PYTHON BASIC

What is Python Programming ?

Python is an Interpreted,Object oriented,High-level Programming language with dynamic sematics.

Features:-

- Easy to learn
- Open source
- Broad standard library
- Cross Platform
- Work on Interpreter logic
- Multi language Paradigm(Structural,OOPS,Simple Way)
- High-level Programming
- Extendable Language
- Expressive Programming
- GUI Programming Support

Application:-

- Network Programming
- Data Analysis
- Robotics
- Website and Application Development
- Desktop Application
- Games Development
- Web Scraping
- Data Visualization
- Scientific Calculation(Numpy, ML)
- Machine Learning & Artificial Intelligence
- 3D Application Development
- Audio & Videos S/W Development

Create First Programm:-

CODE:-

```
print('Hello World')
```

ANS:-

```
Hello World
```

Python print() function:-

- The print() function in Python is used to print a specified message on the screen. The print command in Python prints strings or objects which are converted to a string while printing on a screen.
- Suntax;-

```
print(Objects)
```

How to print blank lines:-

- Sometimes you need to print one blank line in your Python program. Following is an example to perform this task using Python print format.
- Syntax:-

```
print(4*\n")
```

Print end command:-

- By default, print function in Python ends with a newline. This function comes with a parameter called 'end.' The default value of this parameter is '\n,' i.e., the new line character. You can end a print statement with any character or string using this parameter. This is available in only in Python 3+.
- Eg:-

```
print("Welcome to ",end='')
print("School",end=!)
```

- Eg:-

```
print("Python",end='@')
```

Keywords:

- Keywords are reserved words which are used in building Python.
- These Keywords has special purpose and special syntax.
- Eg: def, class, if, while, else....

Syntax:

```
Import keyword
k=keyword.kwlist
Print(k)
Len(k)
```

Identifier:

- It identifies an object uniquely.

```
Ord('a')-97
Ord('A')-65
```

```
Chr('65')-A
Chr('97')-a
```

- Identifiers are Case Sensitive.
- Aplhanumeric Eg: Aj10=10
- Never start with number.
- Number Character of identifiers is unlimited.
- No special character in idenitifiers except underscore.

Eg: alp_23

Variable:

- A Python variable is a reserved memory location to store values. In other words, a variable in a python program gives data to the computer for processing.

Eg: a=10, Here 'a' is variable

- There are two types of variables in Python, Global variable and Local variable. When you want to use the same variable for rest of your program or module you declare it as a global variable, while if you want to use the variable in a specific function or method, you use a local variable while Python variable declaration.

Data Type:

FUNDAMENTAL	SEQUENTIAL
INTEGER	LISST
FLOAT	SET

Python Escape Character Sequences:

Escape characters or sequences are illegal characters for Python and never get printed as part of the output. When backslash is used in Python programming, it allows the program to escape the next characters.

Syntax:

\Escape character

Explanation:

Here, the escape character could be t, n, e, or backslash itself.

Types of Escape Sequence

Escape characters can be classified as non-printable characters when backslash precedes them. The print statements do not print escape characters.

Here is a list of Escape Characters

Code	Description
\'	Single quotation
\\	Backslash
\n	New Line
\r	Carriage return
\t	Tab
\b	Backspace
\f	Form feed
\ooo	Octal equivalent
\xhhh	Hexadecimal equivalent

Example Usage of Various Escape Characters

Escape character	Function	Example Code	Result
\n	The new line character helps the programmer to insert a new line before or after a string.	txt = "Guru \n99!" print(txt)	Guru99
\\	This escape sequence allows the programmer to insert a backslash into the Python output.	txt = "Guru \\99!" print(txt)	Guru \99!
\xhh	Use a backslash followed by a hexadecimal number. This is done by printing in backslash with the hexadecimal equivalent in double quotes.	txt = "\x47 \x75\x72 \x75" + "99!" print(txt)	Guru99!
\ooo	To get the integer value of an octal value, provide a backslash followed by ooo or octal number in double-quotes. It is done by printing in a backslash with three octal equivalents in double quotes.	txt = '\107 \125\122 \125'+ "99!" print(txt)	GURU99!
\b	This escape sequence provides backspace to the Python string. It is inserted by adding a backslash followed by "b". "b" here represents backslash.	txt = "Guru \b99!" print(txt)	Guru99!
\f	It helps in the interpolation of literal strings	txt = "Guru \f99!" print(txt)	Guru99!
\r	It helps you to create a raw string	txt = "Guru \r99!" print(txt)	Guru99!
\'	It helps you to add a single quotation to string	txt = "Guru \'99!" print(txt)	Guru'99!

What Does "\t" Do in Python?

The t alphabet in Python represents a space. It enables you to insert space or tab between strings in a code. It helps us to have space in the Python program when there is a need for it. To eliminate the usage of keyboard space, the coders utilize tab escape sequences.

Following is the syntax for a tab escape sequence.

Syntax:

"\t"

Example:

In this example, the string used is "Guru99". The program will put a tab or a space

Data Type:

FUNDAMENTAL	SEQUENTIAL
INTEGER	LISST
FLOAT	SET
COMPLEX	DICTIONARY
BOOLEAN	
STRING	

- Fundamental data types are immutable.
- Variables holding same value reffers to same memory location.
- Id() is used address of memory location where value is stored.
- HETEROGENEOU VALUE: Values with different data types.

► Sequential data type can hold multiple heterogeneous value.
► These values are stored in series of internal memory location.
► These internal memory location are identified with concept indexing.
Eg: x[2]

Fundamental Data Type:

INTEGER:

- +ve/-ve/0 numbers
  Decimal
  Octal
  Hexadecimal
  Binary
- int() is used to convert any form

a. Decimal:

- Default form of numbering which can be used in our life.
- Numbers: 0,1,2,3,4,5,6,7,8,9
- Print() is used to convert in decimal form.
- Eg: 99,12,20

b. Octal:

- Numbers which is represented with 8 digits starting 0 to 8.
- Number: 0,1,2,3,4,5,6,7,8
- Prefix: 0o/0O
- Eg: 0o45/0O56
- oct() is used to convert any form to octal form.

c. Hexadecimal:

- Numbers which is represented with 16 digits.
- Numbers: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- Prefix: 0x/0X
- Eg: 0x23/0X54
- hex() is used to convert any form to hexadecimal form.

d. Binary:

- Number combination of 0 and 1.
- Number: 0,1
- Prefix: 0b/0B
- Eg: 0b0110,0b0011
- bin() is used to convert any form to binary form.

FLOATS:

- An integer with decimal point.
- Float is used to get the accurate value using decimal point.
- It can generate infinite number of values between any two number.
- It can extract lower and higher value between any two n umber.
- float() is used to convert to float form.

BOOLEAN:

- True/False[1/0]
- bool() is used to convert to Boolean form.
  Numeric Value:
  - True: All non-zero number is consider as True.
  - False: Number is equal to zero. Eg: (0000)

  Strings:
  - True: All non-empty string is considered as True.
  - False: Empty string is considered as False. Eg: ("")

COMPLEX:

- Eg: 10+20j
  10=Real Value
  20=Imaginary Value
  j=Complex
- complex() is used to convert any form to complex form.

STRING:

- String is a collection of one or more characters put in a single quote, double quotes or triple quotes.
- Multiple line strings can be denoted as triple quotes.
- Eg: "Hello Good Morning"
- Str() is used to convert any form to string form.

a. String Indexing:

Python index() is an inbuilt function in Python, which searches for a given element from the start of the list and returns the index of the first occurrence.

b. String Slicing:

Syntax:
"\t"

Example:
In this example, the string used is "Guru99". The program will put a tab or a space between Guru and 99.
Python Code:
```
TextExample="Guru\t99"
print (TextExample)
```
Output:
```
Guru 99
```
Explanation:
In the above example, instead of adding space using a keyboard, the program helps us by putting a space or a tab between the string "Guru99". It also provides a space at the precise location where the escape sequence is added.

When to use "\t" in Python?

The escape sequence tab is utilized to put a horizontal tab between words and hence helps manipulate python strings. However, If the escape sequence tab is not used, the programmer has to manually add a space between every word of the string.
You can transform it into a time-consuming exercise. Moreover, the space added between different keywords may or may not be precise in its placement.
Here is an example that displays the manual addition of a space between words and the use of an escape sequence between words.
Python Code:
```
print("Manually Added  space in string Guru   99")
TextExample="Use\tof\ttab\tto\tadd\tspace\tGuru\t99"
print(TextExample)
```
Output:
```
Manually Added space in string Guru   99
Use     of      tab     to      add     space  Guru    99
```
Explanation:
The programmer manually added space between words in the above code, so the placement was not precise. When the escape sequence tab was applied, the program automatically provided the precise location of space between words.

Application of built-in function Chr () and Ord ()

The Chr () function is a built function that takes a single argument as input. The function takes Unicode characters as an input which ranges from 0 to 1,114 and 111, respectively. The function can be used as the substitute for escape sequence "\t" to put a space between two words.
The syntax for the Chr function is represented below: –
Syntax: –
```
Chr(Unicode character)
```
The tab has the Unicode character 9. Use the following Python command to arrive at the Unicode character as shown below: –
Python Code:
```
print("Unicode character of the tab is")
Ord=ord('\t')
print(Ord)
```
Output:
```
Unicode character of the tab is
9
```
Explanation:
The above code provides the Unicode character for the tab. It can be used as an input for the Chr function. Use of Chr (9) would allow us to create a substitute for a tab escape sequence.
This code is an example of how to use Chr (9), as shown below:
Python Code:
```
TextExample="Guru+chr(9)+99"
print(TextExample)
```
Output:
```
Guru    99
```

Type Casting:

- Convert one data type to another data type.

Type Casting	Integer	Float	Complex	Boolean	String
Integer	YES	YES	YES	YES	PARTIALLY
Float	YES	YES	YES	YES	PARTIALLY
Complex	YES	YES	YES	YES	PARTIALLY
Boolean	YES	YES	YES	YES	YES
String	YES	YES	YES	YES	YES

- Python slicing is about obtaining a sub-string from the given string by slicing it respectively from start to end.
- [<Start>:<End>:<Step>]
  - All three are optional.
  - Step is very important, it is decide the direction.
  - It Defines how to display extracted value using start and end.
  - Default value of step is 1.
  - Direction - Left to Right
  - Starting Index is 0.
  - Starting Index < Ending Index (Positive)
  - For Negaive Step:
    - Direction: Right to Left
    - Starting Index: -1
    - Ending Index: -len
    - Staring Index > Ending Index

c. String Function:

a. lower(): All letter converts to lower case.(abc)
b. upper(): All letter converts to upper case.(ABC)
c. title(): First letters of strings convert to upper case.(Apple Banana)
d. capitalize(): First letter of string convert to upper case.(Apple banana cat)
e. swapcase(): It converts the string according to their original string.
f. center(): How much value we give to this function it take that much space of the value of the string and the value is in center. According to the value it take the space form both side.
g. ljust(): It take the value of string to left side and other are space.
h. rjust(): It take the value of string to right side and other are space.
i. strip():

- **Python strip()** function is a part of built-in functions available in the Python library. The strip() method removes given characters from start and end of the original string. By default, strip() function removes white spaces from start and end of the string and returns the same string without white spaces.

j. lstrip():
k. rstrip():
l. find():

- **Python String find()** is a function available in Python library to find the index of the first occurrence of a substring from the given string. The string find() function will return -1 instead of throwing an exception, if the specified substring is not present in the given string.

m. split():

- The split function helps in string manipulation in Python. It allows you to split up a string into different substrings. It returns a list of words present in a line or a string and is separated by a delimiter string.

n. rsplit():
o. lsplit():
p.PARTITION():

- It perform only one split and return 3 values.
- 1st value- Delimiter
- 2nd value- Left side of the delimiter
- 3rd value- Right side od the delimiter
- Delimiter is there.
- It come in Tuple format

- S=ORANGE BANANA APPLE GRAPES APPLES KIWI
- S.split("APPLE**")**
- S.partition("APPLE")- ORANGE BANANA 'APPLE' GRAPES APPLE KIWI

q. REPLACE():

- It is used to alter a part of string.
- This is not a permanent change to orginal string.It creates an instance and stores
  - Str.replace()

r. JOIN():

- It converts a list of values into a String based on delimiter.
- Input into join function should be a sequential data type.
- "<delimiter>".join(<Sequential Data type input>)
- Eg:- ["Blue","Green","Yellow"]
- "-"join["Blue","Green","Yellow"]
- Ans:- Blue-Green-Yellow

s. is():

❖ All the Function is written like: Variable name.Function Name

Operators in Python:

1.Arithimatic Operators
2.Assignment Operators
3.Comparison Operators
4.Logical Operators
5.Identify Operators
6.Membership Operators
7.Bitwise Operators

1.Arithimatic Operators:

+	Addition a+b
-	Substraction a-b
*	Multiplication a*b
/	Division a/b
%	Modules a%b
**	Exponent a**Any Number(2/3/4)
//	Floor Division a//Any Number(2/3/4)

2.Assignment Operators:

=	a=5
+=	a+=5/a=a+5
-=	a-5/a=a-5

3.Comparison Operators:

==	Equal	x==y
!=	Not Equal	x!=y
>	Greater Than	x>y
<	Less Than	x<y
>=	Greater than equal	x>=y
<=	Less than equal	x<=y

4.Logical Operators:

and	Returns True if both statements are True	x<5 and x<10
or	Returns True if one of the statement is True	x<5 or x<4
not	Returns result, returns False if the result is True	Not(x<5 and x<4)

5.Identify Operators:

in	Returns True if a sequence with the specified value is present in the object	x in y
Not in	Returns True if a sequence with the specified value is not present in the object	x not in y

6.Membership Operators:

is	Returns True if both variables are the same objects	x is y
is not	Returns True if both variables are not the same object	x is not y

7.Bitwise Operators:

&	AND	x&y
|	OR	x|y
^	XOR	x^y
~	NOT	~x
>>	Right Shift	x>>
<<	Left Shift	x<<

- In Python bitwise operators are used to perform bitwise calculation on integers. The integers are 1st 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.
- It works only on Integer.
- RightShift Bitwise Operator:

  Removes plays holder from right side of original value.

  Resultant is always less than original value.
- LeftShift Bitwise Operator is the opposite of Right Shift Bitwise Operator.
- Bitwise Not/Compliment:

  It is a uninary value.

  Two steps process

  One's compliment- add one to original value

  Two's Compliment- reverse the sign

INPUT:

- int() is used.
- It is used to take value from the user.