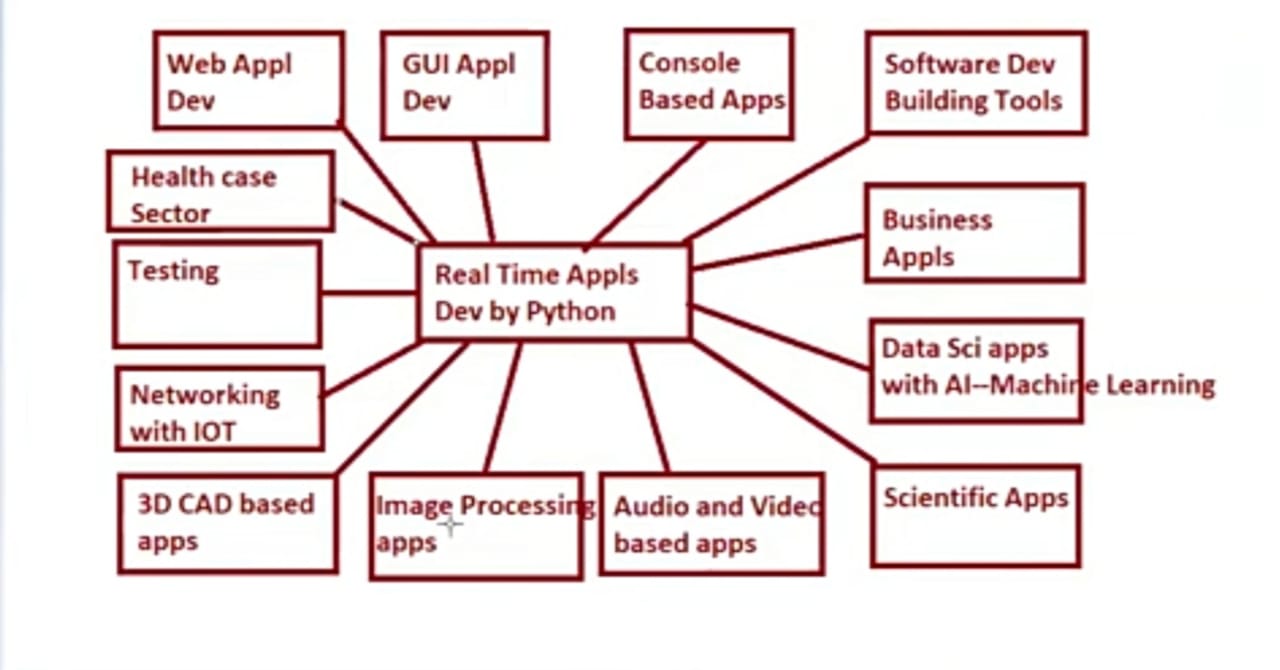
Pythone notes:

Real time apps dev by python:



**What is python:**

=>Python is one of the General Purpose, High Level, object oriented and

Interpreted Programming lang.

=>Developed by **Guido Van Rossum** at CWI in **Nether Lands in year 1991** Feb

=>Downloaded Freely from www.python.org I

=>Python software having two versions

1) Python 2.x—x----- 1,2,3,4....outdated version

2) Python 3.x--x-1,2,3,----9--industry used version 3.9

=> Python does not support Backword Compatability

Python software maintained by non-commercial organization called

Python Software Foundation (PSF)

**Features of Python lang**

=>Features of a lang are nothing but services / facilities provided by lang Developers in the lang, which are used by Lang programmers for developing real time applications.

1 Simple and Easy to Learn

2. Dynamically Programming Lang

3. Platform Independent lang

4. Portable

5. Interpreted Programming Lang

6. High Level Programming Lang

7. Procedural (Functional) and Object Oriented Programming Lang

8. Robust(Strong)

9. Extensible

10. Embedded

11. Supports Third Party APIs(scikit, scipy, numpy, matplotlib etc)

Python lang Inspired From:

1)Functional Programming from C

2)Object orientened Principle from Cpp

3)Scripting Programming from PERL

4)modular Programming from Modulo3.

**1 ) Simple**

Python is Simple Programming lang, bcoz of 3 Tech factors:

a) Python lang provides Rich set Of Modules. So that Python programmer can re-use the pre-defined code without our own code.

Def of module:

A module is a collection of Functions, Variables and classes

Examples:- calendar, random, math,cmath, re, os,io.......etc

b) Python Programming provides in-built facility called "Garbage Collector", which collects un-used memory space and improves the performance of Python Based Applications.

Def.Garbage Collector:

A Garbage Collector is one of software component in python software, which is running in the background of regular python program and whose role is to collect/remove un-used memory space.

Hence Garbage Collector is taking care about Automatic Memory Mgmt

c) Python provides Devloper friendly Syntaxes. So that we can develop error-Free programs in a limited span of time.

2)**Freeware and open source:**

a)Freeware:

b)open source:

=>The father of python Rossum developed CPYTHON and it is treated as standard python software.

=>many companies the CPYTHON and released and they are called python distributions/flavours

a) JPython (OR) Jython->Used for running Java Based Applications.

b) Iron Python-->Used for running C#.net applications.

c) Mocro Python-->Used for devlopping Micro Controllers.

d) Anakonda Python--> Used for running Big Data / Hadoop.

e) Ruby Python-->Used for running Ruby Based Applications.

**3) Dynamically Typed Programming lang**

In Programing languages, we have two types. They are

1. Static Typed Programming Lang

2. Dynamic Typed Programming Lang.

=>In Static Typed Programming Lang, It is mandatory to specify the data type of the variable otherwise we get Compile time error.

Example:- C, CPP, Java-etc

=>In Dynamically Typed Programming Lang, It is not necessary to specify the data type of the variable. ie. Python Environment takes care about assigning the data type to variable automatically depends on type value we place in the variable.

Example: python

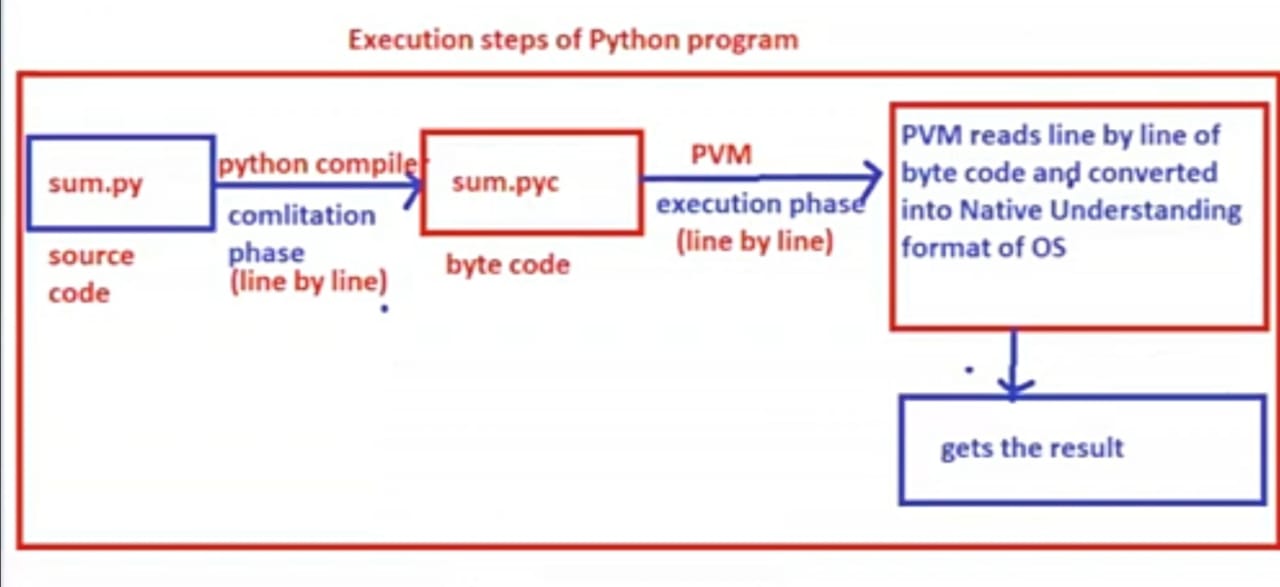
Example: >>> a=10

>>>type(a)------<class, 'int'>

**4)Interpreted Programming Lang**

When we execute the python program, Internally the source code (Ex: sum.py) is compiled with line by line conversion and this compiled version is called byte code (Ex: sum.pyc). The intermediate code called byte code (.pyc) is read by PVM (python Virtual machine) line by line and converted into Native understanding format of OS.

=>Consider the following diagram, it shows exection steps of python program.

****

**5)Platform Independent lang**

concept: A language is said to be Platform Independent if whose applications runs on Every OS.

property:

contain restrictions on size of data"

"All the variable in python are treated as objects and they does not contain restrictions size of data”

**6)High Level Programming Lang**

The statments in python program are looking like english statements and it is treated as "High Level Programming lang"

Extensible:

The other languages (c,cpp,java….etc) are integrating the code/snippets/script of python.The facility is called extensible.

**Embedded:**

The python code can also integrate the code of other languages. This property is called Embedded.

Procedural (Functional) and Object Oriented Programming Lang

Since python program can be developed with Functional Programming and Object Oriented Approaches and hence it is populated as " Functional and object oriented Programming lang".

* **Programming Fundamental in Python**

1) Identifers

2) Literals

3) Object Referencing

4) Data Types

**1) Importance of Identifiers in python**

=>With the help of data types, memory space can be allocated and data can be stored. To process the data which stored in memory, the memory space must be given some distinct/unique names and these unique names makes us to identify the values present in memory and these distinct names are called Identifiers.

Identifiers are also called Variables bcoz identifier values can be changed during program execution.

=>Hence in python, all values must be stored in memory in the form identifers/Variables.

**Def of Variable:**

A variable one of the identifer, whose value can be changed during execution of the program.

Rules for using Identifiers / Variables in Python

1) The variable name is a combination of alphabets, digits and special symbol under score (\_\_)

2) The First letter of Variable Name must starts with either alphabet or underscore (\_)

Example:

>>>a=10---------valid

>>>\_=20--------valid

>>> $=40----invalid

>>>tot$sal=50----invalid

3) Within the Variable names, special symbols are not allowed except under score (\_)

Example:-

>>>tot$sal=6.7-------invalid

>>>tot$sal=2.3-------invalid

>>>tot-sal=4.5------invalid

>>>tot\_sal\_emp=2.3----valid

4) We should not use Keywords as variable names (bcoz keywords are the reserved words and gives some specific meaning)

Example:

>>>if=20- --invalid

>>>else= 40- ------invalid

>>>if\_=50- ---valid

>>>int=40- -------VALID (int is a class but not a kwd)

>>>float=23.45-----------VALID

5) There is restrictions on size of the variable. (recommended to take short and sweet name)

6) Variable names are case sensitive

Example:

=>>>age=99-------valid

>>>AGE=100-----valid

>>>Age=30-------valid

>>>a\_g\_e=44------valid

* **Data Types in PYTHON**

>The purpose of Data Types to allocate memory space in Main Memory for storing input data.

=>In python, we have 14 data types and they are classified into 6 types.

1) Fundamental Categery Data Types:

1. int

2. float

3. bool

4. complex

2)Sequential Categery Data Types:

5. str

6. bytes

7. bytearray

8. range

III)List Category DataTypes:

9.list

10. tuple

IV) Set Categery Data Types:

11. set

12. frozenset

V) Dict Categery Data Types:

13.dict

VI)None Type Categery Data Types:

14.none

***I) Fundamental Categery Data Types:***

purpose:To store single value.

1) int

2) float

3) bool

4) complex

**1)int:**

purpose: To store Integral Values/whole numbers (does not contain. decimal values)

Example:-

>>>a=100

>>> print(a)-------100

>>> type(a)--------<class 'int'>

>>> print(type(a))- ------<class 'int">

>>> print(a, type(a)) ------100 <class 'int'>

>>> b=200

>>> print(b, id(b), type(b))--------> 200 2329064006288 <class 'int'>

=>With this data type we can store diff values of Different Number Systems

=>We have 4 types of Number System. They are

a)Decimal Number Systems( default number System)

Digits:- 0,1,2,3,4,5,6,7,8,9

base:---10

b) Binary Number Systems

digits:- 0,1

base:-2

c) Octal Number System:

digits:0,1,2,3,4,5,6,7

base: 8

d) Hexa Decimal Number System:

digits: 0,1,2,3,4,5,6,7,8,9, ABCDEF

b) Storing Binary Number Systems Data

=>To store Binary Number Systems data, in python environment, binary data must be preceded with 0b or 0B

=> syntax:- varname=0b binarydata

Example:

>>>a=0b1111

>>>print(a)----------15

>>>a=0B1010

>>>print(a)-----------10

>>>print(type(a))----------<class, 'int'>

b) Storing Octal Number Systems Data

=>To store Octal Number Systems data, in python environment, Octal data must be preceded with 0o or 0O

syntax:- varname=0o Octal data

Examples:-

>>>a=0o23

>>>print(a)-----------------19

>>>type(a)---------------<class, 'int'>

>>>a=0o129.------------inavlid, bcoz the digit 9 is not present in Octal number System

c) Storing Hexa Decimal Number Systems Data

1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

=>To store Hexa Decimal Number Systems data, in python environment,

Hexa Decimal data must be preceded with Ox or OX

=> syntax:- varname=0x Hexa Decimal data

Example:

>>>a=0xAB

>>> print(a)

>>> type(a)---------<class 'int'>

>>> a=0xFACE

>>> print(a)- -64206

>>> type(a)- <class 'int'>

>>>a=oxACER- --invalid, the letter 'R' is not present in Hexa Decimal Number System

**Base Conversion Functions:** =>The purpose of Base Conversion Functions is that converting Decimal Number System into Binary, octal and Hexa Decimal Number system. we have 3 Base Conversion Functions. They are

a) bin()

b) oct()

c) hex()

1)bin()

=>This function is used for converting Decimal Number System data into binary number System data.

syntax:- varname= bin(decimal number System data)

Example:-

>>>a=15

>>> b=bin(a)

>>> print(b)--------0b1111

>>> a=10

>>> b=bin(a)

>>> print(b)-----0b1010

b) oct()

>This function is used for converting Decimal Number System data into Octal number System data.

syntax:- varname= oct (decimal number System data)

Examples:

>>>a=19

>>>b=oct(a)

>>>print(b)- -----0o23

c) hexO

=>This function is used for converting Decimal Number System data into

Hexa Decimal number System data.

Examples:

syntax:-

>>>a=171

>>>b=hex(a)

>>>print(b)- -----0xab or 0xAB

>>>print(hex(10))-------------0xa

2) float:

=>'float' is one of the pre-defined class

=>To store floating point/real constant literals(value)

Example: 23.45 here 23 is called Integer part and

0.45 is called decimal part

>This data type does not support to store Binary, octal and hexa decimal

number System data.

>>>a=23.45

>>>type(a)- ---------<class, 'float'>

>>>a=0b1111.0b1010--------error

>>>a=0xab.0xface--------------error

>>>a=0b1111.0b1010------error

>>>a=0xab.0xface--------error

>>>print(a, type(a))-----------300 <class,<'float'>

>>>a=3e2

>>>a=4e-2

>>>print(a, type(a)) ------------0.04 <class,<'float'>

**3) bool**

=>'bool' is one of the pre-defined class

=> The purpose of this data is to store True False values (called boolean literals)

=>Internally True value is treated as 1 False value is treated as 0

Example:

>>>a=True

>>> print(a)-----True

>>> type(a)-----<class 'bool'>

>>> b=False

>>> print(b, type(b))- ---------False <class 'bool'>

>>>print(True+True)------------2 (1+1)

>>>print(True-True)- --------0 (1-1)

>>>print(True+False)--------1 (1+0)

**2) float()**

=> This function is used for converting any valid other type value into float type value.

syntax:- varname2=float(varname1)

Example:

>>> a=12

>>> b=float(a)

>>> print(b, type(b))------------12.0 <class 'float'>

>>> print(float(True)) ---------------1.0

>>> print(float(False)----------------0.0

>>> print(float("12.34"))-----------12.34

>>>print(float("12"))- -------------12.0

Note: print(int(0b1111))--------------------15

print(float(0b1010.0b1010))---------error

print(float(0b1111))- -------------15.0

**3)bool()**

>>This function is used for converting any valid other type value into bool

type value.

syntax:-varname2=bool(varname1)

Examples: (HINT:-Every Non-Zero value is True and Zero val is False)

>>>print(bool(10))------ -True

>>>>print(bool(12.34))-------True

>>>print(bool(0))------------False

>>>print(bool (0.0))--------False

>>>print(bool(2+3j))------------- True

>>>print(bool("python"))--------True

>>>print(bool("0"))------------True

>>>print(bool(" "))-- -----------True

>>>print(bool("")) -------------False

>>>print(bool("0.0"))------------------ True

**4)complex():-**

>This function is used for converting any valid other type value into complex type yalue.

syntax:- varname2=complex(varname1)

Examples:

>>> a=10

>>> b=complex(a)----------(10+0j) <class 'complex'>

>>>print(complex(10.5))- -------------(10.5+0j)

>>>print(complex(True))- -------------(1+0j)

>>>print(complex("12")) (12+0j))

>>> print(complex("12.35"))------------- (12.35+0j)

>>>print(complex("python"))-------------------error---Non-Numerical strings can't convert into complex

>>>print(complex("3x.4y"))---------------------Error

**2)Sequential Categery Data Types:**

**5) str()**

This function is used for converting any valid other type value into str

type value.

syntax:- varname2=str(varname1)

Examples:

>>>a=10

>>>b=str(a)

>>>print(b, type(b))- ------10 <class,'str'>

>>>print(str(12.34))-- --------'12.34'

>>> print(str(True))----------'True'

>>> print(str(-2+4j))-----'-2+4j’

**2**) **bytes data type:-**

="bytes' is one of the pre-defined class

=>An object of bytes allows us to store sequence of +ve integer values with range (0,256), 0 is inclusive and 256 is exclusive

=> To convert one type value into bytes type value, we use bytes()

=>an object bytes is immutable bcoz 'bytes' object does not support item assignment

Examples:

>>>L=[10,20,256]

>>>b=bytes(l)-----------------error bcoz exceds the range(0,256)

>>>L1=[10,20,255]

>>>b=bytes(L1)

>>>type(b----------)<class, 'bytes')

>>> for x in b:

print(x)--------10 20 255

>>>print(b[0])---------------- 10 indexing allowed

>>>for x in b[0:3]:

print(x)--------------10 20 255 slicing allowed

>>>b[0]=122-------------------------Error- bcoz bytes object is immutable.

**3)ByteArray**

**4)Range**

**III) List Categery Data Types**

Purpose:- To store multiple values of same type or different type or both types

=>This Categery contains two data types. They are

1) list

2) tuple

**1) list:**

=>'list' is one of the pre-defined class

=>An object of list allows us to store multiple values of same type or different

type or both types.

=>An object of list type allows us organize both unique and duplicate values. =>The elements of list must be enclosed within Square Brackets [] and the values must be separated by comma (,)

=>An object of list maintains insertion order.

=>On the object of list, we can perform both indexing and sclicing.

=>An object of list is Mutable.

=> To convert one type values into list type values, we use list()

=>Empty list object can be created as follows

Example: >>>L1=[] (or) >>>L1=list()

>>>print(L1)---- [ ] --------empty list

Examples

>>>>L1 [10,20,10,15,-23]

>>>L2=112,"KVR","OUCET",94.25]

>>>print(L1, type(L1))-------->[10,20,10,15,-23] <class, 'list'>

>>>print(L2, type(L2))-------->[12,"KVR","OUCET" 94.25] <class, "list">

Example:

>>>print(L1[0])------------------ 10

>>>print(L1[1:4])---------[20,10,15]

>>>print(L2[-4:-1]---------------[12,"KVR","OUCET"]

Functions in list:

1) append():-

>This function is used for adding an element to the list object at end of list object.

syntax: listobj.append(element)

eg.L1.append(45)

2) insert():

>>This function is used for inserting an element in the list object by specifying valid existing position.

syntax: listobj.insert(validpos, element)

eg.L1.insert(2,78)

3) (a) pop():

>>This function is used for removing an element from the list object by specifying valid existing position.

syntax:- listobj.pop(valid position)

>This function is used for removing an top most / lastest element from the

(b) pop()

list object without specifying valid existing position.

syntax:- listobj.pop()

Examples:

L1=[10,"RS"]

>>>L1.append(12.34)

>>>print(L1)----------------[10,RS,12.34]

>>>L1.insert(2, "NL")-[10,RS, NL, 12.34]

>>>L1.pop(2) -------removed 'NL'

>>>print(L1)- ----------[10,'RS',12.34]

>>>L1.pop(removed 12.34)

>>>print(L1)---------[10,'RS']

4) remove():

This function is used for removing the specific element from list (first occurence only) provided the element must present otherwise we get ValueError

Syntax: listobj.remove(element)

Example:- >>>L1=[10,20,10,90,20,30]

>>>L1.remove(20)

>>>print(11)-------------[10,10,90,20,30]

>>>L1.remove(100)-----ValueError

5) index():

=>This function is used for finding an +ve index of specified element of a

Example:- >>

list provided element present in list otherwise we get ValueError. Syntax: listobj.index(element)

Example: >>>L1=[10,20,10,20,30]

>>>L1.index(200)----------ValueError

**6) copy():**

>>>This function is used copy the content of source list object into destination list object with same content with different address .This type of copy process is called shallow copy.

Syntax:- dest.listobj=sourcelistobj.copy()

Example:- >>>l1=[10,20,”RK”]

>>>L2=L1.copy() //Shallow copy

>>>print(L1,L2)-------[10,20,”RK”] [10,20,”RK”]

>>> Print(id(L1),id(L2)----------------xxxxx234 yyyyyy675

Points of Shallow Copy:

=>Initial content of source and dest are same.

=>Address of source and dest list are different.

=>Modifications are not reflected(Independent).

Points of Deep Copy:

=>Initial content of source and dest are same.

=>Address of source and dest list are Same.

=>Modifications are reflected to each other (dependent)

Syntax:-

Destlistobj=sourcelistobj

Example:

>>>L1=[10,22,”RR”]

>>>L2=L1 //Deep copy

>>>print(L1,L2)------------[10,22,”RR”]

>>>print(id(L1),id(L2))-------------xxx123 xxx123

>>>L1.append(90)

>>>print(L1,L2)-----------------------[10,22,”RR”,90] [10,22,”RR”,90]

Special point:(another way of Shallow copy)

Example:

>>>>l1=[10,22,34]

>>>>l2=l1[:] #shalloew copy with slicing

>>>>print(l1,l2)--------[10,22,34] [10,22,34]

>>>>print(id(l1),id(l2))--------------xxxxx123 yyyyyy456

>>>>l1.append(89)

>>>> print(l1,l2)--------[10,22,34,89] [10,22,34]

**7)Reverse():-**

This fun is used for obtaining reverse of given element of list obj.

Syntax:listobj.reverse()

Ex-

>>>l1=[10,22,33]

>>>print(l1.reverse()) ----------------[33,22,10]

**8)sort():-**

This fun is used for sorting element of list obj in ascending order provides elements must be homogeneous.

Syntax: listobj.reverse()

>>>>l1=[10,30,50,90,40]

>>>>print(l1.sort())-----------[10,30,40,50,90] //ascending order

>>>>print(l1.sort())-----------[90,50,40,30,10] #descending order by usind sort method two times

Special Point:

>>>>print(l1.sort(reverse=True)) ------------Directly display in descending order.

**9) clear():**

=>This function is used for removing / clearing all the elements of list object (object remains same)

syntax:- listobj.clear

Example:- >>>l1=[10,20,30]

>>>print(len(l1))------------- 3

>>>l1.clear()

>>>print(l1.len())-------0

>>>print(l1)----------------[]

Special Point:del()- is one of the pre-defined function present in built-ins module and it is by default imported.

del() is used for deleting the values of any Iterable object(list, str..etc) and object also.

Syntax: del(object name)

(or)

del (objectname [index])

(or)

del(objectname[start:stop])

Example:-

>>>l1=[10,20,30]

>>>del(l1 [0])

>>>print(l1)--------[20,30]

>>>del(l1)

>>>print(l1)------------ Error-->name '11' is not defined

**extend()**

This function is used for adding the content of dest list object to the source

list object content.

syntax:- sourcelistobj.extend(destlistobj)

Example:-

>>> l1 = [10, 20, 30]

>>>l2=[“python”,"AI” ]

>>>l1.extend(12)

>>> print (l1)------------[10,20,30, “python” ,"AI” ]

special point:

We can use operator + for extending the functionality of differnt source list objects into dest list object.

syntax:- destlistobj=sourcelistobj1+sourcelistobj2...+sourcelistobj-n

Example:-

>>>l1 = [10, 20]

>>>l2=["python","java"]

>>>l3=[True,2+3j]

>>>l4=l1+l2 + l3

>>>print(l4)--------------[10,20,”python”,”java”,True,2+3j]

**11)count():**

This is used for counting no. of occurences of a specfic element in a list object.

Syntax:-

listobject.count(element)

Example:-

L1=[10,20,10,10,-100]

>>>l1.count(10)--------3

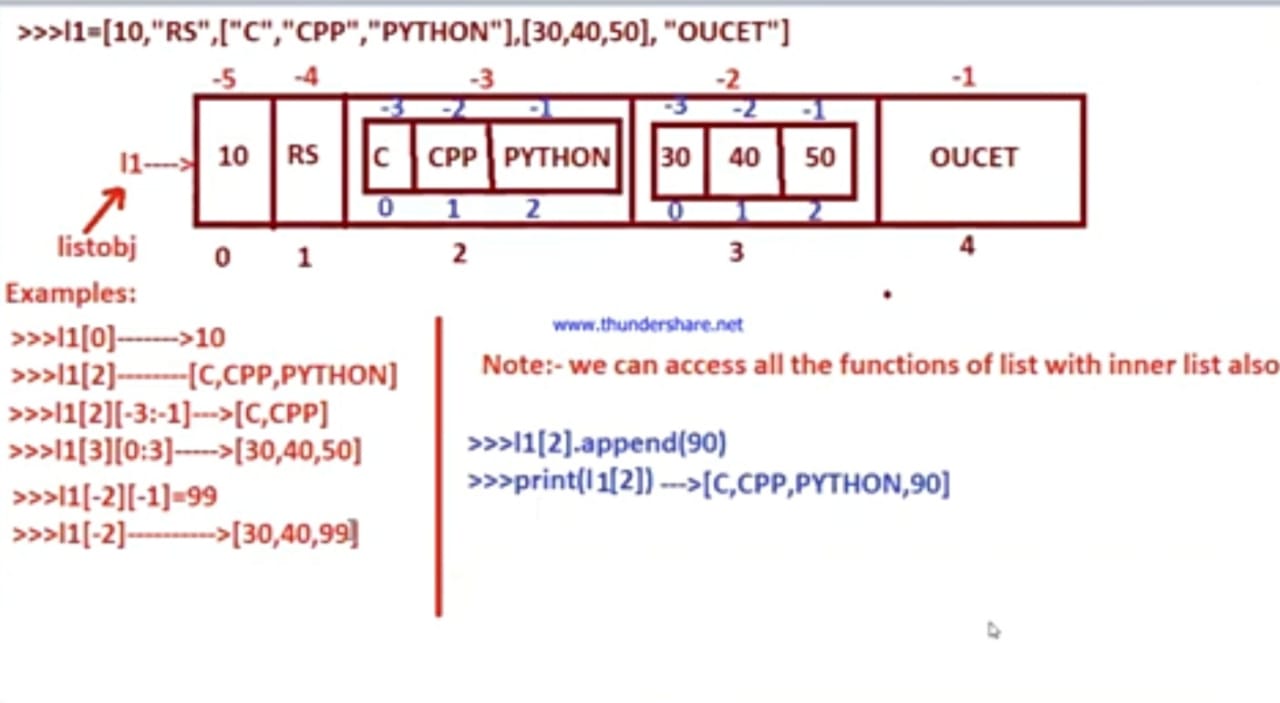
>>>l1.count(-100)-- -----------1

>>>11.count(123)-------- 0

**12)inner/ nested list**

Example:- >>>l1=[10,"RS":["C","CPP","PYTHON"],[30,40,50], "OUCET"]

The diagramatic Representation for list an inner lists is given bellow.



**2) tuple:**

=>'tuple' is one of the pre-defined class

=>An object of tuple allows us to store multiple values of same type or different type or both types.

=>An object of tuple type allows us to organize both unique and duplicate values.

=>The elements of tuple must be enclosed within Braces () and the values must be separated by comma (,)

=>An object of tuple maintains insertion order. www.thundershare.netaA

=>On the object of tuple, we can perform both indexing and sclicing.

=>An object of tuple is Immutable.

=>To convert one type values into tuple type values, we use tuple() =>Empty tuple object can be created as follows

Example: >>>tp=0 (or) >>>tp-tuple() >>>print(tp)----0) <----empty tuple

Example:- >>>tp1=10,"Hyd",34.56

>>>print(tp1, type(tp1))---(10,"Hyd",34.56) <class,"tuple">

**IV) Set Catagery data Types:**

=>This Catagery data Types allows us to store multiple values of same type or different type with Unique Values without maintaining Insertion order.

=>This Catagery two data types. They are

a) set

b) frozenset

**a) set:**

=>'set' is pre-defined class

=>An object of set allows us to store multiple values of same type or different type and both Types.

=>An object of set organize only Unique Elements.

=>An object of set never maintains insertion order

=>Elements of set must be enclosed within curly braces {} and values

separated by comma (,)

=>An object of set is mutable (in the case of add()) and it is immutable in the case of "Item Assignment"

=>On the object of set, we can't apply indexing and sclicing Operations bcoz set object never maintains insertion order.

=>To convert one type of elements into set type, we use set()

=>To create an empty set we use set()

Example:-

>>>s=set()

>>>print(s, type(s)) set() <class, 'set'>

Functions in set:

**1) add()**

>>>This function is used for adding an element to set object

syntax: setobj.add(element)

>>>s={10,"Hyd"}

>>>s.add("India")

>>>print(s) ------------{10,"India","Hyd"}

**2) clear():**

=>This function is used for clearing/removing all elements from set object I

syntax:- setobj.clear()

Example:- >>>s1={10,20"Hyd"}

>>>len(s1)-

>>>s1.clear()

>>>print(s1)-set()->empty

>>>len(s1)---------0

**3) discard():**

=>This function is used for removing the specified value from set object and it won't return any deleted element (if element found the it is removed otherwise not

deleted)

syntax:- setobj.discard(value)

Example: >>>s1={10,20,"Hyd","python"}

>>>s1.discard(10) # here 10 removed

>>>print(s1)----(20,"Hyd","Python"} >>>s1.discard(100) # here 100 not found and not deleted.

**4)remove():**

=>This function is used for removing a value from set object. if the value is unable to remove we get "KeyError"

Syntax:-

setobject.remove(value)

>>>s1={"apple","kiwi","banna"}

>>>s1.remove("apple")

>>>print(s1)------>{"kiwi","banna"}

>>>s1.remove("SB")-----error---->KeyError bcoz "SB" does not

exists in s1

**5) popO**

=>This function removes arbitrarly / random element from given set object.

syntax:- setobj.pop(

>>>s1={10,20,30,40}

>>>s1.pop()---->20 arbitrarly element removed

>>>print(s1)----{10,30,40}

Note:- we can't apply delo on set object for removing an ele

>>>s2.issuperset(s1)-

false

special case:

>>>set().issubset(s1)True >>>s1.issubset(set())------False

>>>set().issubset(set())----True

**6) issubset():**

=>this function returns True provided one set is the sub set of another set otherwise it returns False

setobj1.issubset(setobj2)

**7) issuperset():**

=>this function returns True provided one set is the super set of another set otherwise it returns False

setobj1.issuperset(setobj2)

Example:-

>>>s1={10,20,30,40}

>>>s2={20,10}

>>>s1.issubset(s2)-------------False

>>>s2.issubset(s1)—--------True

>>>s1.issuper(s2)

**8) isdisjoint():**

=>This function returns True provided both sets are having different Elements. This function returns False provided both sets are having at least one common Element.

syntax:- setobj1.isdisjoint(setobj2)

Example:- s1={10,20,30,40}

s2={"java","python"}

s3={10,True}

s4={"java",12.34}

>>>s1.isdis

**8) difference():**

This function is used obtaining only the elements of setobj1 by removing common elements from setobj1 and setobj2.(i.e exclusive elements of setobj1)

syntax:- setobj3=setobj1.difference(setobj2)

Examples:-

>>>s1={10,20}

>>>s2={20,30}

>>>s3=s1.union(s2)

>>>print(s3)-- -{10,20,30}

>>>s4=s1.intersection(s2)

>>>print(s4)--- --{20}

>>>s5=s1.difference(s2)

>>>print($5)-----------{10}

>>>s6 s2.difference(s1)

>>>print(s6)----------{30}

special case: print(

>>>s5=s1.difference(s2)

>>>print($5)

-- -{10}

>>>s6=s2.difference(s1)

>>>print(s6)-- -{30}

special case:

print(s1| s2)------------{10,20,30}---here the symbol (1) is called BitWise OR is used for union

print(s1&s2)-----{20}--here the symbol (&) is called Bitwise AND is used for intersection

print(s1-s2)- -{10}

print(s2s1)--------{30}

**12) symmetric\_difference()**

This function is used for obtaining exclusive elements of both the sets by removing common elements and placing those elements in resultant set obj.

Syntax: resultantsetobj=setobj1.symmetric\_difference(setobj2)

Example:- s1={10,20,30,40}"

s2={30,40,45,65}

$3=s1.symmetric\_difference(s2) print(s3)- -----{10,20,45,65}

**13) symmetric\_difference\_update()**

This function is used for obtaining exclusive elements of both the sets and update sourse set object by removing common elements and never place the elements in resultant set obj.

syntax: sourcesetobj.symmetric\_difference\_update(s2) Example:-

-->>>s1={10,20,30,40}

>>>s2={30,40,45,653

>>>s3=s1.symmetric\_difference\_update(s2)

>>> print(s3)- ->None

>>> print(s1)- ->{65, 10, 45, 20}

**14) update():-**

=>This function updates the elements of suorce set with elements of destnation set object.

syntax: sourcesetobj.update(destsetobj) Example:

>>>s1={10,20}

>>>s2={"java,"python"}

>>>s1.update(s2)

>>>print(s1)----[20,10,java,"python"]

15) copy():

=>This function copies the elements source set object into destination set object (performs shallow copy)

syntax:- dest set obj=sourcesetobj.copy

Example:-

>>>s1={10,20}

>>>s2=s1.copyO www.thundershare.net

>>>print(s1,s2)-[10,20]

[10,20]

>>>s2.add("python")

>>>print(s1,s2)-[10,20]

[10,"python",20]

2)frozenset:

=>'frozenset' is pre-defined class

=>An object of frozenset allows us to store multiple values of same type or

different type and both Types,

=>An object of frozenset organize only Unique Elements.

=>An object of frozenset never maintains insertion order

=>An object of frozenset immutable

=>On the object of frozenset, we can't apply indexing and sclicing Operations. =>To convert one type of elements into frozenset type, we use frozenset()

=>To creaate an empty frozenset we use frozenset()

Example:- >>>fs=frozenset()

>>>print(fs, type(fs))->frozenset() <class, 'frozenset'>

Note:- The functionality of frozenset is similar to set but an object of frozenset belongs to immutable and object of set is both immutable (item assignment) and mutable(add())

Note:- Frozenset contains the following functions.

a) issuperset

b) issubset()

c) disjoint()

d) union()

e) intersection()

f) copy()

g) difference()

h) symmetric\_difference()

**V) dict categery data type:**

=>This categery contains only one data type.

a) dict

**a) dict**

=>This data type allows us to store the data in the form of (key, value)

=>In (Key,value), the values of Key represents Unique

the values of Value represents may or may not be Unique =>The elements dict object must be enlcosed within curly braces {} and www.thundershare.net whose (key, value) must separated by colon(:) and terminated by comma( ;). =>To convert one type of values into dict type we use dict() and it can also use to create empty dict object.

=>An object of dict is mutable.

create a dict:-

=>We create two types of dict objects. They are

b) non-empty dict

a) empty dict

syntax for empty dict:

dictobj={}

(or)

dictobj=dict()

Example

>>>d1=dict{}

Or

>>>d1=dict()

>>>print(d1,type(d1))-----------{} <class,’dict>

Syntax for adding (keys and values) to empty dict object

dictobj[keyname-1]=val-1

dicyobj[keyname-2]=val-2

…..

dicyobj[keyname-n]=val-n

here if keyname is str type then keyname must written within single/double Quotes. if keyname is numerics, we write keyname directly without quotes.

Example:

d1['stno']=123

d1[12]="OUCET"

print(d1)-----------{'stno': 123, 12:"OUECT"}

d1['place']="Hyd"

print(d1)----------{'stno': 123, 12:"OUECT",'place":"Hyd"}

Syntax for creating non-empty dict:

dictobj={keyname-1 : vall, keyname-2 : val2,...keyname-n: val-n}

Example: d1={'stno':100, 12:"OUCET", 'place' : "hyd"} print(d1)---{'stno' : 100, 12:"OUECT",'place':"hyd"}

updating the values of dict:

syntax:- dictobj[keyname]=new value undershare.net

=> If keyname is already existing in dict obj then old value is replaced with new value. if keyname is not existing in dict obj then (keyname,new value) is inserted newly in dict obj.

>>>d1={10:"AI", 11:"Python"}

>>>d1[10]="DS"----------- # updated entry

>>>print(d1)->{10:"DS",11:"Python"}

>>>d1[12]="Web Dev" # new entry

>>>print(d1)-------[10:"DS",11:"Python",12:"Web Dev"}

Functions in dict:

1) get()

2) values()

3) keys()

4) items()

5) popO

6) popitem()

Functions in dict:

1) get():

=>This function is used for obtaining value of value from dict object by passing value of Key. If key does not does not exists we get KeyError.

Var=dictobj.get(key)

Example:

>>>d1=[10:"OUCET",11:"JNTUH"}

>>>val=d1.get(10)

>>>print(val)-------OUCET

2) values():

=>This function is used for obtaing all the values of dict object.

syntax: var=dictobj.values()

Examples:

>>>vals=d1.values()

>>>print(vals)------------------>["OUCET","JNTUH"]

3) keys():

=>This function is used for obtaing all the keys of dict object.

syntax: var dictobj.keys()

Examples:

>>>keys=d1.keys()

>>>print(keys)------------------>[10,11]

4) items():

=>This function is used for obtaing all the key and values of dict object. syntax: var-dictobj.items()

Examples: kvs=d1.items()

print(kvs) ----------------------(10,"OUCET"), (11,"JNTUH")]

5) pop():

=>This function removes (key, value) from dict object by passing key

syntax: dictobj.pop(key)

Example:

>>>d1.pop(11)-------->"JNTUH"

>>>print(d1)------>{10:"OUCET"}

6) popitem():

This function removes latest element / top most element. syntax: dictobj.popitem()

example: d1.popitem()

7) copy():

This function performs shallow copy of dict object

Syntax: dictobj2-dictobj1.copy()

Example:-

d2=d1.copy()

>>>print(d1)---------------->{10:"OUCET"}

>>>print(d2)---------------->{10:”OUCET”}

8) clear():

This function clears/removes all elements from dict object it becomes

empty.

syntax:-dictobject.clear()

Example: d1.clear()

9) update():

This function updates the one dict obj elements with another dict object elements.

syntax: dictobj1.update(dictobj2) -------here dictobj1 is updated with dictobj2 elements

Example:

>>>d1={10,"OUCET"}

>>>d2={11:"JNTU",12:"KU"}

>>>d1.update(d2)

>>>print(d1)---------------(10,"OUCET",11:"JNTU",12:"KU"}

**VI) None categery Data Type:**

=>This data type is used for string a value which is not any valid python data

type value i.e None

=>The value None is of type <class, 'NoneType">

Example:

>>>x=None

>>>print(x, type(x))----- None <class, 'NoneType">

**No.of approaches to develop python program**

=>We can develop python program in 2 ways.

1)By Using Interactive Mode approach

2) By Using Batch Mode Approach

1. By Using Interactive Mode approach:

=>In this approach, we give one statement at a time and we get one output at a time.

=>The limitation of this approach is that unable execute batch of statments. Example:- python cmd prompt

python IDLE shell

2) By Using Batch Mode Approach:

=>This approach always recommed to develop python based applications / program, where it can contain block of statements and those statements must be saved on some file name with an extension .py (source code).

=>The Batch Mode Programming can be with the following.

a) Python IDLE shell

b) Edit plus (from 4.0 version) www.thundershare.net

c) IDE's (Integrted Development Environment)

i) PyCharm

ii) spider

iii) Jupiter Anakonda...etc

a) Batch mode programming by using Python IDEL Shell:

=> Python IDEL Shell is coming along with Python Software Installation.

steps: 1) launch Python IDLE shell

2) Choose File->New File

(ensure that a window will be opned and there we can develop/ write the program)

3) Write the source and save it on file name with an extension .py (Ex First.py)

(OR)

4) Run The program. Choose Run->Run Module (15) and ensure the result.

OR

=>perform (1), (2) and (3) steps

=>Run the python program by going to cmd prompt and use tool 'py'

syntax: py filename.py

python filename.py

Example:

f:\\python-kvr-4pm\basics> py mul.py

obtain the result

b) By using Edit plus (from 4.0 version)

steps:

1) lauch Edit Plus

2) Choose File--->New->others-->python (click) [ensure that window will be opened]

3) Write the source code and save it one some file name with an extension .py [ example:--- sum12.py]

4) execute the python program by coming to the command Prompt

Example:-f:\python-kvr-4pm\basics>py sum.py

ensure that we get result

steps:

1) launch Edit Plus

2) Choose File->New->others--->python (click) [ensure that window will be opened]

3) Write the source code and save it one some file name with an extension .py

[ example:--- sum12.py]

4) execute the python program by coming to the command Prompt

Example:-f:\python-kvr-4pm\basics>py sum.py

ensure that we get result.

**Reading the data from the key board**

=>To read the data from the key board, in python we have 2 pre-defined functions. They are

a) input()

b) input(message)

a) input()

=>This function is used for reading any type of data dynamically from key board and returns in the form str.

syntax: varname = input()

b) input(message):

=>This function is used for reading any type of data dynamically from key board and returns in

the form str.

syntax:varname = input()

b) input(message):

=>This function is used for reading any type of data dynamically from key board and returns in the form str by prompting the message to end-user on the console.

Syntax:-varname=input(User Prompting Message)

print("Enter any Value:")

a=input()

• Example:-

(OR)

a=input("Enter a any value:")