# Python

Python is a high level, interpreted & object oriented scripting language(backend).

## What is the difference b/w Compiler & Interpreter:

1. Compiler scans the source code in one go where as Interpreter scans each line at a time.
2. Compiler generates intermediate machine code first & then generated output whereas Interpreter doesn’t generate intermediate machine code, it directly generates output.
3. The compiler is used by programming languages such as C, C++, Java, etc., whereas An interpreter is used by programming languages such as Python, PHP, Perl, Ruby etc,

## Note:-

* Floor Division -- // it gives only integer output excludes the decimal part.

Ex:- 5 //2 ---output -----2

* Remainder --- % it gives the remainder

Ex:- 5 % 2 –output -------1

* Exponential calculation : \*\* means it performs the power of operation

Ex: 2 \*\*3 -----output ----8 (2^3)

## *Data types in Python:-*

* Integer
* Float
* String

2+3 =5---here 5 is an integer

5/2 = 2.5 – Here 5 is floating value

String is combination of characters. (must be in quotes either single or double.)

## *Variables in Python*

* Integer variables
* Float variables
* String variables

## *Integer variables example:-*

Assign x = int (x = 5) ----Here 5 is an integer & assigned to variable ‘x’

Likewise

We can assign integers to different variables & perform BODMAS operations also

y = 6

x+y = 11 ---11 is output here.

x/y = 2.5 & so on…

We can change/reassign the variable value

Example :- x= 9

Previously x=5 but now x=9

## *Note :*

Likewise we can assign float values to the variables & change them easily.

## *String Variables :*

Like integers we can assign string to a variable.

x = ‘shilpa’

y = ‘ Asha’

Here we can only perform addition of these.. Which means it combines both the string values.

x + y = ‘shilpaAsha’

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

s h i l p a

0 1 2 3 4 5

If we want to print characters of the variable from forward –it starts from 0

From backward it starts from -1

## *Examples:-*

X[from:to]---- give index values in the place from & to

X [0:2] ----Here 2 value is excluded. Prints till 2 but not 2

Output------‘sh’

X [0: ]

Output--- ‘shilpa’ -----Starts from 0 it takes everything…

X[ :4]

Output : ‘shil’ ---Starts from the beginning if we don’t mention

X [2: 10]

Output : ‘ilpa’ -----It doesn’t show error..It takes till the values are there

## *Note:*

String variable is immutable. We can’t change the characters in the string once assigned but we can change the assigned string completely.

Example : x =’Advaitha’ --Now x = ‘Advaitha’ previously it was ‘shilpa’

But we can’t do this as

X [0] = ‘k’

Output -- Error 'str' object does not support item assignment

X [0:3] = ‘Ram’ -----Error

## len( )

len( ) is in-built function in Python.

We can either directly give string name (or) we can give variable name

len ( variable name) ==count the characters &gives output

Ex: name = ‘Advaitha’

len (name) ----- Output ---8

len (‘Advaitha’)------- Output ---8

## Lists

A list can accept anything integer, float, string.

Represented with square brackets ---[ ]

It is Mutable. –Means we can change the values. (Using index values)

## *Examples:*

* Numbers = [2,45,32,56]
* Names = [‘shilpa’, ‘asha’, ‘pranaya’, ‘Advaitha’]
* Mis = [2.5, ‘Ramisetty’, 34,’Rama’]

We can perform operations on these variable Lists.

## *In-built operations:*

For example take a variable as Data =[‘shilpa’, 3.5, ‘Advaitha’, 25]

Rev: -4 -3 -2 -1

Shilpa 3.5 Advaitha 25

Index: 0 1 2 3

Data.append(‘Asha’) ------It adds in the last place

Data.insert(index,value) -------Ex:- data.insert(0, 50)------It adds 50 in the first place

Data.extend([values differentiated by coma]) -----data.extend([‘Pranaya’, 75])

It adds these values at the last place.

Data.remove(value)-----data.remove(75)----It deletes 75 from the list

Data.reverse()-----It shows the data in reverse order

Data.pop()------LIFO-----data.pop()-----removes lastly inserted value

(Or)

We can mention index value in pop

Data.pop(index value)-----data.pop(0)-----it deletes the first value.

## *Note:-*

If the list consists only integers & float values we can sort it using-----data.sort()

It shows in ascending order.

We can clear the list completely using ---nums.clear()

It prints-----[]

## Tuple

Tuple is same as list but represented with ()

We can identify a tuple with this brackets only.

Tuples are more memory efficient. The immutability is considered as the identifying feature of Tuples.

Example :

Tup = (23,34,56,56)

## *Difference b/w List, Tuple & set:*

List is Mutable….. we can change the assigned values using index values

Tuple is Immutable……..Can’t be changed once assigned. ---we can’t change the values using index numbers.

## Set { }

set is an unordered collection of distinct immutable objects.-----set is represented by { }

Set doesn’t allow duplicates as well.

Set = {23,45,56}

## Dictionaries

It is a key:value pair. It is mutable. We can change the values using keys.

Represented with { }

Dictionaries can be nested or we can give lists in the dictionaries etc.,

Examples:--

Di ={‘Shilpa’ : ‘Java’, ‘Asha’:’C’, ‘Advaitha’ : ‘Perl’}

Dictionaries have many in built functions. Like

di.get(‘key\_name’) ----results a value name that the key is representing.

di.copy() ----returns the dictionary copy.

di.keys()------returns all the keys in the dictionary

di.values()------returns all the values in the dictionary.

di.pop(key\_name)-------removes the key:value pair

di.popitem()------removes the last inserted one.

di.clear------clears the dictionary.

## Data types

* None
* Numeric
  + - 1. Int 5
      2. Float 5.6
      3. Boolean (True/False)---True-1 & False-0
      4. Complex 5+6j
* Sequence
  + - 1. Lists
      2. Tuple
      3. Set
      4. String
      5. Range
* Dictionary