**Python**

Ex:- To Print "HelloWorld"

Java:-

----

public class HelloWorld

{

p s v main{String[] args}

{

SOP("Hello World")

}

}

C:-

--

#include<stdio.h>

void main()

{

print("Hello World")

}

Python:-

print("Hello World")

The name Python was selected from the Tv show "The complete Monty Python's Circus",Whuch was broadcasted in BBC 1969 to 1974

Where we can use Python:-

1.For Developing Desktop Application.

2.For Developing Web Application.

3.For Deceloping Database Application.

4.For Network Programming.

5.For developing Games.

6.For Data Analysis Applications.

7.For Machine Learning.

8.For IOT.

9.For Developing AI Applications.

Limitation of Python:-

1.Not using For Mobile applications.

2.Not using For Enterpris app,Backing sector.

Features Of Python:-

1.Simple and easy to learn

2.Freeware and Opensource

3.High Level Programming language

4.Platform Independent.

5.Portability.

6.Dynamically Typed

7.Both POPs(Procedure Oriented) and OOPs(Object Oriented).

8.Extensive Library.

Flavors Of python:-

1.CPython:- Python to C

2.Jython or JPython :- Python to Java

3.IronPython:- Python to .net

4.RubyPython :- Python to Ruby

Python Version:-

Python 1.0V introduced in Jan 1994

Python 2.0V introduced in Oct 2000

Python 3,0V introduced in Dec 2008

python 3.10.1 Python 2.9.13

Identifiers

A name in Python Program is called Identifiers.

It can be Variable Name or function name or module name or class name.

Ex:-

variable\_name:- a = 10

Rules to define identifiers in Python:-

1.The only allowed Characters in python are.

a.alphabets symbol(either lower case and upper case)

b.digit(0 to 9)

c.underscore symbol(\_)

Ex:-

>>> Bank=9

>>> bank

Traceback (most recent call last):

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

bank

NameError: name 'bank' is not defined

>>> bank\_ui =9

>>> 78bank = 78

SyntaxError: invalid syntax

>>> bank78 =90

>>> bank&tu=89

SyntaxError: cannot assign to operator

>>>

2.Identifiers should not starts with digit.

-> 123total ----------wrong

-> total123 ----------right

3.Identifiers are case sensitive ,of couse python language is case sensitive language.

ex:-

>>> total = 10

>>> print(TOTAL)

Traceback (most recent call last):

File "<pyshell#8>", line 1, in <module>

print(TOTAL)

NameError: name 'TOTAL' is not defined

>>> print(total)

10

>>>

Identifiers Points:-

1.Alphabets symbols(both upper and lower)

>>> Total = 10

>>> total = 10

2.Only (\_) underscore symbol is allowed.if identifiers is start with underscore(\_) then it's private or other mode.

>>> abc\_abc = 10---normal

>>> \_abc = 10-----protected

>>> \_\_abc = 10-----private

3.Identifiers should not start with digits.

>>> 78bank = 78

SyntaxError: invalid syntax

4.Identifiers are case sensitive.

>>> total = 10

>>> print(TOTAL)

Traceback (most recent call last):

File "<pyshell#8>", line 1, in <module>

print(TOTAL)

NameError: name 'TOTAL' is not defined

5.we can not use keyword or reserved word as Identifiers.-----

6.there is no limit for python Identifiers.But not recommended to use too lengthy Identifiers.

7.except \_ underscore symbol,other symbol are not alloweed.

bank&tu=89

SyntaxError: cannot assign to operator

Note:-

1. Normal variable => x = 10

2.if Identifiers start with single(\_) then it indicates that it is protected variable.

\_x = 10

3.If indetifiers start with double(\_) then it indicates that it is Private variable.

\_\_x = 10

Casing:-

Casing are two types snake casing and camel casing.

Camel Casing

First leter of word is Uppercase and rest of the letters lower case and here no symbols are allowed,if we define two words in one identifiers then we write First letter of both wors are in Upper Case.

ex:-

BankReciverName = "Santhosh Deshmukh"

Snake Casing:-

Here we can not use upper case,and only(\_) underscore is allowed.

ex:-

bank\_reciver\_name ="Santhosh Deshmukh"

Reserved Words/Keywords

In python some word are reserved to reprsent some meaning of functionality.Such type of words are called Reseved words.

There 35 reserved words available in python.

>>> import keyword

>>> keyword.kwlist

['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']

Data Types

----------

Data type represent the type of data present inside a variable.

In python we are not reqyuired to specify the type explicitly.Based on vlue provided the type will be assigned automatically.Hence python is Dynamically Typed lang.

There are 14 types od Data Present in Python.

1.Integer(int)

2.Float(float)

3.Complex(complex)

4.Boolean(bool)

5.String(str)

6.Bytes(bytes)

7.Bytearray(bytearray)

8.Range(range)

9.List(list)

10.Tuple(tuple)

11.Set(set)

12.Frozenset(frozenset)

13.Dictionary(dict)

14.None(None)

Garbage Collection :-

Usually Programmer taking very much care while creating object,but negleeting destruction of useless object.Because og his negletance,total memory can be filled with useless object which creates memory problem and total application will be down

with out of memory error.

But in python,We have some assitance which is always running in the background to destory useless object,Because this assistance the chance of falling python program with memory problem is very less.The assistance is nothing but Garbage Collection.

ex:-

>>> a = 10

>>> b = 10

>>> a

10

>>> b

10

>>> id(10)

2136897448528

>>> id(a)

2136897448528

>>> id(b)

2136897448528

>>>

>>> a = 20

>>> id(a)

2136897448848

>>>

Basic In built Functions:-

1.print()---To print value

2.type()----knowing the data type.

3.dir()-----function of the data.

4.help()-----how to use the fuction of data.

5.id()------location of data.

Integer(int)

we can use int data type to represent whole number(integral value).

>>> a = 10

>>> type(a)

<class 'int'>

>>>

We can represent int values form.

1.decimal form

2.Binary form'

3.Octal form

4.Hexa Decimal Form.

int to Base form:-

Base Conversion:-

1.Decimal(float)-:(Base:-10)

It is the default number system in python:- float

The allowed digits are 0 to 9

ex:-

>>> a = 10

>>> float(a)

10.0

>>>

2.Binary form(Base:-2)

Inbuitl function:- bin()

The allowed digits are:- 0 and 1

Literal value should be prefixed with 0b.

ex:-

>>> a = 10

>>> b = bin(a)

>>> b

'0b1010'

>>> type(b)

<class 'str'>

>>>

3.Octal form(Base:-8)

Inbuilt function:- oct()

The allowed digits are:- 0 to 7

Literal value should be prefixed with 0o.

ex:-

>>> a = 10

>>> b = oct(a)

>>> b

'0o12'

>>> type(b)

<class 'str'>

>>>

4.Hexa Decimal Form:-(Base-16)

Inbuilt function:- hex()

The allowed digits are:- 0 to 9 and a to f

Literal value should be prefixed with 0x.

ex:-

>>> a = 10

>>> b = hex(a)

>>> b

'0xa'

>>>

we can use int data type to represent whole number(integral number).

>>> a = -100

>>> type(a)

<class 'int'>

>>>

>>> b = 100

>>> type(b)

<class 'int'>

we can represent int values in----Base conversion

Float Data type:-

we can use float data type to represent floating point value(decimal values)

>>> f = 1.234

>>> type(f)

<class 'float'>

>>>

Exponential form:-

We can also represent floating point value by using exponential form.(scientific natation)

>>> f = 1.2e3

>>> f

1200.0

The main adv of exponential form is we can represent big value in less moemory.

Complex Number:-

A complex number is of the form a+bj

a:- Real number(int + float)

b:- Imaginary number(j is include and j is unsloved value):- int and float

ex:-

>>> a = 3+5j

>>> type(a)

<class 'complex'>

>>> b = -9-7j

>>> type(b)

<class 'complex'>

Addtion and substration of complex number:-

Real only add or sub with another Real numeber

Imaginary only add or sub with another imaginary numeber

add:-

>>> a = 7+9j

>>> b = 8+7j

>>>

>>> a+b

(15+16j)

>>>

>>> c = -9+8j

>>> d = 8-10j

>>> c+d

(-1-2j)

sub:-

>>> a = 7+4j

>>> b = 10+9j

>>>

>>> a-b

(-3-5j)

>>>

4.Boolean(bool()):-

we can use this data type to represent boolean values. For only yes or no answer.

The only allowed values for this data type are:- True and False.

Internally Python reprsents True as 1 and False as 0.

>>> True + True

2

>>> True + False

1

>>> False + False

0

>>>

>>> a = True

>>> type(a)

<class 'bool'>

>>>

>>> b = False

>>> type(b)

<class 'bool'>

>>> s = ['a',889,'b',989]

>>> 1 in s

False

>>> 'b' in s

True

>>>

String(str()):-

str represent String Data type.

A string is sequence of any keyboard letter enclosed within single or double or triple quotes.

>>> a = ''

>>> type(a)

<class 'str'>

>>>

>>> b = '()'

>>> type(b)

<class 'str'>

>>>

>>> c = '!@()-=\*&^%%WE^bsancbnmsabcj,sbcj'

>>> type(c)

<class 'str'>

>>>

>>> d = ""

>>> type(d)

<class 'str'>

>>>

>>> e = "hasjhcgusayhcy78q27ei2he27180eyhjdbsajkcgasuiyc9dqw"

>>> type(e)

<class 'str'>

>>>

Triple Quotes(Multiple Quotes):-

By using single quote or double quote we cannot represent multiple4 ;lines string literals.

Function:-

def add(a,b):

''' This function is using for addition purpose '''

c = a+b

return c

class:-

class AdlofHitler(self):

''' This class was written for Germani Chansellor Adlof Hitler '''

def jhkwjkl\_bwjbd():

evknekv

Types of Triple quotes:-

There are 2 types of triple quote are present in python

1.Single Triple quote :- ''' '''

2.Double Triple quote :- """ """

Note1:- When we can use in a worksheet a single Triple Quote,We can use in the whole Worksheet same things and for double triple quote follow the same rule.

Note2:- When we can use in variable store string data so here we can start with single quote then we can use in the whole worksheet same things, and for double quote follow the same rule.

Exception:-

s = 'how's this'-----------wrong

s = "how's this"-----------right

Slicing Of string:-

slice means a piece

+ve-:- 0 1 2 3 4 5 6

s = ' S c o d e e n'

-ve:- -7 -6 -5 -4 -3 -2 -1

[] operator is called slice operator,which can be used to retrive parts of string

-In python string follows zero based index.

Here is both +ve and -ve index are allowed.

+ve index means forward direction from left to right.

-ve index means backward direction from right to left.

Formula :-

variable\_name[start\_index:end\_index+1:steps]

>>> s[0:3]

'Sco'

>>>

>>> s[4:]

'een'

>>> s[:4]

'Scod'

>>> s[:]

'Scodeen'

>>> name = 'Rahul Dravid'

>>> dob = '20-08-2013'

>>> first\_sl = name[0:2]

>>> sec\_sl = name[6:8]

>>> thr\_sl = dob[6:]

>>>

>>> first\_sl+sec\_sl+thr\_sl

'RaDr2013'

Negetive:-

>>> s = 'Scodeen'

>>> s[-7:-4]

'Sco'

>>> s[-4:-1]

'dee'

>>> s[3:6]

'dee'

>>>

steps:-

>>> a = 'i love my country'

>>> a[0:-2:3]

'io u'

>>> a[0:-1:3]

'io ur'

>>> a[0:-2:3]

'io u'

>>>

count of string:-

>>> s = 'Scodeen'

>>> len(s)

7

>>>

Function or method of string:-

dir():- check the data type or class existing function.

a = 'thomson'

>>> dir(a)

['capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format\_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']

Imp:- capitalize,count,endswith,find,format,index,join,lower,replace,split,startswith,strip,swapcase,upper

How to use any function:-

dot method:-

class\_name.function\_name()

data\_type\_of\_variable\_name.function\_name()

Imp:- capitalize,count,endswith,find,format,index,join,lower,replace,split,startswith,strip,swapcase,upper

1.capitalize():- Make the first character is upper case and rest of the string is lower case.

>>> s = 'SoCdEEn'

>>> s.capitalize()

'Socdeen'

2.count():- count a specific charcater how may times is used in a string.

>>> a = 'siskhsya O Anusandhan'

>>> a.count('s')

4

3.endswith():-It's show the Boolean value,if we will give the right character to end the string then it's return True otherwise False.

>>> s = 'SoCdEEn'

>>> s.endswith('n')

True

>>> s.endswith('7')

False

4.startswith():-It's show the Boolean value,if we will give the right character to start the string then it's return True otherwise False.

>>> s = 'SoCdEEn'

>>> s.startswith('S')

True

>>> s.startswith('H')

False

>>>

5.replace():-

>>> s = 'jello'

>>> s.replace('j','h')

'hello'

>>> s.replace('jello','hi')

'hi'

>>>

>>> a = 'jello how are you joker'

>>> a.replace('j','h')

'hello how are you hoker'

>>> a.replace('jello','Hello')

'Hello how are you joker'

>>>

6.strip():-To remove white spaces both sides.

>>> s = ' Mahavira '

>>> s.strip()

'Mahavira'

>>>

7.find():- Returns index of first occurance of the given substring,If it is not available then we will get -1.

>>> s = 'Learning Python is very easy'

>>>

>>> s.find('Python')

9

>>> s.find('P')

9

>>> s.find('n')

4

>>> s.find('i')

5

>>> s.find('Java')

-1

>>>

8.index():- index() method is exactly same as find() method except that if the specified substring is not available the will ger ValueError.

>>> s = 'Learning Python is very easy'

>>> s.index('Python')

9

>>> s.index('i')

5

>>> s.index('java')

Traceback (most recent call last):

File "<pyshell#60>", line 1, in <module>

s.index('java')

ValueError: substring not found

>>>

9.upper:- Convert whole string into the upper case.

>>> s = 'Gogo MariNa'

>>> s.upper()

'GOGO MARINA'

>>>

10.lower:- Convert the whole string into the lower case.

>>> s = 'Gogo MariNa'

>>> s.lower()

'gogo marina'

11.swapcase():- It's convert the uppercase to lower and vice vers.

>>> s = 'Gogo MariNa'

>>> s.swapcase()

'gOGO mARInA'

>>>

12.split():-The split() method splits a string into a list.

s = 'Learning Python is very easy'

>>> s.split()

['Learning', 'Python', 'is', 'very', 'easy']

>>> s = 'Learning/Python/is/very/easy'

>>> s.split('/')

['Learning', 'Python', 'is', 'very', 'easy']

>>> s.split()

['Learning/Python/is/very/easy']

>>>

14.join:- The join() method takes all items in the collection of iterable and joins them into one string.

>>> my\_data = ('Hi','Carry','Minati')

>>> x = '-'.join(my\_data)

>>> print(x)

Hi-Carry-Minati

>>> type(x)

<class 'str'>

>>> x = ' '.join(my\_data)

>>> print(x)

Hi Carry Minati

>>> x = '{//} '.join(my\_data)

>>> print(x)

Hi{//} Carry{//} Minati

>>> x = ' @'.join(my\_data)

>>> print(x)

Hi @Carry @Minati

>>>

15.format():- The format() method format the specified values and insert them inside the string's placeholder.

>>> s = 'Counting the string'

>>> 'Counting substring in the given string'

'Counting substring in the given string'

>>> s = 'Counting {0} {1} the {2} string'

>>> s.format('substring','in','given')

'Counting substring in the given string'

>>>

Placeholder {} :- Insert the substring into a complete string.

List Data type(list()):-

>>> l = [12,89,'gh',78.89,[56,89,90],(56,89),89]

>>> type(l)

<class 'list'>

>>> l[0]

12

>>> l[-1]

89

>>> l[1]

89

>>> l[-3]

[56, 89, 90]

>>>

If we want to represent a group of values as a single entity where insertion order required to preserve and duplicate are allowed then we should go for list data type.Here every data are separet by comma.List is enclosed with in squre bracket[].

1.Insertion order is preserved.

2.Heterogeneous object are allowed.

3.duplicate are allowed.

4.Growable in nature(add data).

5.Values should be enclosed with in squre bracket.

Note:- List is dynamic because based on our requirement we can increase the size and decrease the size.

Important Function of list:-

>>> l = [10,20,30,40,50]

>>> dir(l)

['append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']

>>>

len():- Return the number of elements present in the list.

>>> len(l)

5

>>>

1.count():- It returns the number of occurance od specified item in the list.

>>> l = [1,1,2,2,2,3,4,4]

>>> l.count(3)

1

>>> l.count(4)

2

>>> l.count(1)

2

>>> l.count(2)

3

>>>

2.index():- Returns the index of the first occurance of the specified item.

>>> l = [1,1,2,2,2,3,4,4]

>>> l.index(1)

0

>>> l.index(3)

5

>>> l.index(4)

6

>>>

>>> l.index(6)

Traceback (most recent call last):

File "<pyshell#21>", line 1, in <module>

l.index(6)

ValueError: 6 is not in list

3.append():- we can use append() to add item at the end of the list.

>>> l = [10,20,30]

>>> l.append(40)

>>> l

[10, 20, 30, 40]

>>> l.append(50)

>>> l

[10, 20, 30, 40, 50]

>>>

4.insert():- To insert at specified index position.

>>> l = [10,20,30]

>>> l.insert(1,15)

>>> l

[10, 15, 20, 30]

>>>

>>> l.insert(-1,35)

>>> l

[10, 15, 20, 35, 30]

difference between append() and insert():-

append() insert

In list when we add any element In list we acn insert ane element in particular

it will come in last that is index number.

it will ve last element

5.extend():- To add all items of one list to another list.

l1= []

l2 = []

l1.extend(l2):- l2 items will be added to l1

l2.extend(l1):- l1 items will be added to l2

>>> l = [10,20,30]

>>> p = [1,1,2,2,2,3,4,4]

>>>

>>> l.extend(p)

>>> l

[10, 20, 30, 1, 1, 2, 2, 2, 3, 4, 4]

>>>

6.remove():- we can use this function to remove specified item from the list.If the item present multiple times then only first occurance will be removed.

>>> l = [1,1,2,2,2,3,4,4]

>>> l.remove(3)

>>> l

[1, 1, 2, 2, 2, 4, 4]

>>> l.remove(1)

>>> l

[1, 2, 2, 2, 4, 4]

>>> l.remove(1)

>>> l

[2, 2, 2, 4, 4]

>>>

>>> l.remove(9)

Traceback (most recent call last):

File "<pyshell#58>", line 1, in <module>

l.remove(9)

ValueError: list.remove(x): x not in list

>>>

Note:- Hence before using remove() first we have to check specofied element present in the list or not by using in operator.

7.pop():- It removes and returns the last element of the list.

This is only function which manipulates list and returns some element.

>>> l = [10,20,30]

>>> l.pop()

30

>>> l = [10,20,30,80,90,70]

>>> l.pop(2)

30

>>> l.pop(5)

Traceback (most recent call last):

File "<pyshell#69>", line 1, in <module>

l.pop(5)

IndexError: pop index out of range

>>> l.pop(-2)

90

>>>

Differenece betwen remove() and pop()

remove() pop()

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

1.We can use to remove special element 1. We can use to remove last element from the list.

from the list.

2.It can't return any value. 2.It returned removed element.

3.If special element not available then 3.If list is empty then get Error.

we get ValueError.

8.clear():- It's clear the all data from respective list.

>>> l = [10,20,30,80,90,70]

>>> l.clear()

>>> l

[]

9.copy():- copy the same data from a another list.

>> l = [10,20,30,80,90,70]

>>> a = l.copy()

>>> a

[10, 20, 30, 80, 90, 70]

>>>

10.reverse:- we can use to reverse() order of elemet list.

>>> l = [10,20,30,80,90,70]

>>> l.reverse()

>>> l

[70, 90, 80, 30, 20, 10]

>>>

11.sort():- In list by deafult insertion order is preserved.If want to sort the elements of list according to deafult natural sorting order then we should go for sort()

function.

>>> l = [10,34,23,11.78,1,11,12.78,99,456,234]

>>> l.sort()

>>> l

[1, 10, 11, 11.78, 12.78, 23, 34, 99, 234, 456]

>>> l.sort(reverse=True)

>>> l

[456, 234, 99, 34, 23, 12.78, 11.78, 11, 10, 1]

>>>

>>> l = ['a','A','apple','ball','Apple','Ball']

>>> l.sort()

>>> l

['A', 'Apple', 'Ball', 'a', 'apple', 'ball']

l = [1,2,1.56,56,'A','Apple','a','b']

>>> l.sort()

Traceback (most recent call last):

File "<pyshell#13>", line 1, in <module>

l.sort()

TypeError: '<' not supported between instances of 'str' and 'int'

>>>

Tuple Data Type(tuple()):-

Tuple Data Type is exactly same as list data type except that is immutable ,we can not change values.

Tuple elements can be reprsented with in parenthsis.

>>> s = (1,2,3,4)

>>> type(s)

<class 'tuple'>

>>>

>>> s = (1,2,3,4)

>>> dir(s)

[count', 'index']

>>>

Note:- tuple is the read only version of list.

Set Data type(set()):-

if we want to reprsent a group of values without duplicates where order is not important then we should go for set data type.

ex:-

>>> s = {1,1,2,2,2,3,4,'scodeen',True,(78,90)}

>>> s

{1, 2, 3, 4, (78, 90), 'scodeen'}

>>> type(s)

<class 'set'>

>>>

Notes:-

1.Insertion order is not preserved.

2.Duplicates are not allowed.

3.hetrogeneous object are allowed.

4.Index concept is not applicable.

5.Set is mutable.

6.Growable in nature.

set manipulation:-

>>> s = {1,2,3,4,5,6,7}

>>> dir(s)

['add', 'clear', 'copy', 'difference', 'difference\_update', 'discard', 'intersection', 'intersection\_update', 'isdisjoint', 'issubset', 'issuperset', 'pop', 'remove', 'symmetric\_difference', 'symmetric\_difference\_update', 'union', 'update']

imp:- 'add', 'clear', 'copy', 'difference','discard', 'intersection', 'pop', 'remove','union', 'update']

1.add():- Add itmes x to the set.

>>> s = {1,2,3,4,5,6,7}

>>> s.add(8)

>>> s

{1, 2, 3, 4, 5, 6, 7, 8}

2.clear():- To remove all the elements from the set,

>>> s = {1,2,3,4,5,6,7}

>>> s.clear()

>>> s

set()

3.copy():- Create a mirror image in other variable.

>>> s = {1,2,3,4,5,6,7}

>>> a = s.copy()

>>> a

{1, 2, 3, 4, 5, 6, 7}

>>> type(a)

<class 'set'>

4.differenece():- Returns the elements present in x but not in y.

>>> x = {10,20,30,40,50}

>>> y = {30,40,50,60}

>>> a = x.difference(y)

>>> a

{10, 20}

>>> b = y.difference(x)

>>> b

{60}

>>>

5.discard():- It removes the specified elements from the set.

If the specified element not present in the set then we won'get any error.

>>> x = {10,20,30,40,50}

>>> x.discard(40)

>>> x

{50, 20, 10, 30}

>>> x.discard(800)

>>> x

{50, 20, 10, 30}

>>>

6.pop():- It removes and returns some random elemts from the set.

>>> x = {10,20,30,40,50}

>>> x.pop()

50

>>> x.pop()

20

>>>

7,remove():-

It remove the specified element from the set.

If the specofied element not present in the set then we will get error.

>>> x = {10,20,30,40,50}

>>> x.remove(30)

>>> x

{50, 20, 40, 10}

>>> x.remove(300)

Traceback (most recent call last):

File "<pyshell#49>", line 1, in <module>

x.remove(300)

KeyError: 300

>>>

8.intersection():-

Return common elements present in both the set.

x = {10,20,30,40,50}

>>> y = {30,40,50,60}

>>>

>>> x.intersection(y)

{40, 50, 30}

>>> y.intersection(x)

{40, 50, 30}

>>>

9.union(): We can use this function to return all elements present in both set.

x.union(y) or y.union(x)

>>> x = {10,20,30,40}

>>> y = {30,40,50,60}

>>> x.union(y)

{40, 10, 50, 20, 60, 30}

>>

10.update():-

To add multiple items to the set.

>>> s = {10,20,30}

>>> t = (67,78,89)

>>> s.update(t)

>>> s

{67, 10, 78, 20, 89, 30}

>>>

Frozenset data type(frozenset()):-

s = {10,20,30}

>>> fs = frozenset(s)

>>> fs

frozenset({10, 20, 30})

>>> type(fs)

<class 'frozenset'>

>>>

>>> dir(fs)

[ 'copy', 'difference', 'intersection', 'isdisjoint', 'issubset', 'issuperset', 'symmetric\_difference', 'union']

>>>

Dictionary Data type(dict()):-

Name age roll

sakti 22 45

vishal 23 56

shiba 24 57

gago 45 78

.074a = {Name:Sakti,age:22,roll:45}

b = {Name:Vishal,age:23,roll:56}

>>> a = ('Name','age','roll\_no')

>>> b = ('Sakti',22,45)

>>>

>>> c = dict(a,b)

Traceback (most recent call last):

File "<pyshell#3>", line 1, in <module>

c = dict(a,b)

TypeError: dict expected at most 1 argument, got 2

>>>

>>> c = {'Name':'Sakti','age':22,'roll\_no':45}

>>> type(c)

<class 'dict'>

>>> c.keys()

dict\_keys(['Name', 'age', 'roll\_no'])

>>> c.values()

dict\_values(['Sakti', 22, 45])

>>>

>>> d = {['a','b']:1}

Traceback (most recent call last):

File "<pyshell#10>", line 1, in <module>

d = {['a','b']:1}

TypeError: unhashable type: 'list'

>>> d = {67:[67,90]}

>>> d[67]

[67, 90]

>>>

>>>

>>> a = []

>>> type(a)

<class 'list'>

>>> b = ()

>>> type(b)

<class 'tuple'>

>>> c = {}

>>> type(c)

<class 'dict'>

>>> d = set()

>>> type(d)

<class 'set'>

>>>

dictionary

If we want to represent a group of objects as key-value pairs then we should go for dictionary.

we can use list,tuple and set to represent a group of individual object as a single entity.

Imp ponits:-

1.Duplicate keys are not allowed but values can be duplicated.

2.Hetrogeneous object are allowed for both key and values.

3.Insertion order is nor preserved.

4.Dictionary are mutable.

Note:- In C++ and java Dictionary are known as 'Map' and in perl and Rubby known as 'hash'

>>> a = {'Name':'Krishna','Age':24,'Address':'BBSR'}

>>> type(a)

<class 'dict'>

dir(a)

['clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']

>>>

1.clear():- To Remove all entries from the dictionary.

>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}

>>> a.clear()

>>> a

{}

Note:- del a

to delete total dictionary.

>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}

>>> del a

>>> print(d)

Traceback (most recent call last):

File "<pyshell#32>", line 1, in <module>

print(a)

NameError: name 'a' is not defined

>>>

2.copy:- To create exactly duplicate dictionary.

>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}

>>> b = a.copy()

>>> b

{'Name': 'Krishna', 'Age': 24, 'City': 'BBSR'}

>>> type(a)

<class 'dict'>

>>>

3.fromkeys():- The fromkeys() method returns a dictionary with specified keys and the specified values

syntax:- dict.fromekys(a,b)

a = ('Delhi','BLG','HYD','BBSR')

>>> b = ('Ragav')

>>> c = {}

>>> c.fromkeys(a,b)

{'Delhi': 'Ragav', 'BLG': 'Ragav', 'HYD': 'Ragav', 'BBSR': 'Ragav'}

>>>

4.get():- To get the value associated with the key.

>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}

>>> a.get('Name')

'Krishna'

>>>

5.itmes():- It returns list of tuples representing key-value pairs.

[(k,v),(k,v),(k,v)]

ex:-

>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}

>>> a.items()

dict\_items([('Name', 'Krishna'), ('Age', 24), ('City', 'BBSR')])

>>>

6.keys:- It returns all keys associated with dictionary.

>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}

>>> a.keys()

dict\_keys(['Name', 'Age', 'City'])

>>>

7.values():- It returns all values associated with the dictionary.

>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}

>>> a.values()

dict\_values(['Krishna', 24, 'BBSR'])

>>>

8.pop():- It removes the entry associated with the specified key and returns the corresponding values

>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}

>>> a.pop('City')

'BBSR'

>>> a

{'Name': 'Krishna', 'Age': 24}

If type specified key is not available then we will get KeyError

>>> a.pop('zip\_code')

Traceback (most recent call last):

File "<pyshell#59>", line 1, in <module>

a.pop('zip\_code')

KeyError: 'zip\_code'

>>>

9.popitem():- It removes an arbitaryitem(key-value) from the dictionary and returns it,

ex:-

>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}

>>> a.popitem()

('City', 'BBSR')

>>> a

{'Name': 'Krishna', 'Age': 24}

>>>

empty dictionary:-

>>> b ={}

>>> b.popitem()

Traceback (most recent call last):

File "<pyshell#65>", line 1, in <module>

b.popitem()

KeyError: 'popitem(): dictionary is empty'

>>>

10.setdefault():-

If the key is already available then this function returns the corresponding values.

if the key is not available then the specified key-value will be added as new item to the dictionary.

>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}

>>> a.setdefault('zip\_code',1278)

1278

>>> a

{'Name': 'Krishna', 'Age': 24, 'City': 'BBSR', 'zip\_code': 1278}

>>> a.setdefault('City','RKL')

'BBSR'

>>> a

{'Name': 'Krishna', 'Age': 24, 'City': 'BBSR', 'zip\_code': 1278}

>>>

11.update:- All itmes present in the dictionary then we will add more key-value pair in it.

>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}

>>> a.update({'zip\_code':1234,'post\_office':'Khandagiri','PoliceStation':'CRPF'})

>>> a

{'Name': 'Krishna', 'Age': 24, 'City': 'BBSR', 'zip\_code': 1234, 'post\_office': 'Khandagiri', 'PoliceStation': 'CRPF'}

>>>

Operators

Operator is a symbol that performs certain operations.

Python provides 6 main operators.

1.Arithmetic Operators.

2.Realational or Comparision Opeartors.

3.Logical Operators.

4.Bitwise Opearators.(Not Required)

5.Assignment Operators.

6.special Opeartors.

1.Arithmetic Operators:-

+ -->Addition

>>> 12+90

102

>>> 9.89+7.67

17.560000000000002

>>> 6+9j + 7+8j

(13+17j)

>>> 'K'+'S'

'KS'

>>> True + False

1

>>>

- -->substraction

>>> 90-67

23

>>> 7.890-6.89

1.0

>>> 7+9j - 6-9j

(1+0j)

>>> 'H'-'L'

Traceback (most recent call last):

File "<pyshell#9>", line 1, in <module>

'H'-'L'

TypeError: unsupported operand type(s) for -: 'str' and 'str'

>>> True - True

0

>>>

\* -->Multiplication

>>> 56\*78

4368

>>> 6.89\*7.78

53.6042

>>> (9+8j) \* (8+8j)

(8+136j)

>>> 'KS'\*3

'KSKSKS'

>>> True \*556

556

/ -->Division

>>> 67/8

8.375

>>> 68.89/6.45

10.68062015503876

>>> (67+9j) / (8+7j)

(5.300884955752212-3.5132743362831858j)

>>> 'K'/7

Traceback (most recent call last):

File "<pyshell#23>", line 1, in <module>

'K'/7

TypeError: unsupported operand type(s) for /: 'str' and 'int'

>>> True / 7

0.14285714285714285

>>>

% -->modulo

>>> 62%6

2

>>>

// -->Floor Division

>>> 67//5

13

>>>

\*\* -->Exponent

>>> 2\*\*3

8

>>> 4\*\*2

16

>>>

2.Realational or Comparision Opeartors.

Relational operator always return the boolean value.

>,<,>=,<=,==(equal to),!=(Not equal)

Greater than:-

>>> 45>78

False

>>> 67>56

True

>>>

Less than :-

>>> 56<78

True

>>> 78<45

False

>>>

Greater that equal to :-

>>> 78>=90

False

>>> 67>=67

True

>>>

Less than equal to:-

>>>

>>> 67<= 78

True

>>> 78 <= 67

Falses

>>> 78<= 78

True

Equal to:-

>>> 89 == 89

True

>>> 78 ==90

False

>>>

Not equal to:-

>>> 67 != 89

True

>>> 56 != 56

False

>>>

Logical Opeartors:-

and , or , not

and :- If both the arguments are True then only result is True otherwise False

or:- If atleast One argument is True then Result is True Othersie False.

not :- Complement

and:-Always return the biggest value.

T T T

T F F

F F F

>>> 56 and 78

78

>>> 0 and 67

0

or :- If a arguements evaluates to True then result is this arguement otherwise second one.

if one arguement is True then True otherwise False.

always smallest value is return.

T T T

T F T

F F F

>>> 10 or 20

10

>>> 78 or 4550

78

>>> 0 or 67

67

>>>

not:- if a argument is evalutates to False then result is True otherwise False.

>>> not 6

False

>>> not 0

True

>>>

Assignment Operators:-

we can use assignment operator to assign value to the variable

a = 10 ( = is assignment opeartor)

we can combine assignment operator with some other operator to form compound assignment operator,

ex :-

x = x+10 --------------------- x+=10

Possiable list of all compoundment operator

+=

-=

\*=

/=

%=

//=

\*\*=

Special Operator:-

1.identity Opeartors

2.membership Opeartos

1.identity Opeartors

we can use idenity operators for address comparision.

2 idenitity opeartor is available.

1. is(if variable indicate same data)

2. is not(if variable indicate different data)

x is y return True if both x and y are pointing to the same object.

>>> a = 10

>>> b = 10

>>> a is b

True

>>>

>>> a= 89

>>> b = 90

>>> a is b

False

>>>

x is not y return True if x and y are pointing to the differnet object.

>>> a = 90

>>> b = 89

>>>

>>> a is not b

True

>>>

>>> a = 10

>>> b = 10

>>> a is not b

False

>>>

2.Memebership Opeartor:-

we can use membership operators to check whether the given object present in the given collection(String,List,Srt,Tuple or Dict)

in:- Returns True if the given obejct present in the specified collection.

>>> a = 'Vicky'

>>> 'k' in a

True

>>> 'h' in a

False

>>>

>>> b = [10,20,30,40]

>>> 89 in b

False

>>> 40 in b

True

>>>

not in:- Returns True if the given object not present in the specified collection.

>>> a = 'Vicky'

>>> 'k' not in a

False

>>> 'h' not in a

True

>>>