
False True

Taking user input:-

```
name=input("Enter your name")
id=int(input("Enter your college id"))
gender=input("Enter F or M")
print("Your name is" + name)
print("Your id is",id)
print("Your gender is"+gender)
```

Note that: in case of id , it is integer, we cannot concatenate string with integer, so in this case instead of '+' we use ','.
By default python input is string to convert it into int we have written
Id=int(input("Enter id"))

Writing Comments in program

We have to # beside the comment for writing comment in our programs.

```
name=input("Enter your name")        #string input
id=int(input("Enter your college id"))      #integer input
gender=input("Enter F or M")
print("Your name is" + name)
print("Your id is",id)
print("Your gender is"+gender)
```

Python Reserved Word:-

and	**assert**	**break**	**class**	**continue**		**def**	**del**	**elif**	**else**	**except**	
exec	**finally**	**for**	**from**	**global**	**if**	**import**	**in**	**is**	**lambda**	**not**	**or**
	pass	**print**	**raise**	**return**	**try**	**while**	**with**	**yield**			

Indentation:-
Indentation are very important in python. In python program indentation means a specific logic for that line.

Operators and Expressions:-
Following are the types of operators:-

1. Arithmetic Operators
2. Unary operator
3. Comparison operator
4. Bitwise operator
5. Assignment Operators
6. Membership Operators
7. Logical operators
8. Identity Operators

1. Arithmetic Operator:-
   + - * / % ** and //
   + for sum
   - For subtract
   *For multiplication
   ** For Exponential
   // For quotient
   / For division(Floor division)
   % For remainder (Modulus)

What is floor value:-

Floor(2)=floor(2.0001)=floor(2.999999)=2

Floor(3)=floor(3.00001)=floor(3.999999)=3

Floor(-3)=floor(-2.99999)=floor(-2.00001)= -3
Floor (-2)=floor(-1.99999)=floor(-1.00001)= -2

Therefore

9/4 = -9/-4=2.25

But 9//4=-9//-4=floor(2.25)=2

-9/ 4=9/-4=-2.25
-9//4=9//-4=floor(-2.25)=-3

Remainder=dividend-divisor*floor(quotient)

6.3//1.5=-6.3//-1.5=floor(4.2)=4

-6.3%-1.5=-6.3-(-1.5)*4.0=-6.3+6.0=-0.3

Comparison Operator:-

Operator	Description
==	Returns true if two values are same
!=	Returns true if two values are not same
>	Returns true if value at the operand on left side of operator is greater than value at right side
<	Returns true if the value at the operand on the right side of the operator is greater than the value on its left side.
>=	Returns true if the value at the operand on the left side of the operator is either equal or greater on its right side

<=	**Return true if the value at the operand on the right side of the operator is either equal or greater than the value on its left side**

Assignment Operator and in-place operator

Operator	Description
=	Assign value to the operand
+=	Add and assign
-=	Subtract and assign
*=	Multiply and assign
/=	Divide and assign
%=	Modulus and assign
//=	Floor division and assign
**	Exponent and assign

The in place operator can be used on other data types like string also.
Str1=”hello ”
Str2=”welcome”
Str1+=Str2
Print(Str1)
Hello welcome

Unary Operator:-

Unary operator operates on single operand
a=10
b=-a
here – is unary operator as it acts on single operand.

Bitwise Operators:-
Bitwise operator perform operations at the bit level. Bitwise AND, bitwise OR, bitwise Xor, shift operator are some examples.

Bitwise AND (&)
10101010 & 01010101=00000000
Bitwise OR(|)
10101010 |01010101=11111111

Bitwise XOR(^)
10101010 ^ 01010101=11111111

Bitwise NOT (~)
~10=01

Shift operator
There are left shift(<<) and right Shift (>>). These operations are used to shift bits to the left or to the right.

X=0001 1101
X<<1 (x is shifted bit by one place towards left)
0011 1010

X=0001 1101

X<<4
1010 0000

Logical Operator:-

1. **Logical AND (&&)**
2. **Logical OR(||)**
3. **Logical NOT(!)**

1. **Logical AND:-**
   Evaluate two conditions simultaneously with relational operator. If both left and right side expression are true, then the whole expression is true.
2. **Logical OR:-**
   Logical or operator is used simulatenously evaluate two conditions. If one or both the expressions of the logical operator is true, then whole expression is true.
3. Logical Not(!)
   It takes a single expression and negates the value of the expression.
   a=10
   b=!a
   Therefore b=0

Membership Operator:-
Python supports two types of membership operators-in and not in. Membership operator is used in a sequences such as strings,lists, or tuples.
In operator:- The operator returns true if a variable is found in the specified sequences. And false otherwise.
Not in operator:- The operator returns true if a variable is not found in the specified sequences and false otherwise.

Identity Operators:-
Is Operator:- Returns true if operands or values on both sides of the operator point to the same object and false otherwise.
Is not operator:- Returns true if operands or values on both sides of the operator does not point to the same object otherwise false.

Operator Precedence Chart:-

Operator	Description
**	Exponentiation
~, +, -	Complement, unary plus (positive), and minus (negative)
*, /, %, //	Multiply, divide, modulo, and floor division
+, -	Addition and subtraction
>>, <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<=, <, >, >=	Comparison operators
<>, ==, !=	Equality operators
=, %=, /=, //=, -=, +=, *=, **=	Assignment operators
is, is not	Identity operators
in, not in	Membership operators
not, or, and	Logical operators

Functions for type conversion

Function	Description
`int(x)`	Converts x to an integer
`long(x)`	Converts x to a long integer
`float(x)`	Converts x to a floating point number
`str(x)`	Converts x to a string
`tuple(x)`	Converts x to a tuple
`list(x)`	Converts x to a list
`set(x)`	Converts x to a set
`ord(x)`	Converts a single character to its integer value
`oct(x)`	Converts an integer to an octal string
`hex(x)`	Converts an integer to a hexadecimal string
`chr(x)`	Converts an integer to a character
`unichr(x)`	Converts an integer to a Unicode character
`dict(x)`	Creates a dictionary if x forms a (key-value) pair

1. Write a program to enter two integers and then perform all arithmetic operations on them.
2. Repeat the program in Question 1 using floating point numbers.
3. Write a program to perform string concatenation.
4. Write a program to demonstrate printing a string within single quotes, double quotes and triple quotes.
5. Write a program to print the ASCII value of a character. Use ord() function
6. Write a program to print character for an ASCII value. Use chr() function
7. Write a program to take the character in uppercase and convert it into lowercase. A=65 a=97
8. Write a program to swap the contents of two variables.
9. Write a program to read two floating point numbers. Add these numbers and assign the result to an integer and finally display all three variables.
10. Write a program that prompts user to enter two integers x and y. The program then calculates and displays X^y.
11. Write program that prompts user to enter his first and last name and then displays a message " Greetings !!! First name Last name".