**PYTHON**

* Python is the general-purpose high-level programming languages.
* Python is used to develop any type of application either desktop or web application.
* Guido Vam rossam is the father of python which is developed in 1989 while working nri in Netherlands.
* Officially it is made available for the public in February 28th 1991.
* We use python to write very big line of code in very less code.
* C & Java are **statically** typed programming language.
* Python is Dynamically Typed programming language.
* Python name takes from Tv show BBC channel 1969 to 1974 Monty Python’s circus episode.
* We use software python idle 3.8.3

**Ex:-** Print(‘Hello World’)

a,b=10,20

Print(a+b)

Type(a)--------> <class=’int’>

A=True

Type(a)-------><class=’boolean’>

It borrows **functional programming language** from c and **oops** from c++. And **scripting language** from perl and shell script. **Modular features** from Modula-3. Modular means divided into modules. Function al means without have class and object call a function.

**EX: -** def f1():print (‘Good Morning’)

F1() -------------------------->Good Morning.

Most of the syntax are borrowed from C and ABC Language.

**Where we can use Python?**

* We can devlop Desktop Application (Windows Application).
* We can devlop WEb Application(standalone)-------->DJango----->Django is a framework to devlop web Application.
* We can devlop Database Application like jdbc.
* Games
* Data Analysis
* Machine learing
* AI
* For IOT Application.

**Which Company use Python?**

* GOOGLE
* YouTube
* drop box
* NASA
* Facebook
* Instagram
* eBay.

**Features of Python**

* Simple and easy to learn, there are only 33 reserved keywords in python

**ex: -** for I in range (10):

Print(I)

**ex:-** x=10 if a<b else 20

* Free ware and Open Source.
* High Level Programming Languages.
* Platform Independent (Write Once Run Anywhere)
* Portability (moving from one place to another without making any changes)
* Dynamically Typed
* Both Procedure oriented and Object-Oriented Programming language.
* Interpreted
* Extensible (we can use some other language code in python, we can Improve performance of the application)
* Extensive Library
* Embedded (we can use python code in other languages)

EX: -- from random import \*

Print(randint(0,9), randint(0,9), randint(0,9), randint(0,9)) ===4-digit random number

**Limitation of Python: -**

* Performance wise it is not up to the mark because it is interpreted language.
* It is not frequented for mobile application.
* Myth: - python is not suitable for large scale enterprise application

**Flavors of Python:** -

* CPython  
  jython or jpython
* Iron python
* Pypy======> PVM====>JIT
* Ruby Python
* Anaconda Python (to handle big data)
* Stack less (python for concurrency)

**Versions of Python: -**

Python 1.0 introduced in Jan. 1994

Python 2.0 introduced in October 2000

Python 3.0 introduced in Dec 2008

Python 3.6.3 introduced in 2016

Python 3.8.2 introduced in 2019(Any new version provide support for old version)

**Identifiers: -**

Variable name, Method name, Class name

**Rules of Identifiers: -**

* Python only support symbols like a- z , A-Z,0-9, \_
* Identifiers should not Starts with digit
* Case sensitive
* We can’t use a reserved word as identifiers.
* No length limit for python identifier it may be crore.
* If any identifiers start with the \_ x then it would be private and if starts with \_\_x strongly private and if starts with \_\_x\_\_ then it is called language specified.

**Reserved Keywords: -**

33 reserved words are there.

[‘FALSE’,’NONE’,’TRUE’,’AND’,’ASSERT’,’AS’,’BREAK’,’CLASS’,’CONTINUE’,’def’, ‘del’, ‘else’ , ‘elif’, ‘except’ , ‘finally’, ’for’, ’from’, ‘if’, ‘import’ ‘is’ , ‘in’ ,’import’, ‘in’ , ‘is’ , ‘lambda’, ‘nonlocal’, ‘not’, ‘or’, ‘pass’ , ‘raise’, ‘return’, ‘try’, ‘while’, ‘with’ ,‘yield’]

**Datatypes: -**

**Datatypes** represent the type of value represented by the variable.

In python all datatype implemented is an object.

* Int (all Integral value)
* Float (all floating-point value)
* Complex (all complex value scientific calculation)
* Bool (all Logical value)
* Str (all string value)
* Bytes (all byte group value)
* Byte array (Bytes and Byte array are almost same)
* Range (range of numbers)
* List (group of values)
* Tuple (List of values likers mutable immutable)
* Set (list of values without duplicates)
* Frozen set (like set but not modify)
* Dict (dictionary like key value like map and hash)
* None (nothing)

**Inbuilt functions: -**

* In python everything is an object**.**

**Print (): -** to print anything.

**Type (): -**tell about types of something.

**Id (): -**to find address of object.

**Int Data Types:-**

to represent integral value whole number. Either 10 ,12 or 1213233343434.

* **Long is not available in python 3.**
* **Decimal form(base-10)**

0 to 9

A=7986

* **Binary form (base 2)**

O and 1

A=0b1111 (positive number)

B=-0b1010 (negative number)

* **Octal form (base 8)**

O to 7

A=0o1111

* **Hexadecimal form (base 16)**

0 to 9 and A to F or a to f

A=0x1111 or A=0xFace

* In python you will get result only in decimal form.

**Base Conversion: -**

**Bin (): -** to convert any base value to binary

**Oct (): -** to convert any base value to octal

**Hex (): -** to convert any base value to hexadecimal value

**Float Data Types : -**

All the floating-point value.

**Ex: -** 12.3451 is correct

0o123.46 is not correct

* Exponential form 1.2e3 and 1.2e100 is valid.

**Complex Data Types: -**

It is used to represent complex number.

**Ex:-**a+bj = 10+20j or 10.5+2.3j

Where a=real part

B=imaginary part

J^2=-1

**Bool Data types: -**

We can use this datatype to represent Boolean value or logical values .

**True** and **False** where **T** and **F** must be capital if you use small **t** and **f** it will give error.

True+True==>2 (internally **True** is treated as **1** and **false** is treated as **0**)

True+False==>1

**Str Data Types: -**

In python string we can write under ‘ ‘ and also ‘ ’. but ‘ ‘ code is recommended.

**Ex**:- ‘durga’

For multiple line use ‘’’ ‘’’’.

**Ex**:- ‘’’durga

Soft'’’

Ex:- ‘’’Durga sir ‘python’ is very helpful.’’’

**Slicing operator: -**

Slice means one piece.

Ex:- s=‘durga software solution’

s[2:8]

**Typecasting or Type coercion: -**

**Int(): -**

**Ex: -**Int(10.50)------->10

Int(10+20j)------>Type Error because we can't convert complex number into int and float.

Int(True)---->1

Int(False)----->0

Int(‘10’)-------->10

Int(‘10.5’)--------->Value Error

**Float():-**

float (10)------->10.0

float (10+20j) ------>Type Error because we can't convert complex number into int and float.

float (True)---->1.0

float (False)---->0.0

float (‘10’) -------->10.0

float (‘10.5’) --------->10.5

float(‘ten’) --------->Value Error

**Complex (): -**

**It is used tom convert other types to complex.**

**Form 1: -complex(x)--->x+0j**

**Ex: -** complex(10)=====>10+0j

complex(10.5)=====>10.5+0j

complex(True)=====>1+0j

complex(False)=====>0j

complex(‘10’)=====>10+0j

complex(‘10.5’)=====>10.5+0j

complex(‘ten’)=====>Value Error

**Form 2: -complex(x,y) --->x+yj**

**Ex:-** complex(10,20)=====>10+20j

complex(True,False)=====>1+0j

complex(10,20.5)=====>10+20.5j

complex(‘10’,’20’)=====>can’t take second argument if first one is string

**Bool(): -**

**Bool() for int and float argument.**

**Ex: -** Bool (1) ===>True

Bool (0) ====>False

Bool(10) ====>True

Bool(10.5) ==>True

Bool(0.0) ===>False

Bool(0.1) ===>True

**Bool() for complex argument.**

**Ex:-**bool(0+0j) ===>False

Bool(1+0j) ===>True

**Bool() for string argument**

**Ex:-** Bool(‘‘) ====> if empty string value will be False

Bool(‘1.1’) ===>True

Bool(‘ ‘) ====>True

**Str(): -**

**You can convert any form to string type.**

**Ex:-** str(‘4+5j’) ===>'4+5j’

**Mutable and Immutable :-**

All fundamental datatype are immutable. Int, float, complex, string, bool are the fundamental datatypes.

If all varaibles are pointing to the same object then adress will be always same.

V1=’Hyd’ ----|id(v1)

V2=’Hyd’------| id(v2) same address of all variables 49706144

V3=’Hyd’------| id(v3)

V3=’warangle’-----| id(v3)----->address will be 47480192

Ex:- x=10

Y=10

X is y----->True(but it is only applicable for 0 to 256)

X=257

Y=257

X is y------>Fa

X=23.7

Y=23.7

X is y------>False(for floating value it is not applicable)

X=25+10j

Y=25+10j

X is y------>False

**Byte Data Type: -**

It represents a group of byte number just like an array.

If you want to represent binary data like images and videos then internally it uses byte datatype.

Ex: - x=[10,20,30,40]

B=bytes(x)

Type(b)------------>Bytes

B[0]=10 , b[1]=20, b[-1]=40

B[0 : 3 ]-------->b ‘\n\x14\xle’

For x in b: print(x)------------------>10,20,30,40

1.byte must be in the range 0 to 256.

**Ex**: -

X=[20,40,256,98] -----------> if it is 257 or more it will give error.

2.bytes datatype is immutable. If you try to change you will get error.

Bytes object does not support item assignment.

**Ex**: - X=[20,40,256,98]

B=bytes(x)

B[0]------>20

B[0]=115------------->it will give error.

Bytes datatype and byte array are almost same .

But difference is byte array is mutable. And bytes datatype is immutable

**Ex**: - X=[20,40,256,98]

B=bytearray(x)

Type(b)---->byte array

B[0]=253

For x in b: print(x)------------------> 253,40,256,98

**List Data Types: -**

If you wants to represent the group of objects in the single entity where,

Insertion order is preserved, and duplicates are allowed.

It is heterogeneous and growable in nature.

Value should be enclosed with []

Ex: - l=[]

Type(l)------>list

l.append(10)

l.append(20)

l.append(30)

l.append(10)

Print(l)------------>[10,20,30,10]

l.append(‘durga’)

l.append(None)

Print(l)----------> [10,20,30,10,’durga’,None]

L[0]--->10, l[-1]--->none

L[1:5]------->[20,30,40,’durga’]

l.remove(10)----->[20,30,40,’durga’,none]

**Note**:- in python \* operator is applicable but one element should be number and another should be later.

Ex: - s=[10,’durga’,’pawan’]

S1=s\*2

Print(s1)----->[10,’durga’,’pawan’, 10,’durga’,’pawan’]

**Tuple Data Type: -**

List and Tuple is almost same, but List is mutable, and Tuple is immutable.

By using () we represent the tuple.

Ex: - t= (10,20,30,’ Durga’, True)

T [0] =100----->error tuple doesn't support item assignment.

T=(10,20,[30,40])---->valid

**Range () Data Type: -**

Range datatype represent a sequence of values.

Range is only applicable for number not also for floating value.

Elements represents inside range is immutable.

**Form 1:** -r= range (10) ,,,,,,,r=range (100)

Range represent 0 to 9

**Form 2:** - r=range (10,30), if I represent 10 to 29.

**Form 3:** - r=range (10,50,5), if I represent 10 to 50 incremented by 5 .----------->10,15,20....45,50

**Set Data Type: -**

It is group of individual objects.

Set datatype is represented by {}.

Set doesn’t allow duplicates and insertion order.

Indexing slicing such type of terminology not applicable in set.

Set is mutable.

Ex: - s= {10,20,30,10,20,30}

S= {10,20,30} ------it removes the duplicates.

S[0] ---->error et object does not support indexing

S[1:2]---->set object is not subscriptable

s.add(‘durga’) ----> {a=10,20,30,’durga’}

s.remove(10) ----->{20,30,10,20,30,’durga’}

**Frozenset() Data Type: -**

It is used when we have group of token where no can change anything.

It is just as same as set but add and remove is not applicable.

**Ex: -** s={10,20,30,40}

Fs=frozenset(s)

Fs------>frozenset{(10,20,30,40)}

**Dict Data Type:-**

Dict means dictionary which represents a group of key value pairs.

Key must be unique, but value may be duplicates.

It also represents using {}.

{} ----> empty dictionary------>for empty set s=set ()

**Ex: -** d={10:’durga’ , 20:’pawan’, 30:’kajal’} ----->in Jason same way we represent.

D[100] = ‘sunny’

D[200]=’bunny’

D[100]=’chinni’----->it replaced chinni by sunny------>{100:’chinni’ , 200:’bunny’}

**None Data Type: -**

Def f1():

Pass

**Escape characters: -**

\n

\t

\b

\f

\r

\’

\’’

\\

\v

**Constant: -**

In java constant are there but in python it is nothing like final.

Final int x=10

X=11;\\CE:cannot

MAX\_VALUE=10

**Operators :-**

1. Arithmetic Operator

2. Comparison Operator or Relational Operator

3. Logical Operator

4. Bitwise Operator

5. Assignment Operator

6. Special Operator

1. **Arithmetic Operators:-**

**+ ---🡪Addition Operator**

**- ---🡪Subtraction Operator**

**\* --🡪Multiplication Operator**

**/ --🡪Division Operator**

**% --🡪Modulo Operator**

**\*\* -🡪Exponent or Power Operator**

**// --🡪Floor Division Operator**

**Ex: -** a=10

b=2

print (‘a+b=‘,a+b)#12

print (‘a-b=‘,a-b)#2

print(‘a\*b=‘,a\*b)#20

print(‘a/b=‘,a/b)#5.0 -----🡪represent answer in float

print(‘a%b=‘,a%b)#0

print(‘a//b=‘,a//b)#9//2=4, 9.0//2.0=4.0 ----🡪represent answer in int(if both argument are int) and float(if both argument are float) both.

print(‘a\*\*b=‘,a\*\*b)#10^2=100 #10^-2=0.01 #10+2j \*\* 10+5j=-59555……+3245….

**Note:-** In java **+** operator also use for **str** type also which is called **string concatenation**.

But in python it gives error. If want to use then both argument should be string only.

**string concatenation operator:-**

**Ex**:- ‘durga’+3---🡪error

**Ex**:- ‘durga’ + ‘3’ -🡪’durga3’

**Ex**:- ‘durga’ + str(3) 🡪’durga3’

**string multiplication operator:-**

**Ex:-** ‘durga’ \* 3 🡪 durgadurgadurga

‘durga’\*’3’ 🡪 error

‘durga’ \* 3.0 🡪 error

1. **Relational Operators:-**

**<, >, <=, >=, ==, !=**

**Note:-** we can apply relational operator with **number** and **string** also it will compare both string based on **Unicode** or **asicii** value and also with **Boolean** but both value should be Boolean only not also single one string.

**Note:- space matters a lot in python.**

**Ex:-** a=20 //’Durga’ //True

b=10 //’Darga’ //False

print (‘a>b=‘,a>b)#T

print(‘a<b=‘,a<b)#F

print(‘a>=b=‘,a>=b)#T

print(‘a<=b=‘,a<=b)F

**Chaining of Relational operator: -**

If all the condition is satisfied then it gives result true.

**Ex**: -10<20 #True

10<20<30<40<50 #True

10<20<30<40<50>60 #False

**Equality Operator: -**

**== , != (it compare only content not the address)**

**If content is same then True otherwise if not same then False**

**Ex:-** 10==20 #False

10!=20 #True

10==True #False

‘durga’==’durga’ #True

10==’durga’ #False

10==20==30==40==50 #False

10==5+5==3+7==2\*5 #True

‘a’==97 #False

(10+2j)==(10+2j) #True

(10+2j)==’durga’ #False

10==10.0 #True

10==10.1 #False

10==’10’ #False

1==True #True

10.10==10.1 #True

**Logical Operators: -**

**AND ===🡺**if both are true then only true

**OR ====🡺**if atleast one argument is True the True

**NOT ===🡺**

**For non-boolean types:**

0 means False

Non-zero means True

Empty string =🡺 False

X and y

If x evaluates to false then result is x otherwise returns y.

0 and 20 #0

10 and 20 #20

X or y

If x is evaluates to true then returns x otherwise return y

10 or 20 🡺10

0 or 20 🡪20

0 or 0 🡪0

0 or True 🡪 True

10 or 10/0 🡪 10

0 or 10/0 🡪 divide by 0 error

**Not x:**

Not 10 => False

Not ‘’ =>True

Not 0 => True

**Bitwise Operators: -**

**Applicable only for int and boolean**

**& 🡺 if both bits are 1 then only 1 otherwise 0**

**| 🡪if atleast one bit is 1 then 1 otherwise 0**

**^ 🡪x-or 🡪if both bits are different then 1 otherwise 0**

**~ 🡪 bitwise complement operator**

**<< 🡪bitwise left shift( right hand side vacant cells fill with 0s)**

**>> 🡪bitwise right shift (left hand side vacant cells fill with sign bit)**

**For +ve numbers🡺0**

**For –ve numbers 🡺1**

**Ex:-** 4&5 🡪100&101 🡪100 🡪4

4|5 🡪100|101 🡪101 🡪5

4^5 🡪100^101 🡪001 🡪1

~4🡪4🡪000000000..100🡪1111111…011🡪111111111…011+0000001🡪101🡪-5

~True 🡪~1🡪000000000000001🡪1111..0🡪1111…0+1-🡪-2

not True 🡪False

True & True 🡪valid

10.5 & 2.6 🡪Invalid

10<<2🡪10🡪00000..00001010🡪000..0000101000 🡪40

10>>2🡪10🡪0000.001010 🡪000000..0010🡪2🡪sign bit

-10>>2🡪-3

-10<<2🡪-40

Print(True<<2)🡪1<<2🡪00000..0001🡪000…000100🡪4

Print(True>>2)🡪1>>2🡪00000.0001🡪0000000..00🡪0

**Assignment Operators:-**

Increment ++ and decrement – operator are not available in python, if you use it will give error**.**

**=**

**Ex: - x=10**

**a , b , c, d=10,20,30,40**

**a , b , c,d=10,20,30---🡪error**

**+=**

**-=**

**\*=**

**/=**

**%=**

**\*\*=**

**//=**

**Ex:-** a=4

a&=5

print(a**)**

**Ternary Operator : -**

**?:**

X=(condition)?firtsValue:secondvalue

x=firstvlue if condition else secondValue

a,b=10,20

x=30 if a>b else 40

print(x) -🡪40

Ex:- a=int(input(‘Enter First Number:’))

b=int (input(‘Enter Second Number:’))

min =a if a<b else b

print (‘Minimum Value:’, min)

x=firstValue if condition1 else secondValue if condition2 else thirdvalue

x=10 if 20<30 else 40 if 50<60 else 70 🡪10

1. **Special Operators**

**1.Identity operators**

**Is:- It compare the address**

**Ex:-**A=10

B=10

Print (a is b) 🡪True

List1= [10, 20, 30]

List2= [10, 20, 30]

Print (id (list1))🡪57478120

Print (id (list2))🡪18154152

Print (list1 is list2) 🡪False

Print (list1 == list 2) 🡪True

**Is not:-**

**Ex:-**Print (a is not b)🡪 False

1. **Membership operator:-**

**In:-**

**Not in:-**

**Ex:-** List1= [10, 20, 30]

Print (10 in list1) #true

Print (70 not in list1) #true

**Ex: -** s=’Hello Learning python is very very easy!!!’

Print (‘Hello’ in s) #True

Print(‘L’ in s) #True

**Operator Precedence: -**

**() :- highest priority goes to parenthesis in python.**

**\*\* :- exponential operator or power operator**

**~,- unary operator**

**+,/,%,//**

**+,-**

**<<,>>**

**&**

**^**

**|**

**>,>=,<.<=,==,!=**

**=,+=,-+,\*=**

**Is , is not**

**In , not in**

**Not**

**And**

**or**

**Ex**:- a=30

B=20

C=10

D=5

Print((a+b)\*c/d) #100.0

Print((a+b)\*(c/d)) #100.0

Print(a+(b\*c)/d) #70.0

**Module:- a group of functions, variables, classes**

**Bydefault module are available to python but not for our program so we need to import.**

**Math module:-**

**Sqrt()**

**Ceil()**

**Floor()**

**Pow(x,y)**

**Factorial()**

**Gcd()**

**Sin()**

**Cos()**

**EX:-**Import math

Print(math.sqrt(16)) #4

Print(math.pi) #3.14

**Aliasing: -**

If we want then we can change or we can give our own name sowe use aliasing then we don’t need to write math again again.

**Ex: -** *import math m*

*Print(m.sqrt(25))*

*Print(m.pi)*

*Or also you can write like that in one line*

***Ex****:- from math import sqrt,pi or \**

*Print(sqrt(25))*

*Print(pi)*

**Constants:-**

**Pi=🡺3.14**

**E=🡺2.71**

**Inf=🡺Infinity**

**Nan=🡺not of a number**

**Input and Output statements:-**

**Python-2:**

X=raw.input(‘enter some number’)🡺 str

We required to use typecasting functions

Y=input(‘Enter some number’)

Not consider as str type what every type provided

We are not required typecasting

**Python -3:**

**Input function always considered as string . then we have to typecast it.**

**X=input(‘Enter some number:’)**

**Ex:-** x=int(input(‘Enter first number’))

Y=int(nput(‘Enter second number’))

Print(‘The sum:’,x+y)

**Or**

Print(‘The Sum:’, int(input(‘Enter First number:’))+int(input(‘Enter Second Number’)))

**Ex:- Employee Details:**

eno =int(input(‘Enter Employee number:’))

ename =(input(‘Enter Employee Name:’))

esal=int(input(‘Enter Employee salary:’))

eaddr =(input(‘Enter Employee address:’))

married=bool(input(‘Enter Employee married status:[True][False]’))

print(‘Please confirm Information..’)

print(‘Employee Number:’,eno)

print(‘Employee Name:’,ename)

print(‘Employee Salary:’,esal)

print(‘Employee Address:’,eddr)

print(‘Employee Marriage:’,married)

**If you want to take multiple input in one line:-**

**Ex:-** a,b=[int(x) for x in input(‘Enter 2 numbers:’).split()]

print(‘The Product:’, a\* b)

**#Read 2 float values from the keyboard which are specified with, separation and print sum**

10.2, 10.6

a,b,c,d=float(x) for x in Input(‘Enter 2 float values:’).split()] -🡪with split only str type only.split(5)-🡪error but split(‘5’)

print(‘The Sum:’,a+b+c+d)

**for List: eval()**

**Ex:-** x=eval(input(“Enter some data:”)) 🡪1,2,3.4,5 or ‘pwn’,’kajal’,’suraj’ or 1,’pawan,’m’,25

print(type(x))

for x1 in x:

print(x1)

**Command Line Arguments: -**

The argument which are passing from command line this is called command line argument.

**Ex**:- py test.py 10 20 30 40 50(command line argument)

**Argv :-** Predefined variable internally it hold all command line value.

Argv is the list type in python

Argv variable in sys module

**If you want to use argv from sys module then we need to import it.**

**Ex:-**From sys import argv

Print(type(argv)) //list

Print(argv[1:])

Print(argv[:])

Print(argv[0])  
py test.py 10 20 30-----🡪 test.py 10 20 30 -🡪py test.py ‘sunny Leony’

**#read a group of int values from the keyboard as cmd line arguments and print sum**

From sys import argv

Args=argv[1:]

Sum=0

For x in args:

n=int(x)

sum+=n

print(‘The sum:’, sum)

**Output Statements:-**

**Print():-**

**To print output or some statement to the console.**

**Form-1:-**

**Print() without any argument. Purpose to use this a new line or empty line character by default will be inserted.**

**Ex:-**Print(‘Hello’)

Print()

Print(‘Gud Morning’)

Print(‘How are you’)

**Form-2:-**

**Print(string) with string argument.**

**Ex**:-Print(‘Hello\nGudMorning’)--🡪 gud morning in new line

Print(‘Hello\tGudMorning’) -🡪gud morning tab space

Print(‘Durga’+’software’)🡪Durga Software 🡪**both argument should be str only**

Print(‘Durga’ \* 3) 🡪DurgaDurgaDurga 🡪**one argument should be str and other one should int.**

Print(‘durga’+’software’)🡪durgasoftware

Print(‘durga’,’software’)🡪durga software

**Form-3:-**

**Print**() with variable number of arguments means any number of argument.

**Ex**:-a,b,c=10,20,30

Print(‘The values are:’,a,b,c)🡪 The values are 10 20 30

**Sep(separator) attribute:-**

**It is used to separate attribute by space or coma or colon within the attribute.**

**Ex:-** print(a,b,c,sep=’,’)🡪 10,20,30

**Print() with end attribute:-**

**It is used to separate multiple line in one line.**

**Ex:-**print(‘Hello’,end=’ ’)

print(‘Students,end=’ ’)

print(‘Python,end=’ ’)

print(‘easy’,end=’ ’)--🡪Hello Students Python easy

**print(formatting string):-**

**It is used to represent or mix up the varaiable.**

**%i == > int type**

**%d == > int type**

**%f == > float type**

**% s == > str type**

**Print(‘formatted string’ %(variable list))**

**Ex:-** a,b,c=10,20,30

Print(‘a value is %i’ %a)🡪 a valuer is 10

Print(‘a value is %i and b value is %i’ %(a,b))🡪 a value is 10 and b value is 20

**Ex:-** name=‘Durga’

L=[10,20,30]

Print(‘Hello Durga the list is: %s’ (name,l)) 🡪Hello Durga the list is: [10,20,30]

**Print() function with replacement operator:**

**{} == > replacement operator**

**Ex:-** name=’Durga’

Salary=10000

Gf=‘sunny’

Print(‘Hello {0} your salary {1} and your Girl friend {2} is waiting’ .format(name,salary,gf)) **🡪 Hello Durga your salary is 10000 and your Girl friend is waiting**

Print(‘Hello {} your salary {} and your Girl friend {} is waiting’ .format(name,salary,gf)) **🡪 Hello Durga your salary is 10000 and your Girl friend is waiting**

Print(‘Hello {x} your salary {y} and your Girl friend {z} is waiting’ .format(x=name,y=salary,z=gf)) **🡪 Hello Durga your salary is 10000 and your Girl friend is waiting**

**Flow Control:**

1. **Conditional Statements/ selection statements:-**

**If x**

**Option-1**

**else**

**option-2**

**if x**

**action-1**

**elif x**

**action-2**

**elif x**

**action-3**

* **If**
* **If-else**
* **If-elif-else**

**If- statemets:-**

**Ex:-** name=input(‘Enter Name:’)

If name==‘Durga’:

Print(‘Hello Durga Good morning’)

Print(‘How are you’)**🡪 Hello Durga Good morning How are you**

**If-else statements:-**

**Ex:-** name=input(‘Enter Name:’)

If name==‘Durga’:

Print(‘Hello Durga Good morning’)

else

print(‘Hello Guest Good Morning’)

print(‘How are you’)

**if-elif-else statements:-**

**Ex:-**brand=input(‘Enter your favourite Brand’)

If brand==‘RC’:

print(‘It is a childrens brand’)

elif brand==‘KF’:

print(‘It is not that much kick’)

elif brand==‘KO’:

print(‘It is too light’)

elif brand==‘FO’

print(‘Buy one get one free’)

else:

print(‘others brands are not recommended’)

**#program to find biggest of given 3 numbers from the keyboard**

**Ex**:- n1=eval(input(‘Enter First Number’))

n2=eval(input(‘Enter Second Number’))

n3=eval(input(‘Enter Third Number’))

if n1>n2 and n1>n3:

print(‘Biggest number is:’,n1)

elif n2>n3:

print(‘Biggest number is:’,n2)

else:

print(‘Biggest number is:’,n3)

**#write a program to check whether the given number is in between 1 and 100 or not**

If n>=1 and n<=100:

Print(‘The given number’,n,’ is in between 1 and 100’)

else:

print(‘The given number’,n,’is not in between 1 and 100’)

**#write a program to print number between 0 to 9**

n=int(input(‘Enter a digit from 0 to 9:’))

if n==0:

print(‘ZERO’)

elif n==1:

print(‘ONE’)

elif n==2:

print(‘TWO’)

elif n==3:

print(‘THREE’)

elif n==4:

print(‘FOUR’)

elif n==5:

print(‘FIVE’)

elif n==6:

print(‘SIX’)

elif n==7:

print(‘SEVEN’)

elif n==8:

print(‘EIGHT’)

elif n==9:

print(‘NINE’)

else:

print(‘plz enter the number between 0 to 9 only’)

**1.else is always optional**

**If**

**If-else**

**If-elif-else**

**If-elif**

**2.switch is not available in python**

**2.Iterative Statements:**

* **Loops**
* **For loop**
* **While(in python there is no do while loop)**

**for loop:-(for each is mostly recommended in python)**

**If we know the no. of Iteration in advance then we use while loop.**

**syntx: for each element in sequence:**

**do some action**

**Ex**:- s=‘sunny Leone’ or s=[10,20,30,40]

Count=0

for x in s:

count+=1

print(x)

print(‘The number of characters is :’,count)

**Ex**:- s=input(‘Enter some string:’)

i=0

for x in s:

if i>=0:

print(‘The character present at’,i,’index is’,x)

i+=1

**Ex:-** for x in range(21):

If(x%2!=0):

Print(x) -----🡪1 3 5…….21

**Ex**:- for x in range(10,0,-1):

Print(x)----🡪10 9 8 7 6 5 4 3 2 1

**Ex:-**list=eval(input(‘Enter some list:’)) -🡪10,20,30,40

Sum=0

for x in list:

sum+=x

print(‘The Sum is’, sum)-🡪the sum is 100

**While Loop:-**

**If we don’t know no. of Iteration in advance then we use while loop.**

**Syntax:-while condition:**

**body**

**Ex:-** x=1

while x<=10:

print(x) --------🡪1 2 3 4 5 6 7 8 9 10

x+=1

**#find sum of first n number**

**Ex:-** n=int(input(‘Enter some number’)) **#4**

Sum=0

i=1

while i<=n

sum=sum+i

i+=1

print(‘The sum of first’,n,’numbers is:’,sum)**#10**

**Ex:-**name=‘‘

pwd=‘‘

While (name!=‘durga’) or (pwd!=’python’):

name=input(‘Enter Name:’)

pwd=input(‘Enter Password:’)

Print(‘Hello durga thanks for the confirmation’)

**Ex**:- i=0

While True

I+=1

Print(‘Hello:’,i)**------------🡪Infinite loop**

**Ex:-** for i in range(4)

for j in range(4):

Print(‘i={} and j={}’.format(i,j))**----------🡪Nested loop**

**Patterns:-**

**\***

**\* \***

**\* \* \***

**\* \* \* \***

**\* \* \* \* \***

n=int(input(‘Enter the number of rows:’))

for i in range(1,n+1): #i represents row number

#print(‘\* ‘\*i)-------🡪optional

for j in range(1,i+1): #j represents the number of \*

print(‘\*’,end=‘ ‘)

print()

**1🡪\* 2🡪\*\* 3🡪\*\*\***

**\*\* \*\*\***

**\*\*\***

**Ex**:- n=int(input(‘Enter the number of rows:’))

for i in range(n):

for j in range(n):

print(‘\*’,end=‘ ‘)

print()

**3.Transfer Statements:**

**Break**

**Continue**

**Pass**

1. **Break:-**

**Based on some condition if we want to break loop execution**

**Ex:-**for i in range(10):

If i==7:

Print(‘processing is enough…plz break’)

Break

Print(i)**-------🡪1 2 3 4 5 6 7**

**Ex:-**cart=[10,20,600,70,80,90]

For item in cart:

If item>500:

Print(‘sorry we cannot process this order…insurance must be required’)

Break

Print(‘processing item’,item)

**2.continue:-**

**We use continue within loop to Skip the current iteration and go for next iteration**

**While condition:**

**Body**

**Body**

**Body**

**Continue**

**Body**

**Body**

**Body**

**Ex**:- for i in range 10:

If i%2==0:

continue

Print(i)**-----🡪1 3 5 7 9**

**Ex:-** numbers=[10,20,0,5,0,30]

for n in numbers:

if n==0:

print(‘Hey how we can divide with zero…pagal ho gya kya’)

continue

print(‘100/{}={}’.format(n,100/n))**---🡪100/10=10.0 100/20=5.0 hey how we can divide with zero……..**

**note:-when else part will be executed?---🡪if loop executed without break.**

**Ex:-for x in range(10):**

**Print(‘The current item: ’,x)🡪 The current item:0..9**

**else:**

**Print(‘congratulations, all items processed successfully’)🡪 congratulations, all items processed successfully**

**Ex:-for x in range(10):**

**If x>5:**

**break**

**Print(‘The current item: ’,x)🡪 The current item:0..9**

**else:**

**Print(‘congratulations, all items processed successfully’)🡪**

**pass:-**

* **pass is a keyword in python**
* **In our programming syntactically if block is required which won’t do anything then we use pass.**

* **It is an empty statement**
* **It is a null statement like ; in java**
* **It wont do anything**

**Ex**:- def f1():

Print(‘Hello’)

Def f2():

Pass

F1()

F2()

**Ex**:- for i in range(100)

if i%10==0

print(i)

else:pass

**del statement:**

* **del is a keyword in python**
* **del is used to delete a object permanently after using that varaiable when comes to garbage collector.**
* **Advantage of using del statement is memory utilization bydefault will be improved.**

**Ex**:- x=10

Print(x)🡪**10**

del x

print(x)**🡪name ‘x’ is not defined**

**difference between del and None: -**

**🡪in the class del, the varaiable will be removed and we cannot access that variable(unabled operation)**

**🡪But in the case of None assignment the vaiable won’t be removed but the corresssponding object is eligible for Garbage Collection bind operation). Hence after assigning with None value, we can access that variable.**

**Ex**:-s=‘durga’

S=None

Print(s) **#None**

**String Data Type :-**

We can create string object using ‘ ‘ or ‘’ ‘’.

**Ex**:- s=’durga’

s=‘durga’

**Note**:-

There is no char datatype in python either ‘’ or ‘’’’ character is only consider a string only.

**Ex:**-Ch=’a’🡺string

**Multi line string literals or values :**

**Triple quotes**

**Ex**:- s=’’’ durga or s=‘‘‘durga

software’’’ software’’’

print(s) print(s)

**Ex**:- s=”This is ‘ single quote symbol”

S=’This is “ double quote symbol’

S=‘This is “\ double quote symbol’

S=’’’These are ‘ and ‘ symbols’’’

**How to access characters of the string:**

* **By using index( index can be either +ve(left to right(0 to n)) or –ve(right to left(-1 to –n)))**

**Ex**:- s=’durga’

Print(s[0]) 🡪d

Print(s[-1]) 🡪a

Print(s[7]) 🡪error string index out of range

**Ex**:- s=input(‘Enter some string:’)

i=0

for x in s

print(‘The character present at positive index:{} and at negative index:{} is :{}’.format(i,len(s)),x))

i=i+1**🡪 The character present at positive index:0 and at negative index:-5 is :d…u..r…g…a**

* **By using slice operator**

**Syntax:- s[beginindex:endindex:step]**

* Step value can be either +ve or –ve.
* If +ve then it should be forward direction(Left to Right).
* If -ve then it should be backward direction(Right to Left).
* If +ve forward direction from begin to end-1
* If -ve backward direction from begin to end+1.

**Ex:- s=’0123456789’**

**S[2:8:1] 🡪234567**

**S[2:8:-1] 🡪empty output**

**S[2:8:0] 🡪slice step value cannot be 0**

**S[-1:-6:-1]🡪98765**

**S[2:-5:1]🡪234**

**S[1:6:-2]🡪empty string**

**S[0:-5:-5]🡪empty**

**Ex:-s=’0123456789’**

**s[0:7:1] 🡪0123456**

**s[0:7:2] 🡪0246**

**s[0:7] 🡪0123456 because by default step value is 1**

**s[0:] 🡪0123456789**

**s[::] 🡪0123456789 🡪default begin value is 0**

**s[::-1] 🡪987654321 🡪 reverse value**

s[begin:end:step]

step=🡺+ve forward direction from begin to end-1

-ve backward direction from begin to end+1

**In forward direction:**

Default value for begin: 0

Default value for end: length of the string

Default value for step:1

**In backward direction:**

Default value for begin index: -1

Default value for end: -(length of the string+1)

Default value for step:-1

**Mathematical Operation:-**

**+ :-** concatenation add value

**Ex:- ‘durga’+’soft’🡪’durgasoft’**

**\*:-** string repeatation concat multiple string operator

**Ex:- ‘durga’\*3🡪durgadurgadurga**

**Inbuilt function in string :-**

**len(s): -**

**Ex:- len(‘durga’) 🡪5**

**Q. WAP to access each character of string in forward direction and backward direction:**

**S[:] == > forward direction**

**S[::-1] == > backward direction**

**Using while loop 🡺**

**s=input(‘Enter Some String:’) ---- > ‘durga’**

**n=len(s)**

**i=0**

**print(‘Data IN Forward Direction:’) -->‘durga’**

**while i<n:**

**print(s[i],end=’’)**

**i=i+1**

**print()**

**print(‘Data in Backward Direction’) 🡪’agrud’**

**i=n-1**

**while i>=0:**

**print(s[i],end=’’)**

**i=i-1**

**print(‘Data in Backward Direction with –ve index’) 🡪’agrud’**

**i=n-1**

**while i>=-n:**

**print(s[i],end=’’)**

**i=i-1**

**Q.1.WAP to remove duplicate characters from the string**

**2. Find each character how many time it is presenting.**

**Mississippi**

**3.Reverse : Durga Software Solutions-**🡪**agrud erawtfof snoitulos**

S=input(‘Enter some string’)

L=s.split()

L1=[]

For word in l:

L1.append(word[::-1])

Output=’ ‘.join(l1)

Print(output)

**Membership operators in string:-**

**in**

**not in**

**Ex:-** s=’durga’

Print(‘d’ in s)🡪**True**

Print(‘z’ in s)🡪**False**

**Ex:- WAP to check substring is available in string or not**

S=input(‘Enter Main String:’)

Subs=input(‘Enter Substring to search:’)

If subs in s:

Print(subs, ‘Is found in Main String’)

else:

print(subs, ‘is not found in Main String’)

**Comparison operators in string:-**

**>,<,>=,<=**

**Equality operators:-**

**==, !=**

In general **==** equal operator meant for **content** operator

**Is** operator meant for **reference** operator

It is based on alphabetical order based on **Unicode** .

**Ex**:- s1=input(‘Enter First String’) #**ram**

s2=input(‘Enter Second String’) #**ramnarayan**

If s1==s2:

Print(‘Both Strings are equal’)

elif s1<s2:

print(‘First String is smaller than second string’)

else:

print(‘First String is biggest then second string’)

**lstrip():-**

to remove to spaces present at beginning of the string.

**rstrip():-**

to remove to spaces present at end of the string.

**Strip():-**

**Strip is used to remove spaces from beginning and end both side.**

**Ex:-**city=input(‘Enter Your City Name:’)

list=[‘Ranchi’,’Goa’,’Delhi’,’Banglore’]

if city.strip() in list:

print(‘Your city is available and ccc are xxx’)

else:

print(city,’Not available…Plz enter valid city name’)

**find(substring) :-**

to find substring is present in string or not.

If the string is already specified then Find method is always going to return index of first occurance **0**.

If the string is already not specified then Find method is always going to return **-1**.

**Ex:-s=’abababa’**

**Sub=’ab’**

**l=len(b)**

**i=s.find(sub)**

**if i==-1:**

**print(sub,’not found in the given string’)**

**While l!=-1:**

**Count=0**

**Count=count+1**

**Print(sub, ‘present at index:’ i)🡪’ab’ present at index: 0 2 4**

**i=s.find(sub,i+len(sub),l)**

**print(‘The no. of occurances:’,count)-🡪3**

**Ex:-s=input(‘Enter Main String’)🡺durgasoftdurgasoft**

**subs=input(‘Enter substring:’)**

**flag=False**

**pos=-1**

**n=len(s)**

**count=0**

**while True:**

**pos=s.find(subs,pos+1,n)**

**if pos==-1:**

**break**

**print(‘found at index:’,pos)**

**flag=True**

**count=count+1**

**if flag==False:**

**print(‘Not Found:’)**

**print(‘No of occurances :’,count)🡺2**

**index()**

**in find method if not found the substring then it return -1 and in index if found then print the index and if not found then error(value error).**

**rfind():- it also find from right to left.**

**rindex():- it find the substring from right to left.**

**counting substrings in the given string:**

**s.count(substring)**

**s.count(substring,begin,end)**

**Ex**:- s=’durgasoftdurgasoft’

Print(s.count(‘soft’))🡪**2**

Print(s.count(‘soft’,8,len(s))) **🡪1**

**Replacing a string with another string**

**s.replace(oldstring,newstring)**

s=’Learning python is very difficult’

s1=s.replace(‘difficult’,’easy’)

print(s1)**--🡪Learning python is very easy.**

**When we use replace operator it does not effect the existing object instead it create the new object.**

**Ex:-**

S=’abababa’

S1=s.replace(‘a’,’b’)

Print(s, ‘Address is:’,id(s)) **🡪 abababa Address is :35074560**

Print(s1,’Address is:’,id(s1))**🡪bbbbbbb Address is 35074368**

**Splitting of string:**

**s.split(seperator)🡪default separator is space ‘ ‘**

**Ex:-**s=’Durga Software Solutions’

l=s.split()

print(type(l))

print(l) ---- **>[‘durga’ , ‘software’ , ‘Solutions’]**

for x in l:

print(x)-**🡪 durga software solutions**

**Ex:-** s=’02—3-2021’

L=s.split(‘-‘)

Print(l)

**Note:- Split is related to string only and sep is related to print().**

**rsplit():-**I don’t want all but I want max 3 only , it gives in split reverse direction

**Ex**:- s=’Durga Software Solutions hyd india’

l=s.rsplit(‘,’-1)

for x in l:

print(x)**-🡪10,20,30,40,50**

**60**

**70**

**80**

**Join():-**

**Ex**:- l=[‘durga’,’soft’,’solutions’]

S=’-‘.join(l)

Print(s) **🡪durga-soft-solutions**

**Changing case of a string**

**Upper()== >**To convert to upper case

**Lower() == >**To convert to lower case

**Swapcase() == > To convert lower to upper and upper to lower**

**Title() == >**The Python Classes By Durga Sir🡪 every word start with upper case.

**Capitalize() == >**in the total string first character is upper case and other one is lower case.

**Ex:- s=‘The Pyhton Classes By Durga Sir’**

Print(s.upper()) **🡪THE PYTHPON CLASSES BY DURGA SIR**

Print(s.lower()) **🡪the python classes by durga sir**

Print(s.swapcase()) **🡪tHE pYTHON cLASSES bY dURGA sIR**

Print(s.title()) **🡪 The Pyhton Classes By Durga Sir**

Print(s.capitalize()) **🡪The Pyhton Classes By Durga Sir**

**Checking Starting and Ending part of the String:-**

**Startswith()**

**Endswith()**

**Ex:-** s=‘Learning Python is very easy’

Print(s.startswith(‘Learning’))🡪**True**

Print(s.endswith(‘easy’)) **🡪True**

**Q.To print characters at odd position and even position for the given string.**

**1st way:-**S=input(‘Enter some string’)-🡪durga

Print(‘Characters at Even position:’,s[::2])🡪dra

Print(‘Characters at Odd position:’,s[1::2])🡪ug

**2nd way:-** S=input(‘Enter some string’) 🡪durga

i=0

print(‘Characters at Even position:’)

while i<len(s):

print(s[i],end=’,’)

i=i+2

print()

i=1

print(‘Characters at Odd position:’)

while i<len(s):

print(s[i],end=’,’)

i=i+2

**To check type of characters present in a string:**

**Isalnum() 🡺{A-Z ,a-z,0-9}**

**Isalpha() 🡪only alphabet symbols**

**Isdigit()**

**Islower()**

**Isupper()**

**Istitle()**

**Isspace()**

**Ex:-**print(‘Durga786’.isalnum() #**True**

print(‘durga786’.isalpha() #**False**

print(‘durga’.isalpha() #**True**

print(‘durga’.islower() #**True**

print(‘durga’.isupper() #**False**

print(‘Durga Software Solutions’.istitle() #**True**

print(‘Durga Software Solutions’.isspace() **#True**

**isalpha:-**

**Q.A4d4b31--🡪adb4431**

S=input(‘Enter Some string’)🡪DURGASOFT1726746WARE

S1=s2=output=’’

For x in s

If x.isalpha():

S1=s1+x

Else:

S2=s2+x

For x in sorted(s1)

Output=output+x

For x in sorted(s2)

Output=output+x

Print(output)-**--🡪AADEFGORRSS21325434**

**Qa4b3c2**

**Output:-aaaabbbcc**

S=input(‘Enter some string:’)🡪**a7b4z3**

Output=’’

For x in s:

If x.isalpha():

Output=output+x

Previous=x

Else:

Output=output+previous\*(int(x)-1)

Print(output)**🡪aaaaaaabbbbzzz**

**@input:a4k3b2**

**Output:-aeknbd**

S=input(‘Enter Some String:’)

Output=’’

for x in s:

if x.isalpha():

output=output+x

previous=x

else:

newch=chr(ord(previous)+int(x))

output=output+newch

print(output)

**inputs:**

**s1=RAVI**

**s2=TEZA**

**Output:-RTAEVJIA**

S1=input(‘Enter First String’)

S2=input(‘Enter Second String’)

Output=’’

I==j==0

While i<len(s1) or j<len(s2):

If i<len(s1):

Output=output+s1[i]

I=i+1

If j<len(s2):

Output=output+s2[j]

j=j+1

print(output)

**LIST:-**

* **If you want to represent the group of individual object at a single entity.where,**
* **Insertion order is preserved by using index and duplicates are allowed**
* **Hetrogeneous(mutable)**
* **Growable**

**How to create list object:-**

* **l[] 🡪create an empty list**
* **l=[10,20,30,40,50] 🡪 if you already know value**
* **l=eval(input(‘Enter some list’))** 🡪**if we want value dunamically from keyboard.**
* **l=list(sequence)** 🡪**by usning list function**
* **l=s.split(separator)** 🡪**string to list conversion**

**Nested List:-**

A list can contain another list such type of list is called nested list**.**

**Ex:-[10,20,[30,40]]**

**Ex:-x=[[10,20,30],[40,50,60],[70,80,90]]**

**Print(x)**

**Print(‘element row wise:’)**

**For r in x:**

**Print(r)**

**Print(‘elements in matrix style:’)**

**For I in range(len(x)):**

**For j in range(len(x[i])):**

**Print(x[i][j],end=’ ’)**

**Print()**

**List Comprehensions:**

**List=[1,4,9,16,25,36..]**

**Ex:- l1=[x\*x for x in range(1,11)]**

**Print(l1)**

**How you can access list element:-**

**By using index(forward and backward direction)**

**By using slice operator.**

**List vs immobility**

**Traversing the elements of list:**

**By using while loop:-**

List=[1,2,3,4,5,6,3,4,5,7,8,9,0]

i=0

While i<len(list):

Print(list[i])

i=i+1

**by using for loop:-**

list=[1,2,3,4,5,6,7,8,3,4,5,8]

for x in list:

print(x)

**To display only even number:**

list=[1,2,3,4,5,6,7,8,3,4,5,8]

for x in list:

if x%2==0:

print(x)

**To the element at positive and negative index:**

list=[‘A’,’B’,’C’,’D’,’E’]

x=len(list)

for I in range(x):

print(list[i],’is available at positive index:,’,i,’and at negative index:’,i-x)

**Important functions of list:-**

**Functions():- if you define any functions outside of a class is known as function or functional programming**

**Methods():- if you define any functions inside of a class is known as methods.**

**Ex:-** def f1(): ------------- >function

Print(‘Hello,this is function:’)

Class Student:

Def info(self): ------------------ >method

Print(‘It is method and don’t get confuse:’)

f1()

s=Student()

s.info()

**Important functions and Methods of List:**

**1.len(list) --- >python provide inbuilt function len**

**2.count(obj) ---> count no. of occurances of element**

**3.index(obj) -- >return the index of occurance**

**Ex**:-l=[10,20,30,40,10,20,10,10]

Target=int(input(‘Enter the value to search:’))

If target in l:

Print(target,’available and its first occurance is at:’l.index(target))

Else:

Print(target,’not available’)

**Manipulating elements of list:-**

1. **l.append(element) --**🡪**only for list to add element**

**append always add the element at last position.**

**Ex:- To add all elements of list upto 100 which are divisible by 10**

L=[]

For x in range(0,101,10):

l.append(x)

print(l)

**2>l.insert(element) 🡪**

**If you want to insert an element at our specified position then we use insert.**

**Ex:-**l=[]

l.append(10)

l.append(20)

l.append(30)

l.append(40)

print(l) --- >10 20 30 40

l.insert(1,50)

print(l) ---- > 50 10 20 30 40

l.insert(50,777) 🡪

print(l) --- **> 10 20 30 40 777 -- > if the index is greater than max index then the element will be added at the end.**

l.insert(-10,999)

print(l) **--- > 999,10,20,30,40,777 -🡪 if the index is less than minimum index then the element will be added at the begin.**

print(l.index(777)) -- >**5**

print(l.index(999)) -- >**0**

**3.extend():**

**To add all element to the list we use extend method.**

L1=[‘chiken’,’mutton’,’fish’]

L2=[‘KF’,’RC,’FO’]

L1.extend(l2) --- **>‘chiken’,’mutton’,’fish’,’KF’,’RC,’FO’**

Or l3=L1+L2

**Remove**():

To remove the specified element.

**Pop**() or **pop(index):**

By default to remove and return the last element.

**Clear**():

To remove all element present in the list

**Ex**:-l=[10,20,30]

Print(x)

x.clear()

print(x)

.

**Reverse():**

**To reverse the order of the element of the list.**

**l=[20,0,15,5,10]**

**l.sort(reverse=True)**

**print(l) 🡪20,15,10,5,0**

**Sort():**

**To sort accoprding to default assending order based on Unicode and only homogeneous datatype.**

**Ex:-l=[20,0,15,5,10]**

**l.sort()**

**print(l) 🡪0,5,10,15,20**

**aliasing:-**

**aliasing means just duplicates varaiable.**

**Copy:-**

**Copy means cloning.**

**Ex:-x=y**

**X=y.copy()**

**+ operator:**

**Both arguments list objects only**

**\* operator:**

**One argument should be list and another one should be number or integer.**

**Comparing list objects:**

**X==y**

**If you are comparing list object then,**

**1.The number of elements must be equal**

**2.The order should be same**

**3. The contents should be same(including case)**

**Ex:-**X=[‘Dog’,’Cat’,’Rat’]

Y=[‘Dog’,’Cat’,’Rat’]

Z=[‘DOG’,’CAT’,’RAT’]

Print(x==y) #**True**

Print(x==z)#**False**

Print(x!=z) **#True**

Print(x is y) **🡪False🡪because objects are different.**

Print(x[0] is y[0]) **🡪True 🡪because address are same.**

**<,>,<=,>= :**

**X=[50,20,30]**

**Y=[40,90,100,120,170]**

**Print(x>y) 🡪it always going to compare first element. Like 50>40 🡪#True**

**Print(x>=y) 🡪#False🡪if the first element is same the it check for second element.**

**Membership operator (In and not in) :**

**X=[50,20,30]**

**Print(20 in x) 🡪True**

**Print(100 not in x) 🡪True**

**Nested List:-**

A list can contain another list such type of list is called nested list**.**

**Ex:-[10,20,[30,40]]**

**Ex:-**x=[[10,20,30],[40,50,60],[70,80,90]]

Print(x)

Print(‘element row wise:’)

For r in x:

Print(r)

Print(‘elements in matrix style:’)

For I in range(len(x)):

For j in range(len(x[i])):

Print(x[i][j],end=’ ’)

Print()

**List Comprehensions:**

**List=[expression for x in sequence if condition]**

**List=[1,4,9,16,25,36,48,64,81,100]**

**Ex:- l1=[x\*x for x in range(1,11)]**

**L2=[x’ for x in l1 if x%2==0]**

**L3=[x’ for x in l1 if x%2!=0]**

**L4=[x+x for x in range(1,11)]**

**L5=[x\*\*2 for x in range(1,11) if (x\*\*2)%2!=0]**

**Print(l1) 🡪[1,4,9,16,25,36,48,64,81,100]**

**Print(l2) 🡪[4,16,36,48,64,100]**

**Print(l3) 🡪[1,9,25,81]**

**Print(l4) 🡪[2,4,6,8,10,12,14,16,18,20]**

**Print(l5) 🡪[1,9,25,49,81]**

**Ex:-words=‘the quick fox jump over the lazy dog’.split()**

**Print(words) 🡪 [‘the’, ‘quick’, ‘fox’, ‘jump’, ‘over’, ‘the’ ,‘lazy’, ‘dog’]**

**L=[[w.upper(),len(w)] for w in words],**

**Print(l) 🡪 [[‘THE’,3],[ ‘QUICK’,5],[ ‘FOX’,3],[ ‘JUMP’,4],[ ‘OVER’4],[ ‘THE’,3] ,[‘LAZY’,4], [‘DOG’,3]]**

**Tuple:-**

* **Tuple is imutable and list is mutable .**
* **Other than all thing same as tuple.**

**How to create tuple:-**

**T=()**

**T=(10,) 🡪if I don’t take comma it will treated as int value .for single valued tuple it should end with comma.**

**T=(10,20,30) 🡪() are always optional**

**T=tuple(sequence)**

**How to access elements of tuple:**

**Index**

**Slice**

**Mathematical operation:**

**+**

**\***

**Important functions of tupple:**

**1.len():**

**Ex:-t=(10,20,30,40,10,10)**

**Print(len(t)) 🡪4**

**2.count():**

**Ex:- t.count(10) 🡪3**

**3.index():**

**Ex:-t.index(10) 🡪0**

**t.index(90) 🡪value error**

**4.sorted():**

**To sort elements based on natural sorting order.**

**Sort() is not available in tuple because tuple is immutable**

**Ex:-t=(30,10,50,40,20)**

**Print(sorted(t))🡪[10,20,30,40,50]**

**4.min() and max()**

**It is only for homogeneous object not for heterogeneous.**

**Ex:-t1=’durga’**

**Print(min(t1)) 🡪a**

**Print(max(t1)) 🡪u**

**4.cmp()**

**Cmp is available in python 2 only not in python 3**

**Cmp(t1,t2)**

**If t1 is equal to t2 the it return 0**

**If t1 is smaller than t2 then it return -1**

**If t1 is greater than t2 then it return +1**

**Tuple packing and tuple unpacking:**

**For list ,set and tupple these are applicable.**

**Packing :- grouping into single**

**A=20**

**B=20**

**C=30**

**D=40**

**T=a,b,c,d**

**Print(t) -🡪(10,20,30,40)**

**Unpacking:-**

**T=(10,20,30,40)**

**a,b,c,d=t**

**Print(‘a=’,a,’b=’,b,’c=’,c,’d=’,d) 🡪a=10 b=20 c=30 d=40**

**Tuple comprehension:**

**Tuple comprehension is not supported.even it is supported but result is not tuple but is some generator object.**

**T=[x\*x for x in range(1,11)]**

**Print(t)**

**Write a program to take a tuplle of numbers from the keyboard and print sum,avg.**

**Ex:-t=eval(‘enter some tuple of numbers:’))**

**L=len(t)**

**Sum=0**

**For x in t:**

**Sum=sum+x**

**Print(‘The sum:’,sum)**

**Print(‘avg is:’,sum/l)**

**Differences between list and tupple:**

**List Tuple**

**1.[10,20,30,40] 1.(10,20,30,40)**

**2.[] is mandatory 2.() is optional**

**3.mutable 3.immutable**

**4.list object cannot be 4.tuple object can be**

**Used as keys for the used as keys for**

**Dictionary . dictionary.**

**5.comprehension applicable 5.comprehension not applicable**

**Set:-**

* **set does not allowed duplicates.**
* **set does not preserve insertion order.**
* **Slice operator and indexing not allowed by set.**
* **Set objects are mutable.**
* **If we want to create empty set then must use set()**

**S=set()**

**Ex:-s={10,20,30}**

**Important functions of set:-**

**S={10,20,30}**

**1.add(newelement):-**

**If you want to add a single element then use add()**

**s.add(newelement) 🡪add new element**

**Ex:-s=set()**

**s.add(10)**

**s.add(20)**

**s.add(30)**

**print(s) 🡪{10,20,30}**

**2.update(x)**

**Update(x,y,z)**

**If you want to add a group of elements**

**s.update(x): 🡪x will be list tuple anything**

**Ex:-s={10,20,30,40}**

**L=[50,60,70]**

**s.update(l) or s.update(l,range(1,5),’durga’)**

**print(s) 🡪{10,20,30,40,50,60,70}**

**3.copy:-**

**Return a same object or cloned object then we use copy()**

**Ex:-s={10,20,30,40}**

**S1=s.copy()**

**Print(id(s))-🡪5335363**

**Print(id(s1)) 🡪2947477**

**4.pop():-**

**Remove and return the element.**

**It remove some random element.**

**Ex:- s={10,20,30,40}**

**Print(s.pop())🡪first time 40 removed**

**Print(s.pop())**

**Print(s.pop())**

**Print(s.pop())**

**Print(s) 🡪empty set**

**5.remove(x):-**

**It removes the specified element.**

**6.discard(x):**

**If the specified element is not available and we use discard then we do not get error it will remove.**

**S={10,20,30,40}**

**s.discard(40)**

**s.discard(140)**

**print(s)**

**7.clear():**

**Clear all the set.**

**Mathematical operations:**

**1.union()**

**It will return both set elements**

**S1.union(s2) or s1|s2**

**2.intersection**

**It will return common elements**

**S1.intersection(s2) or s1&s2**

**3.difference**

**The element present in s1 but not in s2**

**S1.difference(s2) or s1-s2**

**4.symmetric difference(s1^s2)**

**The element present in s1 but not in s2**

**The element present in s2 but not in s1**

**Except these result is symmetric difference**

**S1.symmetric difference(s2)**

**Membership operator (in or not in)**

**S1=set(‘durga’)**

**Print(s1)**

**Print(‘d’ in s1) #True**

**Print(‘z’ not in s1) #True**

**Set comprehension :**

**S={ x\*x for x in range(1,6)}**

**Print(s)**

**To eliminates duplicates present in list:**

**L=eval(input(‘Enter some list of vales’))**

**S=set(l)**

**Print(s)**

**Or**

**L=eval(input(‘Enter some list of vales’))**

**L1=[]**

**For x in l:**

**If x not in l1:**

**L1.append(x)**

**Print(l1)**

**How you can identify different vowel in word:**

**W=input(‘Enter some word’)**

**S=set(w)**

**V={‘a’,’e’,’I’,’o’,’u’}**

**D=s.intersaction(v)**

**Print(‘The different vowels present in the given word:’,d)**

**Print(‘The number of different words:’,len(d))**

**Dictionary Data Structure:**

* List, tuple,set == >only to hold individual objects
* Dictionary is based on key-value pairs

**Ex**:rollno:name,,, mobno:address

* Key never be duplicate but value may be duplicate.
* Heterogeneous object are allowed for both key value
* Insertion order is not preserved
* Mutable(add or remove)
* Dynamic(Increase and decrease)
* Indexing and slicing notn applicable
* Null is not allowed

**How to create empty dictionary:-**

**D={}**

**D=dict()**

D[key]=value

**Ex**:-D={}

D[100]=’durga’

D[200]=’ravi’

D[300]=’shiva’

D[‘rich’]=’chohen’

Print(d) #{100:’durga’,200:’ravi’,…}

**syntax**:-d={key1:value1,key2:value2}

**Ex**:-d={‘a’:’apple’,’b’:’banana’,’c’:’cat’}

**How we can access the elements of dictionary:**

rec={}

n=int(input(‘Enter number of studnts:’)) 🡪3

i=1

While i<=n:

Name=input(‘Enter Students name:’) 🡪sunny

Marks=input(Enter % of marks:’) 🡪90

Rec[name]=marks

i=i+1

Print(‘name of students’,’\t’,’% of marks’) 🡪sunny 90

For x in rec:

Print(‘\t’,x,’\t\t’,rec[x])

**How to update dictionaries:**

D= {100:’durga’,200:’ravi’,300:’shiva’}

Print(d)

D[400]=’pawan’

Print(d) 🡪{100:’durga’,200:’ravi’,300:’shiva’,400:’pawan’}

D[100]=’vikas’

Print(d) 🡪{100:’vikas’,200:’ravi’,300:’shiva’,400:’pawan’}

**How to delete elements from the dictionary:**

D={100:’durga’,200:’ravi’,300:’shiva’,400:’pawan’}

Print(d)

Del d[200]

Print(d)🡪 {100:’durga’,300:’shiva’,400:’pawan’}

Del d[500] 🡪error

d.clear() 🡪for deleting all elements from the dictionary but still we can access d and we can add another elewment

print(d) 🡪{}

del d 🡪for all delete but we can’t do anything with the dictionary we cant acces this.

**Important functions of dictionary:**

**1.dict() :-**

**To create empty dictionary.**

**Or we can add elements.**

**Ex:-d=dict([(100,’durga’),(200,’ravi’),(300,’shiva’)])#with list of tupples**

**Print(d)**

**2.len():-**

**3.get():-**

**To get the values by key**

**d.get(key)**

**d.get(key,default value)**

**D=dict({(100,’durga’),(200,’ravi’)})**

**Print(d.get(100)) 🡪durga**

**Print(d.get(500)) 🡪none**

**Print(d.get(500,’durga’) 🡪durga**

**4.pop(key):**

**It removes the entry associated with specified key and return the corresponding value**

**D=dict({(100,’durga’),(200,’ravi’)})**

**Print(d)**

**Print(d.pop(100))🡪durga**

**Print(d)🡪{200:’ravi’}**

**4.Popitem()**

**One key value will be remove but random one.**

**Print(d.popitem())🡪200 :’ravi’**

**Print(d.popitem())🡪100:’durga’**

**Print(d)🡪{}**

**5.keys():-**

**It returns all keys associated with the dictionary.**

**Print(d.keys())🡪([100 200])**

**For k in d.keys():**

**Print(x)🡪100**

**200**

**6.values():-**

**It returns all values.**

**Print(d.values()) 🡪([‘durga’,’ravi’])**

**7.items():-**

**It returns all item.**

**D={100:’durga’,200:’shiva’,300:’ravi’}**

**For k,v in d.items():**

**Print(k,’…..’,v)🡪100…..durga 200…shiva 300…ravi**

**8.copy():-**

**If you want cloning or duplicate object.**

**9.setdefault(k,v)**

**If the key the avaialable then new value will be replaced with old value return corresponding value if key is not available please add new value.**

**Ex:-d={100:’durga’,200:’ravi’,300:’shiva’}**

**Print(d.setdefault(400:’sunny’))**

**Print(d)🡪 {100:’durga’,200:’ravi’,300:’shiva’,400:’sunny’}**

**10.update():-**

**To add two dictionary.**

**d={100:’durga’,200:’ravi’,300:’shiva’}**

**d1={‘a’:’apple’,’b’:’ball’}**

**d.update(d1)**

**print(d)--🡪 d={100:’durga’,200:’ravi’,300:’shiva’,’a’:’apple’,’b’:’ball’}**

**Ex:wap to take dictionary from the keyboard and print the sum of values**

**D=eval(input(‘Enter dictionaries’)🡪{‘A’:100,’b’:200,’c’:300}**

**S=sum(d.values())**

**Print(‘The Sum:’,s) 🡪600**

**No of occurance of each later present in the given string?**

**Mississippi🡪m-1,i-4,s-4,p2**

**Word=input(‘Enter some words:’)**

**D={}**

**For x in word:**

**D[x]=d.get(x,0)+1 🡪it will fetch the key is available or not and if key is available then get corresponding value if key is not already available it will considder 0 bydefault.**

**Print(d)**

**For k,v in sorted(d.items()):**

**Print(‘{} occurred {} times’,format(k,v))**

**Functions**

Function is a group of statements.

Function is mainly used for code reusability.

**Ex**:-def wish(name):  
 print(‘good morning:’,name)

print(‘good morning:’,name)

print(‘good morning:’,name)

print(‘good morning:’,name)

wish(‘sunny’)

wish(‘bunny’)

wish(‘chinny’)

**Ex**:-def insertEmployee(name, age,esal,eddr)

Insert into employess table

Insert into payments table

Insert into projects table

Insert into projects table

return x,y,z

For e in employees:

insertEmplpoyee(data)

**parameters**:-

1>.positional

2>.keyword

3>.default

4>.var

* In java if does not have return type and if we return anything then it gives error or we have to provide void.
* If function does not return anything then it will return none.

**Ex**:-def wish():

Print(‘hello’)

Print(wish()) 🡪none

* But in python, Any function can return multiple value.

**Ex**:-def calc(a,b): 🡪**positional parameter**

Sum=a+b

Sub=a-b

Mul=a\*b

Div=a/b

return sum,sub,mul,div

**t=calc(100,50)** 🡪**order and no. of argument is important**

**t= calc(a=100,b=50)** 🡪**keyword** **argument**

**t=calc(100,b=50) 🡪valid**

**t=calc(a=50,100) 🡪Invalid because first argument should be positional**

**t=calc(50,a=100) 🡪error because for a multiple value**

for x in t:

print(x)

**default argument:-**

def wish(msg,name=’Guest’):pass 🡪default argument should be last

def wish(name=’Guest’,msg):pass 🡪error because non-default argument follows default argument.

**Variable length argument:-**

In java If you change number of argument you should change no. of method with different parameter.

So for overcome from this readability problem we use var length argument.

**Ex**:-def sum(\*n):

Sum()

Sum(10)

Sum(10,20)

Sum(10,20,30)

Sum(10,20,30,40)

**wap to find sum of given numbers which I don’t know how many numbers I am passing**

**def sum(\*n,name):**

**result=0**

**for x in n:**

**result=result+x**

**print(‘The sum By:’,name,’:’result)**

**Sum(name=’Durga’)🡪0**

**Sum(10, name=‘Ravi’) 🡪10**

**Sum(10,20, name=‘Pawan’,) 🡪30**

**Sum(10,20,30, name=’shiva’) 🡪60**

**Sum(10,20,30,40, name=’suraj’) 🡪100**

**Types of variables:-**

**Global variables**

**local variables**

**Packages:-**

**Libraries:-**

**Recursive Functions:-**

**A function that’s call itself recursive functions.**

**It reduce the length of the code and improves readability.**

**Very complex problem we can solve easily Towers of Hanoi.**

**Wap to find factorial by using recursion.**

**Def factorial(n):**

**If n==0:**

**Result=1**

**Else:**

**Result=n\*factorial(n-1)**

**Return result**

**Print(factorial(0)) 🡪1**

**Print(factorial(5)) 🡪20**

**Anonymous function:-**

**Sometime we declare a function without name such type of function is known as anonymous function.**

**If we have the requirement instant use(one time use) the we use this.**

**Otherwise if we use multiple time then we use normal function.**

**Ex:-normal function:-**

**Def squareIt(n):**

**Return n\*n**

**Lambda function:**

**S=Lambda n:n\*n🡪 n=input n\*n=expresion**

**Print(s(4))🡪16**

**S=Lambda a,b:a+b**

**Print(‘The sum of {} and {} is:{}’.format(2,4,s(2,4)))**

**Print(‘The sum of {} and {} is:{}’.format(20,40,s(20,40)))**

**Print(‘The sum of {} and {} is:{}’.format(200,400,s(200,400)))**

1. **Filter():-**

**If you want to filter particular from a group the we use filter()**

**Syntax:- filter(function,sequence)**

**Ex:-def iseven(x):**

**If x%2==0:**

**Return True**

**Else:**

**Return False**

**L=[0,5,10,15,20,25,30]**

**L1=list(filter(iseven,l))**

**Print(l1)**

**Or**

**L=[0,5,10,15,20,25,30,35,40]**

**L1=list(filter(lmbda x:x%2==0,1))**

**Print(l1)**

**L2=list(filter(lmbda x:x%2!=0,1))**

**Print(l2)**

**2.Map()**

**If you want to find some function value or double value then we use map().**

**Not for checking condition but for any operation or business logic.**

**Syntax:-map(function,sequence)**

**L=[0,5,10,15,20,25,30,35,40]**

**L=[0,10,20,30,40,50,60,70,80]**

**Ex:- def double(x):**

**Return 2\*x**

**L=[1,2,3,4,5]**

**L3=[10,20,30,40]**

**L1=list(map(double,l)) 🡪[2,4,6,8,10]**

**L2=list(map(lambda x:x\*x,l) 🡪[2,16,36,64,100]**

**L4=list(map(lambda x,y:x\*y,l1,l3))🡪10,40,90,160**

**3.Raduce():-**

**Function aliasing:-**

**Nested Function:**

**The function which decleared inside another function such type of function bydefault consider as nested function.**

**Use:-this type of availability only in python not in java.**

**To define function specific repeatedly required functionality.**

**Ex:-def f1(): 🡪outer function**

**Def inner(a,b): 🡪inner function**

**Print(‘The Sum:’,a+b)**

**Print(‘the Average:’,(a+b)/2)**

**Inner(10,20)**

**Inner(20,30)**

**Inner(40,50)**

**Inner(100,200)**

**F1()**

**If I directly want to execute inner function**

**Def outer():**

**Print(‘outer function started’)**

**Def inner():**

**Print(‘inner function execution’)**

**Print(‘outer function returning inner function’)**

**Return inner**

**F1=outer()**

**F1()**

**F1()=outer()**

**Function Decorators:**

**Input function=== >Decorator Function ===> outputfunction with extended functionality**

**Decorators help to make our code shorter and more pythonic.**

**Ex:-def décor(func):**

**Def inner(name):**

**If name=‘sunny’:**

**Print(‘Hello sunny bad morning’)**

**Else:**

**Func(name)**

**Return inner**

**Decorfunction=décor(wish) or @decor**

**@decor**

**def wish(name):**

**Print(‘hello’,name,’Good morning’)**

**Wish(‘Durga’)**

**Wish(‘ravi’)**

**Wish(‘sunny’)**

**Ex:-def smartdivision(func):**

**Def inner(a,b):**

**If b==0:**

**Print(‘Hello stupid..How we can devide with zero:’)**

**Else:**

**Return func(a,b)**

**Return inner**

**@smartdivision**

**Def division(a,b):**

**Return a/b**

**Print(division(10,2))**

Print(division(10,5))

Print(division(10,0))

**Modules:-**

**A group of functions, classes and variables saved to a file.**

**Advantages of modules**

* **Length of the code will be reduced and readability**
* **code reusability**
* **Maintainability.**

**Import module:-**

**Import module1**

**Import module1, module2, module3**

**Import module1 as m1**

**Ex:-module1.add(10,20)**

**Durgamath.add(10,20)**

**Durgamath.x**

**Ex:-From module1 import add,x,product**

**Ex:-From module1 import \***

**Ex:-form module1 import add as a:**

**Module1.add(10,20)🡺a(10,20) 🡪instead**

**PYC🡺\_pycache\_ 🡪in this file module complile code will be generated and save.**

**Reloading a module:-**

**Either you modify your module but it loaded only one time.but we can remove this dimarits by using reload() .**

**Imp module**

**Reload () module**

**Ex:- import time**

**From imp import reload**

**Import module1**

**Print(‘programe entering into sleeping state’)**

**Time.sleep(30)**

**Reload(module1)**

**Print(‘programe entering into sleeping state again’)**

**Time.sleep(30)**

**Reload(module1)**

**Print(‘This is the last line of program’)**

**Print(‘This is first version of module’)**

**Print(‘This is second version of module’)-🡪after 30 second it will execute.**

**Finding members of the module:-**

**Dir() function :-it is going to show list of current module members.**

**Ex:-x=10**

**Y=20**

**Def f1():**

**Print(‘hello’)**

**Print(dir())**

**Dir(module name) 🡺 Specified module members**

**\_builtins\_**

**\_cached\_**

**\_doc\_**

**\_file\_**

**\_name\_ 🡺to check whether the module executed directly or this module executed from another programe.**

**Math module:-**

**Ex:-Import math**

**help(math)🡪like documentation**

**working with random module:-**

**it used to generate random numbers.**

**Ex:-Games, otp,capcha,lottery**

**1.random():-**

**This function generate float value between 0 and 1(not inclusive)**

**0<x<1**

**Ex:-from random import \***

**For I in range(10):**

**Print(random()) 🡪0.4485……,0.0144…,0.6835…..**

**2.randint():**

**Generate only int value between ranges,.**

**1 and 100 or 1 and 1000(inclusive)**

**Randint(1,100)**

**Ex:-from random import \***

**For I in range(10):**

**Print(randint(1,100)) 🡪1,100,23,3,98…..**

**3.uniform():**

**Generate float value but not inclusive.**

**Uniform(1,10)**

**Ex:- :-from random import \***

**For I in range(20):**

**Print(uniform(1,10))🡪2.144…4.654…5.544….**

**4.randrange(start,stop,step):**

**Returns a random number from the range.(start is incluve stop is exclusive)  
start<= x <stop-🡪step is optional**

**Ex:- :-from random import \***

**For I in range(20):**

**Print(randrange(10))🡪0…..9**

**Print(randrange(1,11))🡪1…..10**

**Print(randrange(1,11,2))🡪1,9,3,…..5,5,….7**

**5.choice():-**

**It won’t return random number**

**It will return a random object.**

**Ex:-from random import \***

**List=[‘sunny’,’bunny’,’chinny’,’vinny’]**

**For I in range(10):**

**Print(choice(list))-🡪sunny..chinni…chinni…bunny…sunny**

**Print(‘durga’) 🡪r..d..r..g..a..**

**Q.WAP to generate a 6-digit random number as OTP.**

**Option1:- From random import \***

**For i in range (10)**

**Print(randint(0,9),randint(0,9) ,randint(0,9) ,randint(0,9) ,randint(0,9) ,randint(0,9),sep=’’)-🡪 453556…148765….232355..**

**Option2:- :- From random import \***

**For i in range (10)**

**Print(randint(100000,999999))🡪 453556…148765….232355..**

**Q.WAP to generate a 6-digit random password .**

**1,3,5 are alphabets symbols**

**2,4,6 are digits**

**From random import \***

**For i in range (10)**

**Print(char(randint(65,65+25)),randint(0,9) , char(randint(65,65+25)),randint(0,9) ,sep=’ ’)🡪U4O7U0…K9P0P4…K8A5F5**

**Exceptions Handling:-**

**Types of Errors:-**

**1.syntax error:-**

**Because of invalid syntax syntax error occurs.**

**Ex:-x=10**

**If x==10 🡪: is not there**

**Print(“x value is 10”)🡪invalid syntax.**

**Ex:-print “hello friends” 🡪syntax error missing () in python 3.**

**2.Runtime error or exception:-**

**At runtime Wrong input , memory problem ,wrong logic then we will will get runtime error or exception. Exception handling is for runtime error but not for syntax error.**

**Ex:-x=int(input(“Enter First Number”) 🡪10 or Ten**

**Y= int(input(“Enter Second Number”) 🡪5**

**print(“The Result:”,x/y) 🡪zerodivisionError:division by zero or**

**ValueError:invalid literal for int() base 10:’ten’**

**Ex:-f=open(‘xyzxyz.txt’)**

**Print(f.read()) 🡪FileNotFoundError: No such file or directory: ‘xyzxyz.txt’**

**2.The 3 most important questions about Exception Handling**

**What is Exception?**

**An Unwanted, unexpected event which disturb the normal flow control of the program is known as exception.**

**Ex:-internet error, sleeping error, TyrePuncture Error, FileNotFound Error, ZeroDivisionError, Value Error.**

**What is the main objective of exception?**

* **It is highly recommended to handle exceptions.**
* **The main objective of exception handling is Graceful or normal termination of the application (i.e. we should not block our resources and we should not miss anything. all the costly resources should be closed.**

**What is the meaning of exception handling?**

* **Exception handling does not mean repairing an exception. we have to define alternative way to continue rest of the program normally.**

* **This way of defining alternative is nothing but exception handling.**

**Ex:-try:**

**Read data from remote file**

**Locating at London**

**Except FilenotFoundException:**

**Use local file and continue rest of the program normally.**

**3.Default Exception Handling**

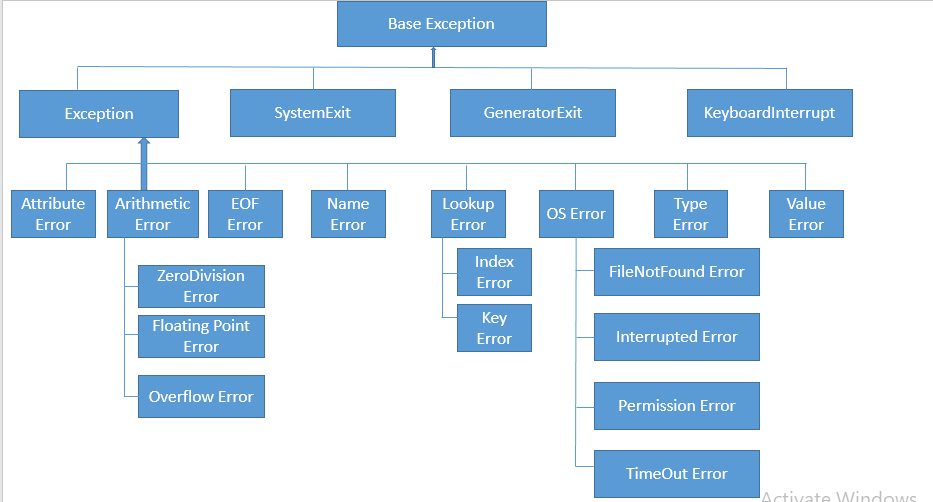
**Every exception in python is an object. For every exception type the corresponding class is available**

**Whenever an exception occurs PVM will create the corresponding exception object and will check for handling code. If handling code is available then it will be executed and rest of the program will be executed normally.**

**If handling code is not available PVM will terminated program abnormally and print corresponding exception information to the console. The rest of the program won’t be executed.**

**To prevent this abnormal termination we should handle the exceptions explicitly. Of course by using try except blocks.**

**Exception Hierarchy:-**



**4. Customized Exception handling by using try-except**

**Try:**

**Risky code**

**Except:**

**Handling code**

**Ex:-print(‘statement-1’) 🡪 statement-1**

**try:**

**print(10/0)**

**except ZeroDivisionError:**

**print(10/2) 🡪5.0**

**print(‘statement-3’) 🡪 statement-3**

**5.Control flow in try-except**

**Ex:- try:**

**Statement-1**

**Statement-2**

**Statement-3**

**Except xxx:**

**Statement-4**

**Statement-5**

**Case1:- if there is not any exception.**

**1,2,3,5,NT will ne excuted**

**Case2:-if an exception raised at stmt-2 and corresponding except block mathched. 🡪 1,4,5,NT will be executed.**

**Case3:- if an exception raised at stmt-2 but corresponding except block not mathed. 🡪1,AT**

**Case4:- if exception raised at stmt-4 or stmt-5. 🡪AT**

* **Within the try block if anywhere an exception raised then rest of the try block won’t be executed even though we handled the exception. Hence inside try block we have to take only risky code and the length of the try block should be as less as possible.**

* **in addition to try block, there may be a chance of raising exceptions inside except and finally block.**
* **If any statement which not part of try block, raising an exception then it is always abnormal termination**

**6.How to print Exception information to the console?**

**Ex:-try:**

**X=int(input(‘Enter First number’))**

**Y=int(input(‘Enter Second number’))**

**Print(‘The Result:’x/y)**

**Except BaseException as msg:**

**Print(‘Exception type:’,type(msg)) 🡪Exception Type:<class ‘zeroDivisionError’>**

**Print(‘The type of exception :’,msg.\_\_class\_\_)🡪 Exception Type:<class ‘zeroDivisionError’>**

**Print(‘The Exception class name:’msg.\_\_class\_\_.\_\_name\_\_)🡪 Exception class name: ZeroDivisionError**

**Print(‘The Discription of exception:’,msg)🡪 Discription of exception: ZeroDivisionError**

**7.try with multiple except block**

* **The way of handling an exception is varied from exception to exception.**
* **Hence for every possible exception type,we have to take a separate except block.try with multiple except blocks is possible and recommended to use.**

**Syntax:-try:**

**………..**

**………...**

**…………..**

**Except ZeroDivisionError:**

**Perform alternative arithametics operations**

**Except FileNotFoundException:**

**Use local file instead of remote file**

* **If try with multiple except blocks available, the based on raised exception the corresponding except block will be executed.**
* **If try with multiple except blocks available, then , The order of these except blocks is important.**
* **Python virtual machine will always consider from top to bottom until matched except block identified.**

**Ex:-try:**

**Print(10/0)**

**Except ArithmaticError:**

**Print(‘Arithmatic Error’) 🡪this block will excute.**

**Except ZeroSDivisionError:**

**Print(‘ZeroDivisionError’)**

**Ex:- try:**

**X=int(input(‘Enter First number’))**

**Y=int(input(‘Enter Second number’))**

**Print(‘The Result:’x/y)**

**Except ZeroDivisionError:**

**Print(‘cannot divide by zero’)**

**Except ValueError:**

**Print(‘please provide int values only’)**

**8.single except block that can handle multiple different exception**

**If handling code is same for multiple exceptions, then instead of taking different except block, we can take single except block that can handle all those exceptions.**

**Syntax:-**

**Except(Exception1,Exception2,….)**

**Except(Exception1,Exception2,….) as msg:**

**Parenthesis are mandatory and this group of exceptions internally consided as tuple.**

**Ex:- try:**

**X=int(input(‘Enter First number’))**

**Y=int(input(‘Enter Second number’))**

**Print(‘The Result:’x/y)**

**Except (ZeroDivisionError, ValueError) as msg:**

**Print(‘The Raised exception :’,msg.\_\_class\_\_)🡪 Exception Type:<class ‘zeroDivisionError’>**

**Print(‘The Discription of exception:’,msg)🡪 Discription of exception: ZeroDivisionError**

**Print(“Please provide valid input only..”)**

**9. Default Exception Block and various except block syntaxes:-**

* **we can use default except block to handle any type of exceptions.**
* **In default except block, generally we can print exception information to the console.**
* **Default except block must be at the last except block otherwise it will give syntax error.**
* **This restriction is applicable only for default except block but not for normal except blocks. i.e. normal except blocks can be in any order.**

**Syntax:-except:**

**Statements**

**Ex:- try:**

**X=int(input(‘Enter First number’))**

**Y=int(input(‘Enter Second number’))**

**Print(‘The Result:’x/y)**

**Except ZeroDivisionError:**

**Print(‘ZeroDivisionError, cannot divided by zero’)**

**Except:**

**Print(‘default except:please provide valid input only’)**

**various except block syntaxes:-**

**except (ZerodivisionError):**

**except ZerodivisionError as msg:**

**except (ZerodivisionError) as msg:**

**except (ZerodivisionError,ValueError):**

**except (ZerodivisionError,ValueError) as msg:**

**except :**

**if except block is defined for only one exception then parenthesis are optional.if multiple exceptions are there then parenthesis are mandatory.**

**If we use parenthesis, then ‘as’ must be outside of parenthesis only.**

**10.finally block purpose and specialty**

* **It is not recommended to place cleanup code(resource deallocation code like closing database connection etc) inside try block, because there is no guarantee for execution for every statement inside try block.**
* **It is not recommended to place cleanup code inside except block, because if there is no exception then except block won’t be executed.**
* **Hence we required some place to maintain cleanup code which should be executed always irrespective of whether exception raised or not raised and whether exception handled or not handled. such type of best place is nothing but finally block.**
* **Hence the main purpose of finally block is to maintain cleanup code.**

**Syntax:-try:**

**Risky code**

**Except:**

**Handling code**

**Finally:**

**Cleanup code**

**The specialty of finally block is, it will be executed always irrespective of whether exception raised or not raised and whether exception handled or not handled.**

**Ex:-**

**Case1:- if there is no exception.**

**try:**

**Print(‘try’)**

**Except :**

**Print(‘Except’)**

**Finally:**

**print(‘finally’)🡪try finally**

**Case2:- if exception raised and handled.**

**try:**

**Print(‘try’)**

**Print(10/0)**

**Except ZeroDivisionError:**

**Print(‘Except’)**

**Finally:**

**Print(‘finally’) 🡪try except finally**

**Case3:- if exception raised but not handled.**

**Ex:- try:**

**Print(‘try’)**

**Print(10/0)**

**Except ValueError:**

**Print(‘Except’)**

**Finally:**

**Print(‘finally’) 🡪try finally zeroDivisionError**

**11.finally block vs os.\_exit(0)**

* There is only one situation where finally block won’t be executed ‘.e. whenever we are using os.\_exit(0)
* Whenever we are using os.\_exit(0) then python virtual machine itself will be shutdown .in this particular case finally block won’t be executed.

**Os.\_exit(0) :-**

* Here zero represent status code.
* Zero means normal termination.
* Non-zero means abnormal termination.
* This status code internally used by pvm.
* Whether it is zero or non-zero there is no difference in the result of the program.

**Ex:-import os**

**Try:**

**Print(‘try’)**

**Os.\_exit(0)**

**except ValueError:**

**Print(‘Except’)**

**finally:**

**Print(‘finally’)🡪try**

**12. Difference between finally and destructor**

**Finally block meant for cleanup activities related to try block. i.e. whatever resources we opened as the part of try block will be closed inside finally block.**

**Destructor meant for cleanup code activities related to object. Whatever resources associated with the object should be deallocated inside destructor, which will be executed before destroying object.**

**13.control flow in try-except-finally**

**Ex:-try:**

**Stmt-1**

**Stmt-2**

**Stmt-3**

**Except:**

**Stmt-4**

**Finally:**

**Stmt-5**

**Stmt-6**

**Case1:-**if no exception raised**🡪1,2,3,5,6,NT**

**Case2**:- if exception raised at stmt-2 and corresponding except block matched**🡪1,4,5,6,NT**

**Case3**:- if exception raised at stmt-2 and corresponding except block not matched**🡪1,5,AT**

**Case4**:- if exception raised at stmt-4 then it is always AT, only before AT finally block will be executed

**Case5**:- if exception raised at stmt-5 and step-6

**Then it is always AT.**

**14.nested try-except-finally theory and demo program**

We can take try-except-finally blocks inside try or except or finally. Hence nesting of try-except-finally blocks is possible.

General risky code we have to take inside outer try block and too much risky code we have to take inside inner try block.

Inside inner try block if an exception raised then inner except block is responsible to handle. If it is unable to handle then outer except block is responsible to handle.

**Try:**

**Try:**

**Except:**

**Finally:**

**Except:**

**Try:**

**Except:**

**Finally:**

**Finally:**

**Try:**

**Except:**

**Finally:**

**Ex:-**if there is no exception **🡪1,2,4,6 print**

if there is exception in inner try (10/0)🡪**1,2,4,5,6**

If there is exception is divide by zero and except block is ValueError **🡪1,2,4,5,6**

if there is exception in outer try (10/0)🡪**1,5,6**

* Here inner finally block will not execute because if control didn’t enter to inner try block then also finally block won’t be executed.
* Once control entered in the try block compulsory the corresponding finally block will be executed.

**Try:**

**Print(‘outer try block’)**

**Try:**

**Print(‘inner try block’)**

**Print(10/0)**

**Except ZeroDivisionError:**

**Print(‘inner Except block’)**

**Finally:**

**Print(‘inner finally block’)**

**Except:**

**Print(‘outer except block’)**

**Finally:**

**Print(‘outer final block’)**

**15.Control Flow in Nested try-Except-finally**

**Try:**

**Stmt-1**

**Stmt-2**

**Stmt-3**

**Try:**

**Stmt-4**

**Stmt-5**

**Stmt-6**

**Except:**

**Stmt-7**

**Finally:**

**Stmt-8**

**Stmt-9**

**Except yyy:**

**Stmt-10**

**Finally:**

**Stmt-11**

**Stmt-12**

**Case1:-** if there is no exception no except will execute**🡪1,2,3,4,5,6,8,9,11,12 print**

**Case2:-** if exception raised at stmt-2 and corresponding except block matched**🡪1,10,11,12,NT**

**Case3**:- if exception raised at stmt-2 and corresponding except block not matched**🡪1,11,AT**

**Case4:-** if exception raised at stmt-5 and inner except block matched **🡪1,2,3,4,7,8,9,11,12,NT**

**Case5:-** if exception raised at stmt-5 and inner except block not matched but outer except block matched. **🡪1,2,3,4,8,11,12,NT**

**Case6**:- if exception raised at stmt-5 and both inner and outer except block not matched **. 🡪1,2,3,4,8,11,AT**

**Case7:-** if exception raised at stmt-7 and corresponding except block matched**🡪12,3,(4,5,6 may or may not),8,10,11,12,NT**

**Case8:-** if exception raised at stmt-7 and corresponding except block not matched **🡪1,2,3, ,(4,5,6 may or may not),8,10,11,AT**

**Case9:-** if exception raised at stmt-8 and corresponding except block matched**🡪1,2,3, ,(4,5,6,7 may or may not),10,11,12,NT**

**Case10**:- if exception raised at stmt-8 and corresponding except block not matched**. 🡪1,2,3, ,(4,5,6,7 may or may not),11,NT**

**Case11:-** if exception raised at stmt-9 and corresponding except block matched**🡪1,2,3, ,(4,5,6,7 may or may not),8,10,11,12,NT**

**Case12:-** if exception raised at stmt-9 and corresponding except block not matched**🡪1,2,3, ,(4,5,6,7 may or may not),8,11,AT**

**Case13:-** if exception raised at stmt-10 then it is always AT but before AT only finally block will be executed.

**Case14:-** if exception raised at stmt-11 or stmt-12 then it is always.

**16. else block with try-except-finally**

**If-else === >if condition is false then only else will be executed.**

**For-else === > if loop executed without break then only else will be executed.**

**While-else == > if loop executed without break then only else will be executed.**

**Try-except-else-finally**

**Try:**

**Risky code**

**Except:**

**Handling code**

**It will be executed if exception in try block.**

**Else:**

**It will be executed if there is no exception in try block.**

**Finally:**

**Cleanup code**

**It will be executed whether exception raised or not.**

**It will be executed whether exception handled or not.**

**Case1:-If there is no exception.**

**Ex:-try:**

**Print(‘try’)**

**Except:**

**Print(‘except’)**

**Else:**

**Print(‘else’)**

**Finally:**

**Print(‘finally’) 🡪try else finally**

**Case2:-If there is exception.**

**Ex:-try:**

**Print(‘try’)**

**Print(10/0)**

**Except:**

**Print(‘except’)**

**Else:**

**Print(‘else’)**

**Finally:**

**Print(‘finally’) 🡪try except finally**

**Case3:-if we use and don’t use else block we will get syntax error.**

**Ex:- try:**

**Print(‘try’)**

**Print(10/0)**

**Else:**

**Print(‘else’)**

**Finally:**

**Print(‘finally’) 🡪syntax error**

**17.else block with try-except finally demo programs**

**Ex:-f=none**

**Try:**

**F=open(‘abc.txt’,’r’)**

**Except:**

**Print(‘some problem while locating and opening the file’)**

**Else:**

**Print(‘file opened successfully’)**

**Print(‘the data present in the file is:’)**

**Print(‘#’\*30)**

**Print(f.read())**

**Finally:**

**If f is not none:**

**f.close()**

**18. various possible combinations of try-except-else-finally**

**1.whenever we are writing try block, compalsary we should write except or finally blocks.**

**i.e. try without except or finally is always invalid.**

**2.Whenever we are writing except block, compalsary try should be there.**

**i.e. except without try is always invalid.**

**3.whenever we are writing finally block, compulsory try block should be there.**

**i.e. finally without try is always invalid.**

**4. whenever we are writing else block, compulsory except block should be there.**

**i.e. else without except is always invalid.**

**5.we can write multiple except blocks for the same try.but we cannot write multiple else blocks and finally blocks.**

**6.in try-except-else-finally order is important.**

**7.we can write try-except-else-finally inside try, except,else and finally blocks. Hence nesting of try-except-else-finally is always possible.**

**1.Try:**

**Print(‘try block’) 🡪invalid**

**2.except:**

**Print(‘except block’) 🡪invalid**

**3.else:**

**Print(‘else block’)🡪invalid**

**4.finally:**

**Print(‘finally block’) 🡪invalid**

**5.try:**

**Print(‘try block’)**

**Except:**

**Print(‘except block’) 🡪valid**

**6.try:**

**Print(‘try’)**

**Else:**

**Print(‘else’) 🡪invalid**

**7.try:**

**Print(‘try’)**

**Finally:**

**Print(‘finally’) 🡪valid(before AT close my connection)**

**8. try:**

**Print(‘try’)**

**Else:**

**Print(‘except’)**

**Except:**

**Print(‘Except’)🡪invalid because except must be first**

**9. try:**

**Print(‘try’)**

**Else:**

**Print(‘except’)**

**finally:**

**Print(‘finally’)🡪invalid(without except no else)**

**10. try:**

**Print(‘try’)**

**Except:**

**Print(‘Except’)**

**Else:**

**Print(‘except’)**

**Finally:**

**Print(‘finally’)🡪valid**

**11. try:**

**Print(‘try’)**

**Except:**

**Print(‘Except’)**

**try:**

**Print(‘try’)**

**Finally:**

**Print(‘finally’)🡪valid**

**12. try:**

**Print(‘try’)**

**Except:**

**Print(‘Except’)**

**try:**

**Print(‘try’)**

**else:**

**Print(‘else’) 🡪invalid**

**13. try:**

**Print(‘try’)**

**Except xxx:**

**Print(‘Except1’)**

**Except yyy:**

**Print(‘Except2’) 🡪valid**

**14. . try:**

**Print(‘try’)**

**Except:**

**Print(‘Except’)**

**Else:**

**Print(‘except1’)**

**Else:**

**Print(‘except2’) 🡪invalid because multiple else not psbl**

**15 . try:**

**Print(‘try’)**

**Except:**

**Print(‘Except’)**

**finally:**

**Print(‘finally1)**

**finally:**

**Print(‘finally2’) 🡪invalid**

**16. try:**

**Print(‘try’)**

**Except:**

**Print(‘Except’)**

**If 10>20:**

**Print(‘if’)**

**Else**

**Print(‘else’) 🡪valid**

**17.try:**

**Print(‘try’)**

**Print(‘hello’)**

**Except:**

**Print(‘except’) 🡪invalid**

**18.try:**

**Print(‘try’)**

**Except xxx:**

**Print(‘except1’)**

**Print(‘heloo’)**

**Except yyy:**

**Print(‘except2’) 🡪invalid**

**19. try:**

**Print(‘try’)**

**Except:**

**Print(‘except’)**

**Print(‘hello’)**

**Else:**

**Print(‘else’) 🡪invalid**

**20. try:**

**Print(‘try’)**

**Except:**

**Print(‘except’)**

**Print(‘hello’)**

**Finally:**

**Print(‘finally’) 🡪invalid**

**21. try:**

**try:**

**Print(‘inner try’)**

**Except:**

**Print(‘inner except’)**

**Finally:**

**Print(‘inner finally’)**

**Except:**

**Print(‘except’)🡪valid**

**22. try:**

**Print(‘inner try’)**

**Except:**

**Try:**

**Print(‘inner try)**

**except:**

**print(‘inner except’)🡪valid**

**Print(‘inner finally’) 🡪valid**

**23. try:**

**Print(‘inner try’)**

**Except:**

**Print(‘inner except’)**

**else:**

**try:**

**Print(‘inner try’)**

**Finally:**

**Print(‘finally’)🡪valid**

**24. . try:**

**Print(‘inner try’)**

**Except:**

**Print(‘inner except’)**

**Finally:**

**Try:**

**Print(‘inner try’)**

**Except:**

**Print(‘inner except’)🡪valid**

**25.try:**

**Try:**

**Print(‘inner try’)**

**Except:**

**Print(‘outer except’)🡪invalid**

**26. try:**

**Try:**

**Print(‘inner try’)**

**Finally:**

**Print(‘inner finally’)**

**Except:**

**Print(‘outer except’) 🡪valid**

**19.Types of exceptions Predefined and User Defined**

**Types of exceptions:**

**1.predefined exception**

* Also known as inbuilt exceptions or PVM Exceptions

* These will be raised automatically by PVM whenever a particular event occurs.

**#zeroDivisionError 🡪print(10/0)**

**#ValueError 🡪x=int(‘10’) or x=int(‘ten’)**

1. **user defined exception or customized exception**

* Also known as customized exceptions or programmatic exceptions.

* Sometimes we have to define and raise exceptions explicitly to indicate that something goes wrong, such type of exceptions are called user defined exceptions or customized exceptions or programmatic exceptions.
* Programmer is responsible to define these exceptions and python virtual machine not having any idea about these. Hence we have to raise explicitly based on our requirement by using ‘raise’ keyword.

**Ex:- InsufficientFundsException**

**InvalidPINException**

**TooYoungException**

**TooOldException**

**20.How to define and raise Customized Exceptions & Demo Programs**

Every exception in python is a class and it should be child class of **BaseException**.

**Syntax: class NameofException(PredefinedException):**

**Def \_init\_(self,msg):**

**Self.msg=msg**

**Ex: class ToYoungException(Exception):**

**Def \_init\_(self,msg):**

**Self.msg=msg**

**TooYoungException** is our exception class name and it is the child class of exception.

We can raise exception by using **raise** keyword

**Ex:- raise TooYpoungexception(‘message’)**

**Ex:-class TooYoungException(Exception):**

**Def \_\_init\_\_(self,msg):**

**Self.msg=msg**

**class TooOldException(Exception):**

**Def \_\_init\_\_(self,msg):**

**Self.msg=msg**

**Age=int(input(‘Enter Age:’))**

**If age>60:**

**Raise TooYoungException(‘Please wait some more time , definitely you will get best match’)**

**Elseif age<18:**

**Raise TooOldException(‘Your age already crossed marriage age, no chance of getting marriage’)**

**Else:**

**Print(‘You will get match details soon by email!!!’)**

**Logging :-**

**Ex:-log book ,internet center**

* **For storing record we use log file.**
* **By using log files we can debugging.**
* **Provide statistics information.**

**By logging python record some event of applications.**

**Logging levels:**

**We can able to log information by using cbrp**

**6 levels**

**1>.CRITICAL🡺50🡪numbers**

**2>.ERROR🡪40 🡪serious but not critical**

**3>.WARNING🡪30🡪alert to the programmer**

**4>.INFO🡪20🡪with some important information**

**5>.DEBUG🡪10**

**6>.NOTSET🡪0🡪may not support something**

**How to implement logging:**

**Name of the file**

**Level messages**

**basicConfig() of logging module**

**logging.basicConfig(filename=’log.txt’,level=logging.WARNING)**

**logging.debug(message)**

**logging.info(message)**

**logging.warning(message)**

**logging.error(message)**

**logging.critical(message)**

**Q.wapp to create a log file and write WARNING and higher level mess.**

**Import logging**

**logging.basicConfig(filename=’log.txt’,level=logging.WARNING)**

**print(“python logging demo”)**

**logging.debug(“debug message”)**

**logging.info(“info message”)**

**logging.warning(“warning message by default”)**

**logging.error(“error message”)**

**logging.critical(“critical message”)**

**ex:- Import logging**

**logging.basicConfig(filename=’log.txt’,level=logging.INFO)**

**logging.info(“a new request came”)**

**try:**

**x=int(input(“Enter First Number:”))**

**y=int(input(“Enter Second Number:”))**

**print(x/y)**

**Except ZeroDivisionError as msg:**

**Print(“cannot divide with zero”)**

**Logging.exception(msg)**

**Except ValueError as msg:**

**Print(“Enter only int values”)**

**Logging.exception(msg)**

**logging.info(“Requested Processing Compleated”)**

**debugging:   
defect/bug**:Mismatch between expected result and actual result.

**Debugging is done before given project to client. After dev and test.**

**The most common way of debugging is uses of print().**

**But the problem with print() that all print will be excute either error or not error means memory wastage.so print() is not recommended.**

**We should go for assert statements fur debugging puposes.**

**Assert statement we can add any no. but after execution assert statement is removed or not execute like print.**

**Types of assert statement:-**

**1>simple version**

**2>.very simple version( Argumented version)**

**FileHandling:-**

**File:-File is used to store our data permanently we should go for file if our data is very small.**

**Types of Files:-**

**1>.Text Files:-**

**To store character data.**

**Eg.abc.text**

**2>.Binary Files:-**

**Images, videos, audios files, zip files. Etc.**

**Open files:-**

**F=open(filename,mode)🡪mode should be read write and append**

**F=open(abc.text,’w’)**

**The allowed values for mode are:-**

**r, w, a, r+, w+, a+, x**

**applicable only for text files.**

**1.r:**

**Open an existing file for read operation.**

**F=open(‘abc.text’)🡪this is the default mode**

**F=open(‘abc.text’,’r’)**

**If the specified file is not available then you got filenotfound error.**

**2.w:**

**Open an existing file for write operation.**

**F=open(‘abc.text’,’w’)**

**If the specified file is available but contains some data..overwrite.**

**If the specified file is not available then the required file will be created automatically.**

**3.a: 🡺apend**

**Open an existing file for apend operation.**

**F=open(‘abc.text’,a)**

**If the specified file is available your old data will be there not override**

**If the specified file is not available then the required file will be created automatically.**

**If I want to append from a particular location**

**f.seek()**

**4>.r+: 🡪read and write**

**5>.w+: 🡪write and read(override data)**

**6>a+: 🡪append and read**

**7>.x: 🡪**

**To open a file in excusive mode for written operation.**

**Open an existing file for write operation.**

**F=open(‘abc.text’,’x’)**

**the specified file should not be already available but first create new file.**

**If the specified file is available then FileExistsError**

**For binary files:-**

**Rb, wb, ab, r+b, w+b, a+b, xb**

**Getting properties of File:-**

**f.name🡪name of the append file**

**f.mode🡪In which mode file is opened**

**f.closed 🡪returns Boolean value**

**f.readable()**

**f.writeable()**

**#open a file and close that file:-**

**F=open(‘abc.txt’,’w’)**

**Print(‘Name:’,f.name) 🡪abc.txt**

**Print(‘Mode:’,f.mode) 🡪w**

**Print(‘Closed:’,f.closed) 🡪False**

**Print(‘is readable:’,f.readable()) 🡪False**

**Print(‘is writeable’,f.writeable()) 🡪True**

**f.close()**

**Print(‘Closed:’,f.closed) 🡪True**

**#writing data to the the file:**

**F=open(‘abc.txt’,’w’)**

**f.write(string) 🡪to write one word or one line**

**f.writelines(list of lines)🡪to write list of lines**

**#Reading character data from the file:**

**f.read()🡺To read total data**

**f.read(n)🡪to read ‘n’ characters from the file**

**f.readline()🡺to read only one single line**

**f.readlines()🡺to read all lines into a list**

**ex:-f=open(‘abc.text’,’r’)**

**data=f.read()**

**print(data)**

**f.close()**

**Q.if read() can read all data then why we use readlines()?**

**f.read(10) can use with character count but we don’t know the how many character are there.**

**Q.how to access the content from files?**

**F=open(‘abc.txt’,’r’)**

**Lines=f.readlines()**

**For line in lines:**

**Print(line,end=’’)**

**With statement:-**

**If you open a file with with statement then you don’t need to close file it automatically closed even exception is there.**

**With open(‘abc.txt’,’w’) as f:**

**f.write(‘Durrga\n’)**

**f.write(‘softwares\n’)**

**f.write(‘solutions \n’)**

**print(‘Is File Closed:’,f.closed)**

**tell():-**

**to return current position of the cursor(file pointer) from beginning of the file.**

**The first position is zero index.**

**if mode is r 🡪begin**

**w🡪begin**

**a🡪last**

**Ex:-f=open(‘abc.txt’,’r’)**

**Print(tell()) 🡪0**

**Print(f.read(2)) 🡪 DU**

**Print(tell())🡪2**

**Print(f.read(3)) 🡪 rga**

**Print(tell())🡪5**

**seek():-**

**it changes the current position means cursor movement backward and forward**

**f.seek(17)🡪from the beginning the cursur will be movied to 17th position**

**f.seek(0) 🡪 from the end the cursur will be movied to 0th position**

**ex:-data=’All Students are STUPIDS’**

**f=open(‘abc.txt’,’w’)**

**f.write(data)**

**with open(‘abc.txt’,’r+’) as f:**

**text=f.read()**

**print(text) 🡪All students are STUPID**

**print(‘The current position:’f.tell())**

**f.seek(17)**

**print(‘The Current Cursor Position:’,f.tell())**

**f.write(‘GEMS!!!’)**

**f.seek(0)**

**text=f.read()**

**print(‘data After Modification:’)**

**print(text)🡪 All students are GEMS!!!**

**Os module contains one method i.e. isfile():**

**By using isfile() the specified file is available or not.**

**Ex:-import os**

**Fname=input(‘Enter file Name:’)**

**If os.path.isfile(fname)**

**Print(‘File Exists:’,fname)**

**F=open(fname,’r’)**

**Print(‘The content of the file is:’)**

**Print(f.read())**

**Else:**

**Print(‘file does not exists’,fname)**

**Wap in python to check file is present or not and count the no. ogf words,lines and characters present in that file**

**Ex:-import os**

**fname=input(‘Enter file Name:’)**

**If os.path.isfile(fname)**

**Print(‘File Exists:’,fname)**

**F=open(fname,’r’)**

**Lcount=wcount=ccount=0**

**For line in f:**

**Lcount=lcount+1**

**Words=line.split()**

**Wcount=wcount+len(words)**

**ccount=ccount+len(line)**

**print(‘The number of Lines:’,lcount)**

**print(‘The number of words:’,wcount)**

**print(‘The number of characters:’,ccount)**

**Else:**

**Print(‘file does not exists’,fname)**

**How to open read and write binary files?**

**F1=open(‘guido.mp4’,’rb’)**

**F2=open(‘newpicpython.mp4’,’wb’)**

**B=f1.read()**

**F2.write(b)**

**Print(‘New Image is available:newpicpython.jpg’)**

**Handling csv files: -🡪means excel files**

**Csv🡪comma separated values**

**Ex:- import csv**

**With open(‘actors.csv’,’w’,newline=’’) as f:**

**W=csv.writer(f)**

**w.writerow([‘NAME’,’ROLLNO’,’MARKS’,’ADDR’])**

**while True:**

**name=input(‘Enter Student Name:’)**

**roolno=int(input(‘Enter Student roll no:’))**

**marks=int(input(‘Enter Student marks:’))**

**addr=input(‘Enter Student address:’)**

**w.writerow([name,rollno,marks,addr])**

**option=input(‘Do you want to insert one more record[yes|no]:’)**

**if option.lower() == ‘no’:**

**break**

**print(‘Total student data written to csv file successfully’)**

**Zipping and Unzipping Files:-**

**1>.to improve memory utilization**

**2>.we can reduce transfer time**

**zipfile module**

**It contain one special class i.e. ZipFile Class**

**To create a zip file:**

**ZIP\_DEFLATED**

**f=ZipFile(‘files.zip’,’w’,’ZIP\_DEFLATED’)**

**Ex:-** **from zipfile import \***

**f=ZipFile('files7.zip','w',ZIP\_DEFLATED)**

**f.write('file1.txt')**

**f.write('file2.txt')**

**f.write('file3.txt')**

**f.close()**

**print('files7.zip created successfully')**

**to check file is created or not :-**

**dir files7.zip**

**unzip files:-**

**f=ZipFile('files7.zip','r',ZIP\_STORED)**

**namelist() 🡪to get list of files**

**Ex:- from zipfile import \***

**f=ZipFile('files7.zip','r',ZIP\_STORED)**

**names=f.namelist()**

**for name in names:**

**print(‘File Name:’,name)**

**print(‘The Content of this file is:’)**

**f1=open(name,’r’)**

**print(f1.read())**

**print(‘\*’\*10)**

**Directories:-**

**osmodule**

**Case1:- To know current working directory**

**Ex:-import os**

**Cwd=os.getcwd()**

**Print(‘Current Working Directory:’,cwd)**

**Case2:- To create a directory movies**

**Ex:-import os**

**os.mkdir(‘movies’)**

**os.mkdir(‘movies/prabhasmovies’)🡪create subdirectories**

**os.mkdir('E:\\moviess') 🡪inside another drive we can create dirctory.**

**os.makedirs(‘videos/testingvideos/seleniumvideos/durgavideos’)🡪create multiple subdirectories**

**Print(‘Directory created successfully:’)**

**Case3:- To remove a directory movies**

**Ex:-import os**

**os.rmdir(‘videos/testingvideos/seleniumvideos/durgavideos’)**

**os.removedir(‘videos/testingvideos/seleniumvideos/durgavideos’)🡪to remove all directories**

**Print(‘Directory deleted successfully:’)**

**Case3:- To rename a subdirectory movies**

**Ex:-import os**

**os.rename(‘javavideos’,’javacompletevideos’)**

**print(‘renamed successfully’)**

**case4:- To getting list of files**

**Ex:-import os**

**print(os.listdir('.'))**

**List=os.listdir()**

**for name in list:**

**print(name)**

**print('The number of files and directories:',len(list))**

**case4:- To getting list of hidden files**

**walk()**

**it is used to print hidden files.**

**Ex:-import os**

**for dirpath,dirnames,filenames in os.walk(‘.’):**

**print(‘Current Working Directories:’,dirpath)**

**print(‘ Directories:’,dirnames)**

**print(‘File Names:’,filenames)**

**print()**

**print()**

**print(type(os.walk(‘.’))🡪<class ‘generator’>**

**Running other programs from python program:**

**Os.system() to run programs and commands**

**Os.system(‘cmd string’)**

**Ex:-os.system(‘py zipfiles.py’)**

**os.system(‘dir’)**

**os.system(‘javac test.java’)**

**os.system(‘java test’)**

**os.system(‘notepad’)🡪for opening notepad**

**or directly write into cmd notepad for open notepad**

**Process API**

**We can able to get in java 9.**

**How to get information about a file:-**

**Size,last accessed time,last modified time etc**

**Stat() function 🡪the statistics of a file**

**St\_mode 🡪protection bits**

**St\_ino 🡪Inode number**

**St\_dev 🡪device**

**St\_nlink 🡪no of hard links for this file**

**St\_uid 🡪userid of owner**

**St\_gid 🡪group id of owner**

**St\_size 🡪size of file in bytes**

**St\_atime🡪 time in most recent times**

**St\_mtime🡪time in most recent modification**

**St\_ctime🡪time in most recent meta data change**

**Date and time directly**

**Fromtimestamp()**

**Ex:-Import os**

**from datatime import \***

**stats1=os.stat(‘abc.txt’)**

**print(‘File Size In Bytes:’,stats1.st\_size)**

**Print(‘File Last Accessed time:’,datetime.fromtimestamp(stats1.st\_stime)**

**Print(‘File Last Modified time:’,datetime.fromtimestamp(stats1.st\_mtime)**

**Pickling and unpickling of objects:-**

**It is equal to java’s serialization and deserialization.**

**Pickle**

**dump()**

**Load()**

**e=Employee(----)**

**pickle.dump(e,f)🡪pickling(writing)**

**e2=pickle.load(f)🡪unpicking(reading)**

**import pickle**

**Class Employee:**

**Def \_init\_(self,eno,ename,esal,eaddr):**

**Self.eno=eno**

**Self.ename=ename**

**Self.esal=esal**

**Self.eaddr=eddr**

**Def display(self):**

**Print(self.eno,’\t’,self.ename,’\t’,self.esal,’\t’,self.eaddr)**

**With open(‘emp.dat’,’wb’) as f:**

**e=Employee(100,’sunny’,1000,’Mumbai’)**

**pickle.dump(e,f)**

**print(‘Pickling of Employee Object completed’)🡪save by emp.dle**

**With open(‘emp.dat’,’rb’) as f:**

**Obj=pickle.load(f)**

**print(‘printing employee informarion after unpickling’)🡪100 sunny 1000 mumbai**

**obj.display()**

**pickling of multiple objects:-**

**Class Employee: 🡪save by emp.py**

**Def \_init\_(self,eno,ename,esal,eaddr):**

**Self.eno=eno**

**Self.ename=ename**

**Self.esal=esal**

**Self.eaddr=eddr**

**Def display(self):**

**Print(self.eno,’\t’,self.ename,’\t’,self.esal,’\t’,self.eaddr)**

**Import pickle,emp 🡪save by pick.py**

**F=open(‘emp.dat’,’wb’)**

**n=int(input(‘Enter number of Employess:’))**

**for I in range(n):**

**eno=int(input(‘Enter Employee number:’))**

**ename=int(input(‘Enter Employee name:’))**

**esal=float(input(‘Enter Employee Number:’))**

**eaddr=input(‘Enter Employee Address:’)**

**e=emp.Employee(eno,ename,esal,eaadr)**

**pickle.dump(e,f)**

**print(‘Employee objects pickled successfully’)**

**import pickle,emp 🡪unpick.py**

**f=open(‘emp.data’,’rb’)**

**print(‘Employee Details:’)**

**while True:**

**try:**

**obj=pickle.load(f)**

**obj.display()**

**except EOFError:**

**print(‘All Employees compleated’)**

**break**

**f.close()**

**Object Oriented Programming(Oops):-**

**Class**

**Object**

**Reference variable**

**Class:-**

**Class is nothing but it is a blueprint, template, mold, dye, design, plan or master copy for objects. Ex: -construction of a building**

**Object:-**

**The physical existence of a class Is nothing but object(one to many)**

**Reference variable:-**

**Object reference variables🡺one to one|one to many**

**Self variable:-**

* **Self is a reference variable which is always pointing to current object.**
* **Within the python class to refer current object we should use self variable**
* **Self is same like this keyword in java.**
* **The first argument to the constructor and instance method must be self**
* **PVM is responsible to provide value for self-argument and we are not required to provide explicitly.**
* **By using self we can declare instance variables.**
* **By using self we can access instance variables.**
* **Instead of self we can use any name…but recommended to use self.**

**Ex:-print(id(self)) 🡪3228112🡪within the class how you can access variable**

**S=Student() 🡪 outside the class how you can access variable**

**Print(id(s)) 🡪3228112**

**Ex:-class Student:**

**’’’This class developed by Durga for Demo purpose’’’ #doc\_string**

**Print(Student.\_\_doc\_\_)🡪this will print student class description**

**help(Students)🡪all information about student class**

**Def \_\_init\_\_(self,rollno,name): 🡪\_\_init\_\_ is a constructor**

**Self.rollno=rollno**

**Self.name=name**

**Def talk(self):**

**Print(“Hello My Name is:”,self.name)**

**Print(“My RollNo is:”,self.rollno)**

**S= Student(100,’sunny’)**

**Print(s.name)🡪sunny**

**Print(s.rollno)🡪100**

**s.talk()**

**S1= Student(200,’bunny’)**

**S1.talk()**

**S3= Student(300,’chinny’)**

**S3.talk()**

**Ex:-**

**class Employee:**

**‘’’doc string(description)’’’**

**Def \_init\_(self,eno,ename,esal,eaddr):**

**self.eno=eno**

**self.ename=ename**

**self.esal=esal**

**self.eaddr=eaddr**

**def info(self):**

**print(‘\*’\*20)**

**print(‘Employee number’,self.eno)**

**print(‘Employee Name’,self.ename)**

**print(‘Employee Salary’,self.esal)**

**print(‘Employee address’,self.eaddr)**

**print(‘\*’\*20)**

**e1=Employee(100,’Durga’,10000,’Hyd’)**

**e2=Employee(200,’pawan’,20000,’Chennai’)**

**e1.info()**

**e2.info()**

**Constructor:-**

* **Constructor is a special method in python.**
* **The name should be always: \_\_init\_\_() .**
* **Whenever we creating object constructor will be executed automatically and we are not required to call explicitly.**
* **The main objective to declare and initialize variables.**
* **For every object constructor will be executed only once.**
* **Constructor should have atleast one argument i.e. self.**
* **Under constructor variable name should be same as instance variable or it may be anything i.e. called dummy variable.**
* **It is not mandatory to take a constructor in every python programe, because if you are not taking any constructor then PVM bydefault generate a default constructor.**
* **In python constructor overloading or method overloading is not possible.**
* **Python support multiple constructor but it will execute only last one.**

**Ex:-class Test:**

**Def \_\_init\_\_(self):**

**Print(‘Constructor execution..’)**

**t=Test()**

**t1=Test()**

**t2=Test()**

**Ex:- class Test:**

**Def \_\_init\_\_(self):**

**Print(‘no -arg..’)**

**Def \_\_init\_\_(self,x):**

**Print(‘one-arg..’)**

**t=Test(20)**

**t1=Test(10)**

**Difference between constructors and methods:-**

**Methods Constructors**

1. **The name can be anything 1.It should be \_\_init\_\_() only**
2. **Method will be executed if we call 2.will be executed automatically if we we are creating objects.**
3. **Per object method can be called any 3. per object only once**

**no of time.**

**4. Business logic 4.to declare and initialize instance v variables**

**Ex:-class Student:**

**Def \_\_init\_\_(self,name,rollno):**

**Print(‘constructor execution..’)**

**Self.name=name**

**Self.rollno=rollno**

**Def display(self):**

**Print(‘method execution..’)**

**Print(‘Hello myself is:’,self.name)**

**Print(‘My RollNo is:’mself.rollno)**

**S=student(‘durga’,100)**

**s.display()**

**s.display()**

**s.display()**

**s.display()**

**Types of variables:-**

* **Instance Variables/object level variables**

**🡪The value of object varies from object to object i.e. instance variables**

**🡪The variables which are declare within the class or inside constructor using self.**

**🡪For every object a separate instance variable will be created.**

* **Static variables/class level variables**

**🡪 The value of object which not varies from object to object i.e. static variables. Ex:-college student’s name**

**🡪For all object one copy will be created at class level share for every object of class.**

**🡪It is declare inside the class directly but outside the methods and constructors.**

* **Local Variables/method variables**

**🡪The variable which is declare inside constructor or methods or without using self-keyword is called local variable.**

**Types of methods:-**

* **Instance methods/object related method**

**🡪The method where we are using instance variable.**

* **Class methods**

**🡪when we use @classmethod decorator then it is called class level method and cls must be required and when we use class level data**

* **Static methods**

**🡪when we use @staticmethod decorator and we can access by using class name or object name.**

**Self🡪current object**

**Cls🡪current class level object**

**Ex:-** **class Student:**

**cname='DURGASOFT'**

**def \_\_init\_\_(self,x,y):**

**self.name=x**

**self.rollno=y**

**def display(self):**

**print('hello Myself is:',self.name)**

**print('My RollNo is:',self.rollno)**

**@classmethod**

**def getCollegeName(cls):**

**print('College Name:',cls.cname)**

**@staticmethod**

**def findAverage(x,y):**

**print('Average:',(x+y)/2)**

**s1=Student('durga',100)**

**s1.findAverage(10,20)**

**s1.display()**

**Student.findAverage(10,20)**

**1.Instance variables:-**

**🡪If the value of a variable varied from object to object such type of variables are called instance variables.**

**🡪The variables which are declare within the class or inside constructor using self.**

**🡪For every object a separate copy of instance variable will be created.**

**Where we have to declare instance variables:-**

**1.Inside constructor by using self within the class**

**2.Inside instance method by using self within the class**

**3.from outside of the class by using object references**

**Ex:-class Student:**

**Def \_\_init\_\_(self,name,rollno):**

**Self.name=name**

**Self.rollno=rollno**

**Def display(self):**

**Print(‘Hello myself is:’,self.name)**

**Print(‘My RollNo is:’,self.rollno)**

**s=student(‘durga’,100)**

**s.display()**

**print(s.name,s.rollno)**

**s1=student(‘ravi’,101)**

**s1.display()**

**print(s1.name,s1.rollno)**

**How to access instance variables:**

* **Within the class by using self**
* **From outside of the class by using object reference**

**Ex:- class Student:**

**def \_\_init\_\_(self,name,rollno):**

**self.name=name**

**self.rollno=rollno**

**def info(self):**

**self.marks=60**

**s1=Student('durga',101)**

**s1.info()**

**s1.age=24 🡪a new variable will be added to s.**

**print(s1.\_\_dict\_\_)**

**s2=Student(‘pawan’,102)**

**s2.wife=’reshu’**

**print(s2.\_\_dict\_\_)**

**How to delete instance variables:**

* **del self.variablename**
* **del objectreference.variablename**

**Ex:- class Test:**

**Def \_\_init\_\_(self):**

**Self.a=10**

**Self.b=20**

**Self.c=30**

**Def delete(self):**

**Del self.b,self.c # del self**

**T1=Test()**

**T1.delete()**

**#del t1**

**#del t2.a**

**#del t2.b**

**Print(t1.\_\_dict\_\_)**

* **Static variables/class level variables**

**🡪 The value of object which not varies from object to object i.e. static variables. Ex:-college student’s name, one faculty for all student**

**🡪For all object one copy will be created at class level share for every object of class.**

**🡪It is declare inside the class directly but outside the methods and constructors.**

**🡪you can access static variable using class name and object name.**

**Ex:- class Student:**

**Cname=’DurgaSoft’**

**def \_\_init\_\_(self,name,rollno):**

**self.name=name**

**self.rollno=rollno**

**s1=Student('durga',101)**

**s2=Student(‘pawan’,102)**

**print(s1.name,s1.rollno,s1.cname)**

**print(s2.name,s2.rollno,s2.cname)**

**What are various places are there to declare static variables:-**

**1. Within the class directly but outside of any method.**

**2. inside constructor by using class name**

**3. Inside instance method by using class name**

**4. inside classmethod by using cls variable or classname**

**5.inside static method by using class name.**

**6.From outside of class by using classname.**

**Ex:-** **class Test:**

**a=10**

**def \_\_init\_\_(self):**

**Test.b=20**

**def m1(self):**

**Test.c=30**

**@classmethod**

**def m2(cls):**

**cls.d=40**

**Test.e=50**

**@staticmethod**

**def m3():**

**Test.f=60**

**t=Test()**

**t.m1()**

**Test.m2()**

**Test.m3()**

**Test.g=70**

**Print(Test.\_\_dict\_\_)** **)#a=10, g=70 only because object not created for other but after creating object you can print all.**

**How we can access static variables:-**

**We can access static variable either by class name or object reference.**

**Inside of the class:**

**Classname,self,cls**

**Outside of the class:**

**Object reference,classname**

**Ex:-class Test:**

**a=10**

**def \_\_init\_\_(self):**

**print(‘Inside Constructor’)**

**print(Test.a)**

**print(self.a)**

**def m1(self):**

**print(‘Inside instance method’)**

**print(Test.a)**

**print(self.a)**

**@classmethod**

**Def m2(cls):**

**Print(‘Inside classmethod’)**

**Print(Test.a)**

**Print(cls.a)**

**@staticmethod**

**Def m3():**

**Print(‘inside static method’)**

**Print(Test.a)**

**t=Test()**

**t.m1()**

**t.m2()**

**t.m3()**

**print(‘from outside of the class’)**

**print(Test.a)**

**print(t.a)**

**How to modify static variable:-**

**Within the class we should use classname, cls variable**

**From outside of the class:only classname**

**Ex:- class Test:**

**a=10**

**def \_\_init\_\_(self):**

**Test.a=20**

**@classmethod**

**Def m1(cls):**

**Cls.a=30**

**Test.a=40**

**@staticmethod  
 def m2():**

**Test.a=50**

**t=Test()**

**t.m1()**

**t.m2()**

**Test.a=60**

**print(Test.a) #50**

**print(t.a)#60**

**Ex:-class Test:**

**X=10**

**Def \_\_init\_\_(self):**

**Self.y=20**

**T1=Test()**

**T2=Test()**

**Test.x=888**

**T1.y=999**

**Print(t1.x,t1.y)**

**Print(t2.x,t2.y)**

**how to delete static variables:**

**within the class we should use**

**classname, cls variable**

**from outside of the class:only classname**

**Ex:-class Test:**

**a=10**

**def m1(self):**

**del cls.a**

**print(Test.\_\_dict\_\_)**

**t=Test()**

**print(Test.\_\_dict\_\_)**

**Ex:- class Test:**

**a=10**

**t=Test()**

**del Test.a**

* **Local Variables/method variables**

**🡪The variable which is declare inside constructor or methods or without using self-keyword is called local variable.**

**🡪method level variables**

**We should not use self,cls,classname**

**Ex:-class Test:**

**Def m1(self):**

**A=100**

**Print(a)**

**Def m2(self):**

**A=999**

**B=200**

**Print(a)**

**Print(b)**

**t=Test()**

**t.m1()**

**t.m2()**

**Ex:-** **class Test:**

**def average(self,list):**

**result=sum(list)/len(list)**

**print("The Average avlue:",result)**

**t=Test()**

**t.average([10,20,30,40])**

**Global variable:-**

**The variable which are declare inside a method of a class but with global keyword.**

**From the class we can access global variable directly**

**If you want to make local variable as global then you should should use global keyword.**

**Ex:-class test:**

**Def m1(self):**

**Global x**

**X=888**

**Print(x)**

**Def m2(self):**

**Print(X)**

**t=Test()**

**t.m1()**

**t.m2()**

**Banking Application:-**

import sys

class Customer:

'''Customer class with bank related operations'''

bankname='DURGASOFT'

def \_\_init\_\_(self,name,balance=0):

self.name=name

self.balance=balance

def deposit(self,amt):

self.balance=self.balance+amt

print('After deposit the balance:',self.balance)

def withdraw(self,amt):

if amt>self.balance:

print('Insufficient Funds..cannot perform this operation')

sys.exit()

self.balance=self.balance-amt

print('After withdraw The Balance:',self.balance)

print('Welcome to',Customer.bankname)

name=input('Enter Your Name:')

c=Customer(name)

while True:

print('d-Deposit\nw-Withdraw\ne-Exit')

option=input('Choose Your Option:')

if option=='d' or option=='D':

amt=float(input('Enter the amount to depost:'))

c.deposit(amt)

elif option=='w' or option=='W':

amt=float(input('Enter the amount to withdraw:'))

c.withdraw(amt)

elif option=='e' or option=='E':

print('thanks For Banking')

sys.exit()

else:

print('choose valid option:')

**output**:- Welcome to DURGASOFT

Enter Your Name:pawan

d-Deposit

w-Withdraw

e-Exit

Choose Your Option:d

Enter the amount to depost:12000

After deposit the balance: 12000.0

d-Deposit

w-Withdraw

e-Exit

Choose Your Option:w

Enter the amount to withdraw:5000

After withdraw The Balance: 7000.0

d-Deposit

w-Withdraw

e-Exit

Choose Your Option:e

thanks For Banking

**Types of methods:-**

* **Instance methods/object related method**

**🡪The method where we are using atleast one instance variable.**

* **Class methods**

**🡪if we are using only static variables but not instance variables then this is called class methods.**

**🡪when we use @classmethod decorator then it is called class level method and cls must be required and when we use class level data**

* **Static methods**

**🡪if we are not using any instance or static variable then we should go for static method.**

**🡪static method are general utility method.**

**🡪when we use @staticmethod decorator and we can access by using class name or object name.**

Ex:-def m1(self)

Self.x

@classmethod

Def m1(cls):

cls.x

Test.x

@static method

Def add(x,y):

Print(x+y)

* **Instance methods/object related method**

**🡪The method where we are using atleast one instance variable.**

**🡪 compulsory the first argument to the instance method is always self which is pointing to current object.**

**🡪 inside of the class if we want to call the instance method then we do by using self and outside of the class by using reference variable.**

**🡪 best example of instance method is setter and getter method**

**🡪the disadvantage of instance method is instance method is very costly because if you want to call any instance variable we have to compulsory create and object of class.**

**Ex:-** **class Student:**

**def \_\_init\_\_(self,name,marks):**

**self.name=name**

**self.marks=marks**

**def display(self):**

**print('Hi',self.name)**

**print('Your Marks Are:',self.marks)**

**def grade(self):**

**if self.marks>=60:**

**print('First Grade')**

**elif self.marks>=50:**

**print('Second Grade')**

**elif self.marks>=35:**

**print('You Got Third Division')**

**else:**

**print('you Are Failed')**

**n=int(input('Enter number of Students:'))**

**for i in range(n):**

**name=input('Enter Student name:')**

**marks=int(input('Enter Student marks:'))**

**s=Student(name,marks)**

**s.display()**

**s.grade()**

**print('\*'\*20)**

**Setter and getter methods:-**

* **We are using setter method to set the data to the object and we are using getter method to get the data from the object.**
* **The alternative of constructor is setter and getter method to set and get the value.**

**🡪the biggest advantage of getter and setter is security.**

**🡪hiding data behind method it is also called encapsulation.**

**🡪in constructor we cannot validate values**

**Syntax:-def setVariableName(self,variableName):**

**Self.variableName=variableName**

**def getVariableName(self):**

**return self.variableName**

**Ex:-def \_\_init\_\_(self,name,marks):**

**self.name=name**

**self.marks=marks**

**s=Student(‘Durga’,100)**

**Ex:-** **class Student:**

**def setName(self,name):**

**self.name=name**

**def getName(self):**

**return self.name**

**def setMarks(self,marks):**

**self.marks=marks**

**def getMarks(self):**

**return self.mark**

**n=int(input('Enter number of Students:'))**

**for i in range(n):**

**name=input('Enter Student name:')**

**marks=int(input('Enter Student marks:'))**

**s=Student()**

**s.setName(name)**

**s.setMarks(marks)**

**#print(s.name) -->for getting name directly**

**print('Hi',s.getName()) #for hiding data behind the method**

**print('Your Marks Are:',s.getMarks())**

**print('\*'\*20)**

* **Class methods :-**

**🡪if we are using only static variables but not instance variables then this is called class methods.**

**🡪when we use @classmethod decorator then it is called class level method and cls must be required and when we use class level data.**

**🡪for this we donot require to create object so it is not costly.**

🡪**we can call class method by using classname directly we don’t need to create object.**

**🡪we use @classmethod to differentiate it is a class method but non static method for that it is compalsary.**

**Ex:-@classmethod**

**Def m1(cls):**

**Print(cls.collegename)**

**Print(cls.bankname)**

**Ex:-** **class Animal:**

**legs=4**

**count=0**

**def \_\_init\_\_(self):**

**Animal.count=Animal.count+1**

**@classmethod**

**def getNoOfObjects(cls):**

**print('The number of objects created:',cls.count)**

**@classmethod**

**def walk(cls,name):**

**print('{} walks with {} legs'.format(name,cls.legs))**

**Animal.walk('Dog')**

**Animal.walk('Cat')**

**a1=Animal()**

**a2=Animal()**

**a3=Animal()**

**a4=Animal()**

**a5=Animal()**

**Animal.getNoOfObjects()**

* **Static methods**

**🡪if we are not using any instance or static variable then we should go for static method.**

**🡪static method are general utility method/helper method.**

**🡪when we use @staticmethod decorator and we can access by using class name or object name.**

**Ex:-** **class DurgaMath:**

**@staticmethod**

**def add(x,y):**

**print('The Sum:',x+y)**

**@staticmethod**

**def product(x,y):**

**print('The Product:',x\*y)**

**@staticmethod**

**def average(x,y):**

**print('The Average Value:',(x+y)/2)**

**DurgaMath.add(10,20)**

**DurgaMath.product(10,20)**

**DurgaMath.average(10,20)**

**'''The Sum: 30**

**The Product: 200**

**The Average Value: 15'''**

**How to access members of one class inside another class:-**

**class Employee:**

**def \_\_init\_\_(self,eno,ename,esal):**

**self.eno=eno**

**self.ename=ename**

**self.esal=esal**

**def display(self):**

**print('Employee Number:',self.eno)**

**print('Employee Name:',self.ename)**

**print('Employee Salary:',self.esal)**

**class Test:**

**def modify(emp):**

**emp.esal=emp.esal+10000**

**emp.display()**

**e=Employee(872425,'Durga',70000)**

**Test.modify(e)**

**'''Employee Number: 872425**

**Employee Name: Durga**

**Employee Salary: 80000'''**

**Inner class:-**

**🡪The class which is declared inside another class is called Inner classes.**

**🡪 Without existing one type of object if there is no chance of existing another type of object then we should go for inner classes.**

**Ex:-engine class inside car class, without existing class object engine object have no existence.**

**Department class inside university class, head class inside human class**

**🡪 Without existing outer class object there is no chance of existing inner class object.**

**Inner class object is always associated with outer object.**

**Ex:-** class Outer:

def \_\_init\_\_(self):

print('Outer class object creation...')

def m2(self):

print('Outer class method')

class Inner:

def \_\_init\_\_(self):

print('Inner class object creation...')

def m1(self):

print('Inner class method')

o=Outer()

i=o.Inner()

i.m1()

o.m2()

#i=Outer().Inner()

#Outer().Inner().m1()

**Nested classes:-**

**Ex:-** **class Human:**

**def \_\_init\_\_(self):**

**self.name='Durga'**

**self.head=self.Head()**

**def display(self):**

**print('Name:',self.name)**

**self.head.talk()**

**self.head.brain.think()**

**class Head:**

**def \_\_init\_\_(self):**

**self.brain=self.Brain()**

**def talk(self):**

**print('Talking..')**

**class Brain:**

**def think(self):**

**print('Thinking....')**

**h=Human()**

**h.display()**

**Nested Methods:-**

**We use nested method for code reusability. We don’t need to write method again and again.**

**class Test:**

**def m1(self):**

**def sum(a,b):**

**print('first Argument:',a)**

**print('second Argument:',b)**

**print('The Sum:',a+b)**

**print('The Product:',a\*b)**

**print('\*'\*20)**

**sum(10,20)**

**sum(100,200)**

**sum(1000,2000)**

**t=Test()**

**t.m1()**

**Garbage Collections:-**

**🡪 The main purpose of Garbage Collection is to destroy useless object, my python never going to fail due to memory problem. The assistance who destroy useless object it is called garbage collector.**

**🡪Based on our requirement we can enable and disable garbage collection.**

**🡪gc module**

**1. gc.isenabled()**

**2. gc.disable()**

**3.gc.enable**

**Ex:-import gc**

**Print(gc.isenabled())🡪True**

**Gc.disable()**

**Print(gc.isenabled() 🡪False**

**Gc.enable()**

**Print(gc.isenabled() 🡪 True**

**Destructor:-**

**\_\_del\_\_(self):**

* **Destructor is used to perform cleanup activity and resource deallocation activity.**
* **Destructor is used to close db connection.**
* **Destructor is used to close network connection**
* **Garbage Collector will call destructor automatically**
* **Gc called destructor just before destroying object for closing all the activity.**
* **In java instead of destructor there is finalize method.**

**Ex:-** **import time**

**import gc**

**import sys**

**class Test:**

**def \_\_init\_\_(self):**

**print('Object Initialization...')**

**def \_\_del\_\_(self):**

**print('Fullfilling last wish and performing cleanup activities..')**

**'''gc.disable()**

**print(gc.isenabled())**

**t1=Test()**

**t1=None**

**time.sleep(10)**

**print('End of Application')'''**

**'''t1=Test()**

**t2=t1**

**t3=t2**

**t4=t3**

**del t1**

**time.sleep(10)**

**print('After deleting t1 object not destroyed')**

**del t2**

**del t3**

**time.sleep(10)**

**print('still object are not eligible for gc')**

**time.sleep(10)**

**del t4**

**time.sleep(10)**

**print('End of Application')'''**

**'''list=[Test(),Test(),Test()]**

**time.sleep(10)**

**list=None**

**time.sleep(10)**

**print('end of application')'''**

**'''t1=Test()**

**print(sys.getrefcount(t1))'''**

**module 3:-**

**polymorphism:-**

* **poly means many**
* **morphs means forms**
* **one name but multiple forms is called polymorphism.**

**Overloading:-**

**Method name is same but argument is different.**

**Operator Overloading:-(any operator)**

**In java we are not having operator overloading.**

**Operator overloading by using magic methods**

**+ is only for string object and int object**

**Whenever we are calling + operator than \_add\_\_() method will be called.**

**Whenever we are printing Book object reference then \_\_str\_\_() method will be called.**

**+=================🡺 \_\_add()\_\_(self, other)**

**+🡺\_\_add\_\_(self, other)**

**-🡺 \_\_sub\_\_(self, other)**

**\*🡺 \_\_mul\_\_(self, other)**

**/🡺\_\_div\_\_(self, other)**

**% 🡺 \_\_mod\_\_(self, other)**

**//🡺\_\_floordiv\_\_(self, other)**

**\*\*🡺\_\_pow\_\_(self, other)**

**+=🡺\_\_iadd\_\_(self, other)**

**-=🡺 \_\_isub\_\_(self, other)**

**\*=🡺 \_\_imul\_\_(self, other)**

**/=🡺\_\_idiv\_\_(self, other)**

**%= 🡺 \_\_imod\_\_(self, other)**

**//=🡺\_\_ifloordiv\_\_(self, other)**

**\*\*=🡺\_\_ipow\_\_(self, other)**

**>🡺\_\_gt\_\_(self, other)**

**>=🡺 \_\_ge\_\_(self, other)**

**<🡺 \_\_lt\_\_(self, other)**

**<=🡺\_\_le\_\_(self, other)**

**== 🡺 \_\_eq\_\_(self, other)**

**!=🡺\_\_ne\_\_(self, other)**

**Ex:-** **class Book:**

**def \_\_init\_\_(self,pages):**

**self.pages=pages**

**def \_\_str\_\_(self):**

**return ‘The number of pages:’+ str(self.pages)**

**def \_\_add\_\_(self,other):**

**return self.pages+other.pages**

**def \_\_sub\_\_(self,other):**

**return self.pages-other.pages**

**def \_\_mul\_\_(self,other):**

**return self.pages\*other.pages**

**def \_\_div\_\_(self,other):**

**return self.pages/other.pages**

**b1=Book(100)**

**b2=Book(200)**

**b3=Book(300)**

**b4=Book(700**

**print(b1+b2)#300**

**print(b1+b3)#400**

**print(b2+b3)#500**

**print(b1-b2)#-100**

**print(b1\*b2)#20000**

**print(b1/b2)#unsupported operand type(s) for /: 'Book' and 'Book'**

**#print(b1+b2+b3+b4)# unsupported operand type(s) for +: 'int' and 'Book'**

**Ex:-** **class Book:**

**def \_\_init\_\_(self,pages):**

**self.pages=pages**

**def \_\_str\_\_(self):**

**return 'The No. of pages:'+str(self.pages)**

**def \_\_add\_\_(self,other):**

**total=self.pages+other.pages**

**b=Book(total)**

**return b**

**def \_\_mul\_\_(self,other):**

**total=self.pages\*other.pages**

**b=Book(total)**

**return b**

**b1=Book(100)**

**b2=Book(200)**

**b3=Book(300)**

**b4=Book(400)**

**print(b1)**

**print(b2)**

**print(b1+b2)**

**print(b1\*b2)**

**print(b1+b2+b3+b4)**

**print((b1+b2)\*(b3+b4))**

**'''The No. of pages:100**

**The No. of pages:200**

**The No. of pages:300**

**The No. of pages:2000**

**The No. of pages:1000**

**The No. of pages:210000'''**

**Method Overloading:-**

* **In java method name is same but different argument is called method overloading.**
* **In python won’t support method/constructor overloading because in python type casting or typing is not possible.**
* **If there is multiple method with same argument then here it will only give output for last method.**

**Ex:-** **class Test:**

**def m1(self):**

**print('no-arg method')**

**def m1(self,x):**

**print('one-arg method')**

**t=Test()**

**t.m1(10)#one-arg method**

**Constructor Overloading:-**

**Multiple constructor with same name but different argument.**

**In python won’t support constructor overloading.**

**Ex:-** **class Test:**

**def \_\_init\_\_(self):**

**print('no-arg argument')**

**def \_\_init\_\_(self,a):**

**print('one-arg argument')**

**def \_\_init\_\_(self,a,b):**

**print('two-arg argument')**

**t=Test(10,20)#two-arg argument**

**Overriding:-**

**Method overriding:-**

**Same method name with same argument but different class using super keyword and inheritance.**

**Ex:-** **class P:**

**def property(self):**

**print('cash+land+gold+power')**

**def marry(self):**

**print('subbalaxmi')**

**class C(P):**

**def marry(self):**

**super().marry()**

**print('Katrina..')**

**c=C()**

**c.property()**

**c.marry()**

**'''cash+land+gold+power**

**subbalaxmi**

**Katrina..'''**

**Constructor overriding :-**

**Duck Type Phylosophy of python:-**

**Is-A vs Has-A Relationship :-**

**Composition vs Aggregration:-**

**Composition(strong association between the object)**

**Ex:-university vs Department**

**Aggregration(weak association between the object)**

**Ex:-Department vs Proffesor(Department Has-A Proffesor)**

**Ex:-** **class Student:**

**collegeName='DURGASOFT'**

**def \_\_init\_\_(self,name):**

**self.name=name**

**print(Student.collegeName)**

**print(s.name)**

**Inheritance:-**

**Inheritance means parent class has bydefault child class information.**

* **Code reusability**
* **Existing Functionality we can extend**

**Syntax:-**

**Class P:**

**10 methods**

**Class C(p1,p2,p3):**

**5 methods**

**Ex:- class P:**

**def property(self):**

**print('cash+land+gold+power')**

**def marry(self):**

**print('subbalaxmi')**

**class C(P): pass**

**c=C()**

**c.property()**

**c.marry()**

**Types of inheritance:-**

**1.Single Inheritance**

**2.Multi Level Inheritance**

**3.Hierarchical Inheritance**

**4.Multiple Inheritance**

**5.Hybrid Inheritance**

**6.Cyclic Inheritance**

1. **Single Inheritance:-**

**🡪The process of inheriting property from single parent to single child.**

**Ex:-class P:**

**def m1(self):**

**print('parent method')**

**class C(P):**

**def m2(self):**

**print('child method')**

**c=C()**

**c.m1()**

**c.m2()**

1. **Multi Level Inheritance:**

**Single parent and single child but multiple level.**

**Ex:-** **class P:**

**def m1(self):**

**print('parent method')**

**class C(P):**

**def m2(self):**

**print('child method')**

**class CC(C):**

**def m3(self):**

**print('Sub child methd')**

**c=CC()**

**c.m1()**

**c.m2()**

**c.m3()**

**3.heirarchical inheritance:-**

**One parent but multiple child at same level.**

**Ex:-** **class P:**

**def m1(self):**

**print('parent method')**

**class C1(P):**

**def m2(self):**

**print('child1 method')**

**class C2(P):**

**def m3(self):**

**print(' child2 methd')**

**c=C1()**

**c.m1()**

**c.m2()**

**print()**

**c1=C2()**

**c1.m1()**

**c1.m3()**

**Multiple Inheritance:-**

**Single child multiple parent**

**In multiple inheritance method resolution is always based on parent order. First priority to p1 then p2**

**Java does not support multiple inheritance because of ambiguity problem or diamond access problem.**

**Ex:- class P1:**

**def m1(self):**

**print('parent1 method')**

**class P2:**

**def m1(self):**

**print('parent2 method')**

**class C(P1,P2):pass**

**c=C()**

**c.m1()**

**'''parent1 method'''**

**5.Hybrid Inheritance(combination of two or more inheritance)**

**Ex:-class A(A):**

**Class A(B):**

**Class B(A):**

**6.Cyclic Inheritance:-**

**It is not required**

**Method Resolution Order:-**

**If the child doesn.t contain then try in the parent class the automatically object class.**

**Ex:-** **class P:**

**def m1(self):**

**print('parent method')**

**class C(P):pass #if child contain same method then child class execute**

**p=P()**

**p.m1()**

**print(P.mro())**

**'''**

**parent method**

**[<class '\_\_main\_\_.P'>, <class 'object'>]**

**'''**

**C3 Algoritham:-**

**mro(p)=p + Merge(mro(X),mro(Y),mro(C),XYC**

**mro(P)=p + Merge(XABO,YBCO,CO,XYC)**

**p + X+ Merge(ABO,YBCO,CO,YC)**

**p + X+ A+ Merge(BO,YBCO,CO,YC)**

**p + X+ A+ Y+ Merge(BO,BCO,CO,C)**

**p + X+ A+ Y+ B+ Merge(O,CO,CO,C)**

**p + X+ A+ Y+ B+ C+ Merge(O,O,O)**

**p + X+ A+ Y+ B+ C+ O**

**Ex:-** **class D:pass**

**class E:pass**

**class F:pass**

**class B(D,E):pass**

**class C(D,F):pass**

**class A(B,C):pass**

**a=A()**

**a.m1()#A+B+C+D+E+F+O**

**''' File "C:\Users\bikram chandra\AppData\Local\Programs\Python\Python38-32\C3ALGORITHAM.PY", line 8, in <module>**

**a.m1()#A+B+C+D+E+F+O**

**AttributeError: 'A' object has no attribute 'm1''''**

**super():-**

* **super() is helpful only for inheritance concept only.**
* **From child class To call parent class members(constructor,methods) we should go for super().**
* **Advantage is code reusability.**

**Ex:-** **class Person:**

**def \_\_init\_\_(self,name,age):**

**self.name=name**

**self.age=age**

**def display(self):**

**print('name:',self.name)**

**print('age:',self.age)**

**class Student(Person):**

**def \_\_init\_\_(self,name,age,rollno,marks):**

**super().\_\_init\_\_(name,age)**

**self.rollno=rollno**

**self.marks=marks**

**def display(self):**

**super().display()**

**print('Roll Number:',self.rollno)**

**print('Marks:',self.marks)**

**class Teacher(Person):**

**def \_\_init\_\_(self,name,age,salary,subject):**

**super().\_\_init\_\_(name,age)**

**self.salary=salary**

**self.subject=subject**

**def display(self):**

**super().display()**

**print('Salary:',self.salary)**

**print('Subject:',self.subject)**

**s=Student('Ravi',23,101,90)**

**p=Teacher('Durga',62,10000,'Python')**

**s.display()**

**p.display()**

**'''name: Ravi**

**age: 23**

**Roll Number: 101**

**Marks: 90**

**name: Durga**

**age: 62**

**Salary: 10000**

**Subject: Python'''**

**How to call a particular parent class method by using super()**

**2 ways:-**

**Parentclassname.methodname(self)**

**B.m1(self)**

**Super(D,self).m1()**

**1.From child class by using super() we cannot call parent class instance variables we should use self only.**

**2.From child class by using super() method we can call parent class static variables.**

**Ex:-** **class A:**

**def m1(self):**

**print('A class Method')**

**class B(A):**

**def m1(self):**

**print('B class method')**

**class C(B):**

**def m1(self):**

**print('C class method')**

**class D(C):**

**def m1(self):**

**print('D class method')**

**class E(D):**

**def m1(self):**

**super(D,self).m1()**

**e=E()**

**e.m1()**

**#C class method**

**MultiThreading:-**

**Multitasking:-**

1. **Process Based Multi Tasking**

**🡪 Executing several task simultaneously where each task is a separate independent process.**

**Ex:- typing program in editor, listening songs from audio player from same system, download new song from browser from the internet.**

**🡪This is bast suitable with os level.**

1. **Thread Based Multitasking**

**🡪 Executing several task simultaneously where each task is an independent part.**

**Thread:-**

* **An independent part of program**
* **A flow of execution is considered as a thread**
* **It is a python object.**
* **For every thread independent job is available**
* **Ex. Are multimidea , graphics, games,servers**
* **Every python program contain one thread i.e. MainThread.**

**threading module**

**Ex:-import threading or from threading import \***

**Print(‘Current Executing Thread:’,threading.current\_thread().getName()) 🡪Mainthread**

**3 ways to define thread:-**

**1>. Creating a Thread without using any class**

**2>.Creating a thread by extending Thread class**

**3>.Creating a thread without extending Thread class**

**1>. Creating a Thread without using any class:-**

**🡪this is also called functional programming way.**

**Ex:-** **from threading import \***

**def display():**

**for i in range(10):**

**print('Child thread')**

**t=Thread(target=display) #creation a thread object to execute display()**

**t.start() #starting the thread**

**for i in range(10):**

**print('Main Thread') #main thread is responsible to execute remaining execution.**

**2>. Creating a thread by extending Thread class:-**

**Ex:-** **from threading import \***

**class MyThread(Thread):**

**def run(self):**

**for i in range(10):**

**print('child Thread')**

**t=MyThread()**

**t.start()**

**for i in range(10):**

**print('Main thread')**

**3>.Creating a thread without extending Thread class**

**Ex:- from threading import \***

**Class Test:**

**def m1(self):**

**for i in range(10):**

**print('child Thread')**

**obj=Test()**

**t=Thread(target=obj.m1)**

**t.start()**

**for i in range(10):**

**print('Main thread')**

**Ex:-** **import time**

**from threading import \***

**def doubles(numbers):**

**for n in numbers:**

**time.sleep(1)**

**print('Double values:',2\*n)**

**def squares(numbers):**

**for n in numbers:**

**time.sleep(1)**

**print('Square values:',n\*n)**

**numbers=[1,2,3,4,5,6]**

**begintime=time.time()**

**t1=Thread(target=doubles,args=(numbers,))**

**t2=Thread(target=squares,args=(numbers,))**

**t1.start()**

**t2.start()**

**t1.join()//if thread wants to wait**

**t2.join()**

**endtime=time.time()**

**print('The total time taken:',endtime-begintime)**

**'''Square values: 1**

**Double values: 2**

**Square values: 4**

**Double values: 4**

**Square values: 9**

**Double values: 6**

**Square values: 16**

**Double values: 8**

**Square values: 25**

**Double values: 10**

**Square values: 36**

**Double values: 12**

**The total time taken: 6.009063720703125'''**

**Setting and getting name of a thread:-**

**To get name:-**

**1>t.name**

**2>t.getName()**

**To set name:-**

**1>t.name=’durga’**

**2>t.setName(‘Durga’)**

**Ex:-** **from threading import \***

**print(current\_thread().getName())**

**#current\_thread().setName('Sunny Leone')**

**#print(current\_thread().getName())**

**current\_thread().name='Durga'**

**print(current\_thread().name)**

**Thread Identification Number:**

**By using implicit variable ‘ident’**

**Ex:-** **from threading import \***

**def test():**

**print('Child Thread')**

**t=Thread(target=test)**

**t.start()**

**print('Main Thread Identification Number:',current\_thread().ident)**

**print('Child Thread Identification Number:',t.ident)**

**'''Child Thread**

**Main Thread Identification Number: 21720**

**Child Thread Identification Number: 15856'''**

**Active\_count():**

**from threading import \***

**import time**

**def display():**

**print(current\_thread().name,'...started')**

**time.sleep(3)**

**print(current\_thread().name,'...ended')**

**print('The number of active Threads:',active\_count())**

**t1=Thread(target=display,name='ChildThread-1')**

**t2=Thread(target=display,name='ChildThread-2')**

**t3=Thread(target=display,name='ChildThread-3')**

**t1.start()**

**t2.start()**

**t3.start()**

**print('The number of active threads:',active\_count())**

**time.sleep(10)**

**print('The number of active threads:',active\_count())**

**enumerate():**

**calculate or list out all current running thread**

**Ex:-** **from threading import \***

**import time**

**def display():**

**print(current\_thread().name,'...started')**

**time.sleep(3)**

**print(current\_thread().name,'...ended')**

**print('The number of active Threads:',active\_count())**

**t1=Thread(target=display,name='ChildThread-1')**

**t2=Thread(target=display,name='ChildThread-2')**

**t3=Thread(target=display,name='ChildThread-3')**

**t1.start()**

**t2.start()**

**t3.start()**

**l=enumerate()**

**for t in l:**

**print('Thread name:',t.name)**

**print('Thread Identification Number:',t.ident)**

**print()**

**time.sleep(10)**

**l=enumerate()**

**for t in l:**

**print('Thread name:',t.name)**

**print('Thread Identification Number:',t.ident)**

**print()**

**'''The number of active Threads: 1**

**ChildThread-1 ...started**

**ChildThread-2 ...started**

**ChildThread-3 ...started**

**Thread name: MainThread**

**Thread Identification Number: 1592**

**Thread name: ChildThread-1**

**Thread Identification Number: 6092**

**Thread name: ChildThread-2**

**Thread Identification Number: 18712**

**Thread name: ChildThread-3**

**Thread Identification Number: 15220**

**ChildThread-1 ...ended**

**ChildThread-2 ...ended**

**ChildThread-3 ...ended**

**Thread name: MainThread**

**Thread Identification Number: 1592'''**

**isAlive()**

**is this thread dead or in alive state.**

**from threading import \***

**import time**

**def display():**

**print(current\_thread().name,'...started')**

**time.sleep(3)**

**print(current\_thread().name,'...ended')**

**t1=Thread(target=display,name='ChildThread-1')**

**t2=Thread(target=display,name='ChildThread-2')**

**t1.start()**

**t2.start()**

**print(t1.name,'is Alive :',t1.isAlive())**

**print(t2.name,'is Alive :',t2.isAlive())**

**time.sleep(10)**

**print(t1.name,'is Alive',t1.isAlive())**

**print(t2.name,'is Alive',t2.isAlive())**

**join() method:-**

**if a thread wants to wait until completing some other thread then we should go for join.**

**from threading import \***

**import time**

**def display():**

**for i in range(10):**

**print('Seetha Thread')**

**time.sleep(3)**

**t=Thread(target=display)**

**t.start()**

**t.join(10)**

**for i in range(10):**

**print('Rana Thread')**

**thread synchronization**

**bounded semaphore**

**inter threaded communication**

**types of queues**

**Daemon Threads:-**

**🡪Background executing threads are called daemons threads.**

**🡪The main purpose of daemon thread to provide support for non-daemon threads(main thread).**

**🡪whenever last non-daemon thread terminates automatically all daemon threads will be terminated we are not required to terminate explicitely.**

**Ex:-garbage collector**

**How to change daemon thread nature:-**

**🡪 before starting thread you can change the nature of thread , once thread started you can’t change its daemon nature of active thread.**

**RuntimeError: cannot set daemon thread status of active thread**

**t.daemon**

**t.setDaemon(True)#daemon thread**

**t.setDaemon(False)#non daemon thread**

**Ex:-** **from threading import \***

**mt.setDaemon(true)**

**mt=current\_thread()**

**print(mt.isDaemon())**

**print(mt.daemon)**

**Default nature:-**

**Only By default Main thread is always non-daemon.**

**But for all other threads non-daemon nature, inherited from the parent to child.**

**from threading import \***

**def job():**

**print('Executed by child thread')**

**t=Thread(target=job)**

**print(t.isDaemon())#False**

**t.setDaemon(True)**

**print(t.isDaemon())#True**

**Ex:-** **from threading import \***

**import time**

**def job1():**

**print('Executed by t1')**

**t2=Thread(target=job2)**

**print('t2 is Daemon:',t2.isDaemon())**

**t2.start()**

**def job2():**

**print('Executed by t2')**

**t1=Thread(target=job1)**

**t1.setDaemon(True)**

**print('t1 is daemon:',t1.isDaemon())#False**

**t1.start()**

**time.sleep(10)**

**Decorators:-**

**It is always going to take some input using input function and gives output by using output function.**

**Generators:-**

* **Generator won’t take any input it is responsible to generate a sequence of values.**
* **If I want to store values one by one then we use generator it will improve performance and memory utilization**
* **When compared with normal iterators generators are easy to use**
* **Generators best suitable for web scraping.**
* **We can use list instead generator but there is memory wastage unnecessarily.**

**Yield keyword.**

**Ex:-** **def mygen():**

**yield 'A'**

**yield 'B'**

**yield 'C'**

**g=mygen()**

**print(type(g))#<class 'generator'>**

**print(next(g))#A**

**print(next(g))#B**

**print(next(g))#C**

**print(next(g))#StopIteration**

**to implement countdown:-**

**import time**

**def countdown(num):**

**print('Count Down Starting..')**

**while num>0:**

**yield num**

**num=num-1**

**values=countdown(5)**

**for x in values:**

**print(x)**

**time.sleep(3)**

**print('stop')#5 4 3 2 1 stop**

**to generate first n numbers:-**

**def firstn(num):**

**n=1**

**while n <= num:**

**yield n**

**n=n+1**

**for x in firstn(10):**

**print(x)#1 2 3 4 5 6 7 8 9 10**

**to generate Fibonacci numbers:-**

**def fib():**

**a,b=0,1**

**while True:**

**yield b**

**a,b=b,a+b**

**for n in fib():**

**if n>100:**

**break**

**print(n)#1 1 2 3 4 8 13 21 34 55 89**

**count time taken by generator vs list:-**

**'''def mygen():**

**yield 'A'**

**yield 'B'**

**yield 'C'**

**g=mygen()**

**print(type(g))#<class 'generator'>**

**print(next(g))#A**

**print(next(g))#B**

**print(next(g))#C**

**print(next(g))#StopIteration'''**

**'''import time**

**def countdown(num):**

**print('Count Down Starting..')**

**while num>0:**

**yield num**

**num=num-1**

**values=countdown(5)**

**for x in values:**

**print(x)**

**time.sleep(3)**

**print('stop')#5 4 3 2 1 stop'''**

**#to genarate first n numbers**

**'''def firstn(num):**

**n=1**

**while n <= num:**

**yield n**

**n=n+1**

**for x in firstn(10):**

**print(x)#1 2 3 4 5 6 7 8 9 10'''**

**#to generate fibonacci numbers**

**'''def fib():**

**a,b=0,1**

**while True:**

**yield b**

**a,b=b,a+b**

**for n in fib():**

**if n>100:**

**break**

**print(n)#1 1 2 3 5 8 13 21 34 55 89'''**

**#people generator:-**

**import random**

**import time**

**names=['sunny','bunny','chinni','vinny']**

**subjects=['python','java','blckchain']**

**def people\_list(num):**

**results=[]**

**for i in range(num):**

**person={**

**'id':i,**

**'name':random.choice(names),**

**'subject':random.choice(subjects)**

**}**

**results.append(person)**

**return results**

**def people\_generators(num):**

**for i in range(num):**

**person={**

**'id':i,**

**'name':random.choice(names),**

**'subject':random.choice(subjects)**

**}**

**yield person**

**t1=time.process\_time() #or time.perf\_counter()#because module 'time' has no attribute clock**

**people=people\_list(1000000)**

**t2=time.process\_time()**

**print('Time taken by list:',t2-t1)**

**t1=time.process\_time()**

**people=people\_generators(1000000)**

**t2=time.process\_time()**

**print('Time taken by generator:',t2-t1)**

**'''Time taken by list: 2.296875**

**Time taken by generator: 0.109375'''**

**Regular Expression:-**

**If we wants to represent a group of string according to particular pattern then we should go for REs.**

**Regular expression language independent concept.**

**Main important application areas of regular expression:-**

**1. validations (java script)**

**2. pattern matching applications ctrl f, grep**

**3.To devlop Translator like comiplers, interpresters,assemblers**

**lexical Analysis(tokenization🡺RE),**

**syntax analysis(parsing),**

**symantic analysis,**

**ICG,CO,TCG**

**4.to devlop digital circuits**

**FA with output == > Moore and Melay**

**Binary Incrementor**

**Binary order**

**5.communication protocols TCP/IP etc**

**re module**

**1.compile() function:**

**Ex:-** **import re**

**pattern=re.compile('python')**

**print(type(pattern))#<class 're.Pattern'>**

**2.finditer():**

**Ex:-matcher=pattern.finditer(‘Learning Python is very easy..’)**

**3.start():start index of the match**

**4.end():end+1 index of the match**

**5.group():returns matched item**

**Ex:-** **import re**

**count=0**

**pattern=re.compile('ab')**

**matcher=pattern.finditer('abaababa')**

**for m in matcher:**

**count+=1 #count=count+1**

**#print('match is available at start index:',match.start())**

**print('start:{},end:{},group:{}'.format(m.start(),m.end(),m.group())**

**print('The number of occurance:',count)**

**'''match is available at start index: 0**

**match is available at start index: 3**

**match is available at start index: 5 The number of occurance: 3'''**

**Character classes:**

**[abc] == > either a or b or c**

**[^abc] == >Except a and b and c**

**[a-z] == > Any lower case alphabet symbols**

**[a-zA-Z] == >Any alphabet symbol**

**[0-9] == >Any numeric character**

**[a-zA-Z0-9] == >any alphanumeric character**

**[^a-zA-Z0-9] == >except alphanumeric character**

**Ex:-** **import re**

**matcher=re.finditer('[a-zA-Z]','a7b@k9z')**

**for m in matcher:**

**print(m.start(),'....',m.group())**

**Predefind Character classes:**

**\s 🡪space character**

**\S 🡪Except space character**

**\d 🡪 any digit**

**\D 🡪 except digits**

**\w 🡪Any word character(alpha numeric character)**

**\W 🡪Any character except word(special characters)**

**. 🡪Every character**

**import re**

**matcher=re.finditer('\S','a7b @k9z')**

**for m in matcher:**

**print(m.start(),'....',m.group())**

**Quantifiers:**

**The numbers of occurances**

**a🡪exactly one ‘a’**

**a+ 🡪atleast one ‘a’**

**a\* 🡪any number of a’s including zero number also**

**a? 🡪 atmost one a**

**either one a or zero number of a’s**

**a{n} 🡪Exactely n number of a’s**

**a{m,n} 🡪 minimum m number of a’s and maximum n number of a’s**

**a{2,4} … a+ … a{2}a\* … a{9}a\***

**[^a] == > except a**

**^a 🡪it will check whether the given target string starts with a or not**

**A$ 🡪it will check whether the target string rnds with a or not.**

**Ex:-** **import re**

**matcher=re.finditer('a\*','aabaabbabbaab')**

**for m in matcher:**

**print(m.start(),'....',m.group())**

**Important functions of re module:-**

**1.match()**

**2.fullmatch()**

**3.search()**

**4.findall()**

**5.finditer()**

**6.sub()**

**7.subn()**

**8.split()**

**9.compile()**

**Ex:-** **import re**

**s=input('Enter pattern to check:')**

**m=re.search(s,'abaabaaab')**

**if m!=None:**

**print('Match is available')**

**print('First occurance with start index:{} and end index:{}'.format(m.start(),m.end()))**

**else:**

**print('full string not matched')**

**Ex:-** **import re**

**s=input('Enter mobile number to validate:')**

**m=re.fullmatch('[6-9]\d{9}',s)**

**if m!=None:**

**print(s,'is valid mobile number')**

**else:**

**print(s,'is not vaild mobile number')**

**10 digit or 11 digit or 12 digit or 13 digit also:**

**10:6 to 9, 9 digits**

**11:the first digit should be 0**

**12:the first 2 digit should be 91**

**13:the first 3 digits should be +91**

**Python Database Programming(PDBC):**

**For every application, It is very common requirement to save data for the future purpose**

**1.Temporary storage areas**

**🡪PVM Memories areas**

**2.Permanent storage areas**

**🡪1.File Systems**

**2.Databases etc..**

**File Systems:-**

**Best suitable to store very less amount of info**

**Limitation of file systems:-**

**1.Huge data**

**2.no QL support**

**3.Security**

**4.prevent duplicate data**

**5.data inconsistency problems etc..**

**Databases:**

**1.we can store huge amount of info**

**2.QL Support**

**3.Security is more**

**4.tables**

**5.duplications**

**Limitations of databases:-**

**1.cannot hold very huge amount of info like tera bytes of data**

**2.Structured data but not for unstructured data and semi structured data.**

**PDBC:-**

**🡪Python database connectivity**

**🡪python provide inbuilt support for oracle, mysql etc.**

**🡪if I want to communicate with oracle then we use cx\_oracle and for Microsoft database pymssql and for mysql database we use pymysql module.**

**Standard steps to communicate with database:**

1. **import that databse specific module.**

**Import cx\_oracle**

**Import pymsql**

1. **Establish connection between python program and db.**

**con=cx\_Oracle.connect(database information);**

**ex:- con=cx\_Oracle.connect(‘scott/tiger@localhost’)**

1. **Cursor object(if you want to execute sql queries or result)**

**cursor=con.cursor()**

1. **Execute our sql query**

**Cursor having multiple method:-**

**1.cursor.execute(sqlquery) 🡪a single query**

**2.cursor.executescript(sqlqueries) 🡪To execute a string of sql queries separated by semi colon**

**3.cursor.executemany()🡪 To execute a perimetrized query**

1. **Fetch the results**

**Cursor.fetchone() 🡪to fetch only one row**

**Cursor.fetchall() 🡪 to fetch all rows**

**Cursor.fetchmany() 🡪to fetch n rows**

1. **commit()**

**rollback()**

1. **con.close or cursor.close()**

**working with oracle database:**

**cx\_oracle 🡪driver or translator to communication python program to oracle database.**

**Install cx\_Oracle:**

**pip install cx\_Oracle**

**1.wap to connect with oracle databse and print its version**

**import cx\_Oracle**

**connect=cx\_Oracle.connect('scott/tiger@localhost')**

**if con!=None:**

**print('Connection estaablished successfully')**

**print('Version:',con.version)**

**else:**

**print('Connection not established')**

**con.close()**

**2.wap to create employees table in the database**

**Create table employees(eno number,ename varchar2(10),esal number(10,2), eaadr varchar2(10))**

**import cx\_Oracle**

**try:**

**query='create table employees(eno number,ename varchar2(10),esal number(10,2), eaadr varchar2(10))'**

**conn=cx\_Oracle.connect('scott/tiger@localhost')**

**cursor=conn.cursor()**

**cursor.execute(query)**

**print('table created successfully');**

**except cx\_Oracle.DatabaseError as e:**

**if conn:**

**conn.rollback()**

**print('there is a problem:',e)**

**finally:**

**if cursor:**

**cursor.close()**

**if con:**

**con.close()**

**3.To drop table**

**Drop table employees**

**4.insert one row into the database**

**Insert into employess values(100,’durga’,1000,’hyd’)**

**5.insert multiple records into database**

**Executemany() method**

**Insert into employess values(:eno,:ename,:esal,:eaddr)**

**Records=[(200,’sunny’,2000,’Mumbai’),**

**(300,’Bunny’,3000,’Hyd’),**

**(400,’chinni’,4000,’Hyd’)]**

**Cursor.executemany(query,records)**

**6.to insert multiple rows with dynamic input from the keyboard**

**Query=’’insert into employees values(%d,’%s’,%f,’%s’)’’**

**Cursor.execute(query %(eno,ename,esal,eaadr))**

**7.to update employees salary with increament.**

**8.to delete employess with cutoff**

**9.select operation**

**By fetchone()**

**Fetchall()**

**Fetchmany()**

**Django:-**

* Django is a web framework which can be used to build a web application.
* It is FREE and open source web framework.
* Ex:-gmail.com, twitter.com, facebook.com
* It is written in python language.
* Instead of using MVC django follows design pattern MVT(Model view template)
* View means business logic.
* DSF(Django software foundation) is responsible to maintain django , it is non profit foundation.
* Google, youtube, spotify, bitbukcet, dropbox, yahoo, maps, Mozilla, instagram, NASA many more company website uses django.
* Django introduced in 2003 as an iternal project by lowrence journal-world news paper.
* Original authors: Adrian Holovaty , Simon Willison
* After testing this framework with high traffics.. 2005 july 21st, as open source framework.
* There are many web framework for python like Django, Flask, Pyramid, Web2Py, Bottle, Tornado, Dash, Cherrypy.
* Djangoproject.com official website for web framework.
* Django named based on django rainherdt.
* Django was invented to meet fast-moving newsroom deadlines, while satisfying the tough requirements of experienced web developer.

**top 5 Advantages of Django Framework features:-**

1. **Fast** (95% of work can be done by Django Framework.. only 5% of work has to do by developer)

2. **Fully** **Loaded** (authentication, content administration,site maps, Rss feeds, and many more tasks)

3. **Security** (sql injection Attack, Cross Site Scripting, Cross-Site Request Forgery)

4. **Scalability** (100 or 1000 request no changes in performance)

5. **Versatile** ( all rounder ) 🡪 for small or big application or scitific application . companies,govt, you can use django.

**How to configure atom with python?**

1.file🡪setting🡪install🡪search for terminal 🡪platform ide terminal 2.8.3🡪install🡪click on setting 🡪shell Override 🡪C:\\windows\System32\cmd.exe 🡪normal cmd

Then go to + icon 🡪windows powershell is going to open

2.file🡪setting🡪install🡪search for terminal 🡪python🡪autocompletepython 1.16.0🡪install

3.file🡪setting🡪install🡪search for terminal 🡪django🡪atom-django 0.3.2

In python cmd🡪pip install Django==1.11 or anything

**Django Project vs Django Application:-**

**Ex**:- Bank Project 🡪customer registration, loan, insurance, marketing application

🡪Project means a group of application + configuration information

🡪django application can be use in multiple project

**Ex**:-polling application 🡪pluggable django applications

**How to create Django project :-**

**🡪Create one folder djangoprojectsot 🡪go to that location using cmd**

**Django-admin startproject firstProject**

**🡪firtsproject🡪first project folder, manage.py🡪\_init\_,asgi,settings,urls,wsgi**

**Manage.py 🡪it is the command line utility ,it is used to creating application and create migration and run test cases.**

**Runserver is used to run the manage.py file.**

**How end user send information /url pattern**

[**http://127.0.0.1:8000/firstApp/sports**](http://127.0.0.1:8000/firstApp/sports)

[**http://127.0.0.1:8000/firstApp/cinemas**](http://127.0.0.1:8000/firstApp/cinemas)

**🡪Web server gateway interface (wsgi.py):- To move project into live production we can use.**

**🡪To check your project and files:- tree /f**

**🡪Web server provides environment to run our web application.**

**🡪Django internally contain inbuilt server so you don’t need to install any server.**

🡪Django will also provide inbuilt database also.

🡪How to start Django server?

If you run manage.py automatically django server will be start 🡪**py manage.py runserver**

**Django default port number is 8000** [**http://127.0.0.1:8000**](http://127.0.0.1:8000)

**It will work on port** [**http://127.0.0.1:7777/**](http://127.0.0.1:7777/)

**How to create application?**

**Py manage.py startapp firstApp**

**firstApp🡪migration folder(\_init\_.py)🡪\_init\_.py🡪models.py🡪admin.py🡪views.py🡪apps.py🡪test.py**

**Add this application to the project:-**

**By using settings.py file**

**INSTALLED\_APPS**

**Views.py 🡪define view function**

**urls.py🡪**

**How end user send request to the file**

**https://127.0.0.1:8000/hydjobs**

**End user🡪server🡪urls.py🡪views.py**

How to ru your application?

**1**.run server 🡪 py manage.py runserver

**2**.add application to the project 🡪firstProject🡪settings.py🡪INSTALLED\_APPS=[ ‘jobsapp’,

**3**.go to jobsapp🡪views.py

**from django.shortcuts import render**

**from django.http import HttpResponse**

**# Create your views here.**

**def hydjobsinf(request):**

**s='<h1>Hydrabad job information</h1>'**

**return HttpResponse(s)**

**4.go to jobsapp🡪urls.py**

**from django.contrib import admin**

**from django.conf.urls import url**

**from jobsapp import views**

**urlpatterns = [**

**url(r'^admin/', admin.site.urls),**

**url(r'^hydjobs',views.hydjobsinf),**

**]**