Python basics & fundamentals

***Intro:**

1.) Created by Guido van Rossum in 1991.

2.) As general purpose prg lang.

3.) Interactive & interpreted language.

4.) structured & OOP.

5.) Open-source lang under general public license(GPL).

6.) With libraries & built-in functions.

7.) Have simple syntax & easy to learn.

***Following is some of the important features of Python:**

1.)open source: The Python implementation is under an open-source license that makes it freely usable and distributable, even for commercial use.

2.)interpreted: Python is a high-level language which is interpreted by a Python interpreter.

3.)cross platform compatible: Python can be executed on all major platforms like Windows, Linux/Unix, OS/2, Mac and others.

4.)Object-Oriented: In Python, we encapsulate data within the objects as it supports the object-oriented style of programming.

5.)a great choice for new learners: Python is easy to learn and follows a simple syntax, so it is a good choice for beginner programmers. Python also supports wide range of application development.

6.)extensible: Python has a wide range of libraries and built-in functions which helps in easy and rapid development of applications.

7.)interactive: Python users are provided a command prompt where they can interact directly with the interpreter to write programs.

8.)Database connectivity: Python provides interfaces required to connect to all major databases like Oracle, MySQL, PostgreSQL and others.

***Identifiers:** In Python, variables, functions, classes, modules and objects are identified using a name known as an identifier. An identifier can start with an uppercase or lowercase character or an underscore (_) followed by any number of underscores, letters and digits. All identifiers in Python are case sensitive. Example: weight=10

***Keywords:** Keywords are the reserved words in Python. So keywords cannot be used as identifiers or to name variables and functions. Few of the keywords are listed below. Example: if, else, elif, for, where, break, continue

<u>*Declaring a variable:</u>

<u>Syntax:</u> var_name = literal_value

- where var_name is the name given to the container holding the value specified as literal_value in the syntax above. Example: weight=10
- In the above example, weight is the container holding the value 10 which can change during the execution of the program.

- Python may have data belonging to different types. Common data types used in programming are listed below:

-><u>Category:</u> Numeric, Numeric with decimal point, Alphanumeric, Boolean.

-><u>Datatype:</u> int & long, float & double, char & string, Boolean.

-><u>Example:</u> int(123)&long(123456789), float(12.34)&double(12312.23232), char(A)&string(hello), Boolean (true , false).

<u>*Python is a dynamically typed language:</u>

In the above example, no datatype was mentioned at the time of declaring variable. In Python, the datatype of a variable is decided automatically at the time of execution based on the value assigned to it. This is called as dynamic typing.

```
->num=65 #line 1
  num="A" #line 2
```

In line 1, variable num is assigned a value 65 which is an integer, so the data type of num variable is integer in line 1. In line 2, variable num is assigned a value “A” which is a string, so the data type of the num variable is string in line 1.

Note: To check the datatype of the variable we can use type(var_name) which in turn returns the datatype of the variable.

<u>Example:</u>
```
num=65
print(num,type(num))
num="A"
print(num,type(num))
```

<u>Output:</u>
```
65 <class 'int'>
A <class 'str'>
```

<u>*The input() function:</u> Python provides the input() built-in function to read an input from the user using the standard input device (i.e. keyboard). The input() function always returns string data irrespective of the type of data entered through the keyboard.

<u>Syntax:</u> var_name = input([“interactive statement”])

- where,var_name is the variable assigned with the string value which is read using input method.
- Interactive statement is the statement displayed to the user expecting the response from them.

Example: input_var=input("please enter the value")

print(input_var)

Output: please enter the value100

100

***The print() function:** Python provides the print() built-in function to display the output onto the standard output device (i.e., Monitor).

Syntax: print("var_name1, var_name2, …", [end="value1", sep="value2"])

- where, var_name1, var_name2 are the variable names or the literals you want to print or output.
- end is used to specify the separator between two print statements which is '\n' by default; sep is used to specify the separator between multiple variables displayed using a single print statement

Example: a="infy"

b=20.127

print(a,b,c)

print(a,b,c,sep=":")

print(a,b,c,end=" ")

Output: infy 20.127 10

infy:20.127:10 #seperator between variables changed to ':'

infy 20.127 10 infy 20.127 10 #seperator between two print statement changed to " "

***Arithmetic operators:**

Operator	Explanation	Example
+	Used for addition operation.	1+2 = 3
-	Used for subtraction operation.	2-1 = 1
*	Used for multiplication operation.	2*5 = 10
/	Used for division operation.	11/2 = 5.5
//	Used for int division operation.	11//2 = 5
%	Used for modulo operation.	11%2 = 1 (returns remainder)

***Relational operators:**

Operator	Explanation	Example
==	Checking equality of 2 nums.	10==10 -> True
!=	Checking inequality of 2 nums.	10!=10 -> False
>	Check num1>num2	2>5 = False
<	Check num1<num2	3>2 = True
>=	Check num1 greater than equal too num2.	10>=10 -> True
<=	Check num1 less than equal too num2.	100<=10 -> False

***Assignment operators:**

Operator	Explanation	Example
=	Assigns value.	num =2
+=	Short hand operator for addition.	num+=1 is num=num+1
-=	Short hand operator for subtraction.	Num-=1 is num=num-1
*=	Short hand operator for multiplication.	Num*=1 is num=num*1
/=	Short hand operator for division.	Num/=1 is num=num/1
%=	Short hand operator for modulo.	Num%=1 is num=num%1

***Logical Operators:** These operators are used to combine one or more relational expressions.

Operators	Description
AND	Result will be true, if both the expressions are true. If any one or both the expressions are false, the result will be false
OR	Result will be true, even if one of the expression is true. If both the expressions are false, the result will be false
NOT	If the expression is true, result will be false and vice versa

If A and B are two relational expressions, say A = (Num1>2000), B= (Num2>100), the result of combining A and B using logical operator is based on the result of A and B as shown below:

A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

A	B	A or B
True	True	True
True	False	True
False	True	True
False	False	False

A	not A
True	False
False	True

Comments: Comments are the lines which are skipped during execution of a program. There are two types of comments available in Python:

- First one is **single line comment** which starts with '#' symbol and extends till the end of line. Comments can start from the beginning of the line and middle of the line, but it should not be a part of string literal.
- Second one is **multi line comment** which starts with ''' or """ and ends with ''' or """ respectively. This type of comment is mainly used for documentation purpose.