Part-2

==========
Anonymous Function in Python
(OR)
Lambda Functions in Python
==========

=>Anonymous Function are those which does not contain name explicitly.
=>The purpose of Anonymous Function in python is that "To perform Instant Operations". Instant Operations are those which used / performed at that point of time only but not interested in longer point time.
=>Anonymous Function in python contains single statement only but not containing multiple statements.
=>Anonymous Function automatically / implicitly returns the value (No need to use return statement to return the value)
=>To Anonymous Functions in python, we a keyword "lambda".

=>Syntax:- varname=lambda params-list : single statement
Explanation:

=>'Varname' is one of the valid variable name and itself treated as an object of type <class, 'function'> . so that it can be treated as Function name indirectly.
=>"lambda" is a keyword used for defining Anonymous Functions.
=>"params-list is nothing but list of formal params.
=>"Single statement" repersents an executable statement provides solution to the
instrant requirement / Operation.

Explanation: convert cel temp into F. heat temp

```
def   tempconvert( c):   # tempconvert---normal function
          f=1.8*c+32
         return f

#main program
tf=tempconvert(32)
print("Temp in F.Heat=",tf)
```

(OR)

```
tempconvert=lambda c :   1.8*c+32

#main program
tf=tempconvert(34)
print("Temp in F.Heat=",tf)
```

Python Ternary Operator:

```
Varname = expression1   if cond1  else   expression2
```

Explanation :
---

if Cond1 is True then PVM executes Expression1 and whose result placed into varname.
if Cond1 is False then PVM executes Expression2 and whose result placed into varname.

```
#AnynomousFunex1.py

def   tempconvertion(c):   # Normal Function Definition
       ft=1.8*c+32
       return ft

tempconvert=lambda c :   1.8*c+32  # Anonymous Function Definition

#main program
print("Type of tempconvert=",type(tempconvert))
print("Type of tempconvertion=",type(tempconvertion))
print("-----------------------------------------")
c=float(input("Enter the temp in Celcious:"))
tf=tempconvert(c)
print("Temp in F.Heat by anonymous function=",tf)
print("-----------------------------------------------------:")
tf1=tempconvertion(c)
print("Temp in F.Heat by Normal function=",tf1)
```

```
#anonymousfunex2.py
mulop=lambda a,b : a*b   # Anonymous Function Def.

#main program
a=float(input("Enter First Value:"))
b=float(input("Enter Second Value:"))
result=mulop(a,b)
print("Mul({},{})={}".format(a,b,result))
```

```
#anonymousfunex3.py

findbig=lambda a,b  :  a  if a>b else b    # Anonymous Function Def.

# main program
a=float(input("Enter First Value:"))
b=float(input("Enter Second Value:"))
result=findbig(a,b)
print("big({},{})={}".format(a,b,result))
```

```
#anonymousfunex4.py

findbig=lambda a,b  :  "Equals Value" if (a==b)  else   a if (a>b) else b

# main program
a=float(input("Enter First Value:"))
b=float(input("Enter Second Value:"))
result=findbig(a,b)
print("big({},{})={}".format(a,b,result))
```

```
#anonymousfunex5.py
def   findbig(a,b,c):
```

```
        result= "ALL VALUES ARE EQUAL" if (a==b) and (b==c) else a
if(a>b) and (a>c) else b if (b>c) else c
        return result

# Anonymous function for finding biggest of three numbers
big=lambda a,b,c:  "ALL VALUES ARE EQUAL" if (a==b) and (b==c) else a
if(a>b) and (a>c) else b if (b>c) else c
# Anonymous function for finding smallest of three numbers
small = lambda p,q,r : "ALL VALUES ARE EQUAL" if(p==q) and (q==r) else p
if (p<q) and (p<r) else q if (q<r) else  r

# main program
a=float(input("Enter First Value:"))
b=float(input("Enter Second Value:"))
c=float(input("Enter Third Value:"))
bigger=findbig(a,b,c)
print("big({},{},{})={}".format(a,b,c,bigger))
smaller=small(a,b,c)
print("small({},{},{})={}".format(a,b,c,smaller))
```

```
#anonymousfunex6.py
maxvalue=lambda listobj : max(listobj)
minvalue=lambda listobj : min(listobj)

#main program
lst=[10,20,30,50,23,45,-34,23,-4,34,100,-99]
maxv=maxvalue(lst)
minv=minvalue(lst)
print("Max Element({})={}".format(lst,maxv))
print("Min Element({})={}".format(lst,minv))
```

Special Functions in Python filter() programs

```
                ========================================
                            Special Functions in Python
                ========================================
=>In Python Programming, we have 3 special Functions. They are
                        1) filter()
                        2) map()
                        3) reduce()
                ========================================
                            1) filter()
                ========================================
=>This function is used for " filtering out some elements based on some
conditon from any Collection / Iterable objects by applying to the function."

Syntax:-        varname=filter(Function_name, Iterable_object )
---------------------
Explanation:
---------------------
=>'varname' is an object of  <class, 'filter'> and we can convert into any
     iterable_object type.
=>"Function_name" is either normal function and anonymous function and it
must
      return either True or False.
=>"Iterable_object "  can any Sequence type or collection types.
```

```
=>Execution Process of filter() is that " filter() send every element of
iterable_object  to the specified Function. if the function returns True then
Filter() will consider / filter that element . if The Function returns False
then filter() will neglect that element ( not filtered). This Process will be
continued until all elements of Iterable object will complete."

================================================================
```

Special Functions in Python filter() programs

```
#FilterEx1.py
def  positive(n):
       if n>0 :
              return True
       else :
              return False

def   negative(n):
       if(n<0):
              return True
       else:
              return False

#main program
lst=(10,20,-40,-56,0,23,-67,89,-25,45)
filtobj=filter(positive,lst)
print("type of filtobj var=",type(filtobj)) # <class, 'filter'>
#print("Content of filtobj=",filtobj) Content of filtobj=   <filter object at
0x00000204745CF0D0>
pslst=list(filtobj)  # convert filter object into any collection object type
print("-------------------------------------------")
print("Original Elements=",lst)
print("-------------------------------------------")
print("Possitive elements=",pslst)
print("-------------------------------------------")
chi=filter(negative,lst)
nslst=set(chi)
print("Negative elements=",nslst)
print("-------------------------------------------")
-------------------------------------------------------------------------
#FilterEx2.py

posop = lambda n :  n>0  # anonymous function

negop=lambda n:n<0    # anonymous function

#main program
lst=(10,20,-40,-56,0,23,-67,89,-25,45)
filtobj=filter(posop,lst)
print("type of filtobj var=",type(filtobj)) # <class, 'filter'>
#print("Content of filtobj=",filtobj) Content of filtobj=   <filter object at
0x00000204745CF0D0>
pslst=list(filtobj)  # convert filter object into any collection object type
print("-------------------------------------------")
print("Original Elements=",lst)
print("-------------------------------------------")
print("Possitive elements=",pslst)
print("-------------------------------------------")
chi=filter(negop,lst)
```

```
nslst=set(chi)
print("Negative elements=",nslst)
print("-----------------------------------------")
```

```
#FilterEx3.py
lst=(10,20,-40,-56,0,23,-67,89,-25,45)
pslst=list(filter(lambda n :  n>0, lst))
nslst=tuple(filter(lambda n : n<0, lst))
print("-----------------------------------------")
print("Original Elements=",lst)
print("-----------------------------------------")
print("Possitive elements=",pslst)
print("-----------------------------------------")
print("Negative elements=",nslst)
print("-----------------------------------------")
```

```
#FilterEx4.py
#read the elements dynamically
lst=[]
n=int(input("Enter How many elements u want :"))
print("Enter {} elements:".format(n))
print("-----------------------------------------")
for i in range(1,n+1):
        val=float(input())
        lst.append(val)
else:
        print("-----------------------------------------")
        print("Original Elements=",lst)
        print("-----------------------------------------")
        pslst=list(filter(lambda n :  n>0, lst))
        nslst=tuple(filter(lambda n : n<0, lst))
        print("Possitive elements=",pslst)
        print("-----------------------------------------")
        print("Negative elements=",nslst)
        print("-----------------------------------------")
```

```
#FilterEx5.py
#read the elements dynamically
# program filtering Postive and Negative numbers by using filter()
print("Enter the values separated by comma:")
lst=[int (val) for val in input().split(",")]
print("-----------------------------------------")
print("Original Elements=",lst)
print("-----------------------------------------")
pslst=list(filter(lambda n :  n>0, lst))
nslst=tuple(filter(lambda n : n<0, lst))
print("Possitive elements=",pslst)
print("-----------------------------------------")
print("Negative elements=",nslst)
print("-----------------------------------------")
```

```
#Program for obtaining even and odd from list of values by using filter()
#filterex6.py
print("-----------------------------------------------")
print("Enter List of Values separated by space:")
lst=[int (val ) for val in input().split()]
#filter even elements
evenlst=list(filter(lambda n: n%2==0, lst))
oddlst=list(filter(lambda n: n%2!=0, lst))
print("-----------------------------------------------")
print("Original List={}".format(lst))
print("Even numbers List={}".format(evenlst))
print("Odd numbers List={}".format(oddlst))
print("-----------------------------------------------")
```

---

map() programs reduce() programs

```
                ==========================================
                                2) map()
                ==========================================
=>The purpose of map() is that " To get new list from old list by appylying
to the function "
=>Syntax:-   varname=map(Function_name, Iterable_object)

Explanation:
----------------------
=>'varname' is an object of  <class, 'map'> and we can convert into any
     iterable_object type.
=>"Function_name" is either normal function and anonymous function and it
must
      perform some operation based on logic
=>"Iterable_object "  can be any Sequence type or collection types.
=>Execution Process of map() is that "map() applies each value of iterable
object to the function and gets new iterable object. "


------------------------------------------------------------------------------
-------------------------
My Requirement:   oldsals=[10,20,10,30,40]---from old list
                             decided to give 10% hike

                             newsals=[11,22,11,33,44]----new list
```

```
#mapex1.py
def  hike(esal):
       esal=esal+esal*0.1
       return esal

#main program
print("Enter Old Salaries of employees:")
oldsal=[ int(sal) for sal in input().split()]
mapobj=map(hike,oldsal)
print("Type of map obj=",type(mapobj))# Type of map obj= <class 'map'>
print("content of map=",mapobj) # content of map= <map object at
0x000002102AF0F0D0>
newsal=list(mapobj)
print("---------------------------------------")
print("Old Salaries=",oldsal)
```

```
print("New Salaries=",newsal)
print("---------------------------------------")
```

```
#mapex2.py
hikesal=lambda esal :  esal+esal*0.1

#main program
print("Enter Old Salaries of employees:")
oldsal=[ int(sal) for sal in input().split()]
newsal=tuple(map(hikesal,oldsal))
print("---------------------------------------")
print("Old Salaries=",oldsal)
print("New Salaries=",newsal)
print("---------------------------------------")
```

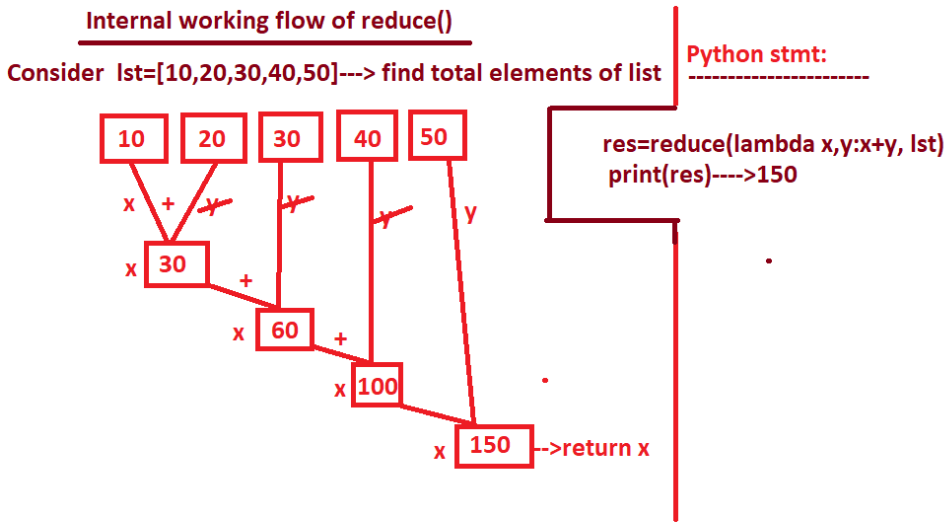
```
#mapex3.py
print("Enter Old Salaries of employees:")
oldsal=[ int(sal) for sal in input().split()]
newsal=tuple(map(lambda esal: esal*1.1,oldsal))
print("---------------------------------------")
print("Old Salaries=",oldsal)
print("New Salaries=",newsal)
print("---------------------------------------“)
```

```
#mapex4.py
print("Enter List of values :")
oldlst=[ float(val) for val in input().split()]
squarelist=list(map(lambda  n: n**2, oldlst))
sqrootlist=list(map(lambda k : k**0.5, oldlst))
print("-----------------------------------------------------")
print("Original values:{}".format(oldlst))
print("Square values:{}".format(squarelist))
print("Square Rool values:{}".format(sqrootlist))
print("--------------------OR-----------------------------")
result=zip(oldlst,squarelist,sqrootlist)
print("-"*50)
print("\tGiven Number\tSquare\tSquareRoot")
print("-"*50)
for on,sqn,sqt in result:
       print("\t{}\t\t{}\t{}".format(on,sqn,sqt))
print("-"*50)
```

```
#mapex5.py
print("Enter  Salaries of employees:")
sallist=[ int(sal) for sal in input().split()]
newsallist1=list(map(lambda sal:sal*1.1,list(filter(lambda sal :
sal>10000,sallist)) ))
newsallist2=list(map(lambda sal:sal*1.2,tuple(filter(lambda
sal:sal<=10000,sallist)) ))
print("Old Salaries :{}".format(sallist))
print("New Salaries having >10000:{}".format(newsallist1))
```

```
print("New Salaries having <=10000:{}".format(newsallist2))
```

```
                ============================================
                                reduce()
                 ===========================================
=>The purpose of reduce() is that "To obtain single result from list of
elements by applying to the function"
=>reduce() present in pre-defined module called 'functools' module.

Syntax:-    varname=functools.reduce(funcname,iterable_object)
=>varname is of type int, float, bool, complex and str.
-----------------------------------------
Internal flow of of reduce()
-----------------------------------------
Step-1) reduce() selects  First two element of any iterable object and
place them                 First variable and Second Variable( say K and V
)
Step-2) reduce() applies the First and Second  Variable values to the
function and              computed  and Resultant Value placed in First
Variable (say K).
Step-3) reduce() select next succeding element from Iterable object and
place it                   into second Variable (say V)
Step-4) Repeate Step-(2) and Step-(3) until all elements of iterable
object                    completed
Step-5) reduce() automatically Returns Result of First Variable ( Say K)
==================================X==================================
```



```
-------------------------------------------------------------------------

#redceex1.py
import functools
print("Enter Salaries of employees:")
sallist=[float(sal) for sal in input().split()]
totsal=functools.reduce(lambda x,y:x+y, sallist)
print("Total Sal=",totsal)
print("type of totalsal=",type(totsal))
print("-------------------------------------------------")

-------------------------------------------------------------------------
#reduceex2.py
import functools
print("Enter Number of values separated by space:")
nums=[ int(val) for val in input().split()]
big=functools.reduce(lambda x,y: x if x>y else y, nums)
print("-----------------------------------------------")
print("Original Elements={}".format(nums))
print("Biggest Element={}".format(big))
```

```
#reduceex3.py
import functools
print("Enter Number of values separated by space:")
nums=[ int(val) for val in input().split()]
big=functools.reduce(lambda x,y: x if x<y else y, nums)
print("-----------------------------------------------")
print("Original Elements={}".format(nums))
print("Smallest Element={}".format(big))
print("-----------------------------------------------")
```

```
#reduceex4.py
import functools
print("Enter Number of values separated by space:")
nums=[ int(val) for val in input().split()]
pssum=functools.reduce(lambda x,y:x+y, list(filter(lambda x: x>0,nums)))
nssum=functools.reduce(lambda x,y:x+y, list(filter(lambda x: x<0,nums)))
print("-----------------------------------------------")
print("Original Elements={}".format(nums))
print("Possitive Element sum={}".format(pssum))
print("Nagative Element sum={}".format(nssum))
print("-----------------------------------------------")
```

```
       ===========================================
              global  and local variables and globals()
        ===========================================
=>When we come acrosss same global Variable names and Local Variable Names
in same function definition then PVM gives preference for local variables
but not for global variables.
```

=>In this context, to extract / retrieve the global variables names along with local variables, we must use globals() and it returns an object of <class,'dict'> and this dict object stores all global variable Names as Keys and global variable values as values of value.

=>Syntax:-

```
var1=val1
var2=val2
--------------
var-n=val-n  # var1, var2...var-n are called global Variables
def    functionname():
  ------------------------
      var1=val11
      var2=val22
      -----------------
       var-n=val-nn  #  var1, var2...var-n are called local Variables
       # Extarct  the global variables
       dictobj=globals()
       globalval1=dictobj['var1']  # dictobj.get("var1") or
globals()['var1'] or globals().get("var1")
       globalval2=dictobj['var2']  # dictobj.get("var2") or
globals()['var2'] or globals().get("var1")
```

```
#globalfunex1.py
a=10
b=20
c=30
d=40   # here 'a' 'b' 'c' and 'd' are called global variables
def  operations():
       global c,d
       c=c+1   # c=31
       d=d+1  # d=41
       a=100
       b=200  # here 'a' and 'b' are called Local Variables
       print("-------------------------------------")
       print("Values of our program")
       print("-------------------------------------")
       print("Val of a (Local )=",a)
       print("Val of b (Local )=",b)
       print("Val of a (global )=",globals()['a'])
       print("Val of b (global )=",globals()['b'])
       print("-------------------------------------")
       res=a+b+c+d +globals()['a']+globals().get('b')  # 100+200+31+41--
>372+10+20
       print("sum=",res)
       print("-------------------------------------")
#main program
operations()
```

```
#globalsfunex2.py
sno=10
sname="Ritche"     # here sno,sname are called Global Variables
def   testing():
       sno=100
```

```
        sname="Rossum"  # here sno,sname are called local Variables
        print("-------------------------------------")
        print("Local Variable Values:")
        print("-------------------------------------")
        print("Student Number:",sno)
        print("Student Name:",sname)
        print("-------------------------------------")
        gv=globals()  # obtains all global variables
        print("type of gv=",type(gv))  # type of gv=<class,'dict'>
        print("Global Variable Values:")
        print("-------------------------------------")
        print("Student Number:",gv['sno'] )
        print("Student Name:",gv['sname'] )
        print("              OR                    ")
        print("Student Number:", gv.get("sno")   )
        print("Student Name:", gv.get("sname"))
        print("              OR                    ")
        print("Student Number:", globals()['sno'])
        print("Student Name:", globals()['sname'])
        print("              OR                    ")
        print("Student Number:", globals().get('sno'))
        print("Student Name:", globals().get('sname'))

        print("-------------------------------------")

#main program
testing()
```

==========

Modules In python

==========

=>We know that Functions concept meant for performing certain Operation and provides Code Re-usability within in the same program but not able to provide code-reusbality across the programs. To over come this problem we use Modules.

=>The purpose of Modules concept is that To provide code-reusbality across the programs.

=>Definition of Module:

=>A Module is a collection of Variables (Global Variables) , Functions and Classes.

=>In Python we have type of Modules. They are

a) Pre-defined Module

b) Programmer-defined module

---

a) Pre-defined Module:

=>These modules are already developed by Python Software developers and available in Python software and whose role is to deal with Universal Requirements.

=>Examples:- builtins functools calendar math cmath re cx_Oracle
mysql-connector, threading, numpy pandas...etc
NOTE:- builtins is the default pre-defined module

---

b) Programmer-defined module:

=>These modules are developed by Python Language Programmers and available in Python Project and whose role is to deal with Common Requirements.

Examples: banking, mathformulas, otpgen.....etc

Creating Programmer-defined Module

=>Creating a Programmer-defined Module is nothing but
a) Define / choose the Variables with Values (Global variables)
b) Define the Functions for performing Common Operations
c) Define the classes (In OOPs we discuss)

Save the above on some file name with an extension .py (FileName.py)

=>Hence FileName.py compiled Version(FileName.cpyhon-310.pyc) is treated as Module Name

=>Once the a module is created , whose module name is placed within a folder which is created automatically by Python Environment.

Examples: __pycache__
filename.cpython-310.pyc

```
#formula.py---file name and acts as module name--->(formula.cpython-
310.pyc)
def  simpleint():
       p=float(input("Enter Principle Amount:"))
       t=float(input("Enter the time:"))
       r=float(input("Enter Rate of Interest:"))
       #cal si
       si=(p*t*r)/100
       totamt=p+si
       print("-"*50)
       print("Simple Interest Details:")
       print("-"*50)
       print("Principle Amount:{}   ".format(p))
       print("Time:{}".format(t))
       print("Rate of Interest:{}".format(r))
       print("-"*50)
       print("Simple Interest :{}".format(si))
       print("Total Amount to Pay:{}".format(totamt))
       print("-"*50)
```

```
#mathsinfo.py---file name and acts as module name
PI=3.14
E=2.71  # here PI and E are called Global Variables
```

```
#SE1.py
from formula import simpleint,hello
simpleint()  # function call
hello("kVR")
```

```
#SE2.py
from mathsinfo import PI,E
print("Val of PI={}".format(PI))
print("Val of E={}".format(E))
```

```
               ========================================
                       Techniques for Re-Using the modules
                ========================================
=>In Python Programming, we have two techniques for Re-Using the modules.
They are
       1) by using import statement.
       2) by using from.... import statement.
-----------------------------------------------
1) by using import statement.
-----------------------------------------------
=>here 'import' is a keyword
=>import statement is used for refering the Variables Names, Function names
and class names  of a module .
-------------------------------------------------------------------------------
------------------
Syntax1:-     import  module name

=>This syntax imports single module name
Examples:    import   formula
                     import mathsinfo
-------------------------------------------------------------------------------
------------------
Syntax2:-     import  modulename1, modulename2,...modulename-n

=>This syntax imports multiple  module names
Examples:     import  formula, mathsinfo
-------------------------------------------------------------------------------
------------------
Syntax-3:   import  modulename as alias name
=>This syntax imports single module name with ailas name

=>Examples:             import formula as  f
------------------              import mathsinfo as m
-------------------------------------------------------------------------------
------------------

Syntax4:-     import  modulename1 as alias name1, modulename2 as alias name-2
                                    ,...modulename-n as alias name-n

=>This syntax imports multiple module names with alias names.

=>Examples:             import formula as  f, mathsinfo as m
```

```
------------------

=>AFTER IMPORTING A PERTICULAR MODULE, THE VARIABLE NAMES, FUNCTION NAMES AND
CLASS NAMES MUST BE ACCESSED W.R.T MODULE NAME OTHERWISE WE GET ERROR.
Syntax:-          Module Name.Variable Name
                       Module Name.Function Name
                       Module Name.Class Name
                              (OR)
                       Module  Alias Name.Variable Name
                       Module Alias  Name.Function Name
                       Module  Alias Name.Class Name
================================X=================================
2) by using from.... import statement:
-------------------------------------------------------------
=>here 'form'  , 'import' are the keywords
=>This approach also used for re-using the variable names, function names and
class names of a perticular modules in the current python program
----------------
Syntax1:
----------------
        import  module name as variable names,function names, class names
=>This syntax imports the variables names, function names and class names
from a specified module name.
Examples:-  from hyd import stateinfo,capinfo,hello,show
-------------------------------------------------------------------------------
-------------------------
Syntax2:
----------------
        import  module name as variable names as alias names,function names
as alias names , class names as alias names.
=>This syntax imports the variables names, function names and class names
from a specified module name with alias names.

 Examples:-
 from hyd import stateinfo as sf1,capinfo as cf1,hello as h1,show as s1
 from bang import stateinfo as sf2,capinfo as cf2,hello as h2,show as s2
-------------------------------------------------------------------------------
-------------------------
Syntax3:
------------
from module name  import *
=>This syntax imports all variable names, function names and Class names
=>This syntax is not recommended to to use bcoz it provides un-necessary
information to the current python program along required information.

Examples:      from hyd   import *
                       from bang import *
-------------------------------------------------------------------------------
--------------------
=>WITH THIS APPROACH , AFTER IMPORTING A PERTICULAR MODULE, THE VARIABLE
NAMES, FUNCTION NAMES AND CLASS NAMES CAN BE ACCESSED DIRECTLY WITHOUT USING
MODULE NAME

Syntax:-          Variable Name  (OR) aliasname of variable name
                       Function Name (OR) aliasname of function name
                       Class Name      (OR) aliasname of Class name
```

```
================================================================================
#syntax1.py
import  formula
import mathsinfo
formula.simpleint()
print("val of pi=",mathsinfo.PI)
print("val of e=",mathsinfo.E)
```

```
#syntax2.py
import  formula, mathsinfo
formula.simpleint()
print("subject =",formula.sub1)
print("val of pi=",mathsinfo.PI)
print("val of e=",mathsinfo.E)
print("subject =",mathsinfo.sub1)
```

```
#syntax3.py
import  formula as f
import  mathsinfo as hyd
f.simpleint()
print("val of pi=",hyd.PI)
print("val of e=",hyd.E)
```

```
#syntax4.py
import  formula as f , mathsinfo as k
f.simpleint()
print("val of pi=",k.PI)
print("val of e=",k.E)
```

```
#fromimportsyntax1.py
from hyd import stateinfo,capinfo,hello,show
print("-----------------------------------")
print("state name=",stateinfo)
print("capital name=",capinfo)
print("-----------------------------------")
hello("Rossum")
show()
```

```
#fromimportsyntax2.py
from hyd import stateinfo as sf1,capinfo as cf1,hello as h1,show as s1
from bang import stateinfo as sf2,capinfo as cf2,hello as h2 ,show as s2
print("-----------------------------------")
print("state name=",sf1)
print("capital name=",cf1)
print("-----------------------------------")
print("state name=",sf2)
print("capital name=",cf2)
print("-----------------------------------")
```

```
h1("Rossum")
s1()
print("-----------------------------------")
h2("Ritche")
s2()
```

```
#fromimportsyntax3.py
from bang import *
from hyd import *
print("-----------------------------------")
print("state name=",stateinfo)
print("capital name=",capinfo)
print("-----------------------------------")
hello("Rossum")
show()
```

```
#bang.py---file name and treated as module name ( bang.cpython-310.pyc)
stateinfo="Karnataka"
capinfo="BANGLORE"
def   hello(s):
       print("{}, Good Eevening from hello()--bang module ".format(s))

def  show():
       print("i am from show()-bang module")
```

```
#hyd.py---file name and treated as module name ( hyd.cpython-310.pyc)
stateinfo="Telangana"
capinfo="HYD"
def   hello(s):
       print("{}, Good Morning from hello()--hyd module ".format(s))

def  show():
       print("i am from show()-hyd module")
```

Case Study
- menuop.py
- arithop.py
- aopdemo.py

reloading a modules
programs

```
#menuop.py----file name and treated as module name
def   menu():
       print("-"*50)
       print("\tArithmetic Operations")
       print("-"*50)
       print("\t1.Addition")
       print("\t2.Substraction")
       print("\t3.Multiplication")
```

```
        print("\t4. Division")
        print("\t5. Modulo Division")
        print("\t6. Exponentiation")
        print("\t7. Exit")
        print("-"*50)
```

```
#arithop.py--------------file name and treated as module name
def   addop():
        a=float(input("Enter First Value for Addition:"))
        b=float(input("Enter Second Value for Addition:"))
        print("\tsum({},{})={}".format(a,b,a+b))

def  subop():
        a=float(input("Enter First Value for Substraction:"))
        b=float(input("Enter Second Value for Substraction:"))
        print("\tsub({},{})={}".format(a,b,a-b))

def mulop():
        p=float(input("Enter First Value for Multiplication:"))
        q=float(input("Enter Second Value for Multiplication:"))
        print("\tmul({},{})={}".format(p,q,p*q))

def  divop():
        p=float(input("Enter First Value for Division:"))
        q=float(input("Enter Second Value for Division:"))
        print("\tDiv({},{})={}".format(p,q,p/q))
        print("\tFloor Div({},{})={}".format(p,q,p//q))

def  modop():
        a=float(input("Enter First Value for Modulas :"))
        b=float(input("Enter Second Value for Modulas:"))
        print("\tmod({},{})={}".format(a,b,a%b))

def  expoop():
        a=float(input("Enter Value for Base:"))
        b=float(input("Enter Value for power:"))
        print("\texp({},{})={}".format(a,b,a**b))
```

```
#aopdemo.py-----------main program
from menuop import menu
import sys
from arithop import *
while(True):
        menu()
        ch=int(input("Enter Ur Choice:"))
        match ch:
                case 1:
                                        addop()
                case 2:
                                        subop()
                case 3:
                                        mulop()
                case 4:
                                        divop()
```

```
                case 5:
                                        modop()
                case 6:
                                        expoop()
                case 7:
                                print("Thanks for Using this App!")
                                sys.exit()
                case _ :
                                        print("Ur Selection of Operation is
wrong-try again")
```

```
               ========================================
                       realoding a modules in Python
                ========================================
=>To reaload a module in python , we use a pre-defined function called
reload(), which is present in imp module and it was deprecated in favour
of importlib module.
=>Syntax:-    imp.reload(module name)
                                        (OR)
               importlib.reload(module name) -----recommended
---------------------------------
=>Purpose / Situation:
---------------------------------
=>reaload() reloads a previously imported module. if we have edited the
module source file  by using an external editor and we want to use the
changed values
                (OR)
       To get the new version of previously loaded module then we use
reload().
================================X=========================

#shares.py---file and treated as module name
def sharesinfo():
       d={"Tech":19,"Pharma":11,"Auto":1,"Finance":00}
       return d


#main program
#sharesdemo.py

import shares
import time
import importlib
def disp(d):
       print("-"*50)
       print("\tShare Name\tValue")
       print("-"*50)
       for sn,sv in d.items():
              print("\t{}\t\t:{}".format(sn,sv))
       else:
              print("-"*50)
#main program
d=shares.sharesinfo()  #previously imported module
disp(d)
```

```
time.sleep(15)
importlib.reload(shares)  # relodaing previously imported module
d=shares.sharesinfo() # obtaining changed / new values ofpreviously
imported
module
disp(d)
```

```
#sharesinfo.py---file name and treated as module name
shinfo={"IT":1010,"Pharm":1170,"automobiles":100,"textiles":180}
```

```
#sharesdisplay.py
import time,importlib
import sharesinfo
print(sharesinfo.shinfo)
time.sleep(15)
importlib.reload(sharesinfo)
print(sharesinfo.shinfo)
time.sleep(15)
importlib.reload(sharesinfo)
print(sharesinfo.shinfo)
```

```
#shares.py---file name and terated as module name
def sharesinfo():
       d={"Tech":1999,"Pharma":1111,"Auto":111,"Finance":1120}
       return d
```

```
#sharesdemo.py
import shares,time,importlib

def   disp(d):
       print("-"*40)
       print("\tShare Name\tShare Value")
       print("-"*40)
       for sn,sv in d.items():
              print("\t{}\t\t\t{}".format(sn,sv))
       else:
              print("-"*40)
#main program
d1=shares.sharesinfo()
disp(d1)
time.sleep(15)
importlib.reload(shares)  # reloading the module name
d1=shares.sharesinfo()
disp(d1)
```

Packages in Python

examples

```
       ================================================
                      Packages in Python
       ================================================
```

=>We know that FUNCTIONS concept is used for Performing Certain operation and provides Code Re-usability within the program but not able to provide Code Re-usability across the programs.

=>We know that MODULES concept is used for re-using the code across the programs provided the modules must present in same Folder but not able to get Code Re-usability across Folders / Drives / Environments / Networks..etc
=>The PACKAGES concept is used for getting the Code Re-usability across Folders / Drives / Environments / Networks..etc through modules where modules contains Variables, Functions and Classes.

Def. of Packages:

=>A Package is a collection of Modules.

=>Creating a package:

Step-1: Create a folder
Step-2: Define / place an an empty python file on the name of __init__.py in folder to make the folder name as Package Name.
Step-3: Define / place a module (s) in the package(Folder Name)

==========

Number of approaches to re-use the modules of Packages

=>We have two approaches re-use the modules of Packages. They are
1) By using sys.path.append()
2) By using PYTHONPATH Environmental Variable.

---

1) By using sys.path.append():

Syntax:-

```
sys.path.append("Absolute path of Package Name")
```

Examples:

```
#kvr1.py
import sys
sys.path.append("E:\KVR-PYTHON-7AM\PACKAGES\BANK")
from formula import simpleint
simpleint()
```

(OR)

```
#kvr1.py
import sys
sys.path.append("E:\KVR-PYTHON-7AM/PACKAGES/BANK")
from formula import simpleint
simpleint()
```

---

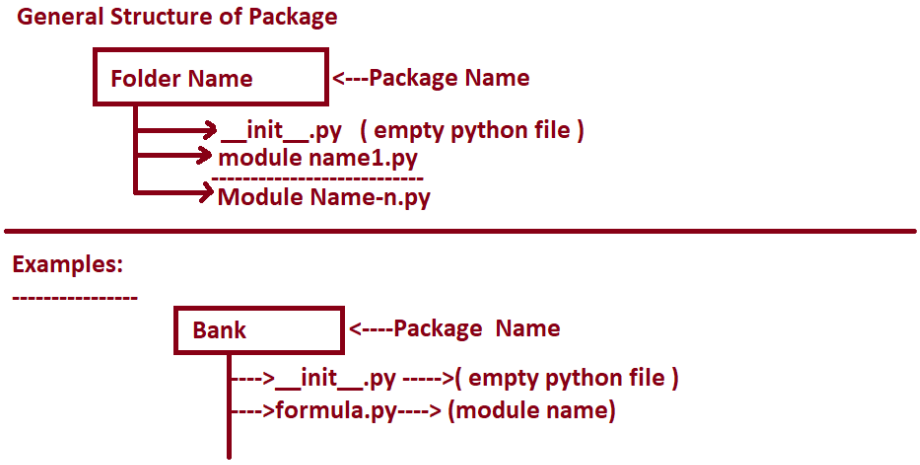
2) By using PYTHONPATH Environmental Variable.

=>PYTHONPATH is one of the keyword for OS and hence it is called Environmental Variable.

=>To set PYTHONPATH, do the follwing steps

a) Goto start button
b) choose Settings
c) Choose System
d) Search for "environmental Variables"
e) choose "new"
f) Type PYTHONPATH for Variable Name
g) Place Absolute path of package as Variable Value
h) Choose OK and OK

Later Execute the program freshly.
==========X==========



==========
Introduction to Exception Handling
and
Types of Errors
==========

=>The purpose of Exception Handling is that "To develop Robust (Strong) Applications"
=>In Real Time , To develop any project, we need to choose a language and by using it we can develop, compile and execute various Programs. During this process, we get 3 types of Errors. They are

a) Compile Time Errors
b) Logical Errors
c) Runtime Errors

---
a) Compile Time Errors:

```
----------------------------------
=> Compile Time Errors are those, which are occuring at Compile Time

                      (.py--->.pyc)
=> Compile Time Errors are occuring due to Syntaxes are not folowed by the

     Programmer.
=>These erros solved by Programmers at development Level.
-------------------------------------------------------------------------------
-------------------------------
b) Logical  Errors:
----------------------------------
=>Logical Errors are those, which are occuring at Execution Time
=>Logical Errors are occuring due to wrong representation of Logic.
=>Logical Errors are always gives Wrong / Inconsistant Result.
=>These errors solved by Programmers at development level.
-------------------------------------------------------------------------------
-------------------------------
c) Runtime Errors
----------------------------------
=>Runtime Errors are those, which are occuring at Execution Time
=>Runtime Errors are occuring due to "Wrong Inputs / Invalid Inputs
entered  by Application Users / End Users".
=>Runtime Errors must be handled by Programmer during Development time
with Forecasting Knowledge.
==========================================
Points to be remembered in Exception Handling
==========================================
1) When the application / end  user enters wrong input / Invalid invalid
then we get
    Runtime Errors
2) Runtime Errors by default gives Technical Errors Messages. These
messages     are understanble Programmers but not by End Users. Industry
always
recommends convert Technical Error Message into User Friendly Error
Messages by Using Exception Handling

3) Definition of exception:- Every Runtime error is called Exception
                      (Invalid Input--->Runtime Error--->Exception)
4) Every exception by default gives Technical Error Message.

5) Definition of exception handling:- The Process of converting Technical
Error Messages into User Friendly Error Messages is called Exceptional
Handling.

6)When the exception occurs in Python Program, Three steps takes place
internally. They are
       a) PVM Terminates the Program execution abnormally.
       b) PVM comes out of Program flow without executing rest of the
statements
       c) PVM by default generates Technical Error Messages.

7) To do the steps (a),(b) and (c) , PVM internally creates an object of
appropriate exception classes.
```

```
8) Hence Every Invalid Input gives exception and every exception is
treated as object and it is created w.r.t appropriate exception classes.
(Invalid  Input--->exception--->object---->appropriate exception class )
=========================X=====================================
```

Types of exceptions in Python

Handling the exceptions in Python

```
                ===================================
                       Types of exceptions in Python
                ===================================
=>In Python, we have two types of exceptions. They are
                1. Pre-defined / Built-in exceptions.
                2. Programmer / User / Custom-defined Exceptions.

1. Pre-defined / Built-in exceptions:
-----------------------------------------------------
=>Pre-defined / Built-in exceptions are developed By Python Language
Developers and they are available in Python API(Library)and whose role is
to deal with Universal Problems.
=>Some of the Universal Problems are
                a) Division by Zero Problems ( ZeroDivisionError)
                b) Invalid number formats ( ValueError)
                c) Invalid arguments / Operations (TypeError)
                d) searching the value of value in in dict
                     by passing invalid key  (KeyError)
                e) Invalid Variable Names (NameError)......etc
-------------------------------------------------------------------------
--
2. Programmer / User / Custom-defined Exceptions.
-------------------------------------------------------------------------
--
=>These exceptions developed by Programmer and they are avalible in Python
Project and they used used for dealing with Common Problems.
=>Some of the Common Problems are
                a) Attempting to enter invalid PIN in atm applicatioins.
                b) Attempting to Wrong User Name and Password.
                c) Attempting to withdraw Invalid amount from existing
account
                .........................etc
=>Every Valid Input gives successful output
=>Every Invalid Input gives Exceptions.
```

```
                ========================================
                       Handling the exceptions in Python
                ========================================
=>Handling the exceptions in Python are nothing but converting Technical
error messages into User-Friendly Error Messages.
=>To convert Technical error messages into User-Friendly Error Messages,
we have 5 keywords. They are
                      1) try
                      2) except
```

```
                    3) else
                    4) finally
                    5) raise
-----------------------------------------------------
=>Syntax for handling the exceptions:
-----------------------------------------------------
             try:
                  Block of statements
                  Generating Exceptions
             except  <exception-class-name-1>:
                   Block of statements
                   generating User-Friendly Error Messages
             except  <exception-class-name-2>:
                   Block of statements
                   generating User-Friendly Error Messages
             ----------------------------------------
             except  <exception-class-name-n>:
                   Block of statements
                   generating User-Friendly Error Messages
             else:
                   Block of statements
                   generating Result of the Program
              finally:
                   Block of statements
                   executed by PVM Compulsorily
```

```
#This Program demonstartes how to cal division of two numbers
#by accepting two integer values from KBD
#div1.py
s1=input("Enter First Value:")
s2=input("Enter Second Value:")
a=int(s1) #  ----- exception generated statement
b=int(s2)  # ----- exception generated statement
c=a/b  # -------exception generated statement
print("Val of a={}".format(a))
print("Val of b={}".format(b))
print("Div={}".format(c))
```

```
#This Program demonstartes how to cal division of two numbers
#by accepting two integer values from KBD
#div2.py
try:
       s1=input("Enter First Value:")
       s2=input("Enter Second Value:")
       a=int(s1)
       b=int(s2)
       c=a/b
except ZeroDivisionError:
       print("\nDON'T ENTER ZERO FOR DEN...")
except ValueError:
       print("\nDON'T ENTER  strs/symbols/alpha-numeric values:")
else:
       print("\nResult---else block")
       print("-"*50)
```

```
        print("Val of a={}".format(a))
        print("Val of b={}".format(b))
        print("Div={}".format(c))
        print("-"*50)
finally:
        print("\ni am from finally block")
```

Explanation for the keywords used in Syntax for handling the exceptions:

Various forms of except blocks

```
#This Program demonstartes how to cal division of two numbers
#by accepting two integer values from KBD
#div2.py
try:
        s1=input("Enter First Value:")
        s2=input("Enter Second Value:")
        a=int(s1)
        b=int(s2)
        c=a/b
except ZeroDivisionError:
        print("\nDON'T ENTER ZERO FOR DEN...")
except ValueError:
        print("\nDON'T ENTER  strs/symbols/alpha-numeric values:")
else:
        print("\nResult---else block")
        print("-"*50)
        print("Val of a={}".format(a))
        print("Val of b={}".format(b))
        print("Div={}".format(c))
        print("-"*50)
finally:
        print("\ni am from finally block")
```

```
#This Program demonstartes how to cal division of two numbers
#by accepting two integer values from KBD
#div3.py
try:
        s1=input("Enter First Value:")
        s2=input("Enter Second Value:")
        a=int(s1)
        b=int(s2)
        c=a/b
except (ZeroDivisionError,ValueError):
        print("\nDON'T ENTER ZERO FOR DEN...")
        print("\nDON'T ENTER  strs/symbols/alpha-numeric values:")
else:
        print("\nResult---else block")
        print("-"*50)
        print("Val of a={}".format(a))
```

```
        print("Val of b={}".format(b))
        print("Div={}".format(c))
        print("-"*50)
finally:
        print("\ni am from finally block")
```

```
#This Program demonstartes how to cal division of two numbers
#by accepting two integer values from KBD
#div4.py
try:
        s1=input("Enter First Value:")
        s2=input("Enter Second Value:")
        a=int(s1)
        b=int(s2)
        c=a/b
except ZeroDivisionError as  k:
        print(k) # division by zero
except ValueError as v:
        print(v) # invalid literal for int() with base 10: 'abc'
else:
        print("\nResult---else block")
        print("-"*50)
        print("Val of a={}".format(a))
        print("Val of b={}".format(b))
        print("Div={}".format(c))
        print("-"*50)
finally:
        print("\ni am from finally block")
```

```
#This Program demonstartes how to cal division of two numbers
#by accepting two integer values from KBD
#div5.py
try:
        s1=input("Enter First Value:")
        s2=input("Enter Second Value:")
        a=int(s1)
        b=int(s2)
        c=a/b
        s="PYTHON"
        print(s[10])
except ZeroDivisionError:  # specific except block
        print("\nDON'T ENTER ZERO FOR DEN...")
except ValueError:   # specific except block
        print("\nDON'T ENTER  strs/symbols/alpha-numeric values:")
except IndexError:   # specific except block
        print("Plz check the index :")
except :  # default except block
        print("exception occurs---Some went wrong!")
else:
        print("\nResult---else block")
        print("-"*50)
```

```
        print("Val of a={}".format(a))
        print("Val of b={}".format(b))
        print("Div={}".format(c))
        print("-"*50)
finally:
        print("\ni am from finally block")
```

```
#This Program demonstartes how to cal division of two numbers
#by accepting two integer values from KBD
#div6.py
try:
        s1=input("Enter First Value:")
        s2=input("Enter Second Value:")
        a=int(s1)
        b=int(s2)
        c=a/b
        s="PYTHON"
        print(s[10])
except ZeroDivisionError:  # specific except block
        print("\nDON'T ENTER ZERO FOR DEN...")
except ValueError:   # specific except block
        print("\nDON'T ENTER  strs/symbols/alpha-numeric values:")
except IndexError:   # specific except block
        print("Plz check the index :")
except BaseException:  # default except block
        print("exception occurs---Some went wrong!")
else:
        print("\nResult---else block")
        print("-"*50)
        print("Val of a={}".format(a))
        print("Val of b={}".format(b))
        print("Div={}".format(c))
        print("-"*50)
finally:
        print("\ni am from finally block")
```

==========

Explanation for the keywords used

in

Syntax for handling the exceptions:

==========

1) try block:

=>It is the block, in which we write block of statements generating exceptions. In otherwords what are all the statements are generating exceptions then those statements must be written in try block and hence try block is called exception monitering block.

=>When the exception occurs in try block then PVM comes out of try block and excecutes appropriate except block.

=>When the pvm executes appropriate except block, PVM never goes to try block for executing rest of the statements in try block.

=>Every try block must be immediately followed by except block(Otherwise we get error).

=>Every try block must contain atleast one except block and it is recommeded to write multiple except blocks for generating multiple user-frinedly error messages.

---

2) except block:

=>It is the block, in which write Block of statements generating User-Friendly Error Messages. In otherwords, except block supresses the technical error messages and generates user-friendly error Messages and except block is called exception processing block.

Note:- Handling the Exception=try block+except block.

=>except block will execute when there is an exception occurs in try block.

=>Even though we write multiple except blocks , at any point of time only appropriate except block will execute depends on type of exception occurs in try block.

=>The place of writing except block is after try block and before else block (if we wrirte else block).

---

3) else block:

=>It is the block , In which we write block of statements generating Result of the program

=>else block will execute when there is no exception exception occurs in try block

=>Writing else block is optional

=>we write else block after except block and before finally block (if we write finally block)

---

4) finally block:

=>It is the block, in which we write block of statements relinquish (release / close / give-up/ clean-up) the resources (Files/ Database) which are obtained in try block.

=>finally block will execute compulsorily irrespective of type of exception occurs or not

=>Writing finally block is optional.

=>finally block to be written after else block

==========X==========

==========

Various forms of except blocks

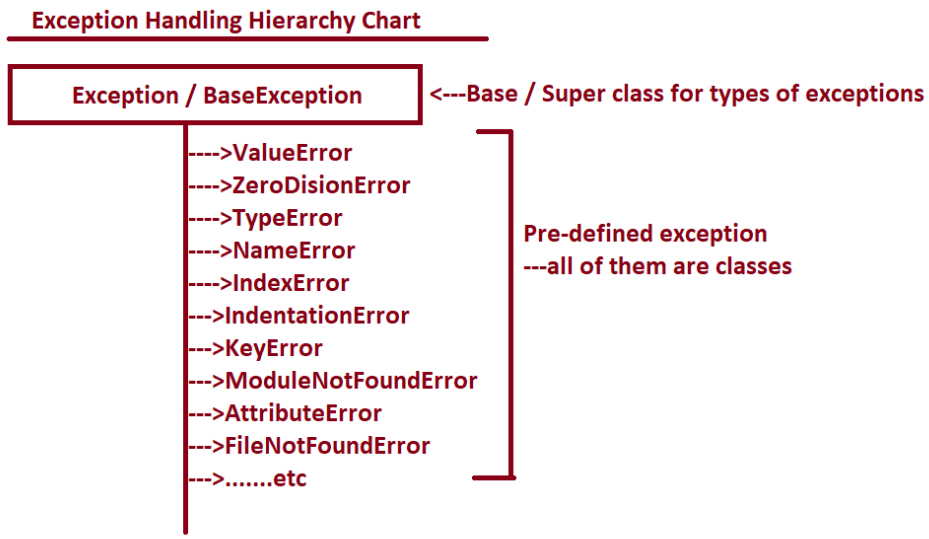
==========

Form-1 (Handling One Exception at a time):

```
        try:
            blcok of statements
            generating exceptions
        except <Exception-Class-Name>
            Block of statements
```

```
            generating user frinedly error Messages
-------------------------------------------------------------------------
----------------------
Form-2 (Handling Multiple Exception at a time--multi exception handling
block):
-------------------------------------------------------------------------
-----------------------------
try:
    blcok of statements
    generating exceptions
except (exception class name-1, exception class name-2.. exception class
name-n)
    Block of statements
    generating user frinedly error Messages
-------------------------------------------------------------------------
---------------------------------
Form-3:  (Handling Multiple Exception with alias name)
-------------------------------------------------------------------------
-------------------------------
=>With this Syntax, we are obtaining messages causing due to exception
occurence

try:
    blcok of statements
    generating exceptions
except <Exception-Class-Name-1> as <alias-name-1>
    Block of statements
    generating user frinedly error Messages
except <Exception-Class-Name-2> as <alias-name-2>
    Block of statements
    generating user frinedly error Messages
-------------------------------------------------
-------------------------------------------------
except <Exception-Class-Name-n> as <alias-name-n>
    Block of statements
    generating user frinedly error Messages
-------------------------------------------------------------------------
-------------------------------
Form-4:  (Handling Multiple Exceptions with deafult except block)
-------------------------------------------------------------------------
-------------------------------
try:
    blcok of statements
    generating exceptions
except  :
    Block of statements
    generating user frinedly error Messages

NOTE:- Industry is always recommended to write default except block after
specific except blocks but before spefic except blocks.
-------------------------------------------------------------------------
-----------------------------------
Form-5:  (Handling Multiple Exceptions with Base class exception)
```

```
-------------------------------------------------------------------------------
------------------------------
try:
    blcok of statements
    generating exceptions
except Exception  :
    Block of statements
    generating user frinedly error Messages
```



development of programmer-defined exceptions

raise keyword concept

```
        ====================================================
                              raise keyword
         ====================================================
=>raise keyword is used for hitting / raising / generating the exception
when certain condition is satisfied.
=>PVM uses raise keyword for hitting Pre-defined exceptions automatically.
where as programmer makes the PVM  to use raise keyword  to hit
programmer-defined exceptions when certain condition is satisfied.

=>Syntax:-
                        if(test cond):
                               raise exception-class-name
```

```
=>Syntax:
               def   function-name(list of formal params if any):
                      if(test cond):
                             raise exception-class-name
                      ------------------------------------------
-------------------------------------------


==============================X================================

               ========================================
                             Development
                                    of
                      Programmer / User / Custom-defined Exceptions.
               ========================================
=>These exceptions developed by Programmer and they are avalible in Python
Project and they used used for dealing with Common Problems.
=>Some of the Common Problems are
               a) Attempting to enter invalid PIN in atm applicatioins.
               b) Attempting to enter Wrong User Name and Password.
               c) Attempting to withdraw Invalid amount from existing
account
               ........................etc
=>Every Valid Input gives successful output
=>Every Invalid Input gives Exceptions.
=======================================================
Steps for developing Programmer-defined Exceptions:
=======================================================
1) Choose the Programmer-Defined Class Name
2) The Programmer-Defined Class Name must inherit Either from "Exception"
or
     BaseException for obtaining exception handling features.
3) Save the above development on some filename with an extension .py (
treated as module name)

Examples:
----------------
# pin.py----------(3)
               (1)          (2)
class PinError(Exception):pass

Examples:
----------------
#login.py--------(3)
                (1)                  (2)
class LoginError(BaseException):pass
-----------------------------------------------------------------
We follow 3 phases for development of Programmer-defined exception. They
are
               a) Develop Programmer-defined Exception classes
               b) Hit / Generate the Programer-defined Exceptions (raise )
               c) Handle the Programmer-defined exceptions(try,except,
else, finally)
```

```
-------------------------------------------------------------------------------
--------------------------------
a) Develop Programmer-defined Exception classes

Example:
------------------
#Phase-I--Development Programmer-defined Exception
#kvr.py---file name and treated as module name
class KvrDivisionError(Exception):pass
-------------------------------------------------------------------------------
----------------------------
b) Hit / Generate the Programer-defined Exceptions (raise )
-------------------------------------------------------------------------------
-------------
#Phase-II:  Hitting the exception by using raise keyword.
#program cal of division of two numbers
#divop.py---file name and treated as module name
from kvr import  KvrDivisionError
def  division(a,b):
       if(b==0):
              raise KvrDivisionError  # hitting / generating Programmer-
defined
              exception
       else:
              c=a/b
              return c
-------------------------------------------------------------------------------
--------------------------
c) Handle the Programmer-defined exceptions(try,except, else, finally)
-------------------------------------------------------------------------------
----------------------------
#Phase-III: Handling the exceptions by using try and except kwds.
#divopdemo.py  ---main program
from divop import division
from kvr import KvrDivisionError
try:
       a=int(input("Enter Value of a:"))
       b=int(input("Enter Value of b:"))
       result=division(a,b)
except KvrDivisionError:
       print("\nDon't Enter Zero for Den...")
except ValueError:
       print("\nDon't enter strs / alpha-numerics/ special symbols:")
else:
       print("\nDiv({},{})={}".format(a,b,result))
finally:
       print("\nI am from finally Block")
=====================================X=========================
```

case studies of Programmer-defined exceptions

Case-1

Case-2

ATM

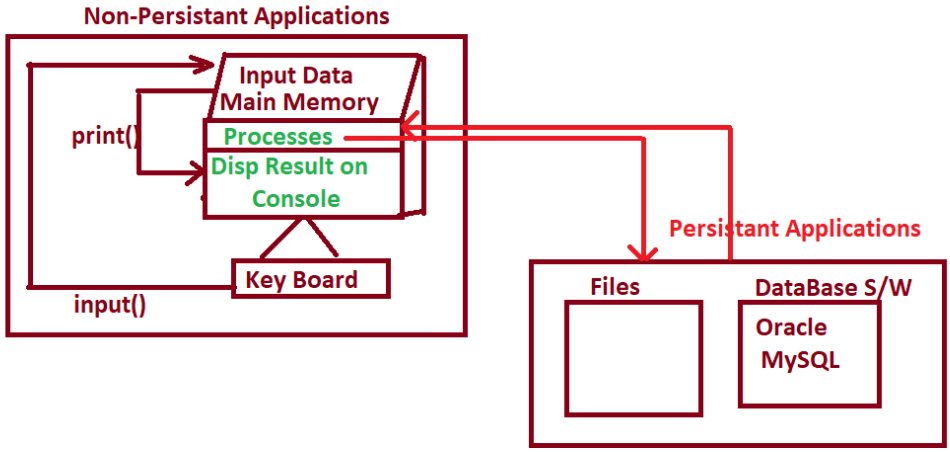
compressed archive

Types of Applications

Introduction to Files

Operations on Files

```
                ====================================
                        Types of Applications
                 ====================================
=>The purpose of Files in any programming Language is that "To Store the
data Permanently (Data Persistency )".
=>In the context of Files, we can develop two types of Applications /
Programs.. They are
               a) Non-Persistant Applications
               b) Persistant Applications

=>In Non-Persistant Application Development, we read / accept  the from
input data from KBD, Stores in main memory (temp storage) , process the
inputs and displays its result on the console.

=>Examples:  All previous program are comes under non-persistant
applications.

=>In Persistant Application Development, we read / accept  the from input
data from KBD, Stores in main memory (temp storage) , process the inputs
and stores the result Permanently.
=>In Industry , we have two types of approaches to store the data
permanently. They are
               a) By Using Files
               b) By using Database softwares
```



Main menu

Python @ 7.00 AM | Mr. K.V.Raofrom 17th Nov 2021

Announcement

Kv Rao

•

27 Jan

Types of Applications

Introduction to Files

Operations on Files

Types of Applications.txt

Text

non-persistant and persistant applications.png

Image

Data Organization in Files and Main memory.png

Image

Introduction to Files.txt

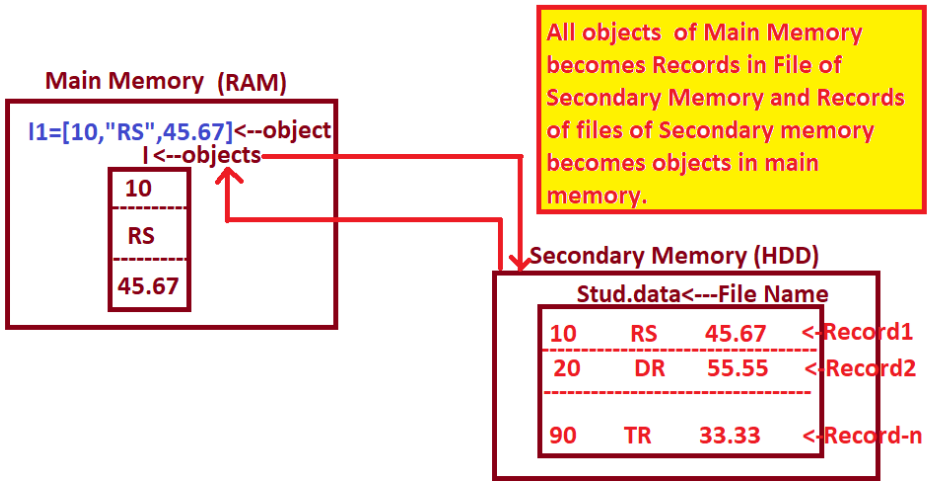
Text

Operations on Files.txt

Text

Class comments

Add class comment…



==========

```
                    Introduction to Files
                ========================================
=>The purpose of Files in any programming Language is that "To Store the
data Permanently (Data Persistency )".
----------------------
=>Def. of File:   A File is a collection of Records.
----------------------
=>Def. of Stream:   The flow of Data between Main memory and secondary
memory
-------------------------   is called Stream.

=>All the objects data of main memory can be stored  permanently in the
form of Files of Secondary memory and Records of file of Secondary Memory
will become objects in Main Memory.
```

```
                ====================================
                        Operations on Files
                 ====================================
=>On Files, we can perform two types of Operations. They are
                 a) Write Operation.
                 b) Read Operation.
------------------------------
a) Write Operation.
------------------------------
=>This operation is used for transfering Temporary data from Main Memory
into the file of secondary memory .
=>Steps:
-------------
        1) Choose the file name.
        2) Open the file name in write mode
        3) Perform Cycle of Write Operations.

=>During Write Operation, we get exceptions like
FileExistError,IOError..etc.
------------------------------
b) Read Operation.
------------------------------
=>This operation is used for transfering the Data of files from Secondary
Memory into object of Main Memory .
=>Steps:
-------------
                 1) Choose the file name.
                 2) Open the File Name in Read Mode
                 3) Perform Cycle of Read Operations
=>During Read Operation, we get exceptions like
FileNotFoundError,EOFError..etc
==========================x===================================
```

Types of Files in Python

File Opening Modes

```
                ========================================
                              Types of Files in Python
                 ========================================
=>In Python, we have two types of Files. They are
```

1) Text Files
2) Binary Files

---

1) Text Files:

=>A Text File always contains the data in the form Alphabets, Digits and Special Symbols.
=>In Python Programming, text files are denoted by a letter 't'.
=>By default,When we deal with Files, File is considered as text file.
=>Examples: .txt .doc .py .java .cpp ...etc

---

2) Binary Files

=>A Binary File always contains the data in the form Binary Format.
=>In Python Programming, Binary files are denoted by a letter 'b'.
=>Examples: .jpeg, .jpg, .gif, audio, video .pdf...etc
.exe. .png.....etc

==========

File Opening Modes

==========

=>File Opening Modes are used for opening the file in certain file Mode.
=>In otherwords , File Opening Modes makes us to understand, in which mode the file is opening.
=>In Python Programming we have 7 file opening modes. They are

1) r :

=>This mode is used for opening the File in read mode.
=>When we open the file in 'r' mode and if the file does not exist in Secondary Memory then we get FileNotFoundError.
=>The default file mode is 'r' .

---

2) w :

=>This mode is used for Opening the File always in write mode newly.
=>If we open new file in 'w' mode then it open in write mode and data written to file from the begining.
=>If we open the existing file in 'w' mode then it open in write mode, existing data is replaced with new data.(Overlapping)

---

3) a :

=>This mode is used for Opening the File always in write mode.
=>If we open new file in 'a' mode then it open in write mode and data written to file from the begining.
=>If we open the existing file in 'a' mode then it open in write mode and new data adding at end of existing data(know as appending)

---

4) r+ :

```
=>This mode is used for opening the file in read mode.
=>When we open the file in 'r' mode then we can perform First Read
Operation and latter we can perform Write Operation(But not reverse
order).
=>When we open the file in 'r+' mode and if the file does not exist in
Secondary Memory then we get FileNotFoundError.
--------
5) w+ :
--------
=>This mode is used for Opening the File always in write mode newly.
=>f we open new file in 'w' mode then it open in write mode and data
written to file from the begining and latter we can perform read
operation.
=>If we open the existing file in 'w+' mode then it opens in write mode,
existing data is replaced with new data.(Overlapping)
=>When we write the data on file with this mode and if we have
insufficient memory in Seconadry Memory then we get IOError.
--------
6) a+ :
--------
=>This mode is used for Opening the File always in write mode.
=>If we open NEW FILE in 'a' mode then it open in write mode and data
written to file from the begining and later we can read the data.
=>If we open the EXISTING FILE  in 'a' mode then it open in write mode and
new data adding at end of existing data( know as appending) and later we
can perform read operation.
--------
7) x :
--------
=>This Mode is used for opening the file Exclusively in Write Mode only
once.
=>If we open the existing file in x mode then we get FileExistError.
```

Opening the Files in Python programs

Writing the data to the files

programs

```
                ==================================
                       Opening the Files in Python
                 ==================================
=>To do any operation on files, we have open the file.
=>To open the file , we have two syntaxes. They are
                1) By using open()
                2) By using    " with  open()  as ".

-----------------------------
1) By using open():
-----------------------------
=>Syntax:-        VarName=open("File Name", "File Mode")
=>here "VarName" repersent a pointer to the file and it called "File
Pointer " and it is an object of type <class, _io.TextIOWrapper>
```

=>open() is pre-defined function, which is used for opening the file in specified File Mode.
=>"File Name" represents Name of File
=>File Mode represents r,w,a,r+,w+,a+ and x.
=>Once we open the files by using open(), we need to close the files explicitly by using close() . In otherwords open() approach is unable to provide auto-closable property.

---

2) By using   " with open() as "

Syntax:-

```
                    with  open("File Name", "File Mode") as <varname>:
                                 ----------------------------------------
--------
                                 ----------------------------------------
-------
                                 Block of Statements--Operation
                                 ----------------------------------------
------
                                 ----------------------------------------
------

                    ---------------------------------------------------
-------
                    Other Statements in Python Program
                    ---------------------------------------------------
-------
```

Explanation:

=>"with" and " as " are keywords
=>open() is a function used to open the file in specified file mode
=>here "VarName" repersent a pointer to the file and it called "File Pointer " and it is an object of type <class, _io.TextIOWrapper>.
=>The main advatange of this syntax is that It provides auto-closable of files. In otherwords, we need not to close any files explicitly.

==========

Writing the data to the files

==========

=>Once we open the file, it is necessary to perform the operations on the files.
=>On Files, we can perform two types of Operation and they can write and read.
=>To write the data to the files, we have two pre-defined functions which are
    present in the object of TextIOWrapper. They are

1) write()
2) writelines()

```
1) write():
----------------
Syntax:-            filepointer.write(data)

=>here  "filepointer" is an object, it always points to the file.
=>" write() " is a pre-defined function, which is used to write the data
to the file
    always in the form of str.
=>"data" represents always any type of value but it should be always in
the form
     of str.
--------------
Example:
--------------
#FileWriteEx1.py
with open("addr.data","a") as fp:
       fp.write("Dennis Ritche\n")
       fp.write("FNO:130,,Fort Side\n")
       fp.write("Bell Labds \n")
       fp.write("USA\n")
       print("\nData Written to the File:")
-------------------------------------------------------------------------
-----------------------------------
2) writelines():
---------------------------
=>This function is used for writing the data int the Iterbable object and
data of iterable object must be of type str.
=>If any iterable object contains other than str then we must convert
entire iterable object into str type by using str()

=>Syntax:-    filepointer.writelines( Iterable object )

Examples:
-----------------------------------------------------------
#FileWriteEx2.py
d={10:"Mango",20:"Apple",30:"Kiwi"}
with open("addr1.info","a") as fp:
       fp.writelines(str(d)+"\n")
       print("Data written to the file")
-----------------------------------------------------------
#FileWriteEx2.py
sti={40,"Sree Devi",88.48,"HCU"}
with open("addr1.info","a") as fp:
       fp.writelines(str(sti)+"\n")
       print("Data written to the file")
-----------------------------------------------------------
#FileWriteEx2.py
tpl=(20,"Babani",88.88,"JNTU")
with open("addr1.info","a") as fp:
       fp.writelines(str(tpl)+"\n")
       print("Data written to the file")
-----------------------------------------------------
#FileWriteEx2.py
lst=[10,"Rutuja",88.88,"OU"]
```

```
with open("addr1.info","a") as fp:
	fp.writelines(str(lst)+"\n")
	print("Data written to the file")

==============================X================================
#This program demonstrates for obtaining the information about files.
#FileInfoEx1.py
try:
	fp=open("stud.info","r")
except FileNotFoundError:
	print("File does not exists:")
else:
	print("id of fp=",id(fp))
	print("File Opened in read mode successfully")
	print("-"*40)
	print("Mode used={}".format(fp.mode))
	print("Is readable={}".format(fp.readable()))
	print("Is writable={}".format(fp.writable()))
	print("Line--14: is file closed={}".format(fp.closed))
	print("-"*40)
finally:
	print("\nI am from finally block")
	fp.close()
	print("Line--19: is file closed={}".format(fp.closed))


"""
Note:
--------
=>"mode" is an attribute in the object of TextIOWrapper and it is used for
giving
		the file mode
=>"closed" is an attribute in the object of TextIOWrapper and it returns
False in
		 the case file is active in open otherwise  it gives True
=>"readable()" is a pre-defined function in the object of TextIOWrapper
and it
			returns True in the case of reading operation on
the file otherwise it returns False
=>"writable()" is a pre-defined function in the object of TextIOWrapper
and it
					returns True in the case of writing
operation on the file otherwise it returns False

"""
```

```
#This program demonstrates for obtaining the information about files.
#FileInfoEx2.py
try:
	fp=open("stud.info","r+")
except FileNotFoundError:
	print("File does not exists:")
else:
	print("File Opened in read mode successfully")
```

```
        print("-"*40)
        print("Mode used={}".format(fp.mode))
        print("Is readable={}".format(fp.readable()))
        print("Is writable={}".format(fp.writable()))
        print("Line--13: is file closed={}".format(fp.closed))
        print("-"*40)
finally:
        print("\nI am from finally block")
        fp.close()
        print("Line--18: is file closed={}".format(fp.closed))


"""
Note:
--------
=>"mode" is an attribute in the object of TextIOWrapper and it is used for
giving
                  the file mode
=>"closed" is an attribute in the object of TextIOWrapper and it returns
False in
                   the case file is active in open otherwise  it gives True
=>"readable()" is a pre-defined function in the object of TextIOWrapper
and it
                        returns True in the case of reading operation on
the file otherwise it returns False
=>"writable()" is a pre-defined function in the object of TextIOWrapper
and it
                                      returns True in the case of writing
operation on the file otherwise it returns False

"""
```

```
#This program demonstrates for obtaining the information about files.
#FileInfoEx3.py
fp=open("stud3.info","a+")
print("File Opened in append mode successfully")
print("-"*40)
print("Mode used={}".format(fp.mode))
print("Is readable={}".format(fp.readable()))
print("Is writable={}".format(fp.writable()))
print("Line--13: is file closed={}".format(fp.closed))
print("-"*40)


"""
Note:
--------
=>"mode" is an attribute in the object of TextIOWrapper and it is used for
giving
                  the file mode
=>"closed" is an attribute in the object of TextIOWrapper and it returns
False in
                   the case file is active in open otherwise  it gives True
=>"readable()" is a pre-defined function in the object of TextIOWrapper
and it
```

```
                        returns True in the case of reading operation on
the file otherwise it returns False
=>"writable()" is a pre-defined function in the object of TextIOWrapper
and it
                                        returns True in the case of writing
operation on the file otherwise it returns False

"""
```

```
#This program demonstrates for obtaining the information about files.
#FileInfoEx4.py
try:
       fp=open("stud4.info","x")
       print("File Opened in x mode successfully")
except FileExistError:
       print("File alerady exist:")
else:
       print("-"*40)
       print("Mode used={}".format(fp.mode))
       print("Is readable={}".format(fp.readable()))
       print("Is writable={}".format(fp.writable()))
       print("Line--13: is file closed={}".format(fp.closed))
       print("-"*40)
finally:
       print("\nI am from finally block")
       fp.close()
       print("Line--18: is file closed={}".format(fp.closed))
"""
Note:
--------
=>"mode" is an attribute in the object of TextIOWrapper and it is used for
giving
                  the file mode
=>"closed" is an attribute in the object of TextIOWrapper and it returns
False in
                   the case file is active in open otherwise  it gives True
=>"readable()" is a pre-defined function in the object of TextIOWrapper
and it
                        returns True in the case of reading operation on
the file otherwise it returns False
=>"writable()" is a pre-defined function in the object of TextIOWrapper
and it
                                        returns True in the case of writing
operation on the file otherwise it returns False

"""
```

```
#This program demonstrates for obtaining the information about files.
#FileInfoEx5.py----- with  open()  as
try:
       with open("stud.info","r+") as fp:
              print("-"*40)
              print("File opened Successfully in read mode")
              print("-"*40)
              print("Mode used={}".format(fp.mode))
              print("Is readable={}".format(fp.readable()))
```

```
                print("Is writable={}".format(fp.writable()))
                print("Line--10: is file closed={}".format(fp.closed))
                print("-"*40)

        print("\nLine--14: is file closed after out of 'with open()'
={}".format(fp.closed))

except FileNotFoundError:
        print("File does not exists:")
finally:
        print("\nI am finally block:")
        print("Line--19: is file closed  ={}".format(fp.closed))
```

```
#This program demonstrates opening the file in read mode
#FileOpenEx1.py
try:
        fp=open("stud.info", "r")
        print("type of fp=",type(fp)) # type of fp= <class
'_io.TextIOWrapper'>
        print("File Opened in read mode successfully")
except FileNotFoundError:
        print("File does not exists:")
#This program demonstrates opening the file in write mode
#FileOpenEx2.py
kvr=open("stud1.info", "x")



print("File Opened in write mode successfully--verify")
```

```
#FileWriteEx1.py
with open("addr.data","a") as fp:
        fp.write("Dennis Ritche\n")
        fp.write("FNO:130,,Fort Side\n")
        fp.write("Bell Labds \n")
        fp.write("USA\n")
        print("\nData Written to the File:")
```

```
#FileWriteEx2.py
lst=[10,"Rutuja",88.88,"OU"]
with open("addr1.info","a") as fp:
        fp.writelines(str(lst)+"\n")
        print("Data written to the file")
```

Reading the Data from the Files
Random Access Files

```
                ========================================
                        Reading the Data from the Files
                 ======================================
=>To read the data from the files , we have 4 pre-defined Functions which
are present in an object of TextIoWrapper . They are
                        1) read()
                        2) read(no.of chars)
                        3) readline()
                        4) readlines()
```

```
-----------------------------------------------------
1) read():
-----------------------------------------------------
=>This function is used for reading entire data of file in the form of
str.
=>Syntax:          varname=filepointer.read()

Examples:
--------------
#This program reads entire data of the file by using read()
#FileReadEx1.py
try:
       fname=input("Enter File Name to read its content:")
       with open(fname,"r") as fp:
              filedata=fp.read()
              print("Complete Content of the File:")
              print("-------------------------------------")
              print(filedata)
              print("-------------------------------------")
except FileNotFoundError:
       print("File does not exists")
============================================================
2) read(no.of chars):
------------------------------
=>This Functions is used for reading specified number of characters.
=>Syntax:-
--------------
                      varname=filepointer.read(no. of chars)
=>Here "no.of chars" represents How many Characters u want to erad.

Examples:
----------------
#This progfram demostarates how to read specified number of chars.
#FileReadEx2.py------read(no. of chars)
try:
       with open("emp.info","r") as fp:
              print("Initial Index of fp=",fp.tell())  # 0
              filedata=fp.read(6)
              print("File Data=",filedata)
              print("-"*40)
              print("Now Index of fp=",fp.tell())# 6
              filedata=fp.read(16)
              print("File Data=",filedata)
              print("-"*40)
              print("Now Index of fp=",fp.tell())#23
              filedata=fp.read(20)
              print("File Data=",filedata)
              print("-"*40)
              print("Now Index of fp=",fp.tell())#
              filedata=fp.read()
              print("File Data=",filedata)
              print("-"*40)
              print("Now Index of fp=",fp.tell())#159 --last index
              print("-"*40)
```

```
                fp.seek(0) # here 0 says initial index of file
                print("Line-23: Initial Index of fp=",fp.tell())  # 0
                filedata=fp.read()
                print("File Data=",filedata)

except FileNotFoundError:
        print("File does not exists:")
```

```
==================================================
3) readline():
-----------------
=>This Function is used for reading one line at a time in the form of str.
=>Syntax:-
                        varname=filepointer.readline()

Examples:
------------------
#This progfram demostarates how to read line by line from the file .
#FileReadEx3.py------readline()
try:
        fp=open("emp.info","r")
        filedata=fp.readline()
        print(filedata,end="")
        filedata=fp.readline()
        print(filedata,end="")
        filedata=fp.readline()
        print(filedata,end="")
        filedata=fp.readline()
        print(filedata,end="")

except FileNotFoundError:
        print("File does not exists:")
==================================================
4) readlines():
----------------------
=>This function is used for reading all the lines of the file in the form
of list type.
=>Syntax:-    varname=filepointer.readlines()
Examples:
------------------
#This program demostarates how to reading all the flies from the file.
#FileReadEx4.py------readlines()
try:
        fp=open("emp.info","r")
        filedata=fp.readlines()
        for line in filedata:
                print(line,end=" ")
        print()
except FileNotFoundError:
        print("File does not exists:")
```

```
                ====================================
                        Random Access Files
                ====================================
```

=>Random Access Files are those which allows us to read the data randomly accoding to programmer choice.
=>To read the data randomly, we have two pre-defined functions present in TextIoWrapper. They are

a) tell()
b) seek()

a) tell():

=>This Function is used for obtaining the Position / Index of the file pointer.

=>Syntax:- Index=file pointer.tell()

b) seek():

=>This Function is used for re-positioning the file pointer withing file content by passing Index value.
=>Syntax:- filepointer.seek(index)

```
#FileReadEx2.py------read(no. of chars)
try:
        with open("emp.info","r") as fp:
                print("Initial Index of fp=",fp.tell())  # 0
                filedata=fp.read(6)
                print("File Data=",filedata)
                print("-"*40)
                print("Now Index of fp=",fp.tell())# 6
                filedata=fp.read(16)
                print("File Data=",filedata)
                print("-"*40)
                print("Now Index of fp=",fp.tell())#23
                filedata=fp.read(20)
                print("File Data=",filedata)
                print("-"*40)
                print("Now Index of fp=",fp.tell())#
                filedata=fp.read()
                print("File Data=",filedata)
                print("-"*40)
                print("Now Index of fp=",fp.tell())#159 --last index
                print("-"*40)
                fp.seek(0) # here 0 says initial index of file
                print("Line-23: Initial Index of fp=",fp.tell())  # 0
                filedata=fp.read()
                print("File Data=",filedata)

except FileNotFoundError:
        print("File does not exists:")
```

```
#This program reads entire data of the file by using read()
#FileReadEx1.py
try:
        fname=input("Enter File Name to read its content:")
        with open(fname,"r") as fp:
                filedata=fp.read()
                print("Complete Content of the File:")
```

```
                print("--------------------------------------")
                print(filedata)
                print("--------------------------------------")
except FileNotFoundError:
        print("File does not exists")
```

```
#This progfram demostarates how to read specified number of chars.
#FileReadEx2.py------read(no. of chars)
try:
        with open("emp.info","r") as fp:
                print("Initial Index of fp=",fp.tell())  # 0
                filedata=fp.read(6)
                print("File Data=",filedata)
                print("-"*40)
                print("Now Index of fp=",fp.tell())# 6
                filedata=fp.read(16)
                print("File Data=",filedata)
                print("-"*40)
                print("Now Index of fp=",fp.tell())#23
                filedata=fp.read(20)
                print("File Data=",filedata)
                print("-"*40)
                print("Now Index of fp=",fp.tell())#
                filedata=fp.read()
                print("File Data=",filedata)
                print("-"*40)
                print("Now Index of fp=",fp.tell())#159 --last index
                print("-"*40)
                fp.seek(0) # here 0 says initial index of file
                print("Line-23: Initial Index of fp=",fp.tell())  # 0
                filedata=fp.read()
                print("File Data=",filedata)

except FileNotFoundError:
        print("File does not exists:")
```

```
#This progfram demostarates how to read line by line from the file .
#FileReadEx3.py------readline()
try:
        fp=open("emp.info","r")
        filedata=fp.readline()
        print(filedata,end="")
        filedata=fp.readline()
        print(filedata,end="")
        filedata=fp.readline()
        print(filedata,end="")
        filedata=fp.readline()
        print(filedata,end="")

except FileNotFoundError:
        print("File does not exists:")
```

```
#This program demostarates how to reading all the flies from the file.
#FileReadEx4.py------readlines()
try:
        fp=open("emp.info","r")
        filedata=fp.readlines()
        for line in filedata:
```

```
                print(line,end=" ")
        print()
except FileNotFoundError:
        print("File does not exists:")
```

```
#This Program will copy the content of one file(Source File) into another
file(destination File)
#FileCopy.py
try:
        sfile=input("Enter Source File:")
        with open(sfile,"r") as rp:
                dfile=input("Enter Destination File:")
                with open(dfile,"a") as wp:
                        filedata=rp.read()
                        wp.write(filedata)
                        print("\nFile Copied --verify")

except FileNotFoundError:
        print("File Does Not exists")
```

Programs on files

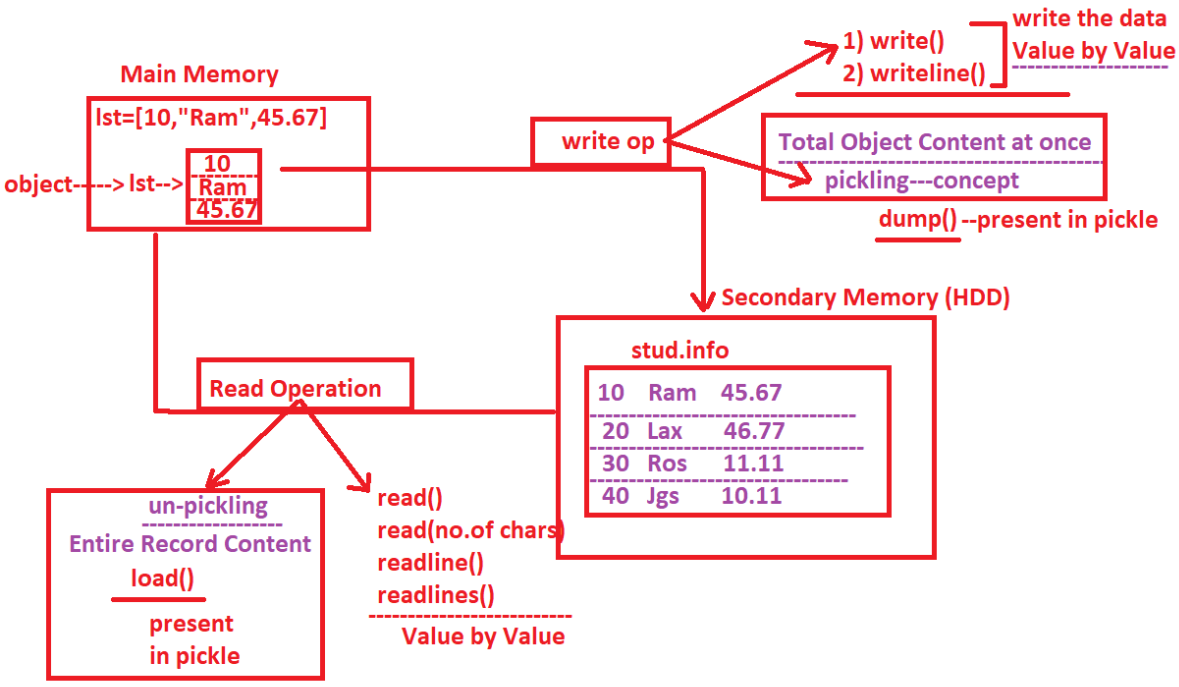
```
#Program for reading the data from key board dynamically and write it to
the file
#DynamicData.py
import sys
with open ("hyd.info","a") as fp:
        print("Enter The data for writing into the file(press 'stop' to
terminate):")
        print("-"*40)
        while(True):
                filedata=input()
                if(filedata!="stop"):
                        fp.write(filedata+"\n")
                else:
                        print("-"*40)
                        sys.exit()
```

```
#Program for finding number of lines, number of words and no. of chars
from a given file.
#FileCountInfo.py
try:
        nl,nw,nc=0,0,0
        fname=input("Enter File Name :")
        with open(fname) as fp:
                for line in fp:
                        print(line,end="")
                        nl=nl+1
                        nw=nw+len(line.split())
                        nc=nc+len(line)
                else:
                        print()
                        print("-"*40)
```

```
                        print("No. of lines in file={}".format(nl))
                        print("No. of Words={}".format(nw))
                        print("No. of Characters={}".format(nc))
                        print("-"*40)
except FileNotFoundError:
        print("File does not exists:")
```

```
#Program for finding number of lines, number of words and no. of chars from a
given file.
#FileCountInfo1.py
try:
        nl,nw,nc=0,0,0
        fname=input("Enter File Name :")
        with open(fname) as fp:
                fileinfo=fp.readlines()    # here fileinfo is an obj. if list
                for line in fileinfo:
                        print(line,end="")
                        nl=nl+1
                        nw=nw+len(line.split())
                        nc=nc+len(line)
                else:
                        print()
                        print("-"*40)
                        print("No. of lines in file={}".format(nl))
                        print("No. of Words={}".format(nw))
                        print("No. of Characters={}".format(nc))
                        print("-"*40)
except FileNotFoundError:
        print("File does not exists:")
```

Pickling and Un-Pickling concepts



==========

Pickling and Un-Pickling

(OR)

Object Serialization and Deserialization

==========

Pickling (Object Serialization):

==========

=>Let us assume, there exist an object in mamin memory and it contains multiple values. To save / write the object data into the file of secondary memory by using write() and writelines() , internally these functions will write Value by Value and it is considered as Time Consuming Process . To overcome this problem, we must another concept called "PICKLING"

=>The Advantage of Pickling concept is that with single write operation, we can write / save the entire object data into the file of secondary memory.

=>Definition of Pickling:

=>The process of writing / saving the entire object content into the file of Secondary
Memory with single write operation is called Pickling.

=>The Pickling always participates in write operation.

=>While we are implementing Pickling Concept, we must ensure that the file must be
binary.

=>Steps for implementing Pickling Concept:

Step-1: import pickle module

step-2: Create an object with data

Step-3: use dump() of pickle module for writing entire object content into the file of
secondary memory.
Syntax: pickle.dump(object, filepointer)

==========

Un-Pickling((Object De-Serialization):

=>Let us assume there exist a record in the file of secondary Memory. To read the record data from the file by using read(), read(no.of chars), readline() and realines(), Internally these functions reads value by value and it is considered as Time Consuming Process.To overcome this process, we must use the concept of UN-PICKLING".

=>The advantage of un-pickling concept is that with single read operation, we can read entire record data from the file of secondary memory.

=>Definition of Un-Pickling:

=>The Process of Reading entire record content from the file of secondary memory into the object of memory with single read operation is called Un-Pickling.

=>Un-Pickling participates in read operation.

=>While we are implementing Un-Pickling Concept, we must ensure that the file must be binary.

```
-----------------------------------------------------------
=>Steps for implementing Un-Pickling Concept:
-----------------------------------------------------------
Step-1:  import pickle module
Step-2:  use load() of pickle module , for reading entire object content
into the object of                main memory
              Syntax:    objectdata=pickle.load()
Step-3:  Display object data (De-serialized data)
-------------------------------------------------------------------------
-----
=>hence Pickling and Un-pickling concepts Enhaces the performnace of
Normal File Programming of Python.
=>To deal with Pickling and Un-pickling concepts , we use a pre-defined
module "pickle".
==========================X==================================
```

Random Access Files

```
=>Random Access Files are those which allows us to read the data randomly
accoding  to programmer choice.
=>To read the data randomly, we have two pre-defined functions present in
TextIoWrapper. They are
                     a) tell()
                     b) seek()

a) tell():
-------------
=>This Function is used for obtaining the Position / Index of the file
pointer.

=>Syntax:-                   Index=file pointer.tell()

b) seek():
-----------------
=>This Function is used for re-positioning the file pointer withing file
content  by passing Index value.
=>Syntax:-          filepointer.seek(index)
-------------------------------------------------------------------------
-------
#FileReadEx2.py------read(no. of chars)
try:
       with open("emp.info","r") as fp:
              print("Initial Index of fp=",fp.tell())  # 0
              filedata=fp.read(6)
              print("File Data=",filedata)
              print("-"*40)
              print("Now Index of fp=",fp.tell())# 6
```

```
                filedata=fp.read(16)
                print("File Data=",filedata)
                print("-"*40)
                print("Now Index of fp=",fp.tell())#23
                filedata=fp.read(20)
                print("File Data=",filedata)
                print("-"*40)
                print("Now Index of fp=",fp.tell())#
                filedata=fp.read()
                print("File Data=",filedata)
                print("-"*40)
                print("Now Index of fp=",fp.tell())#159 --last index
                print("-"*40)
                fp.seek(0) # here 0 says initial index of file
                print("Line-23: Initial Index of fp=",fp.tell())  # 0
                filedata=fp.read()
                print("File Data=",filedata)

except FileNotFoundError:
        print("File does not exists:")
```

Pickling and Un-Pickling programs

```
#emppick.py-----Program-(A)
#This Program reads employee data from KBD and Save employee data into the
file
import pickle
noe=int(input("Enter How Many Employee Data u have:"))
if(noe<=0):
        print("{} is invalid Input".format(noe))
else:
        with open("emp.data","ab") as fp:
                for i in range(1,noe+1):
                        print("-"*40)
                        print("Enter {} Employee Information:".format(i))
                        print("-"*40)
                        #accept employee from KBD
                        eno=int(input("Enter Employee Number:"))
                        ename=input("Entyer Employee Name:")
                        sal=float(input("Enter Employee Salary:"))
                        #create an empty list and append data
                        lst=list()
                        lst.append(eno)
                        lst.append(ename)
                        lst.append(sal)
                        #save lst data into the file
                        pickle.dump(lst,fp)
                        print("-"*40)
                        print("{} employee record saved successfuly in
file:".format(i))
```

```
#empunpick.py-------------Program-(B)
#This program reads the employee records from the file
import pickle
try:
```

```
        with open("emp.data","rb") as fp:
                print("-"*40)
                print("\tE m p l o y e e  D e t a i l s")
                print("-"*40)
                while(True):
                        try:
                                record=pickle.load(fp)
                                for val in record:
                                        print("\t{}".format(val),end="")
                                print()
                        except EOFError:
                                print("-"*40)
                                break
except FileNotFoundError:
        print("File does not exists:")
```

```
#studpick.py-----Program-(A)
#This Program reads student data from KBD and Save student data into the
file
import pickle
def   savestuddata():
        with open("stud.data","ab") as fp:
                #accepting student data from KBD
                while(True):
                        print("-"*50)
                        try:
                                sno=int(input("Enter Student Number:"))
                                sname=input("Enter Student Name:")
                                marks=float(input("Enter Student Marks:"))
                                print("-"*50)
                                #create an empty list and append
                                lst=[]
                                lst.append(sno)
                                lst.append(sname)
                                lst.append(marks)
                                #save lst data into the file
                                pickle.dump(lst,fp)
                                print("-"*50)
                                print("\nStudent Data Saved Successfully ina
file:")
                                print("-"*50)
                                usrchoice=input("\nDo u want to Insert
another record(yes/no):")
                                if(usrchoice=="no"):
                                        print("Thans for using this
Application:")
                                        break
                                if(usrchoice!="yes"):
                                        print("\nU Please Learn Typing!")
                                        print("Thanks for using this
Application:")
                                        break
                        except ValueError:
                                print("\nDon't Enter strs/alpha-numeric vals
/ symbols for stno and marks:")
```

```
#main program
savestuddata()
```

```
#studunpick.py-------------Program-(B)
#This program reads the student records from the file
import pickle
def  readstudrecord():
       try:
              with open("stud.data","rb") as fp:
                     print("-"*40)
                     print("\tS t u d e n t  I n f o r m a t i o n:")
                     print("-"*40)
                     while(True):
                            try:
                                   rec=pickle.load(fp)
                                   for val in rec:
                                          print("\t{}".format(val),
end="")
                                   print()
                            except EOFError:
                                   print("-"*40)
                                   break
       except FileNotFoundError:
              print("File does not exists")

#main program
readstudrecord()
```

```
#This Program copy the image (pjstudent.png) into another image.
#ImageCopy.py
def  saveimage():
       try:
              with open("D:\KVR-PY\pjstudent.png","rb") as rp:
                     with open("C:\PIC\\robo.png","wb") as wp:
                            fileimg=rp.read()
                            wp.write(fileimg)
                            print("\nImage Copied---Verify:")
       except FileNotFoundError:
              print("File does not exists:")

#main program
saveimage()
```

OS Module concept

Programs

```
              ====================================
                            OS Module in Python
               ====================================
=>"os" is one of the pre-defined module in python
=>The Purpose of "os" module  is that "To interact with Operating System
and Performs Some Operations with OS".
=>Some of the Operation , we do on Operating System are
              a) obtaining Current Working Folder
              b) Creating a Folder
              c) Creating Root Folder, Sub Folder, sub-sub folder..etc
```

```
              d) Removing Folder
              e) Removing Root Folder, Sub Folder, sub-sub folder..etc
              f) list the files of Folders.
              g) renaming the folder.
-------------------------------------------------------------------------
----
a) obtaining Current Working Folder/ Directory:
-----------------------------------------------------------------
=>To get Current Working Folder, we use a pre-defined Function called
getcwd(), which is present in os module.
=>Syntax:-  varname=os.getcwd()
--------------
Examples:
--------------
#cwdex1.py
import os
cwdname=os.getcwd()
print("\nCurrent Working Folder=", cwdname)
=========================================================
b) Creating a Folder:
--------------------------------
=>To create a folder, we use mkdir() of os module.
=>Syntax:-     os.mkdir("FolderName")
=>This Function can create one folder at a time and unable create Folders
hierarchy(Root Folder\sub-folder...etc ) and we get OSError
=>If folder Name already exist and if try to create then we get
FileExistsError
---------------
Examples:
---------------
#Creating a Folder
#folderex.py
import os
try:
       os.mkdir("C:\apple\banana\kiwi")
       print("Folder Created --Verify")
except FileExistsError:
       print("Folder already created:")
except OSError:
       print("Folders Hierarchy Can't be created")
=============================================
c) Creating Root Folder, Sub Folder, sub-sub folder..etc
-------------------------------------------------------------------------
-----
=>To create Root Folder, Sub Folder, sub-sub folder..etc(Folders
Hierarchy) , we use a pre-defined function makedir() of os module.
=>Syntax:-       os.makedir("Folders Hierarchy")
=>Folders Hierarchy represents Root Folder, Sub Folder, sub-sub
folder..etc
=>This Function gives FileExistsError when folders Hierarchy already
exists.
--------------------
=>Example:
#createfolders.py
```

```
import os
try:
	os.makedirs("C:\India\Hyd\Ampt\Python")
	print("Folders Hierarchy Created--verify")
except FileExistsError:
	print("\nFolders Hierarchy already Created:")
```

---

d) Removing Folder:

=>To remove a folder, we use rmdir() of os module.
=>Syntax:- os.rmdir("Folder Name")
=>This Function remove at a time only one folder but it can't remove folders hierarchy.
=>If we try to remove a folder and if it does not exists then we get FileNotFoundError.
=>rmdir() can't remove a folder when that folder contains some files and we get OSError.

Examples:

```
#removefolder.py
import os
try:
	os.rmdir("G:\KVR-PYTHON-7AM\FILES")
	print("Folder removed / deleted:")
except FileNotFoundError:
	print("Folder Does not exists:")
except OSError:
	print("Folder contains some files-u can't remove")
```

==========

e) Removing Root Folder, Sub Folder, sub-sub folder..etc

=>To remove Root Folder, Sub Folder, sub-sub folder..etc(Folders Hierarchy) , we use removedirs() of os module.
=>Syntax:- os.removedirs("Folders Hierarchy")
=>Folders Hierarchy represents Root Folder, Sub Folder, sub-sub folder..etc
=>If Folders Hierarchy does not exists then we get FileNotFoundError
=>If any folder in Folders Hierarchy contains some file then we get OSError.

Examples:

```
#removefolders.py
import os
try:
	os.removedirs("C:\India\Hyd\Ampt\Python\KVR")
	print("Folders Hierarchy removed--verify")
except FileNotFoundError:
	print("File does not exists")
except OSError:
	print("Folder in Folder Hierachy conatins some file--so we can't remove")
```

```
============================================================
f) list the files of Folders.:
-----------------------------------
=>To list the files of the folders, we use listdir() of os module and all
files are available in the form of list object
=>Syntax:-       varname=os.lisdir("FolderName")
=>if the folder name does not exists then we get FileNotFoundError

Examples:
---------------
#listfiles.py
import os
try:
       lst=os.listdir("G:\KVR-PYTHON-8PM\MODULES")
       for name in lst:
              print("\t{}".format(name))

except FileNotFoundError:
       print("File does not exists")
-------------------------------------------------------------------------
g) renaming the folder.
-----------------------------------
=>To rename a folder, we use rename() of os module.
=>Syntax:     os.rename("Existing Folder Name", "New Folder Name")
=>If the Existing Folder Name is not present then we get
FileNotFoundError.
----------------
Examples:
----------------
#renameex.py
import os
try:
       os.rename("C:\Dev","C:\INDIA")
       print("Folder Re-named--verify")
except FileNotFoundError:
       print("File does not exists")
```

```
#Creating Root Folder, Sub Folder, sub-sub folder..etc
#createfolders.py
import os
try:
       os.makedirs("C:\India\Hyd\Ampt\Python\KVR")
       print("Folders Hierarchy Created--verify")
except FileExistsError:
       print("\nFolders Hierarchy already Created:")
```

```
#obtaining Current Working Folder/ Directory
#cwdex1.py
import os
cwdname=os.getcwd()
print("\nCurrent Working Folder=", cwdname
```

```
#Creating a Folder
#folderex.py
import os
try:
       os.mkdir("D:\KVR")
```

```
        print("Folder Created --Verify")
except FileExistsError:
        print("Folder already created:")
except OSError:
        print("Folders Hierarchy Can't be created")
```

```
#list the files of Folders
#listfiles.py
import os
try:
        lst=os.listdir("G:\KVR-PYTHON-8PM\MODULES")
        for name in lst:
                print("\t{}".format(name))

except FileNotFoundError:
        print("File does not exists")
```

```
#Removing Folder
#removefolder.py
import os
try:
        os.rmdir("G:\KVR-PYTHON-7AM\FILES")
        print("Folder removed / deleted:")
except FileNotFoundError:
        print("Folder Does not exists:")
except OSError:
        print("Folder contains some files-u can't remove")
```

```
#Removing Root Folder, Sub Folder, sub-sub folder..etc
#removefolders.py
import os
try:
        os.removedirs("C:\India\Hyd\Ampt\Python\KVR")
        print("Folders Hierarchy removed--verify")
except FileNotFoundError:
        print("File does not exists")
except OSError:
        print("Folder in Folder Hierachy conatins some file--so we can't
remove")
```

```
#renaming the folder.
#renameex.py
import os
try:
        os.rename("C:\Dev","C:\INDIA")
        print("Folder Re-named--verify")
except FileNotFoundError:
        print("File does not exists")
```

```
==========================================================================
```

Advantages of OOPs
OOPS principles

```
        ========================================================
          Advantages of Object Oriented Principles / Features / Concetps
        ========================================================
=>In Rea Time, To develop any project we need a language and it can
satisfy two types of Principles. They are
                1) Procedure Oriended Principles
                2) Object Oriended Principles
```

=>Python Programming satisfies Both Procedure Oriended (Functional Programming) and Object Oriended Principles.

"Every Thing is an object in Python"

(or)

Benifits of Object Oriented Principles

=>The objects allows us to store un-limited amount of data and achives Platform
Indepenent.

=>The large volume of data can be transfered between two remote machines all at
once and we can achieve Effective Communication.

=>The Confidential Data can be transfered between two remote machines in the form
of object where it can be available in the form of cipher text / encrypted format . so that we can achieve the security.

=>The Data always stored / available in the form of objects and on the objects data
we can perform the operations by using Functions.

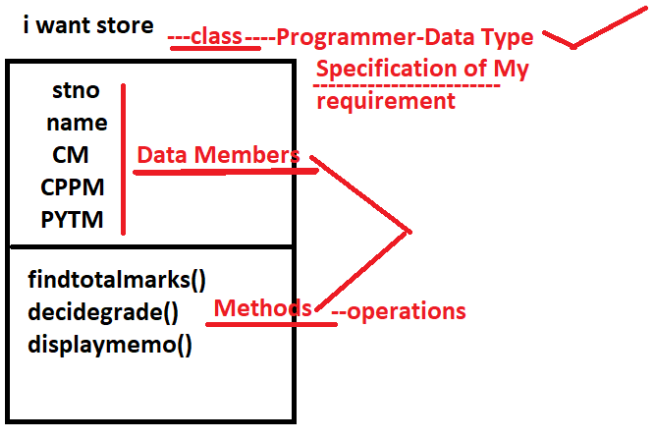
==========X==========

Object Oriented Principles / Features / Concetps

=>To Say a programming Language is Object Oriented then it has to satisfy the
following object oriented principles.

1. classes
2. objects
3. Data Encapsulation
4. Data Abstraction
5. Inheritance
6. Polymorphism
7. Message Passing (we already discussed)

=>The above Object Oriented Principles are common in all Object Oriented Programming Languages But their syntaxes differs from one Object Oriented Programming Language to another Object Oriented Programming Language.



lst=[10,"Rossum",45,67,89,"Python"] here 'lst' is called object

=> here lst is an object of <class, 'list'> here 'list' is one of the pre-defined class

My requirement

=> I want find total marks--no findtotal()

object---class name--list

=>I want decide the grade of the student--no----- decidegrade()

=>I want to display marks memo=====> no---displaymemo()

Fullfilling the need of Universal Properties

But unable to fullfill the needs of Specific Requirements/ Customized Requirements---U Must develop UR OWN CLASS with UR OWN FUNCTIONS

Procedure Orientation

sub program of main program---Function

Object Orientation

sub program of main program---Method

==========

1. classes

==========

=>The purpose of classes concept is that "To develop Programmer Defined Datatype
and Real Time Application with Customized Requirements".
=>The main purpose of developing Programmer Defined Datatype is that to store
mutiple values and performs Customized Operations by using Programmer-defined Functions (called Methods).
=>To develop Programmer Defined Datatype and Real Time Applications with Customized Requirements with classes , we must use a keyword called "class".
=>In Object Oriented Programming, Every Program must starts with a concept called
class. Without classes concept we can't develop a single program.
=>Every Class Name is considered as "programmer-defined data type".

Definition of Class:

=>A class is a collection of Data Members(Instance Data Members and Class Level Data Members) and Methods (Instance Methods, Class Level Methods and Static Methods).
=>Once we define a class, there is no memory space created for Data Members and Methods(bcoz Class definition is treated as specification) but whose memory space is created when we create an Object w.r.t Class Name.(Most Important)
=>Hence Every Program in Python must starts with Classes Concept and data can be stored in the form of objects and it can be created w.r.t Class Name

==========

Syntax for defining a class

==========

=>We know that every Program of Python by using OOPs must starts with the concept of classes.
=>The Syntax for defining a class is shown bellow.

```
class  <clsname>:
       Class  Level Data Members
       def  instancemethodname(self,list of formal params if any ):
                ------------------------------
              Block of stattaments--specific Operations
              Specify Instance Data members
              ------------------------------
       @classmethod
       def  classlevelmethodname(cls,list of formal params if any ):
             ------------------------------
            Block of statements--class level operations
            Specify Class  Level Data Members
            ------------------------------
       @staticmethod
       def  staticmethodname(list of formal params if any):
            ------------------------------
```

```
                                    Block of statements--Utility Operations
                                    -------------------------------
```

Types of Data Members in class

Types of Methods in class

```
                ========================================
                        Types of Data Members in class
                 ======================================
=>In a class of Python, we have two types of Data Members. They are
                 1) Instance Data Members
                 2) Class Level Data Members

-------------------------------------
1) Instance Data Members:
-------------------------------------
=>Instance Data Members are those whose memory space is created each and
every
    time when an object is created.
=>Instance Data Members are used for storing Specfic Values / Perticular
Values,
    which are suitable for one programmer.
=>Instance Data Members must be ACCESSED / PROCESSED w.r.t Object Name
    (Or) self.
                          Object Name.Instance Data Member Name
=>Instance Data Members MUST SPECIFIED in two places. They are
                        a) through object name
                        b) though Instance Methods

----------------------------------------
2) Class Level Data Members:
----------------------------------------
=>Class Level  Data Members are those whose memory space is created only
once .
=>Class Level  Data Members are used for storing Common Values
    which are suitable for all objects
=>IClass Level  Data Members must be ACCESSED / PROCESSED w.r.t Class Name
Object Name (Or) self (OR) cls
=>Syntax:-
                        Class Name.Class Level Data Member Name
                                      (OR)
                          Object Name.Class Level Data Member Name
                                      (OR)
                        self.Class Level Data Member Name
                                      (OR)
                        cls.Class Level Data Member Name
=>Class Level Data Members MUST SPECIFIED in two places. They are
                        a) through Inside class  definition
                        b) though Inside of Class Level Methods
```

```
                 ========================================
                          Types of Methods in class
                 ========================================
=>In a class of python, we can have 3 types of methods. They are
```

```
                    1. Instance Methods
                    2. Class Level Methods
                    3. Static Methods
-------------------------------------------------------------------------
-------------
1. Instance Methods
-------------------------------------------------------------------------
------------
=>Instance Methods are used for performing specific Operations on objects
and
    Intance methods are also called Object Level Methods.
=>Programatically, Instance Methods always takes "self" as First Formal
Parameter for storing address(id) / reference of object.
=>Syntax:-
              def  instancemethodname(self , list of formal params if any
):
                    ------------------------------
                    Block of stattaments--specific Operations
                    Specify Instance Data members
                    ------------------------------
=>In Python Programming, all Instance Methods must be accessed
  w.r.t Object Name (or) self.
              ObjectName.Instance Method Name()
                              (OR)
              self.Instance Method Name()
-------------------------------------------------------------------------
-------------
2. Class Level Methods
-------------------------------------------------------------------------
-------------
#This Program stores stno,sname, marks and course with OOPS
#studentex1.py
class Student:pass  # Here Student is called Class Name and treated as
Prog-def Data type.


#main program
s1=Student()
print("Id of s1=",id(s1))
print("content of s1 before adding Data members=",s1.__dict__) # {  }
print("Number of Values in s1=", len(s1.__dict__))# 0
print("-"*40)
#adding Instance Data memebers through an object
s1.stno=10
s1.sname="RS"
s1.marks=88.88
print("Id of s1=",id(s1))
print("content of s1 after adding Data members=",s1.__dict__) # { -------
}
print("Number of Values in s1=", len(s1.__dict__))# 3
print("-------------OR-------------------")
print("Student Number=",s1.stno)
print("Student Name=", s1.sname)
print("Student Marks=", s1.marks
```

```
#This Program stores stno,sname, marks and course with OOPS
#studentex2.py
class Student:
       crs="PYTHON"  # Here crs is called Class Level Data Member
       addr="HYD"  # Here addr is called Class Level Data Member
#main program
s1=Student()
s2=Student()
#add the Instance  Data Members to s1
s1.stno=100
s1.sname="RS"
s1.marks=23.45
#add the Instance  Data Members to s2
s2.stno=200
s2.sname="JG"
s2.marks=11.11
print("-"*40)
print("Content of s1:")
print("-"*40)
print("Student Number:{}".format(s1.stno))
print("Student Name:{}".format(s1.sname))
print("Student Marks:{}".format(s1.marks))
print("Student Course:{}".format(Student.crs))
print("Student Lives in: {}".format(Student.addr))
print("-"*40)
print("Content of s2:")
print("-"*40)
print("Student Number:{}".format(s2.stno))
print("Student Name:{}".format(s2.sname))
print("Student Marks:{}".format(s2.marks))
print("Student Course:{}".format(Student.crs))
print("Student Lives in: {}".format(Student.addr))
print("-"*40)
```

```
#This Program stores stno,sname, marks and course with OOPS
#studentex3.py
class Student:
       crs="PYTHON"  # Here crs is called Class Level Data Member
       addr="HYD"  # Here addr is called Class Level Data Member

#main program
s1=Student()
s2=Student()
#add the Instance  Data Members to s1 by reading the data from key board
s1.stno=int(input("Enter Student Number for First Student object:"))
s1.sname=input("Enter Student Name for First Student object:")
s1.marks=float(input("Enter Student Marks for First Student object:"))
print("-"*60)
#add the Instance  Data Members to s2 by reading the data from key board
s2.stno=int(input("Enter Student Number for Second Student object:"))
s2.sname=input("Enter Student Name for Second Student object:")
s2.marks=float(input("Enter Student Marks for Second Student object:"))
print("-"*40)
```

```
print("Content of s1:")
print("-"*40)
print("Student Number:{}".format(s1.stno))
print("Student Name:{}".format(s1.sname))
print("Student Marks:{}".format(s1.marks))
print("Student Course:{}".format(Student.crs))
print("Student Lives in: {}".format(Student.addr))
print("-"*40)
print("Content of s2:")
print("-"*40)
print("Student Number:{}".format(s2.stno))
print("Student Name:{}".format(s2.sname))
print("Student Marks:{}".format(s2.marks))
print("Student Course:{}".format(Student.crs))
print("Student Lives in: {}".format(Student.addr))
print("-"*40)
```

```
#This Program stores stno,sname, marks and course with OOPS by using
methods
#studentex4.py
class Student:
       def  getstudentdetails(self):#here self is called Impilcit object
for current class object
              print("Id of self in getstudentdetails()=",id(self))




#main program
s1=Student()
print("Id of s1 in main=",id(s1))
s1.getstudentdetails()
print("-------------------------------------------------")
s2=Student()
print("Id of s2 in main=",id(s2))
s2.getstudentdetails()
```

```
#This Program stores stno,sname, marks and course with OOPS by using
methods
#studentex5.py
class Student:
       crs="PYTHON"  # class Level data member
       def  getstudentdetails(self):#here self is called Impilcit object
for current class object
              print("-"*60)
              self.sno=int(input("Enter Student Number:"))
              self.sname=input("Enter Student Name:")
              self.marks=float(input("Enter Student Marks:"))
              print("-"*60)
       def dispstudentdetails(self):
              print("-"*60)
              print("Student Number:{}".format(self.sno))
              print("Student Name:{}".format(self.sname))
```

```
                print("Student marks:{}".format(self.marks))
                print("Student Course:{}".format(Student.crs))
                print("-"*60)
#main program
s1=Student()   # create an object
print("Enter First Student Object Information")
s1.getstudentdetails()
s2=Student()  # create an object
print("Enter Second Student Object Information")
s2.getstudentdetails()
print("First Student Object Information")
s1.dispstudentdetails()
print("Second Student Object Information")
s2.dispstudentdetails()
```

```
#This Program stores stno,sname, marks and course with OOPS by using
methods
#studentex6.py
class Student:
        crs="PYTHON"  # class Level data member
        def  getstudentdetails(self):#here self is called Impilcit object
for current class object
                print("-"*60)
                self.sno=int(input("Enter Student Number:"))
                self.sname=input("Enter Student Name:")
                self.marks=float(input("Enter Student Marks:"))
                print("-"*60)
        def dispstudentdetails(self):
                print("-"*60)
                print("Student Number:{}".format(self.sno))
                print("Student Name:{}".format(self.sname))
                print("Student marks:{}".format(self.marks))
                print("Student Course:{}".format(Student.crs))
                print("-"*60)
#main program
s1=Student()   # create an object
print("Enter First Student Object Information")
s1.getstudentdetails()
print("First Student Object Information")
s1.dispstudentdetails()

s2=Student()  # create an object
print("Enter Second Student Object Information")
s2.getstudentdetails()
print("Second Student Object Information")
s2.dispstudentdetails()
```

```
#This Program stores stno,sname, marks and course with OOPS by using
methods
#studentex7.py
class Student:
        crs="PYTHON"  # class Level data member
        def  getstudentdetails(self):#here self is called Impilcit object
for current class object
                print("-"*60)
```

```
                self.sno=int(input("Enter Student Number:"))
                self.sname=input("Enter Student Name:")
                self.marks=float(input("Enter Student Marks:"))
                print("-"*60)
                self.dispstudentdetails() # calling another instance method
        def dispstudentdetails(self):
                print("-"*60)
                print("Student Number:{}".format(self.sno))
                print("Student Name:{}".format(self.sname))
                print("Student marks:{}".format(self.marks))
                print("Student Course:{}".format(Student.crs))
                print("-"*60)
#main program
s1=Student()   # create an object
print("Enter First Student Object Information")
s1.getstudentdetails()
s2=Student()  # create an object
print("Enter Second Student Object Information")
s2.getstudentdetails()
```

Types of Methods in class (Fully Completed)

programs

```
                    ========================================
                              Types of Methods in class
                    ========================================
=>In a class of python, we can have 3 types of methods. They are
                      1. Instance Methods
                      2. Class Level Methods
                      3. Static Methods
-------------------------------------------------------------------------
-------------
1. Instance Methods
-------------------------------------------------------------------------
------------
=>Instance Methods are used for performing specific Operations on objects
and
    Intance methods are also called Object Level Methods.
=>Programatically, Instance Methods always takes "self" as First Formal
Parameter for storing address(id) / reference of object.
=>Syntax:-
                def  instancemethodname(self , list of formal params if any
):
                      -------------------------------
                      Block of stattaments--specific Operations
                      Specify Instance Data members
                      -------------------------------
=>In Python Programming, all Instance Methods must be accessed
  w.r.t Object Name (or) self.
                ObjectName.Instance Method Name()
```

```
                              (OR)
              self.Instance Method Name()
-------------------------------------------------------------------------
-------------
2. Class Level Methods
-------------------------------------------------------------------------
-------------
=>Class Level Methods are used for performing Common Operations on Class
Level
    Data Members and they named as Class  Level Methods and we can specify
/ define Class Level Data Members.
=>The Class Level Method Definition must preceded with a pre-defined
decorator
    called @classmethod and it must take the First Formal parameter as
"cls".
=>Syntax:-
                              @classmethod
                              def  classlevelmethodname(cls,list of formal
params if any ):
                                    ---------------------------------
                                   Block of statements--class level
operations
                                   Specify Class  Level Data Members
                                   ---------------------------------
=>In Python Programming, all Class Level Methods must be accessed
  w.r.t Class Name (or) cls (or) Object Name (or) self.

              Class Name. Class Level Method Name()
                              (OR)
              ObjectName.Class Level Method Name()
                              (OR)
              cls.Class Level Method Name()
                              (OR)
              self.Class Level Method Name()

-------------------------------------------------------------------------
-------------
3. Static Methods
-------------------------------------------------------------------------
------------
=>Static Methods are used for performing Universal / Utility Operations.
=>=>The Static Methods Definition must preceded with a pre-defined
decorator
    called @staticmethod and it never takes the First Formal parameter as
"cls or self".

=>Syntax:-
                              @staticmethod
                              def  staticmethodname(list of formal params
if any ):
                                    ---------------------------------
                                   Block of statements--Utility / Universal
operations
                                   ---------------------------------
```

=>=>In Python Programming, all Static Methods must be accessed w.r.t Class Name (or) cls (or) Object Name (or) self.

```
Class Name. Static Method Name()
                (OR)
ObjectName.Static Method Name()
                (OR)
cls.Static Method Name()
                (OR)
self.Static Method Name()
```

```
#This program demonstrates the specification of Class Level Data Members
#EmployeeEx1.py
class Employee:
        @classmethod
        def  getcompname(cls):    # Class Level Method
                cls.cname="InfoSys"
        @classmethod
        def getcompaddr(cls):      # Class Level Method
                Employee.city="HYD"
        def getempdet(self):  # Instance Method
                print("-"*40)
                self.eno=int(input("Enter Employee Number:"))
                self.ename=input("Enter Employee Name:")
                print("-"*40)
        def dispempdet(self):  # Instance Method
                print("-"*40)
                print("Employee Number:{}".format(self.eno))
                print("Employee Name:{}".format(self.ename))
                print("Employee Company Name:{}".format(Employee.cname))
                print("Employee Company address:{}".format(Employee.city))
                print("-"*40)

#main program
Employee.getcompname() # calling class level method
Employee.getcompaddr() # calling class level method
eo1=Employee()
eo2=Employee()
eo1.getempdet()  # calling Instance Method
eo2.getempdet() # calling Instance Method
eo1.dispempdet() # calling Instance Method
eo2.dispempdet() # calling Instance Method
```

```
#This program demonstrates the specification of Class Level Data Members
#EmployeeEx2.py
class Employee:
        @classmethod
        def  getcompname(cls):    # Class Level Method
                cls.cname="InfoSys"
                Employee.getcompaddr() # calling class level method
                #A Class Level Method can call another Class Level Method but
not Instance Method
        @classmethod
        def getcompaddr(cls):      # Class Level Method
                Employee.city="HYD"
        def getempdet(self):  # Instance Method
                print("-"*40)
```

```
                self.eno=int(input("Enter Employee Number:"))
                self.ename=input("Enter Employee Name:")
                print("-"*40)
        def dispempdet(self):  # Instance Method
                print("-"*40)
                print("Employee Number:{}".format(self.eno))
                print("Employee Name:{}".format(self.ename))
                print("Employee Company Name:{}".format(Employee.cname))
                print("Employee Company address:{}".format(Employee.city))
                print("-"*40)

#main program
Employee.getcompname() # calling class level method
eo1=Employee()
eo2=Employee()
eo1.getempdet()  # calling Instance Method
eo2.getempdet() # calling Instance Method
eo1.dispempdet() # calling Instance Method
eo2.dispempdet() # calling Instance Method
```

```
#This program demonstrates the specification of Class Level Data Members
#EmployeeEx3.py
class Employee:
        @classmethod
        def  getcompname(cls):    # Class Level Method
                cls.cname="InfoSys"
                cls.getcompaddr() # calling class level method
                #A Class Level Method can call another Class Level Method but
not Instance Method
        @classmethod
        def getcompaddr(cls):      # Class Level Method
                Employee.city="HYDERABAD"
        def getempdet(self):  # Instance Method
                print("-"*40)
                self.eno=int(input("Enter Employee Number:"))
                self.ename=input("Enter Employee Name:")
                print("-"*40)
        def dispempdet(self):  # Instance Method
                print("-"*40)
                print("Employee Number:{}".format(self.eno))
                print("Employee Name:{}".format(self.ename))
                print("Employee Company Name:{}".format(Employee.cname))
                print("Employee Company address:{}".format(Employee.city))
                print("-"*40)

#main program
Employee.getcompname() # calling class level method
eo1=Employee()
eo2=Employee()
eo1.getempdet()  # calling Instance Method
eo2.getempdet() # calling Instance Method
eo1.dispempdet() # calling Instance Method
eo2.dispempdet() # calling Instance Method
```

```
#This program demonstrates the specification of Class Level Data Members
#EmployeeEx4.py
class Employee:
        @classmethod
```

```
        def  getcompname(cls):    # Class Level Method
                cls.cname="InfoSys"
                #A Class Level Method can call another Class Level Method but
not Instance Method
        @classmethod
        def getcompaddr(cls):      # Class Level Method
                Employee.city="HYDERABAD"
        def getempdet(self):  # Instance Method
                print("-"*40)
                self.eno=int(input("Enter Employee Number:"))
                self.ename=input("Enter Employee Name:")
                print("-"*40)
        def dispempdet(self):  # Instance Method
                Employee.getcompaddr() # calling Class Level Method
                print("-"*40)
                print("Employee Number:{}".format(self.eno))
                print("Employee Name:{}".format(self.ename))
                print("Employee Company Name:{}".format(Employee.cname))
                print("Employee Company address:{}".format(Employee.city))
                print("-"*40)

#main program
Employee.getcompname() # calling class level method
eo1=Employee()
eo2=Employee()
eo1.getempdet()  # calling Instance Method
eo2.getempdet() # calling Instance Method
eo1.dispempdet() # calling Instance Method
eo2.dispempdet() # calling Instance Method
```

```
#This program demonstrates the specification of Class Level Data Members
#EmployeeEx5.py
class Employee:
        @classmethod
        def  getcompname(cls):    # Class Level Method
                cls.cname="InfoSys"
                #A Class Level Method can call another Class Level Method but
not Instance Method
        @classmethod
        def getcompaddr(cls):      # Class Level Method
                Employee.city="HYDERABAD"
        def getempdet(self):  # Instance Method
                print("-"*40)
                self.eno=int(input("Enter Employee Number:"))
                self.ename=input("Enter Employee Name:")
                print("-"*40)
        def dispempdet(self):  # Instance Method
                self.getcompaddr() # calling Class Level Method
                print("-"*40)
                print("Employee Number:{}".format(self.eno))
                print("Employee Name:{}".format(self.ename))
                print("Employee Company Name:{}".format(Employee.cname))
                print("Employee Company address:{}".format(Employee.city))
                print("-"*40)

#main program
Employee.getcompname() # calling class level method
eo1=Employee()
```

```
eo2=Employee()
eo1.getempdet()  # calling Instance Method
eo2.getempdet() # calling Instance Method
eo1.dispempdet() # calling Instance Method
eo2.dispempdet() # calling Instance Method
```

```
#This program demonstrates the specification of Class Level Data Members
#EmployeeEx6.py
class Employee:
        @classmethod
        def  getcompname(cls):    # Class Level Method
                cls.cname="InfoSys"
                #A Class Level Method can call another Class Level Method but
not Instance Method
        @classmethod
        def getcompaddr(cls):      # Class Level Method
                Employee.city="HYDERABAD"
        def getempdet(self):  # Instance Method
                print("-"*40)
                self.eno=int(input("Enter Employee Number:"))
                self.ename=input("Enter Employee Name:")
                print("-"*40)
        def dispempdet(self):  # Instance Method
                Employee.getcompname() # calling class level method
                self.getcompaddr() # calling Class Level Method
                print("-"*40)
                print("Employee Number:{}".format(self.eno))
                print("Employee Name:{}".format(self.ename))
                print("Employee Company Name:{}".format(Employee.cname))
                print("Employee Company address:{}".format(Employee.city))
                print("-"*40)

#main program
eo1=Employee()
eo2=Employee()
eo1.getempdet()  # calling Instance Method
eo2.getempdet() # calling Instance Method
eo1.dispempdet() # calling Instance Method
eo2.dispempdet() # calling Instance Method
```

```
#This Program demonstrates the static methods usage
#staticmethodex1.py
class Employee:
        def  getempdet(self):
                self.eno=int(input("Enter Employee Number:"))
                self.ename=input("Enter Employee Name:")

class Student:
        def  getstuddet(self):
                self.stno=int(input("Enter Student Number:"))
                self.sname=input("Enter Student Name:")
                self.cname=input("Enter Student College Name:")
# I decided to develop  a class name with a Method to display the content of
any object (Utility Method / Universal Operation)
class ObjectDisplay:
        @staticmethod
        def  displaydata(obj):
                print("-"*40)
```

```
                for k,v in obj.__dict__.items():
                        print("\t{}-->{}".format(k,v))
                print("-"*40)

#main program
eo=Employee()
eo.getempdet()
so=Student()
so.getstuddet()
#calling static method of ObjectDisplay class name
print("Employee Details")
ObjectDisplay.displaydata(eo) # calling static method w.r.t Correspondiong
class name
print("Student Details")
ObjectDisplay.displaydata(so)
```

```
#This Program demonstrates the static methods usage
#staticmethodex2.py
class Employee:
        def  getempdet(self):
                self.eno=int(input("Enter Employee Number:"))
                self.ename=input("Enter Employee Name:")

class Student:
        def  getstuddet(self):
                self.stno=int(input("Enter Student Number:"))
                self.sname=input("Enter Student Name:")
                self.cname=input("Enter Student College Name:")
# I decided to develop  a class name with a Method to display the content of
any object (Utility Method / Universal Operation)
class ObjectDisplay:
        @staticmethod
        def  displaydata(obj):
                print("-"*40)
                for k,v in obj.__dict__.items():
                        print("\t{}-->{}".format(k,v))
                print("-"*40)

#main program
eo=Employee()
eo.getempdet()
so=Student()
so.getstuddet()
#calling static method of object of ObjectDisplay class name
od=ObjectDisplay()
print("Employee Details")
od.displaydata(eo) # calling static method w.r.t Correspondiong class objct
name
print("Student Details")
od.displaydata(so)
```

```
#This Program cal area  and perimeter of Circle
#Circle.py
class Circle:
        @classmethod
        def  getPI(cls):
                cls.PI=3.14
                return cls.PI
```

```
        def  readrad(self):
               self.r=float(input("Enter Radious:"))
        def area(self):
               self.ac=Circle.getPI()*self.r**2
               print("Area of Circle={}".format(self.ac))
        def peri(self):
               self.pc=2*self.getPI()*self.r
               print("Peri of Circle={}".format(self.pc))

#main program
co=Circle()
co.readrad()
co.area()
co.peri()
```

```
#CircleDemo.py
from CircleWithModules import Circle
co=Circle()
co.readrad()
co.area()
co.peri()
```

```
#This Program cal area  and perimeter of Circle
#CircleWithModules.py--File Name and treated as module name
class Circle:
        @classmethod
        def  getPI(cls):
               cls.PI=3.14
               return cls.PI
        def  readrad(self):
               self.r=float(input("Enter Radious:"))
        def area(self):
               self.ac=Circle.getPI()*self.r**2
               print("Area of Circle={}".format(self.ac))
        def peri(self):
               self.pc=2*self.getPI()*self.r
               print("Peri of Circle={}".format(self.pc))
```

Programs on Classes and Objects

```
#This program computes sum of two numbers by using classes and objects.
#sum.py
class Sum:
        def  getvalues(self):
               self.a=float(input("Enter First Value:"))
               self.b=float(input("Enter Second Value:"))
        def findsum(self):
               self.c=self.a+self.b
        def dispvalues(self):
               print("sum({},{})={}".format(self.a,self.b,self.c))

#main program
so=Sum()
so.getvalues()
so.findsum()
so.dispvalues()
```

```
#This program computes sum of two numbers by using classes and objects.
#sum1.py
```

```
class Sum:
	def  getvalues(self):
		self.a=float(input("Enter First Value:"))
		self.b=float(input("Enter Second Value:"))
		self.findsum()  # calling Instance Method
	def findsum(self):
		self.c=self.a+self.b
		self.dispvalues()  # calling Instance Method
	def dispvalues(self):
		print("sum({},{})={}".format(self.a,self.b,self.c))

#main program
so=Sum()
so.getvalues()
```

```
#This Program reads two numerical values  and arithmetic operator from KBD
and find operation result.
#aop.py
class Values:
	def  getvalues(self):
		self.a=float(input("Enter First Value:"))
		self.b=float(input("Enter Second Value:"))
		self.op=input("Enter any arithmetic operator:")

class Calculator:
	@staticmethod
	def  compute(obj):
		try:
			vo.getvalues()
			match(obj.op):
				case
"+":print("sum({},{})={}".format(obj.a,obj.b,obj.a+obj.b))
				case "-
":print("sub({},{})={}".format(obj.a,obj.b,obj.a-obj.b))
				case
"*":print("mul({},{})={}".format(obj.a,obj.b,obj.a*obj.b))
				case
"/":print("div({},{})={}".format(obj.a,obj.b,obj.a/obj.b))
				case "//":print("floor
div({},{})={}".format(obj.a,obj.b,obj.a//obj.b))
				case
"%":print("mod({},{})={}".format(obj.a,obj.b,obj.a%obj.b))
				case
"**":print("pow({},{})={}".format(obj.a,obj.b,obj.a**obj.b))
				case _: print("\n {} is not a Arithmetic
Operator:".format(obj.op))
		except ValueError:
			print("\tDon't Enter strs/alpha-numeric/special
symbols for numerics")

#main program
vo=Values()
Calculator.compute(vo)
```

```
#This Program reads two numerical values  and arithmetic operator from KBD
and find operation result.
#aop1.py
class Values:
       def  getvalues(self):
              try:
                     self.a=float(input("Enter First Value:"))
                     self.b=float(input("Enter Second Value:"))
                     self.op=input("Enter any arithmetic operator:")
              except ValueError:
                     print("\nDon't Enter strs/alpha-numeric/special
symbols for numerics")

class Calculator:
       @staticmethod
       def  compute(obj):
              try:
                     match(obj.op):
                                   case
"+":print("sum({},{})={}".format(obj.a,obj.b,obj.a+obj.b))
                                   case "-
":print("sub({},{})={}".format(obj.a,obj.b,obj.a-obj.b))
                                   case
"*":print("mul({},{})={}".format(obj.a,obj.b,obj.a*obj.b))
                                   case
"/":print("div({},{})={}".format(obj.a,obj.b,obj.a/obj.b))
                                   case "//":print("floor
div({},{})={}".format(obj.a,obj.b,obj.a//obj.b))
                                   case
"%":print("mod({},{})={}".format(obj.a,obj.b,obj.a%obj.b))
                                   case
"**":print("pow({},{})={}".format(obj.a,obj.b,obj.a**obj.b))
                                   case _: print("\n {} is not a Arithmetic
Operator:".format(obj.op))
              except AttributeError:
                     print("U have Missed Operator in ur typing:")
              except ZeroDivisionError:
                     print("\nDon't enter zero for Den...")
#main program
vo=Values()
vo.getvalues()
Calculator.compute(vo)
```

```
#this program reads all the records from employee table of oracle data
base
#employeedatabase.py
import cx_Oracle
class Employee:
       def  gettablename(self):
              self.tname=input("Enter table name:")
       def getrecords(self):
              self.gettablename() # calling instance method
              try:
                     con=cx_Oracle.connect("scott/tiger@localhost/orcl")
                     cur=con.cursor()
```

```
                        cur.execute("select * from %s" %self.tname)
                        #print col names
                        print("-"*60)
                        for colname in [ colnames[0] for colnames in
cur.description]:
                                print("{}".format(colname),end="   ")
                        print()
                        print("-"*60)
                        #display records
                        records=cur.fetchall()
                        for record in records:
                                for val in record:
                                        print("{}".format(val),end="  ")
                                print()
                        print("-"*60)
                except cx_Oracle.DatabaseError as db:
                        print("Problem in database :",db)

#main program
eo=Employee()
eo.getrecords()
```

Object concept in Python

```
                ====================================
                        objects  in Python
                ====================================
Importance of Object:
------------------------------
=>When we define a class , Memory space is not created for Data Members
and
    Methods But whose Memory Space is created when e create an object.
=>When we define a class and class name can be treated as Data Type but we
can't
    store the data. To Store the data , we must create an object w.r.t
Class Name.
=>To do any data processing, we must create an object.
=>To create an object, there must exist a class definition otherwise we
get error.
------------------------------
Definition of an object:
------------------------------
=>Instance of a class is called object. ( Instance is nothing but
allocating sufficient memory space for Data Members and Methods ).
------------------------------
Syntax for creating an object:
--------------------------------------
       objectname=ClassName()
----------------------------------------
Example: create an object of  Employee
              eo=Employee()
Example: create an object of  Student
              so=Student()
Example: create an object of  Book
```

```
            bo=Book()
```

Index of Network Programming

Network Programming Concept , definitions and steps

Network Programming module name

```
                ===================================
                            Network Programming
                 ===================================
Index:
---------
=>Purpose of Network Programming--->(Django->MVT->Web Application
Framework)
=>Definition of Network
=>Types of Applications / Programs we develop in Network Programming
                a) Server Side Application
                b) Client Side Application
=>What is DNS / IP Address, Port Number
=>Steps for Developing  Server Side Application
=>Stesp for Developing  Client Side Application
=>Pre-defined Module for Network Programming
                a) socket <----Module Name
=>Programming Examples.
```

```
                    ================================
                              Network Programming
                     ================================
=>The Purpose of Network Programming is that "To share the data /
information between two machines "
=>In Network Programming, we develop two types of Programs. They are
                1) Server Side Program
                2) Client Side Program
-----------------------------------------------
1) Definition of Server Side Program:
-----------------------------------------------
=>A Server Side Program is one which accept Client Request, Process the
Client Request and Gives Response back to Client Side Program.
-----------------------------------------------
2) Definition of Client Side Program:
-----------------------------------------------
=>A Client Side Program is one, which sends request to the server side
propgram and gets the response from Server Side Program.
-------------------------------------------------------------
3) Definition of DNS ( Domain Naming Service):
-------------------------------------------------------------------------
--------------------------
=>The DNS is the name of Physical Machine where Server Side Program
Resides
=>The default DNS of every Computer is "localhost"
-------------------------------------------------------------------------
--------------------------
4) Definition of IP Address ( Internet Protocal Address):
```

=>The IP Address is the name of Address of the Physical Machine where Server Side Program Resides.
=>The default IP Address of every computer is 127.0.0.1(Loop Back address)
---
5) Definition of Port Number:

=>A Port Number is a Logical numerical Id, where Server Side Program is Running.

Steps for developing Server Side Application:
---
1) Every Server Side Program must run at Certain DNS / IP Address and Unique Port
    Number.
2) Every Server Side Program ACCEPT the Client Side Program Request,
3) Every Server Side Program must READ client Side Program request, PROCESS
    the client side program request (decode())
4) Server Side Program SEND the response back to Client Side Program (encode())

Note:- As long as Client Side Program makes request , The Server Side Program pefroms steps(2), (3) and (4)

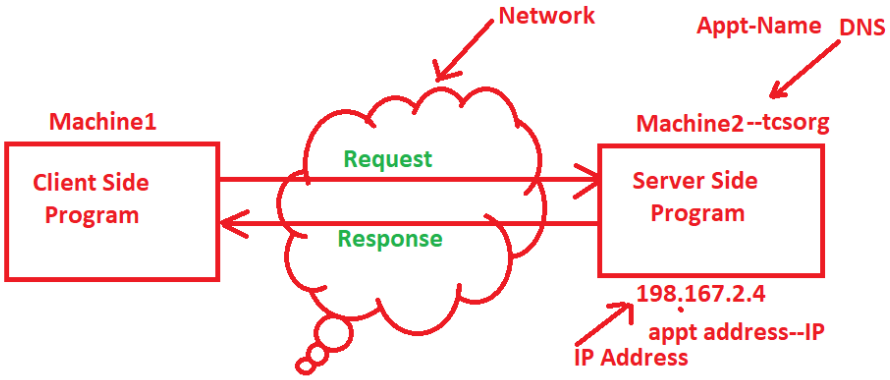
Steps for developing Client Side Application:
---
1) Every Client Side Program must get CONNECTION from Server Side Program by
    passing (DNS / IP Address , Portno)
2) Every Client Side Program must SEND a request to the Server Side Program
    (encode())
3) Every Client Side Program must RECEIVE the response from Server Side Program
   (decode)

Note:- When Client Side Program want make many request to the server side program then it has repeat steps (2) and (3)
==========X==========

Pre-defined Module for Network Programming

=>To develop Network Programming Applications, we have a pre-defined Inbuilt mdule called "socket".
=>This Module "socket" contains Various Functions for Server Side Program and Client Side Program Development.

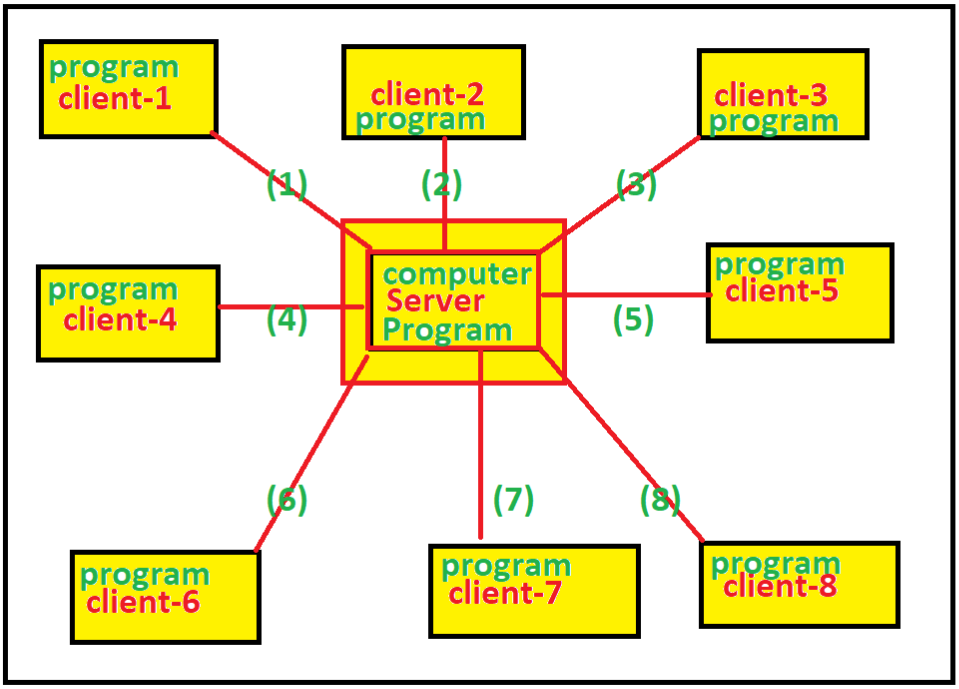
```
-------------------------------------------------------------------------
--
=>Functions in "socket" module used Server Side Program
-------------------------------------------------------------------------
--
1) socket():
----------------------
=>This Function used for creating an object socket class at Both Server
side and Client Side Programs. So that It acts as Bi-Directional
Communication Entity.
Syntax:-    varname=socket.socket()
Example:    s=socket.socket()
-------------------------------------------------------------------------
-----------------
2) bind():
-------------------------------------------------------------------------
---------------
=>This Function is used making the Program as server side program by
binding at Certain DNS and Port Number
=>Syntax:-   socketobj.bind( (DNS/IPAddress, portno) )
=>Examples:     s.bind( ("localhost", 8558))
                    print("Server Side is Ready to accept Reauests:")
-------------------------------------------------------------------------
-------
3) listen():--Tomorrow

4) accept()
5) recv()<--decode()
6) send()-->encode()
-------------------------------------------------------------------------
--
=>Functions in "socket" module used Client Side Program
-------------------------------------------------------------------------
--
1) socket()
2) connect()
3) send()--->encode()
4) recv()<---decode()
```



Django

Server
- Web Server
  - Tomcat
  - JBoss
  - Jetty
- Application Server
  - JBoss
  - web logic
  - web sphere
  - Glassfish Srver

Physical Networking Enviroment



Network Programming module name

Programs

```
                    ================================================
                        Pre-defined Module for Network Programming
                    ================================================
=>To develop Network Programming Applications, we have a pre-defined
Inbuilt mdule called "socket".
=>This Module "socket" contains Various Functions for Server Side Program
and Client Side Program Development.
-------------------------------------------------------------------------
--
=>Functions in "socket" module used Server Side Program
-------------------------------------------------------------------------
--
1) socket():
----------------------
=>This Function used for creating an object socket class at Both Server
side and Client Side Programs. So that It acts as Bi-Directional
Communication Entity.
Syntax:-    varname=socket.socket()
Example:    s=socket.socket()
-------------------------------------------------------------------------
-----------------
2) bind():
-------------------------------------------------------------------------
---------------
=>This Function is used making the Program as server side program by
binding at Certain DNS and Port Number
=>Syntax:-   socketobj.bind( (DNS/IP Address, portno) )
=>Examples:     s.bind( ("localhost", 8558))
                           print("Server Side is Ready to accept
Requests:")
                              (OR)
=>Examples:     s.bind( ("127.0.0.1", 8558))
                           print("Server Side is Ready to accept
Requests:")
-------------------------------------------------------------------------
-------
3) listen():--
-------------------------------------------------------------------------
-------
=>This Function is used for configuring the server side program in such a
way that to how many client, The server Side Program can take requests and
Responds.
Syntax:-     socketobj.listen(no. of client side Programs)
Example:     s.listen(2)
-------------------------------------------------------------------------
-------
4) accept()
-------------------------------------------------------------------------
-------
=>This Function is used for accepting the request of Client Side Program
```

```
=>Syntax:           conn, addr =socketobj.accept()
       =>Here "conn" is an object, which makes us understand, Server Gave
the connection conformation to client side program
       =>here "addr" is an object and it contains address of Client Side
program.
-------------------------------------------------------------------------
---------------------------
5) recv()<--decode()
-------------------------------------------------------------------------
---------------------------
=>This Function is used for receiving the client side data at Server Side
Program and receives the server side program data at client side .
=>Syntax:-          varname=socketobj.recv(1024 / 2048/ 4098).decode()---
-client side
                     Here varname is containing  server data in the form
str at client side
                            (OR)

=>Syntax:-          varname=conn.recv(1024 / 2048/ 4098).decode()----
server side
                      Here varname is containing client  data in the form
str at server side

-------------------------------------------------------------------------
---------------------------------
6) send()-->encode():
---------------------------------
=>This Function used for sending request to the server by client  /
sending response to the client by server.
=>Syntax:-       socketobj.send(str.encode())  # client side program
                            (or)
                     conn.send(str.encode() )  # server side program
-------------------------------------------------------------------------
--
=>Functions in "socket" module used Client Side Program
-------------------------------------------------------------------------
--
1) socket():
-----------------------------
=>This Function used for creating an object socket class at Both Server
side and Client Side Programs. So that It acts as Bi-Directional
Communication Entity.
Syntax:-    varname=socket.socket()
Example:    s=socket.socket()
-------------------------------------------------------------------------
-------------
2) connect()
----------------------
=>This function is used for getting the connection from server side
program.
=>Syntax:-         socketobject.connect( ("DNS/IP address",portno)  )
=>Example:-      socketobject.connect( ("localhost",8558))
              print("Client Side Program got connection from Server Side
Progam")
```

```
-------------------------------------------------------------------------
-------------
3) send()--->encode()
----------------------------------------
=>This Function used for sending request to the server by client  /
sending response to the client by server.
=>Syntax:-       socketobj.send(str.encode())  # client side program
                              (or)
                      conn.send(str.encode() )  # server side program
-------------------------------------------------------------------------
-------------
4) recv()<---decode():
-------------------------------------------------------------------------
--------
=>This Function is used for receiving the client side data at Server Side
Program and receives the server side program data at client side .
=>Syntax:-           varname=socketobj.recv(1024 / 2048/ 4098).decode()---
-client side
                      Here varname is containing  server data in the form
str at client side
                              (OR)

=>Syntax:-           varname=conn.recv(1024 / 2048/ 4098).decode()----
server side
                       Here varname is containing client  data in the form
str at server side
============================X==================================
#SquareClient.py-----Program-(B)
import socket
s=socket.socket()
#step-1
s.connect( ("localhost",8558))
print("Client Side Program Got Connection from Server Side Program:")
#step-2
n=input("\nEnter a Value for computing its square:")
s.send(n.encode())
#step-3
result=s.recv(1024).decode() # here result is the value coming from server
side program
print("square({})={}".format(n,result))
```

```
#SquareServer.py-----Program-(A)
import socket
#step-1
s=socket.socket()
s.bind( ("localhost",8558))
s.listen(2)
print("Sever Side Program is ready to accept any request of client:")
while(True):
       try:
              clientcon,clientaddr=s.accept()  # step-2
              clientdata=clientcon.recv(1024).decode() # step-3
              print("Val of client at Server={}".format(clientdata))
```

```
                val=float(clientdata)
                result=val**2 #process
                clientcon.send(str(result).encode())  #step-4
        except ValueError:
                clientcon.send("Don't Enter strs/alpha-
numerics/symbols".encode())
```

Programs on networking

```
#This Program considered as Client Side Program, It Sends the Messages
Server  Side Program and gets Answer as Response by client suide Program
from server side program
#ChatClient.py-----Program-(B)
import socket,sys
while(True):
        s=socket.socket()
        s.connect( ("localhost",9999) )
        csdata=input("Client-->")
        if(csdata.lower()=="quit") :
                s.send("Bye Server, I have some work!".encode())
                sys.exit()
        else:
                s.send(csdata.encode())
                ssdata=s.recv(1024).decode()
                print("Server-->{}".format(ssdata))
```

```
#This Program considered as Server Side Program, It receives the Messages
from Client Side Program and Gives Answer as Response to client suide
Program
#ChatServer.py-----Program-(A)
import socket
s=socket.socket()
s.bind( ("localhost",9999) )
s.listen(1) # step-1
print("Sever Side Program is ready to accept any request of client:")
print("-"*40)
while(True):
        con,addr=s.accept()   # step-2
        csdata=con.recv(1024).decode()#step-3
        print("Client-->{}".format(csdata))
        sdata=input("Server-->")
        con.send(sdata.encode())
```

```
#This program receives Emp Name, Sal,Designation and Comp Name from Server
side program by connecting to employee table
#EmpDataClient.py----Program-(A)
import socket
while(True):
        s=socket.socket()
        s.connect(("127.0.0.1",8888))
        print("CSP Connected to SSP:")
        eno=input("\nEnter Employee Number :")
        s.send(eno.encode())
        emprec=s.recv(1024).decode()
        print("Employee data: {}".format(emprec))
```

```
        ch=input("\nDo u want to continue(yes/no):")
        if(ch=="no"):
                print("Thanks for using this program:")
                break
```

```
#This program receives Emp number from client side program , connect
employee table , reading other details of employee and send to client side
program.
#EmpDataServer.py----Program-(A)
import socket
import cx_Oracle
#step-1
s=socket.socket()
s.bind( ("127.0.0.1",8888) )
s.listen(2)
print("SSP is ready to accept any CSP Request:")
#step-2
while(True):
        try:
                conn,addr=s.accept()
                eno=int(conn.recv(1024).decode())
                #PDBC Code
                oracon=cx_Oracle.connect("scott/tiger@localhost/orcl")
                cur=oracon.cursor()
                cur.execute("select ename,sal,dsg,cname from employee where
eno=%d" %eno)
                emprec=cur.fetchone()
                if(emprec==None):
                        conn.send("Employee Record Does not
Exists:".encode())
                else:
                        conn.send(str(emprec).encode())
        except ValueError:
                conn.send("Don't Enter strs/symbols/alpha-
numerics".encode())
        except cx_Oracle.DatabaseError as db:
                conn.send("Problem in Database:"+str(db).encode
```

Index of Regular Expression in Python
Regular Expression in Python Concept

```
==========================================
                Regular Expression in Python--Most Imp
                ==========================================
Index:
----------
=>Purpose of Regular Expression
=>Definition of Regular Expression
=>Module Name used in Regular Expression
                => re
=>Functions used in "re" module
                1) finditer()
                2) start()
                3) end()
                4) group()
                5) findall()
```

```
              6) search()
=>Programmer-Defined Character Classes
              =>Programming Examples
=>Pre-Defined Character Classes
              =>Programming Examples
=>Quantifiers in Regular Expressions
              =>Programming Examples
=>=>Programming Examples( Combination of Programmer, pre0defined classes
and Quatifiers)
```

```
                    ========================================
                      Regular Expression in Python--Most Imp
                     ========================================
=>Regular Expression is one of Indepenedent Concept of all Programming
    Language.
=>Regular Expressions can be used in PYTHON, java, .Net...etc
=>The purposes of Regular Expression  are:
       1. Regular Expression used in development of Language Compilers and
                  Interpreters.
       2. Regular Expression used in development of Operating System
       3. Regular Expression are used in Pattern Matching
       4. Regular Expression are used in Electronics Circuits
       5. Regular Expression are used in Universal Protocols
-------------------------------------------------------------------------
-----------------------------------
Definition of Regular Expression:
-------------------------------------------------------------------------
-----
=>Regular Expression is one of String Pattern( Combination of
Alphabets,Digits and Special Symbols ) which is used to search in Given
Data for obtaining desired result.
-------------------------------------------------------------------------
-----------------------------------
=>To deal with Regular Expression, we use a pre-defined module called
"re".
```

```
#RegExprex1.py
#This program searches for a word "Python" in given data
import re
GivenData="Python is an oop lang and Python is also Fun Prog lang"
reg="Python"
mattab=re.finditer(reg,GivenData)
print("Type of mattab=",type(mattab))# Type of mattab= <class
'callable_iterator'>
print("-"*40)
for one in mattab:
       print("Start Index:{}   End Index:{}
Value:{}".format(one.start(),one.end(),one.group()) )
print("-"*40)
```

```
#RegExprex2.py
#This program searches for a word "Python" in given data and find number
of occurences.
import re
GivenData="Pythonic is an oop lang and Python is also Fun Prog lang"
```

```
reg="Python"
mattab=re.finditer(reg,GivenData)
noc=0
print("-"*40)
for one in mattab:
        print("Start Index:{}   End Index:{}
Value:{}".format(one.start(),one.end(),one.group()) )
        noc=noc+1
print("-"*40)
print("Number of Occurences of {}={}".format(reg,noc))
print("-"*40)
```

```
#RegExprex3.py
#This program searches for a word "Python" in given data and find number
of occurences.
import re
GivenData="Python is an oop lang and Python is also Fun Prog lang"
reg="Python"
mattab=re.findall(reg,GivenData)
print("type of mattab=",type(mattab))
print(mattab)
print("Number of occurences of {}={}".format(reg,len(mattab)))
```

```
#RegExprex4.py
#This program searches for a word "Python" in given data
import re
GivenData="Python is an oop lang and Python is also Fun Prog lang"
reg="Python"
mattab=re.search(reg,GivenData)
if(mattab!=None):
        print("type of mattab=",type(mattab))
        print(mattab)
else:
        print("{}  does not exist in Given Data:".format(reg))
```

```
import re
GivenData="Python is an oop lang and Python is also Fun Prog lang"
reg="Python"
  mattab  = re.finditer(reg,GivenData)
```

mattable

Start Index	EndIndex	Value
0	6	Python
26	32	Python

```
for one in mattab:
    print("start index={}".format(one.start())
    print("End Index={}".format(one.end())
    print("Matched value={}".format(one.group())
```

Programmer-Defined Character Classes in Regular Expressions

Pre-defined Character Classes in Regular Expressions
Programs

```
==========================================
                    Pre-Defined Character Classes
                ==========================================
=>Pre-Defined Character Classes are already defined python software and
they helps us to prepare search patterns and they are used to search in
the given data for obtaining desired result.
=>The Syntax for applying Pre-Defined Character Classes is shown bellow.

               " \ Pre-defined character class "

=>We have  6 pre-defined Pre-Defined Character Classes. They are

1) \s ----->It searches for space character only
2) \S----->It searches for all except space character
3) \d----->It searches for digit only    (or)  [ 0-9]
4) \D---->It searches for all except digit (or) [^0-9]
5) \w---->It searchers for word character (or ) [A-Za-z0-9]
6) \W---->It searcher for special symbols (or) [^A-Za-z0-9]
=======================X================================

==========================================
                    Programmer-Defined Character Classes
                ==========================================
```

=>Programmer-Defined Character Classes are prepared by Programmers and they helps us to prepare search patterns and they are used to search in the given data for obtaining desired result.
=>The syntax for Programmer-Defined Character Classes is given bellow.
[Search Pattern-Programmer-Defined Character Classes]
=>We can create 14 Programmer-Defined Character Classes. They are

1) [abc]--->It searches for either 'a' or 'b' or 'c' only

2) [^abc]---->It searches for all except 'a' or 'b' or 'c'

3) [a-z]--->It searches for all lower case alphabets only.

4) [^a-z]--->It searches for all except lower case alphabets .

5) [A-Z]--->It searches for all Upper case alphabets only.

6) [^A-Z]--->It searches for all except Upper case alphabets

7) [0-9]----->It searches for all digits only.

8) [^0-9]----->It searches for all except digits.

9) [a-z 0-9]--->It searches for lower case alphabets and digits only

10) [^a-z 0-9]--->It searches for all except lower case alphabets and digits.

11) [A-Za-z]--->It searches for all alphabets (lower and upper) only

12)[^A-Za-z]--->It searches for all alphabets (lower and upper) only
13) [A-Za-z0-9]-->It searches for all alphabets (lower and upper) and digits (except special symbols)

14) [^A-Za-z0-9]-->It searches for all special symbols

```
#RegExpr5.py
#This program for searching either 'a' or 'b' or 'c'
import re
mattab=re.finditer("[abc]", "kAb6&qPch7v@8%LfP3* 6Kwr")
print("-"*50)
for entry in mattab:
       print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
Output
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr5.py
--------------------------------------------------
Start Index:2  end Index=3  value=b
Start Index:7  end Index=8  value=c
--------------------------------------------------   """
```

```
#RegExpr6.py
```

```
#This program for searching  all except  'a' or 'b' or 'c'
import re
mattab=re.finditer("[^abc]", "kAb6&qPch7v@8%LfP3* 6Kwr")
print("-"*50)
for entry in mattab:
       print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
Output
--------------------------------------------------
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr6.py
--------------------------------------------------
Start Index:0  end Index=1  value=k
Start Index:1  end Index=2  value=A
Start Index:3  end Index=4  value=6
Start Index:4  end Index=5  value=&
Start Index:5  end Index=6  value=q
Start Index:6  end Index=7  value=P
Start Index:8  end Index=9  value=h
Start Index:9  end Index=10  value=7
Start Index:10  end Index=11  value=v
Start Index:11  end Index=12  value=@
Start Index:12  end Index=13  value=8
Start Index:13  end Index=14  value=%
Start Index:14  end Index=15  value=L
Start Index:15  end Index=16  value=f
Start Index:16  end Index=17  value=P
Start Index:17  end Index=18  value=3
Start Index:18  end Index=19  value=*
Start Index:19  end Index=20  value=
Start Index:20  end Index=21  value=6
Start Index:21  end Index=22  value=K
Start Index:22  end Index=23  value=w
Start Index:23  end Index=24  value=r
--------------------------------------------------"""
```

```
#RegExpr7.py
#This program for searching  all lower case alphabets
import re
mattab=re.finditer("[a-z]", "kAb6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
       print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
Output
--------------------------------------------------
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr7.py
--------------------------------------------------
Start Index:0  end Index=1  value=k
Start Index:2  end Index=3  value=b
```

```
Start Index:5  end Index=6  value=q
Start Index:7  end Index=8  value=c
Start Index:10  end Index=11  value=v
Start Index:15  end Index=16  value=f
Start Index:22  end Index=23  value=w
--------------------------------------------------
"""
```

```
#RegExpr8.py
#This program for searching  all except lower case alphabets
import re
mattab=re.finditer("[^a-z]", "kAb6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
        print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
Output
--------------------------------------------------
:\KVR-PYTHON-7AM\REG EXPR>py RegExpr8.py
--------------------------------------------------
Start Index:1  end Index=2  value=A
Start Index:3  end Index=4  value=6
Start Index:4  end Index=5  value=&
Start Index:6  end Index=7  value=P
Start Index:8  end Index=9  value=H
Start Index:9  end Index=10  value=7
Start Index:11  end Index=12  value=@
Start Index:12  end Index=13  value=8
Start Index:13  end Index=14  value=%
Start Index:14  end Index=15  value=L
Start Index:16  end Index=17  value=P
Start Index:17  end Index=18  value=3
Start Index:18  end Index=19  value=*
Start Index:19  end Index=20  value=
Start Index:20  end Index=21  value=6
Start Index:21  end Index=22  value=K
Start Index:23  end Index=24  value=R
--------------------------------------------------
"""
```

```
#RegExpr9.py
#This program for searching  all upper  case alphabets
import re
mattab=re.finditer("[A-Z]", "kAb6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
        print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
Output
--------------------------------------------------
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr9.py
--------------------------------------------------
```

```
Start Index:1  end Index=2  value=A
Start Index:6  end Index=7  value=P
Start Index:8  end Index=9  value=H
Start Index:14  end Index=15  value=L
Start Index:16  end Index=17  value=P
Start Index:21  end Index=22  value=K
Start Index:23  end Index=24  value=R
--------------------------------------------------

"""
```

```
#RegExpr10.py
#This program for searching  all  except upper  case alphabets
import re
mattab=re.finditer("[^A-Z]", "kAb6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
       print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
Output
--------------------------------------------------
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr10.py
--------------------------------------------------
Start Index:0  end Index=1  value=k
Start Index:2  end Index=3  value=b
Start Index:3  end Index=4  value=6
Start Index:4  end Index=5  value=&
Start Index:5  end Index=6  value=q
Start Index:7  end Index=8  value=c
Start Index:9  end Index=10  value=7
Start Index:10  end Index=11  value=v
Start Index:11  end Index=12  value=@
Start Index:12  end Index=13  value=8
Start Index:13  end Index=14  value=%
Start Index:15  end Index=16  value=f
Start Index:17  end Index=18  value=3
Start Index:18  end Index=19  value=*
Start Index:19  end Index=20  value=
Start Index:20  end Index=21  value=6
Start Index:22  end Index=23  value=w
--------------------------------------------------
"""
```

```
#RegExpr11.py
#This program for searching  all  digits
import re
mattab=re.finditer("[0-9]", "kAb6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
       print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
Output
```

```
-------------------------------------------------
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr11.py
-------------------------------------------------
Start Index:3  end Index=4  value=6
Start Index:9  end Index=10  value=7
Start Index:12  end Index=13  value=8
Start Index:17  end Index=18  value=3
Start Index:20  end Index=21  value=6
-------------------------------------------------
"""
```

```
#RegExpr12.py
#This program for searching  all  digits
import re
mattab=re.finditer("[^0-9]", "kAb6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
       print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
Output
-------------------------------------------------
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr12.py
-------------------------------------------------
Start Index:0  end Index=1  value=k
Start Index:1  end Index=2  value=A
Start Index:2  end Index=3  value=b
Start Index:4  end Index=5  value=&
Start Index:5  end Index=6  value=q
Start Index:6  end Index=7  value=P
Start Index:7  end Index=8  value=c
Start Index:8  end Index=9  value=H
Start Index:10  end Index=11  value=v
Start Index:11  end Index=12  value=@
Start Index:13  end Index=14  value=%
Start Index:14  end Index=15  value=L
Start Index:15  end Index=16  value=f
Start Index:16  end Index=17  value=P
Start Index:18  end Index=19  value=*
Start Index:19  end Index=20  value=
Start Index:21  end Index=22  value=K
Start Index:22  end Index=23  value=w
Start Index:23  end Index=24  value=R
-----------------------------------------
"""
```

```
#RegExpr13.py
#This program for searching  all  lower case alphabets and digits
import re
mattab=re.finditer("[a-z0-9]", "kAb6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
       print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)
```

```
"""
Output
-------------------------------------------------
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr13.py
-------------------------------------------------
Start Index:0  end Index=1  value=k
Start Index:2  end Index=3  value=b
Start Index:3  end Index=4  value=6
Start Index:5  end Index=6  value=q
Start Index:7  end Index=8  value=c
Start Index:9  end Index=10  value=7
Start Index:10  end Index=11  value=v
Start Index:12  end Index=13  value=8
Start Index:15  end Index=16  value=f
Start Index:17  end Index=18  value=3
Start Index:20  end Index=21  value=6
Start Index:22  end Index=23  value=w
-------------------------------------------------"""
#RegExpr14.py
#This program for searching  all  except lower case alphabets and digits
import re
mattab=re.finditer("[^a-z0-9]", "kAb6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
        print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)


""""
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr14.py
-------------------------------------------------
Start Index:1  end Index=2  value=A
Start Index:4  end Index=5  value=&
Start Index:6  end Index=7  value=P
Start Index:8  end Index=9  value=H
Start Index:11  end Index=12  value=@
Start Index:13  end Index=14  value=%
Start Index:14  end Index=15  value=L
Start Index:16  end Index=17  value=P
Start Index:18  end Index=19  value=*
Start Index:19  end Index=20  value=
Start Index:21  end Index=22  value=K
Start Index:23  end Index=24  value=R
-------------------------------------------------
"""
```

```
#RegExpr15.py
#This program for searching  all  lower case  and upper case alphabets
import re
mattab=re.finditer("[A-Za-z]", "kAb6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
        print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
```

```
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr15.py
-------------------------------------------------
Start Index:0  end Index=1  value=k
Start Index:1  end Index=2  value=A
Start Index:2  end Index=3  value=b
Start Index:5  end Index=6  value=q
Start Index:6  end Index=7  value=P
Start Index:7  end Index=8  value=c
Start Index:8  end Index=9  value=H
Start Index:10  end Index=11  value=v
Start Index:14  end Index=15  value=L
Start Index:15  end Index=16  value=f
Start Index:16  end Index=17  value=P
Start Index:21  end Index=22  value=K
Start Index:22  end Index=23  value=w
Start Index:23  end Index=24  value=R
-------------------------------------------------
"""
```

Programs

```
#RegExpr16.py
#This program for searching  all  except lower case  and upper case
alphabets
import re
mattab=re.finditer("[^A-Za-z]", "kAb6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
       print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr16.py
-------------------------------------------------
Start Index:3  end Index=4  value=6
Start Index:4  end Index=5  value=&
Start Index:9  end Index=10  value=7
Start Index:11  end Index=12  value=@
Start Index:12  end Index=13  value=8
Start Index:13  end Index=14  value=%
Start Index:17  end Index=18  value=3
Start Index:18  end Index=19  value=*
Start Index:19  end Index=20  value=
Start Index:20  end Index=21  value=6
-------------------------------------------------
"""
```

```
#RegExpr17.py
#This program for searching  all  lower case  and upper case alphabets and
digits
import re
mattab=re.finditer("[A-Za-z0-9]", "kAb6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
```

```
for entry in mattab:
       print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr17.py
--------------------------------------------------
Start Index:0  end Index=1  value=k
Start Index:1  end Index=2  value=A
Start Index:2  end Index=3  value=b
Start Index:3  end Index=4  value=6
Start Index:5  end Index=6  value=q
Start Index:6  end Index=7  value=P
Start Index:7  end Index=8  value=c
Start Index:8  end Index=9  value=H
Start Index:9  end Index=10  value=7
Start Index:10  end Index=11  value=v
Start Index:12  end Index=13  value=8
Start Index:14  end Index=15  value=L
Start Index:15  end Index=16  value=f
Start Index:16  end Index=17  value=P
Start Index:17  end Index=18  value=3
Start Index:20  end Index=21  value=6
Start Index:21  end Index=22  value=K
Start Index:22  end Index=23  value=w
Start Index:23  end Index=24  value=R
------------------------------------------"""
#RegExpr18.py
#This program for searching  all  special symbols
import re
mattab=re.finditer("[^A-Za-z0-9]", "kAb6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
       print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr18.py
--------------------------------------------------
Start Index:4  end Index=5  value=&
Start Index:11  end Index=12  value=@
Start Index:13  end Index=14  value=%
Start Index:18  end Index=19  value=*
Start Index:19  end Index=20  value=
--------------------------------------------------"""
```

```
#RegExpr19.py
#This program for searching  all  space character only
import re
mattab=re.finditer("\s", "kA b6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
```

```
       print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

""" OUTPUT
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr19.py
--------------------------------------------------
Start Index:2  end Index=3  value=
Start Index:20  end Index=21  value=
--------------------------------------------------  """
```

```
#RegExpr20.py
#This program for searching  all  except space character.
import re
mattab=re.finditer("\S", "kA b6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
       print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr20.py
--------------------------------------------------
Start Index:0  end Index=1  value=k
Start Index:1  end Index=2  value=A
Start Index:3  end Index=4  value=b
Start Index:4  end Index=5  value=6
Start Index:5  end Index=6  value=&
Start Index:6  end Index=7  value=q
Start Index:7  end Index=8  value=P
Start Index:8  end Index=9  value=c
Start Index:9  end Index=10  value=H
Start Index:10  end Index=11  value=7
Start Index:11  end Index=12  value=v
Start Index:12  end Index=13  value=@
Start Index:13  end Index=14  value=8
Start Index:14  end Index=15  value=%
Start Index:15  end Index=16  value=L
Start Index:16  end Index=17  value=f
Start Index:17  end Index=18  value=P
Start Index:18  end Index=19  value=3
Start Index:19  end Index=20  value=*
Start Index:21  end Index=22  value=6
Start Index:22  end Index=23  value=K
Start Index:23  end Index=24  value=w
Start Index:24  end Index=25  value=R
--------------------------------------------------"""
```

```
#RegExpr21.py
#This program for searching ALL DIGITS
import re
mattab=re.finditer("\d", "kA b6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
```

```
       print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr21.py
--------------------------------------------------
Start Index:4  end Index=5  value=6
Start Index:10  end Index=11  value=7
Start Index:13  end Index=14  value=8
Start Index:18  end Index=19  value=3
Start Index:21  end Index=22  value=6
--------------------------------------------------------------------------
-------------------------"""
```

```
#RegExpr22.py
#This program for searching ALL  except DIGITS
import re
mattab=re.finditer("\D", "kA b6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
       print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)
"""
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr22.py
--------------------------------------------------
Start Index:0  end Index=1  value=k
Start Index:1  end Index=2  value=A
Start Index:2  end Index=3  value=
Start Index:3  end Index=4  value=b
Start Index:5  end Index=6  value=&
Start Index:6  end Index=7  value=q
Start Index:7  end Index=8  value=P
Start Index:8  end Index=9  value=c
Start Index:9  end Index=10  value=H
Start Index:11  end Index=12  value=v
Start Index:12  end Index=13  value=@
Start Index:14  end Index=15  value=%
Start Index:15  end Index=16  value=L
Start Index:16  end Index=17  value=f
Start Index:17  end Index=18  value=P
Start Index:19  end Index=20  value=*
Start Index:20  end Index=21  value=
Start Index:22  end Index=23  value=K
Start Index:23  end Index=24  value=w
Start Index:24  end Index=25  value=R
--------------------------------------------------"""
#RegExpr23.py
#This program for searching all word char
import re
mattab=re.finditer("\w", "kA b6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
```

```
        print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
:\KVR-PYTHON-7AM\REG EXPR>py RegExpr23.py
--------------------------------------------------
Start Index:0  end Index=1  value=k
Start Index:1  end Index=2  value=A
Start Index:3  end Index=4  value=b
Start Index:4  end Index=5  value=6
Start Index:6  end Index=7  value=q
Start Index:7  end Index=8  value=P
Start Index:8  end Index=9  value=c
Start Index:9  end Index=10  value=H
Start Index:10  end Index=11  value=7
Start Index:11  end Index=12  value=v
Start Index:13  end Index=14  value=8
Start Index:15  end Index=16  value=L
Start Index:16  end Index=17  value=f
Start Index:17  end Index=18  value=P
Start Index:18  end Index=19  value=3
Start Index:21  end Index=22  value=6
Start Index:22  end Index=23  value=K
Start Index:23  end Index=24  value=w
Start Index:24  end Index=25  value=R
--------------------------------------------------"""
```

```
#RegExpr24.py
#This program for searching all special symbols
import re
mattab=re.finditer("\W", "kA b6&qPcH7v@8%LfP3* 6KwR")
print("-"*50)
for entry in mattab:
        print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr24.py
--------------------------------------------------
Start Index:2  end Index=3  value=
Start Index:5  end Index=6  value=&
Start Index:12  end Index=13  value=@
Start Index:14  end Index=15  value=%
Start Index:19  end Index=20  value=*
Start Index:20  end Index=21  value=
--------------------------------------------------
"""
```

Quantifiers in Regular Expression

Programs

```
#RegExpr25.py
#This program for searching  exactly one 'a'
import re
```

```
mattab=re.finditer("a", "abaabaaabab")
print("-"*50)
for entry in mattab:
        print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr25.py
--------------------------------------------------
Start Index:0  end Index=1  value=a
Start Index:2  end Index=3  value=a
Start Index:3  end Index=4  value=a
Start Index:5  end Index=6  value=a
Start Index:6  end Index=7  value=a
Start Index:7  end Index=8  value=a
Start Index:9  end Index=10  value=a
--------------------------------------------------"""
--------------------------------------------------------------------------------
#RegExpr26.py
#This program for searching  for one 'a' or more 'a'
import re
mattab=re.finditer("a+", "abaabaaabab")
print("-"*50)
for entry in mattab:
        print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr26.py
--------------------------------------------------
Start Index:0  end Index=1  value=a
Start Index:2  end Index=4  value=aa
Start Index:5  end Index=8  value=aaa
Start Index:9  end Index=10  value=a
--------------------------------------------------"""
```

```
#RegExpr27.py
#This program for searching zero 'a'  or one  'a'  or  more 'a'
import re
mattab=re.finditer("a*", "abaabaaabab")
print("-"*50)
for entry in mattab:
        print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr27.py
--------------------------------------------------
Start Index:0  end Index=1  value=a
Start Index:1  end Index=1  value=
Start Index:2  end Index=4  value=aa
Start Index:4  end Index=4  value=
Start Index:5  end Index=8  value=aaa
Start Index:8  end Index=8  value=
Start Index:9  end Index=10  value=a
```

```
Start Index:10  end Index=10  value=
Start Index:11  end Index=11  value=
--------------------------------------------------"""
```

```
#RegExpr28.py
#This program for searching zero 'a'  or one  'a'
import re
mattab=re.finditer("a?", "abaabaaabab")
print("-"*50)
for entry in mattab:
        print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)

"""
E:\KVR-PYTHON-7AM\REG EXPR>py RegExpr28.py
--------------------------------------------------
Start Index:0  end Index=1  value=a
Start Index:1  end Index=1  value=
Start Index:2  end Index=3  value=a
Start Index:3  end Index=4  value=a
Start Index:4  end Index=4  value=
Start Index:5  end Index=6  value=a
Start Index:6  end Index=7  value=a
Start Index:7  end Index=8  value=a
Start Index:8  end Index=8  value=
Start Index:9  end Index=10  value=a
Start Index:10  end Index=10  value=
Start Index:11  end Index=11  value=
--------------------------------------------------"""
```

```
#RegExpr29.py
#This program for searching for all
import re
mattab=re.finditer(".", "abaabaaabab")
print("-"*50)
for entry in mattab:
        print("Start Index:{}  end Index={}
value={}".format(entry.start(),entry.end(),entry.group()))
print("-"*50)
```

Additional Programs in Regular Expressions

Note: All of u must create files stud.info and emails.info

```
#This program extract email ids of various people where they present in a
file "emails.info"
#extractmails.py
import re
try:
        with open("emails.info","r") as fp:
                emaildata=fp.read()
                studnames=re.findall("[A-Z][a-z]+",emaildata)
                emaillist=re.findall("\S+@\S+", emaildata)
                print("-"*50)
                print("Student mail list")
                print("-"*50)
                for mail in emaillist:
                        print("{}".format(mail))
                print("-"*50)
```

```
                print("Student Name\tEmail-Id")
                print("-"*50)
                for names,mail in zip(studnames,emaillist):
                        print("{}\t\t{}".format(names,mail))
                print("-"*50)
except FileNotFoundError:
        print("File does not exists:")
```

```
#This program validates Mobile Number by using regular expressions.
#MobileNumberValid.py
import re
while(True):
        mno=input("Enter Ur Mobile Number:")
        if (len(mno)==10):
                result=re.search( "\d{10}" ,mno)
                if(result !=None):
                        print("Ur Mobile Number is Valid:")
                        break
                else:
                        print("Ur Mobile Number must contain only
digits\nshould not contain chars and symbols")
        else:
                print("Ur Mobile Number must contain 10 digits length
only:")
```

```
#This program extracts  Student Names and Marks from given data
#studentnamesmarks.py
import re
gd="Ramu got 55 marks, Raju got 66 marks, Rossum got 88 marks , Gosling
got 44 marks, Ritche got 89 marks and Travis got 47 mraks"
studnames=re.finditer("[A-Z][a-z]+",gd)
print("List of Student Names---finditer():")
print("-"*50)
for name in studnames:
        print("\t{}".format(name.group()))
print("-"*50)
print("---------------------OR-----------------------")
studlist=re.findall("[A-Z][a-z]+",gd)
print("List of Student Names:---findall()")
print("-"*50)
for name in studlist:
        print("\t{}".format(name))
print("-"*50)
```

```
#This program extracts  Student Names and Marks from given data
#studentnamesmarks1.py
import re
gd="Ramu got 55 marks, Raju got 66 marks, Rossum got 88 marks , Gosling
got 44 marks, Ritche got 89 marks and Travis got 47 mraks"
studnames=re.findall("[A-Z][a-z]+",gd)
print("List of Student Names")
print("-"*50)
for name in studnames:
        print("\t{}".format(name))
print("-"*50)
studmarks=re.findall("\d{2}",gd)
print("List of Student Marks")
```

```
print("-"*50)
for marks in studmarks:
	print("\t{}".format(marks))
print("-"*50)
print("\tStudent Names\tStudent Marks")
print("-"*50)
for name,marks in zip(studnames,studmarks):
	print("\t\t{}\t\t{}".format(name,marks))

print("-"*50)
```

```
#This program extracts  Student Names and Marks from given data file data
#studentnamesmarks2.py
import re
with open("D:\KVR-PY\stud.info","r") as fp:
	filedata=fp.read()
	studnames=re.findall("[A-Z][a-z]+",filedata)
	studmarks=re.findall("\d{2}",filedata)
	print("-"*50)
	print("\tStudent Names\tStudent Marks")
	print("-"*50)
	for name,marks in zip(studnames,studmarks):
		print("\t\t{}\t\t{}".format(name,marks))
	print("-"*50)
```

Quantifiers in Reg Expr

```
                    ======================================
                        Quantifiers in Regular Expression
                    ======================================
=>Quantifiers in Regular Expression makes us understand the number of
occurences of search pattern searching in given data for obtaining desired
result.

=> a----->searches only one a
=> a+---->searches one  'a'  or more  'a' s
=>a*-----> Searches for zero 'a' or one 'a' or more 'a' s
=>a?----> Searches for zero 'a' or one 'a'
=> . ------> searches for all the values present in given data
----------
Note:
----------
=>\d+ ------>searches for one or more digits  [0-9]+
=>\w+   (or)  [a-zA-Z]+----->searches for one alphabet or more

=>\ddddd-----------or  d{5}---->searches for five digit number
=>\dd.\dd-----(or) \d{2}.\d{2}---->searches for 2 Integers and 2 decimals
                                                    Ex: 23.45     99.78
--------------------------------------------------------------------------
---------------------------


                    ========================================
                              Functions in re module
                    ========================================
=>The 're' module contains the follwing essential Functions.
--------------------------------------------------------------------------
--------------------
```

1) finditer():

Syntax:- varname=re.finditer("search-pattern","Given data")
=>here varname is an object of type <class,'Callable_Itetaror'>

 =>This function is used for searching the search pattern in given data iteratively and it returns table of entries which contains start index , end index and matched value based on the search pattern.
---
2) group():

=>This function is used obtaining matched value by the findIter()
Syntax:- varname=matchtabobj.group()
---
3) start():

=>This function is used obtaining obtaining starting index of matched value
Syntax: varname=matchobj.start()
---
4) end():

=>This function is used obtaining obtaining end index+1 of matched value
Syntax: varname=matchobj.end()

---
5) search():

Syntax:- varname=re.search("search-pattern","Given data")
=>here varname is an object of <class,'match'>

=>This function is used for searching the search pattern in given data for first occuence / match only.
=>if the search pattern found in given data then it returns an object of match which contains matched value and start and end index values and it indicates search is successful.
=>if the search pattern not found in given data then it returns None and it indicates search is un-successful
---
6) findall():

Syntax:- varname=re.findall("search-pattern","Given data")
=>here varname is an object of <class,'list'>

```
=>This function is used for searching the search pattern in  entire given
data and find all  occurences / matches  and it returns all the matched
values in the form an object <class,'list'>
```

Data Encapsulation and Data Abstraction

Program

```
==========================================
                    Data Encapsulation and Data Abstraction
              ==========================================
Data Encapsulation:
-------------------------------
=>The Process of Hiding the confidential Information / Data / Methods
fropm external Programmers / end users is called Data Encapsulation
=>The Purpose of Encapsulation concept is that "To Hide Confidental
Information / Features of Class (Data Members and Methods ) ".
=>Data Encapsulation can be applied in three levels. They are
              a) At Data Members Level
              b) At Methods Level
              c) At Constructor Level
=>To implement Data Encapsulation in python programming, The Data Members
, Methods and Constructors must be preceded with double under score ( _ _
)

Syntax1:-                class <ClassName>:
                                   def   methodname(self):
                                          self.__Data MemberName1=Value1
                                          self.__Data MemberName2=Value2
                                          ------------------------------
--------------------
                                          self.__Data MemberName-
n=Value-n


Syntax2:-                class <ClassName>:
                                   def   __methodname(self):
                                          self.Data MemberName1=Value1
                                          self.Data MemberName2=Value2
                                          ------------------------------
--------------------
                                          self.Data MemberName-n=Value-n
Syntax3:-                class <ClassName>:
                                   def   ____init__(self):
                                          self.Data MemberName1=Value1
                                          self.Data MemberName2=Value2
                                          ------------------------------
--------------------
                                          self.Data MemberName-n=Value-n

Example1:
---------------------
#account.py----file name and treated as module name
class Account:
       def  getaccountdet(self):
              self.__acno=34567
```

```
                self.cname="Rossum"
                self.__bal=34.56
                self.bname="SBI"
                self.__pin=1234
                self.pincode=4444444
                #here  acno,bal and pin are encapsulated

Example2:
---------------------
#account1.py----file name and treated as module name
class Account1:
       def  __getaccountdet(self):  #here  __getaccountdet() is made is
encapsulated
                self.acno=34567
                self.cname="Rossum"
                self.bal=34.56
                self.bname="SBI"
                self.pin=1234
                self.pincode=4444444

============================================================
Data Abstraction:
------------------------------
=>The Process of retrieving / extracting Essential Details without
considering Hidden Details is called Data Abstraction.

Example1:
-----------------
#others.py---This Program access only cname,bname and pincode only
from account  import Account
ao=Account()
ao.getaccountdet()
#print("Account Number={}".format(ao.acno))  Not Possible to access
print("Account Holder Name={}".format(ao.cname))
#print("Account Bal={}".format(ao.bal))  Not Possible to access
print("Account Branch Name={}".format(ao.bname))
#print("Account PIN={}".format(ao.pin))   Not Possible to access
print("Account Branch Pin Code={}".format(ao.pincode))
-------------------------------------------------------------------------
---------------
Example2:
-----------------
#others1.py--here we can't access method itself. so that we cant access
Instance Data Members.
from account1 import Account1
ao=Account1()
#ao.getaccountdet()---can't access
#print("Account Number={}".format(ao.acno))
#print("Account Holder Name={}".format(ao.cname))
#print("Account Bal={}".format(ao.bal))
#print("Account Branch Name={}".format(ao.bname))
#print("Account PIN={}".format(ao.pin))
#print("Account Branch Pin Code={}".format(ao.pincode))
```

```
#account.py----file name and treated as module name
class Account:
       def  getaccountdet(self):
              self.__acno=34567
              self.cname="Rossum"
              self.__bal=34.56
              self.bname="SBI"
              self.__pin=1234
              self.pincode=4444444
              #here  acno,bal and pin are encapsulated
```

```
#account1.py----file name and treated as module name
class Account1:
       def  __getaccountdet(self):  #here  __getaccountdet() is made is
encapsulated
              self.acno=34567
              self.cname="Rossum"
              self.bal=34.56
              self.bname="SBI"
              self.pin=1234
              self.pincode=4444444
```

```
#others.py
from account  import Account
ao=Account()
ao.getaccountdet()
#print("Account Number={}".format(ao.acno))  Not Possible to access
print("Account Holder Name={}".format(ao.cname))
#print("Account Bal={}".format(ao.bal))  Not Possible to access
print("Account Branch Name={}".format(ao.bname))
#print("Account PIN={}".format(ao.pin))   Not Possible to access
print("Account Branch Pin Code={}".format(ao.pincode))
```

```
#others1.py
from account1 import Account1
ao=Account1()
#ao.getaccountdet()---can't access
#print("Account Number={}".format(ao.acno))
#print("Account Holder Name={}".format(ao.cname))
#print("Account Bal={}".format(ao.bal))
#print("Account Branch Name={}".format(ao.bname))
#print("Account PIN={}".format(ao.pin))
#print("Account Branch Pin Code={}".format(ao.pincode))
```

Pickling and un-pickling programs by using OOPS

```
#student.py---file name and treated as module name
class Student:
       def   getstudentdetails(self):
              self.stno=int(input("Enter Student Number:"))
              self.sname=input("Enter Student Name:")
              self.marks=float(input("Enter Student Marks:"))
       def dispstuddetails(self):

       print("\t{}\t{}\t{}".format(self.stno,self.sname,self.marks))
```

```
#studentpick.py
```

```
from student import Student
import pickle,sys
with open("stud.data","ab") as fp:
       while(True):
              try:
                     so=Student()
                     so.getstudentdetails()
                     pickle.dump(so,fp)
                     print("\nStudent Object data saved successfully in
File:")
                     ch=input("Do u want to insert another student
data:")
                     if(ch=="no"):
                            print("Thx for using program")
                            sys.exit()
              except ValueError:
                     print("Don't enter strs / symbols/ alpha-numerics
for Student Number and Marks:")
```

```
#studentpick1.py
from student import Student
import pickle,sys
class StudentPick:
       def   pickprocess(self):
              with open("stud.data","ab") as fp:
                     while(True):
                            try:
                                   so=Student()
                                   so.getstudentdetails()
                                   pickle.dump(so,fp)
                                   print("\nStudent Object data saved
successfully in File:")
                                   ch=input("Do u want to insert another
student data:")
                                   if(ch=="no"):
                                          print("Thx for using program")
                                          sys.exit()
                            except ValueError:
                                   print("Don't enter strs / symbols/
alpha-numerics for Student Number and Marks:")

#main program
sp=StudentPick()
sp.pickprocess()
```

```
#studentunpick.py
import pickle,sys
try:
       with open("stud.data","rb") as fp:
              print("-"*50)
              print("\tS t u d e n t   D e t a i l s")
              print("-"*50)
              while(True):
                     try:
```

```
                                obj=pickle.load(fp)# here obj is an object of
type student.Student
                                obj.dispstuddetails()
                        except EOFError:
                                print("-"*50)
                                sys.exit()
except FileNotFoundError:
        print("File does not exists
```

```
#studentunpick1.py
import pickle,sys
class StudentUnpik:
        def  unpickprocess(self):
                try:
                        with open("stud.data","rb") as fp:
                                print("-"*50)
                                print("\tS t u d e n t   D e t a i l s")
                                print("-"*50)
                                while(True):
                                        try:
                                                obj=pickle.load(fp)# here obj is
an object of type student.Student
                                                obj.dispstuddetails()
                                        except EOFError:
                                                print("-"*50)
                                                sys.exit()
                except FileNotFoundError:
                        print("File does not exists:")

#main program
sp=StudentUnpik()
sp.unpickprocess()
```

Constructors in Python

Differences between Methods and Constructors

Programs

```
                ========================================
                        Constructors in Python
                ========================================
=>The purpose of Constructors in Python is that "To Initlize the Object".
=>Initlizing the object is nothing but placing our own values without
leaving the
    object empty.
-------------------------------------------
=>Definition of Constructor:
-------------------------------------------
=>A constructor is one of the Special Method which is automatically /
implicitly called by PVM during object creation and it always Initlizes
the object (Placing our own values).
-----------------------------------------------------------------
Rules for using Constructor in Python:
-----------------------------------------------------------------
=>The Constructor called by PVM during Object Creation automatically /
Implicitly.
```

```
=>The Name of the constructor is     def   __init__(self,......):pass
=>The constructors in python can be inherited.
=>The constructors in python can be Overridden
-------------------------------------------------------------------------------
--------------------------------
Syntax for constructor    :
-----------------------------------------
def     __init__(self, list of formal params if any):
         -----------------------------------------
        -----------------------------------------
        Block  of statements--Initlization
        -----------------------------------------
        -----------------------------------------
-------------------------------------------------------------------------------
-------------------
=>Types of Constructors:
-------------------------------------------------------------------------------
--------------------
=>In Python Programming, we have two types of Constructors. They are
              a) Default / Parameter Less Constructors
              b) Parameterized Constructors
-----------------------------------------------------------------
a) Default / Parameter Less Constructors:
-----------------------------------------------------------------
=>Definition:
-------------------
=>Default / Parameter Less Constructor is one, which is not taking any
Parameters / Values.
=>The purpose of Default / Parameter Less Constructors is that "To
Initlize multiple objects of same class with Same Values "
------------------
=>Syntax:-
------------------
           def   __init__(self):
                 --------------------------------
                Block of statements--Initlizatioin
                --------------------------------

Examples:
-----------------
#defaultconstex1.py
class Test:
       def __init__(self):
              print("i am from default constructor:")
              self.a=10
              self.b=20
              print("Val of a={}".format(self.a))
              print("Val of b={}".format(self.b))

#main program
print("Content of t1 object:")
t1=Test()
print("-"*50)
print("Content of t2 object:")
```

```
t2=Test()
print("-"*50)
print("Content of t3 object:")
t3=Test()
```

---

b) Parameterized Constructors:

=> A Parameterized Constructor is one, which is taking Parameters / Values.

=>The purpose of Parameterized Constructors is that "To Initlize multiple objects of same class with Different Values "

=>Syntax:-

```
          def   __init__(self,list of formal params):
                --------------------------------
               Block of statements--Initlizatioin
               --------------------------------
```

Exmaples:

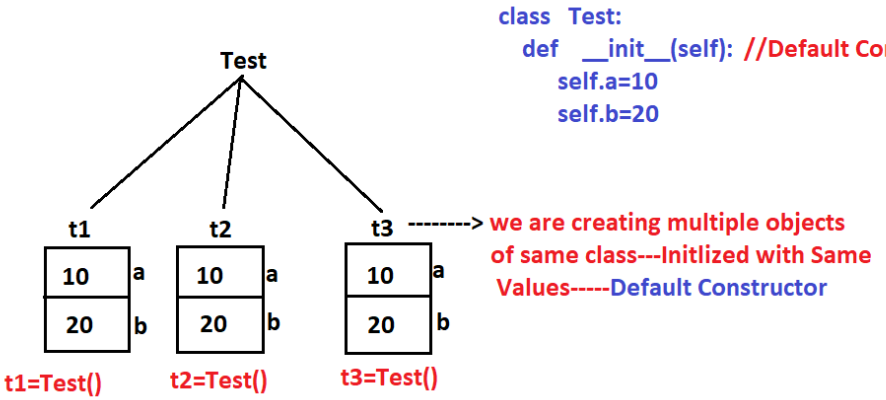
```
#paramconstex1.py
class Test:
       def __init__(self,a,b):
              print("i am from Parameterized constructor:")
              self.a=a
              self.b=b
              print("Val of a={}".format(self.a))
              print("Val of b={}".format(self.b))

#main program
print("Content of t1 object:")
t1=Test(10,20)
print("-"*50)
print("Content of t2 object:")
t2=Test(100,200)
print("-"*50)
print("Content of t3 object:")
t3=Test(1000,2000)
print("-"*50)
```

==========x==========

NOTE:- In a class of python, we can't define both default and parametreised constructor bcoz PVM can remember only latest constructor but not able to remember all types of constructor . To solve this issue, In a class of Python, we can define One Constructor with default parameter mechanism.

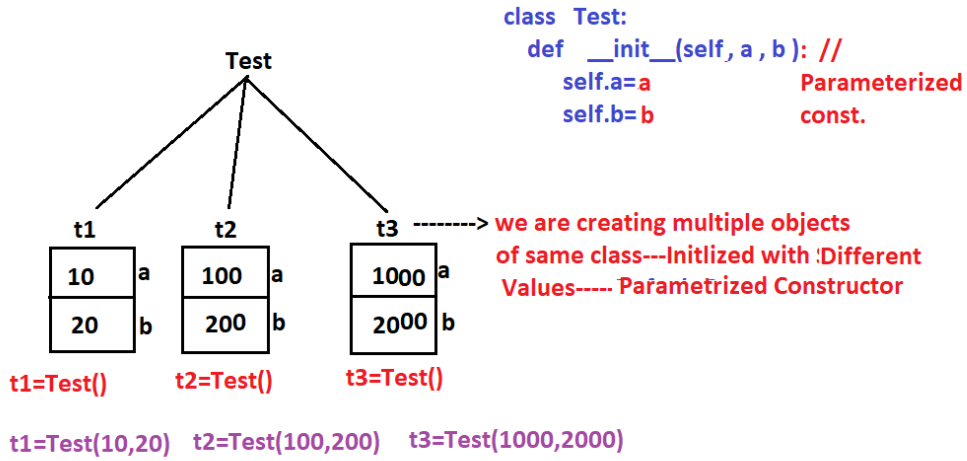
```
#defaultparamconstex1.py
class Test:
       def __init__(self,a=10,b=20):
              self.a=a
              self.b=b
              print("Val of a={}".format(self.a))
```

```
            print("Val of b={}".format(self.b))
#main program
print("Content of t1 object--with default constructor:")
t1=Test()  # calls default constructor
print("-"*50)
print("Content of t2 object---with parameterized ")
t2=Test(100,200) # calls Parameteized constructor
print("-"*50)
print("Content of t3 object--with default constructor:")
t3=Test()  # calls default constructor
print("-"*50)
print("Content of t4 object---with parameterized ")
t2=Test(1000,2000) # calls Parameteized constructor
print("-"*50)
==================================X==========================
```




```
   ================================================
                  Differences between Methods and Constructors
               ================================================
=>Constructors are always used Initlizing the object. Where as Methods are
used for Performing the operations on the object.
=>The name of the constructor is always __init__(self, params list if
any). where as name of the method can be any valid variable name.
=>The Constructors are not recommended to return  the values (They are
suppose to initlize) where methods are recommended to return the values (
They are doing operation).
=>Constructors are calling automatically when an object is created  where
as methods are calling explicitly.
```

Note:- In Python, Both Constructors and Methods can be Inherited and they can
be Overridden.
==========X==========




```
#Constex1.py
class Student:
       def  setstudentvalues(self):  # Ordinary method--need to call
explicitly
              self.stno=10
              self.name="Rossum"
              self.marks=44.44

#main program
so=Student() # object creation--i want place my own values
print("content of so defore setting values=", so.__dict__) #  { }
so.setstudentvalues() # here we are calling explicitly one function
print("content of so after setting values=", so.__dict__) #  {----- }
```

```
#Constex2.py
class Student:
       def  __init__(self):  #Constructor
              print("I am from constructor")
              self.stno=10
              self.name="Rossum"
```

```
                self.marks=66.66


#main program
so=Student() # object creation--i want place my own values--Initlize the
object
print("content of so =", so.__dict__) #  {..... }
```

```
#Contex3.py
class Account:
       def __init__(self):
              self.acno=int(input("Enter Account Number:"))
              self.name=input("Enter Customer Name:")
              self.bal=float(input("Enter Balanace:"))
              self.bname=input("Enter Branch Name:")

#main program
ao1=Account() # object creation
print("content of ao1=", ao1.__dict__)
print("-----------------------------------------")
ao2=Account() # object creation
print("content of ao2=", ao2.__dict__)
```

```
#defaultconstex1.py
class Test:
       def __init__(self):
              print("i am from default constructor:")
              self.a=10
              self.b=20
              print("Val of a={}".format(self.a))
              print("Val of b={}".format(self.b))

#main program
print("Content of t1 object:")
t1=Test()
print("-"*50)
print("Content of t2 object:")
t2=Test()
print("-"*50)
print("Content of t3 object:")
t3=Test()
```

```
#defaultparamconstex1.py
class Test:
       def __init__(self,a=10,b=20):
              self.a=a
              self.b=b
              print("Val of a={}".format(self.a))
              print("Val of b={}".format(self.b))
#main program
print("Content of t1 object--with default constructor:")
t1=Test()  # calls default constructor
print("-"*50)
print("Content of t2 object---with parameterized ")
t2=Test(100,200) # calls Parameteized constructor
print("-"*50)
```

```
print("Content of t3 object--with default constructor:")
t3=Test()  # calls default constructor
print("-"*50)
print("Content of t4 object---with parameterized ")
t2=Test(1000,2000) # calls Parameteized constructor
print("-"*50)
```

```
#paramconstex1.py
class Test:
       def __init__(self,a,b):
              print("i am from Parameterized constructor:")
              self.a=a
              self.b=b
              print("Val of a={}".format(self.a))
              print("Val of b={}".format(self.b))

#main program
print("Content of t1 object:")
t1=Test(10,20)
print("-"*50)
print("Content of t2 object:")
t2=Test(100,200)
print("-"*50)
print("Content of t3 object:")
t3=Test(1000,2000)
print("-"*50)
```

Destructors in Python

programs

```
                    ========================================
                              Destructors in Python
                     ========================================
=>The Name of Destructor is
                                def      __del__(self):
                                       ------------------------
                                    ------------------------
=>Destructors are called by Garbage Collector
=>A Garbage Collector is one of the Python In-built program, which is
executing Internally behind of every Program for collecting / de-
allocating the memory space of all objects which are used in the python
program.
=>In Python Programming Every Garbage Collector contains its own
detsructor.
=>In the destructor , we write code for destorying the memory space of
Objects used in python program and every destructor  called by Garbage
Collector.

=>By Default, Garbage Collector calls Destructor at end execution of
Python Program.
=>Programatically , We can make the Garbage Collector to call Destructor
FORCEFULLY  by nullifying the object ( Example:   obj=None )
```

```
#destructorex1.py
class Employee:
       def __init__(self,eno,ename):
              print("i am from Contructor:")
```

```
                self.eno=eno
                self.ename=ename
                print("\t{}\t{}".format(self.eno,self.ename))

#main program
print("Program Execution Started..")
eo1=Employee(10,"RS")
eo2=Employee(20,"DR")
print("\nProgram Execution Completed..")
#Since program execution completed, GC collects Objects memory spaces and
hand over to OS. So internally to do this process, GC calls its Destructor
Program for de-allocating / destroying the memory space of objects.
```

```
#destructorex2.py
import time
class Employee:
        def __init__(self,eno,ename):
                print("i am from Contructor:")
                self.eno=eno
                self.ename=ename
                print("\t{}\t{}".format(self.eno,self.ename))
        def  __del__(self):  # Programmer-defined Destructor
                print("\nGC is calling Destructor Function")

#main program
print("Program Execution Started..")
eo1=Employee(10,"RS")
eo2=Employee(20,"DR")
eo3=Employee(30,"RR")
print("\nProgram Execution Completed..")
time.sleep(5)
```

```
#destructorex3.py
import time
class Employee:
        def __init__(self,eno,ename):
                print("\ni am from Contructor:")
                self.eno=eno
                self.ename=ename
                print("\t{}\t{}".format(self.eno,self.ename))
        def  __del__(self):  # Programmer-defined Destructor
                print("\nGC is calling Destructor Function")

#main program
print("Program Execution Started..")
eo1=Employee(10,"RS")
print("No Longer Interested to maintain object eo1")
time.sleep(8)
eo1=None # Here Forcefully , GC calls    __def__(self)
eo2=Employee(20,"DR")
eo3=Employee(30,"RR")
print("\nProgram Execution Completed..")
time.sleep(5)
# Here by default  , GC calls     __def__(self)  two times
```

```
#destructorex4.py
```

```
import time
class Employee:
       def __init__(self,eno,ename):
              print("\ni am from Contructor:")
              self.eno=eno
              self.ename=ename
              print("\t{}\t{}".format(self.eno,self.ename))
       def  __del__(self):  # Programmer-defined Destructor
              print("\nGC is calling Destructor Function")

#main program
print("Program Execution Started..")
eo1=Employee(10,"RS")
print("No Longer Interested to maintain object eo1")
time.sleep(8)
eo1=None # Here Forcefully , GC calls        __del__(self)
eo2=Employee(20,"DR")
print("No Longer Interested to maintain object eo2")
time.sleep(8)
eo2=None # Here Forcefully , GC calls         __del__(self)
eo3=Employee(30,"RR")
print("\nProgram Execution Completed..")
time.sleep(5)
# Here by default  , GC calls     __del__(self)  one times
```

```
#destructorex5.py
import time
class Employee:
       def __init__(self,eno,ename):
              print("\ni am from Contructor:")
              self.eno=eno
              self.ename=ename
              print("\t{}\t{}".format(self.eno,self.ename))
       def  __del__(self):  # Programmer-defined Destructor
              print("\nGC is calling Destructor Function")

#main program
print("Program Execution Started..")
eo1=Employee(10,"RS")
print("No Longer Interested to maintain object eo1")
time.sleep(8)
del(eo1) # Here Forcefully , GC calls        __del__(self)
eo2=Employee(20,"DR")
print("No Longer Interested to maintain object eo2")
time.sleep(8)
del eo2 # Here Forcefully , GC calls         __del__(self)
eo3=Employee(30,"RR")
print("\nProgram Execution Completed..")
time.sleep(5)
# Here by default  , GC calls     __del__(self)  one times
```

```
#destructorex6.py
import time
class Employee:
       def __init__(self,eno,ename):
              print("\ni am from Contructor:")
```

```
                self.eno=eno
                self.ename=ename
                print("\t{}\t{}".format(self.eno,self.ename))
        def  __del__(self):  # Programmer-defined Destructor
                print("\nGC is calling Destructor Function")

#main program
print("Program Execution Started..")
eo1=Employee(10,"RS")
eo3=eo2=eo1 # Deep Copy
eo4=Employee(20,"SS")

print("\nProgram Execution Completed..")
time.sleep(5)# Here by default  , GC calls     __del__(self)  one time
only even though there exists two objects and both the objects points to
same memory space
```

```
#destructorex7.py
import time
class Employee:
        def __init__(self,eno,ename):
                print("\ni am from Contructor:")
                self.eno=eno
                self.ename=ename
                print("\t{}\t{}".format(self.eno,self.ename))
        def  __del__(self):  # Programmer-defined Destructor
                print("\nGC is calling Destructor Function")

#main program
print("Program Execution Started..")
eo1=Employee(10,"RS")
eo3=eo2=eo1 # Deep Copy
print("No Longer Interested to maintain object eo1")
time.sleep(8)
del(eo1) # here GC will not call   __del__(self) , bcoz that corresponding
memory pointed by eo2 and eo3
print("No Longer Interested to maintain object eo2")
time.sleep(8)
eo2=None # here GC will not call   __del__(self) , bcoz that corresponding
memory pointed by  eo3
print("No Longer Interested to maintain object eo3")
time.sleep(8)
eo3=None # here GC will  call   __del__(self) , bcoz that corresponding
memory is not pointed by  any objects

print("\nProgram Execution Completed..")
time.sleep(5)# Here by default  , GC calls     __del__(self)  one time only
even though there exists two objects and both the objects points to same
memory space
```

```
#destructorex8.py
import time,gc
class Employee:
        def __init__(self,eno,ename):
                print("\ni am from Contructor:")
                self.eno=eno
                self.ename=ename
                print("\t{}\t{}".format(self.eno,self.ename))
```

```
        def  __del__(self):  # Programmer-defined Destructor
               print("\nGC is calling Destructor Function")

#main program
print("Program Execution Started.. and status of
gc={}".format(gc.isenabled()))
eo1=Employee(10,"RS")
gc.disable()
print("No Longer Interested to maintain object eo1")
time.sleep(8)
del(eo1) # Here Forcefully , GC calls        __del__(self)
eo2=Employee(20,"DR")
print("No Longer Interested to maintain object eo2")
time.sleep(8)
del eo2 # Here Forcefully , GC calls         __del__(self)
eo3=Employee(30,"RR")
print("\nProgram Execution Completed..")
time.sleep(5)
# Here by default  , GC calls      __del__(self)  one times
```

```
#gcex1.py
import gc,time
print("Is GC Running={}".format(gc.isenabled()))
print("\ni am a python Programmer")
time.sleep(5)
gc.disable()
print("Is GC Running after disable={}".format(gc.isenabled()))
print("From Igate MNC Global")
print("In Hyd")
```

Introduction to Inheritance

Types of Inheritances

Definitions of Inheritances

1. Single Inheritance.
2. Multi Level Inheritance.
3. Hierarchical Inheritance.
4. Multiple Inheritance.
5. Hybrid Inheritance.

Syntax for Inheriting the features of Base class into derived class

```
                    ====================================
                            Types of Inheritances
                ====================================
=>Type of Inheritance  is one of the diagram / model always makes us to
     understand How the features are Inherited from Base class into
derived class.
=>In Python Programming, we have 5 types of Inheritances. They are

                        1. Single Inheritance.
                        2. Multi Level Inheritance.
                        3. Hierarchical Inheritance.
                        4. Multiple Inheritance.
                        5. Hybrid Inheritance.
================================X=======================
                        ============================================
                                    Introduction to Inheritance
                        ============================================
```

=>Inheritance is one of the distinct principle of OOPs
=>The purpose of Inheritance is that "To Build / Develop Re-Usable Applications."
=>The advantages of Inheritance concept is that
1) Application Development time Less
2) Application Memory Space is Very Less
3) Application Execution Time is Very Less
4) Application Performnace is Enhanced (Improved)
5) Redundency (Duplication) of the code is minimized.

=>Definition of Inheritance:

=>The process of obtaining the Data Members , Methods and Constructors (Features) from One class into another class is called Inheritance.
=>The class which is giving the Data Memebrs , Methods and Constructors (Features) is called "Base / Super / Parent Class."
=>The class which is Taking the Data Memebrs , Methods and Constructors (Features) is called "Derived / Sub / Child Class."

=>The Inheritance Priciple always Provides Logical Memory Management. This Memory management says that Neither we write Physical Source Code Nor takes Physical Memory Space.

==========X==========

1. Single Inheritance.

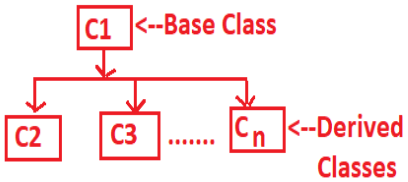
Def:- This Inheritance contains Single Base class and Single Derived Class.

Diagram:-

3. Hierarchical Inheritance

Def: This Inheritance Contains Single Base Class and Multiple Derived Classes.

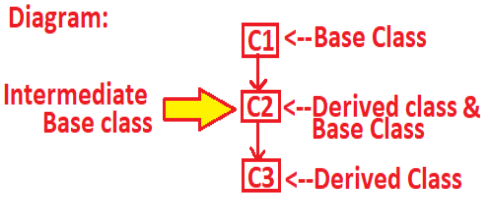
Diagram:

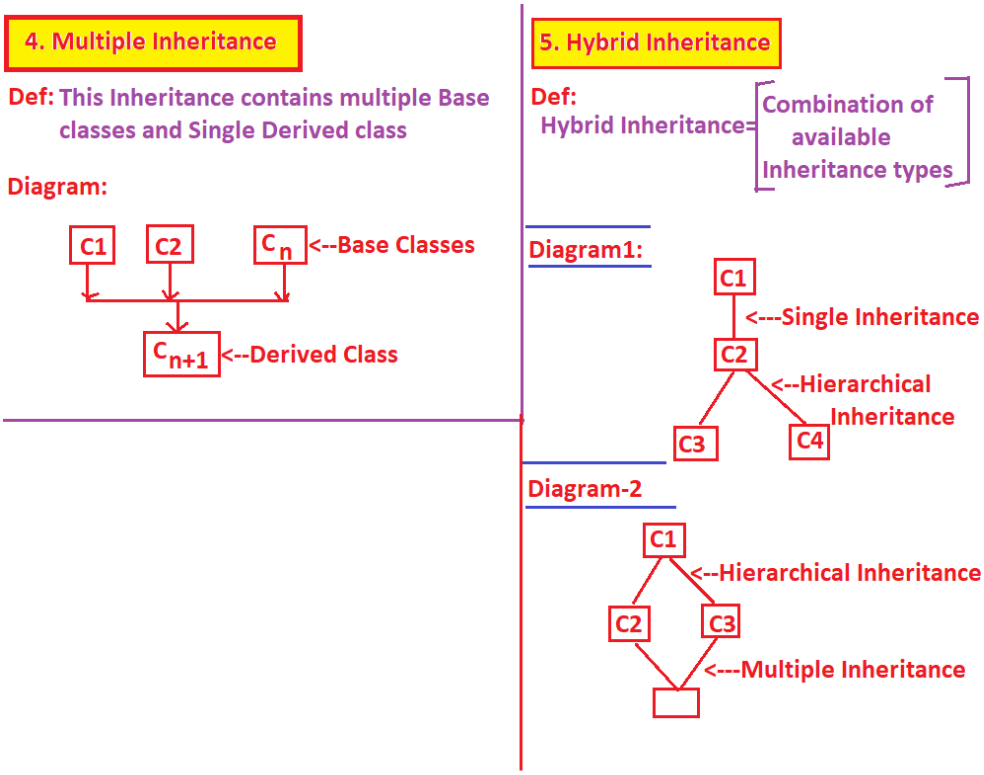


2. Multi Level Inheritance

Def:- This Inheritance contains Single Base, Single Derived Class and One or More Intermediate Base classes.

Diagram:






```
                ============================================================
                 Inheriting the features of Base Class into Derived Class
                ============================================================
=>The features of Base Class are nothing but Data Members , Methods and
Constructors.
=>To Inherit the features of Base Class into Derived Class, we use the
following Syntax:

                class  <class-name-1>:    # Base Class
                         ---------------------
                         ---------------------
                class  <class-name-2>:    # Base Class
                         ---------------------
                         ---------------------
                class  <class-name-n>:    # Base Class
                         ---------------------
                         ---------------------
                class  <class-name-n+1>( Class-Name-1,Class-Name-2...Class-
Name-n):
                         ---------------------
                         ---------------------

-------------------
Explanation:
-------------------
```

=>Here <class-name-1>, <class-name-2>....<class-name-n> are called Base Classes.
=><class-name-n+1> is called Derived Class.
=>When we develop any Inheritance Based Application, It is always recommended to create an object of Bottom most Derived Class bcoz It contains all the features of Intermediate Base Classes and Top Most Base Class.
=>For Every class in python, There exists an implicit pre-defined super class called "object" and It provdes Garbage Collector Program"
==========X==========

Programs on inheritance

```
#This program demonstrates the concept of Inheritance
#InhProg1.py
class C1:
       def  setC1Values(self):
              self.id=10
              self.crs="Python"

class C2(C1):  # Following Single Inheritance
       def setC2Values(self):
              self.sname="Rossum"
              self.cname="OUCET"

#main program
o2=C2()
print("Content of o2=",o2.__dict__)
o2. setC1Values()  # w.r.t Derived Class, we are calling Base Class Method
o2.setC2Values()
print("Content of o2=",o2.__dict__)
```

```
#This program demonstrates the concept of Inheritance
#InhProg2.py
class C1:
       def  setC1Values(self):
              self.id=10
              self.crs="Python"

class C2(C1) : # Following Single Inheritance
       def setC2Values(self):
              self.setC1Values()  # w.r.t Derived Class, we are calling
Base Class Method
              self.sname="Rossum"
              self.cname="OUCET"

       def  dispvalues(self):
              for k,v in self.__dict__.items():
                     print("\t{}\t{}".format(k,v))

#main program
o2=C2()
o2.setC2Values()
print("Student Details")
print("-"*50)
o2.dispvalues()
print("-"*50)
```

```
#InhProg3.py
class Company:
        def   setcompdet(self):
                self.cname=input("Enter the company name:")
                self.cloc=input("Enter the company location:")
        def dispcompdet(self):
                print("-"*50)
                print("Company Name:{}".format(self.cname))
                print("Company Location:{}".format(self.cloc))
                print("-"*50)

class Employee(Company):
        def setempdet(self):
                self.eno=int(input("Enter Employee Number:"))
                self.ename=input("Enter Employee Name:")
                self.sal=float(input("Enter Employee Salary:"))

        def dispempdet(self):
                print("-"*50)
                print("Employee Number:{}".format(self.eno))
                print("Employee Name:{}".format(self.ename))
                print("Employee Salary:{}".format(self.sal))
                print("-"*50)

#main program
eo=Employee() # create an object of Bottom most derived class Employee
eo.setempdet()
eo.setcompdet()
eo.dispempdet()
eo.dispcompdet()
```

```
#InhProg4.py
class Company:
        def   setcompdet(self):
                self.cname=input("Enter the company name:")
                self.cloc=input("Enter the company location:")
        def dispcompdet(self):
                print("-"*50)
                print("Company Name:{}".format(self.cname))
                print("Company Location:{}".format(self.cloc))
                print("-"*50)

class Employee(Company):
        def setempdet(self):
                self.eno=int(input("Enter Employee Number:"))
                self.ename=input("Enter Employee Name:")
                self.sal=float(input("Enter Employee Salary:"))
                self.setcompdet() # calling Base class method

        def dispempdet(self):
                print("-"*50)
                print("Employee Number:{}".format(self.eno))
                print("Employee Name:{}".format(self.ename))
                print("Employee Salary:{}".format(self.sal))
                print("-"*50)
                self.dispcompdet() # calling Base class method

#main program
```

```
eo=Employee() # create an object of Bottom most derived class Employee
eo.setempdet()
eo.dispempdet()
```

Method Overriding and Polymorphism concept program

```
                ==========================================
                              Method Overrding
                 ==========================================
=>To use Method Overriding in the python  Program, we need to apply
Inheritance Principle.
-----------------------------------------
=>Def. of Method Overriding:
-----------------------------------------
=>Method Overriding=Method Heading is Same + Method Body is different
                        (OR)
=>The Process of re-defining the original method of base class into
various derived classes for performing different operations is called
Method Overriding.
=>Method Overrding concept is used for implementing  Polymorphism
Principle.
==================================X==============================
                ==========================================
                              Polymorphism
                 ==========================================
=>Polymorphism is one of the distinct principle of OOPs
=>The purpose of Polymorphism principle is that " To build the Object
oriented Applications with effective Memory space(less memory space) ".
-------------------------------------
=>Def. of Polymorphism:
-------------------------------------
=>The process of representing "One Form in Multiple Forms" is called
Polymorphism.
=>In the definition of Polymorphism, One Form represents Original Method
of Base Class and Multiple Forms represents Overridden Methods of Derived
Class.
=>To Implement Polymorphism Principle, we must use Method Overriding.
```

```
#MethodOverridingEx1.py
#This Program is purely using Inheritance Principle
class Circle:
       def  draw1(self):
              print("Drawing Circle:")

class Rect(Circle):
       def draw2(self):
              print("Drawing Rect:")

#main program
ro=Rect()
ro.draw2()
ro.draw1()
```

```
#MethodOverridingEx2.py
#This Program is purely using Inheritance Principle with Method Overriding
class Circle:
       def  draw(self): # Original Method
```

```
                print("Drawing Circle")

class Rect(Circle):
        def  draw(self):  # Overridden Method
                print("Drawing Rectangle")
                super(). draw()

#main program
ro=Rect()
ro.draw()
```

```
#MethodOverridingEx3.py
#This Program is purely using Inheritance Principle with Method Overriding
class Circle:
        def  draw(self): # Original Method
                print("Drawing Circle")
                #super().draw()---AttributeError: 'super' object has no
attribute 'draw'
class Rect(Circle):
        def  draw(self):  # Overridden Method
                print("Drawing Rectangle")
                super().draw() # calling Base class method name from
derived class
class Square(Rect):
        def  draw(self):  # Overridden Method
                print("Drawing Square")
                super().draw()# calling Intermedaite Base class method name
from derived                                                          class

#main program
so=Square()
so.draw()
```

```
#Program cal area of different Figures such as circle,rect and square by
using method overriding
#MethodOverridingEx4.py
class Circle:
        def  area(self):  # Original Method--one Form
                self.r=float(input("Enter Radious:"))
                self.ac=3.14*self.r**2
                print("Area of Circle={}".format(self.ac))
class Square(Circle):
                def  area(self):  # Overridden Method
                        self.s=float(input("Enter Side:"))
                        self.sa=self.s**2
                        print("Area of Square={}".format(self.sa))
                        print("-"*50)
                        super().area()
class Rect(Square):
                def  area(self):  # Overridden Method
                        self.l,self.b=float(input("Enter Length:")),
float(input("Enter Breadth:"))
                        self.ar=self.l*self.b
                        print("Area of Rect={}".format(self.ar))
                        print("-"*50)
                        super().area()
```

```
#main program
ro=Rect()
ro.area()
```

Number approaches to call Base class methods
from Derived Class Methods

```
          ================================================
              Number approaches to call Base class methods
                     from Derived Class Methods
          ================================================
=>In Python Programming, we have two approaches to calle base class
methods from derived class methods. They are
              a) By Using  super()
              b) By using  ClassName
-------------------------------------------------------------
a) By Using  super():
-------------------------------------------------------------
=>super() is used for calling Base Class Orginal Method (or) Original
Constructor from derived class Overridden Method / Constructor.

=>Syntax1:-      super().MethodName()
                       super().MethodName(list of values)
=>Syntax2:-      super().__init__()
                       super().__init__(list of values)

=>With super() we are able to call single base class method from derived
class method but unable to call multiple base class methods from derived
class methods.
=>To Over this problem, we must use Class Name concept.
-------------------------------------------------------------------------
---------------
b) By using  ClassName:
-------------------------------------------
=>By using Class Name concept we can call  Multiple Base Class Orginal
Methods (or) Original Constructors from derived class Overridden Method /
Constructor.

=>Syntax1:-     Class Name.MethodName(self)
                       Class Name.MethodName(self , list of values)
=>Syntax2:-     Class Name.__init__(self)
                       Class Name.__init__(self , list of values)
=====================================X=================================
#Program cal area of different Figures such as circle,rect and square by
using method overriding for implementing polymorphism
#MethodOverridingEx5.py
class Circle:
       def  area(self, r):  # Original Method--One Form
              self.ac=3.14*r**2
              print("Area of Circle={}".format(self.ac))
class Square(Circle):
              def  area(self, s):  # Overridden Method
                     self.sa=s**2
                     print("Area of Square={}".format(self.sa))
```

```
                    print("-"*50)
                    super().area(float(input("Enter Radious:")))
class Rect(Square):
            def  area(self,l,b):  # Overridden Method
                    self.ar=l*b
                    print("Area of Rect={}".format(self.ar))
                    print("-"*50)
                    super().area(float(input("Enter Side:")))


#main program
l,b=float(input("Enter Length:")), float(input("Enter Breadth:"))
ro=Rect()
ro.area(l,b)
```

```
#Program cal area of different Figures such as circle,rect and square by
using method overriding for implementing polymorphism
#MethodOverridingEx6.py
class Circle:
       def  area(self, r):  # Original Method--One Form
              self.ac=3.14*r**2
              print("Area of Circle={}".format(self.ac))
class Square:
              def  area(self, s):  # Overridden Method
                     self.sa=s**2
                     print("Area of Square={}".format(self.sa))
                     print("-"*50)
class Rect(Circle,Square):
       def  area(self,l,b):  # Overridden Method
              self.ar=l*b
              print("Area of Rect={}".format(self.ar))
              print("-"*50)
              Circle.area(self,5)
              Square.area(self, 10)

#main program
l,b=float(input("Enter Length:")), float(input("Enter Breadth:"))
ro=Rect()
ro.area(l,b)
```

```
#Program cal area of different Figures such as circle,rect and square by
using method overriding for implementing polymorphism
#MethodOverridingEx7.py
class Circle:
       def  area(self, r):  # Original Method--One Form
              self.ac=3.14*r**2
              print("Area of Circle={}".format(self.ac))
class Square:
              def  area(self, s):  # Overridden Method
                     self.sa=s**2
                     print("Area of Square={}".format(self.sa))
                     print("-"*50)
class Rect(Circle,Square):
       def  area(self,l,b):  # Overridden Method
              self.ar=l*b
```

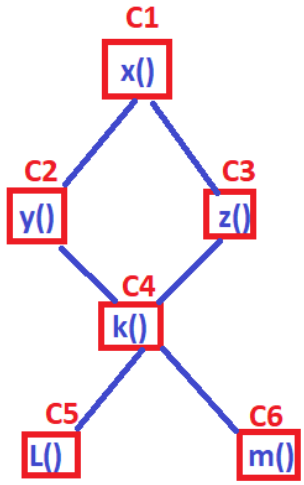
```
                print("Area of Rect={}".format(self.ar))
                print("-"*50)
                Circle.area(self,float(input("Enter Radiuos:")) )
                Square.area(self, float(input("Enter Side:")) )

#main program
l,b=float(input("Enter Length:")), float(input("Enter Breadth:"))
ro=Rect()
ro.area(l,b)
```

```
#Program cal area of different Figures such as circle,rect and square by
using method overriding for implementing polymorphism
#MethodOverridingEx8.py
class Circle:
        def  __init__(self, r):  # Original Constructor--One Form
                self.ac=3.14*r**2
                print("Area of Circle={}".format(self.ac))
class Square:
                def  __init__(self, s):  # Overridden Constructor
                        self.sa=s**2
                        print("Area of Square={}".format(self.sa))
                        print("-"*50)
class Rect(Circle,Square):
        def  __init__(self,l,b):  # Overridden Constructor
                self.ar=l*b
                print("Area of Rect={}".format(self.ar))
                print("-"*50)
                Circle.__init__(self,float(input("Enter Radiuos:")) )
                Square.__init__(self,float(input("Enter Side:")) )

#main program
l,b=float(input("Enter Length:")), float(input("Enter Breadth:"))
ro=Rect(l,b)
```

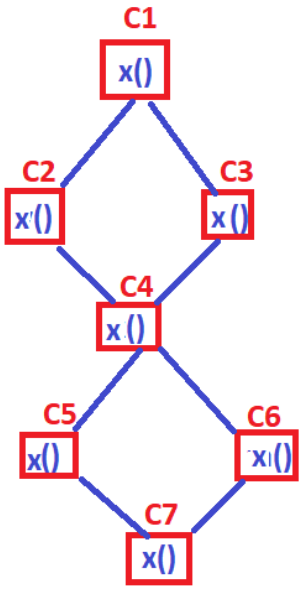
```
#This Program demonstrates Hybrid Inheritane by  using method overriding for
implementing polymorphism
#MethodOverridingEx9.py
class C1:
        def   x(self):
                print("x()--C1")
class C2(C1):
        def   x(self):
                print("x()--C2")
class C3(C1):
        def   x(self):
                print("x()--C3")
class C4(C2,C3):
        def   x(self):
                print("x()--C4")
class C5(C4):
        def   x(self):
                print("x()--C5")
class C6(C4):
        def   x(self):
                print("x()--C6")
class C7(C5,C6):
        def   x(self):
```

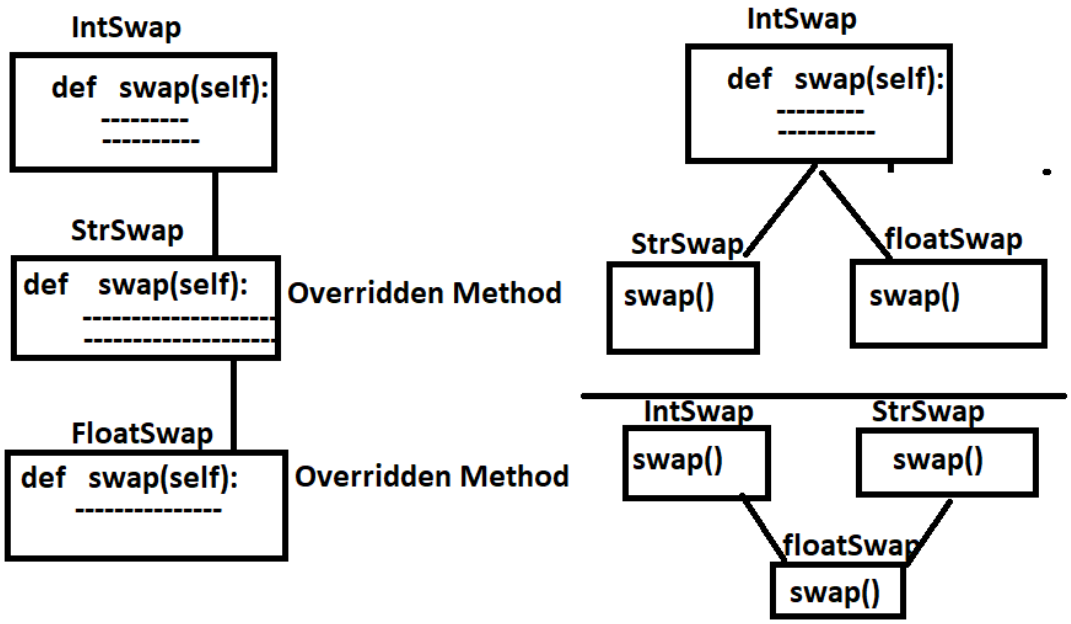
```
        print("x()--C7")
        C5.x(self)
        C6.x(self)
        C4.x(self)
        C3.x(self)
        C2.x(self)
        C1.x(self)

#main program
O7=C7()
O7.x()
```

Inheritance



Polymorphism<--Uses Method Overriding (Uses Inheritance)





String Handling concept

```
                    ========================================
                         String Handling in python (Part-2)
                    ========================================
=>On String data, we have performed alerady Indexing and slicing
=>Along with these operations, we can peform some other operation by using
pre-defined functions which are present in str class.

1) capitalize():
------------------------
=>This converts the first letter the sentense/ Paragraps as capital.
=>Syntax:-     varname=strobj.capitalize()

Examples:
------------------
>>> s="python programming. pytohn is an Fun Programming"
>>> s1=s.capitalize()
>>> print(s1)---Python programming. pytohn is an fun programming
>>> s.capitalize()---'Python programming. pytohn is an fun programming'
>>> print(s)---python programming. pytohn is an Fun Programming
--------------------------------------------------------------------------------
----------------------------------------
2) title()
---------------
=>This function is used for converting First letter of every word of given
string data.
=>Syntax:-    varname=strobj.title()
----------------------
Examples:
----------------------
>>> s="python programming. python is an Fun Programming"
```

```
>>> s1=s.title()
>>> print(s1)--------Python Programming. Python Is An Fun Programming
>>> print(s)---python programming. python is an Fun Programming
------------------------------------------------------------------------
-----------------------------------
3) find()
--------------------
=>This function is used finding the index of First Occurence of specified
word / letter.
=>If the specified word is not present then we get ValueError

Syntax:-    Indexvalue=strobj.index(sub string)
-------------------
Example:
-------------------
>>> s="Python is an oop lang"
>>> s.index('oop')--------13
>>> s.index('o')-----4
>>> s.index('lang')--------17
>>> s.index('ang')-----18
>>> s.index('kvr')--------ValueError: substring not found
------------------------------------------------------------------------
-----------------------------------
4) isalnum():
------------------------
=>This returns True provided str data is a combination of str or digits or
both
=>This returns False provided str data is a combination of str or digits
with special symbols .
Syntax:-     varname=strobj.isalnum()

Examples:
-------------------
>>> s="a1234b"
>>> b=s.isalnum()
>>> print(b)---------True
>>> s="Rossum"
>>> b=s.isalnum()
>>> print(b)----------True
>>> s="12345"
>>> b=s.isalnum()
>>> print(b)------------True
>>> s="a1234b#@"
>>> b=s.isalnum()
>>> print(b)---------False
>>> s="Rossum!"
>>> b=s.isalnum()
>>> print(b)---------False
------------------------------------------------------------------------
---------
5) isalpha()
-------------------
=>This function returns True provided str data must contain purely
alphabets.
```

```
   otherwise we get False
=>Syntax:-     varname=strobj.isalpha()

Examples:
-----------------------
>>> s="Rossum"
>>> b=s.isalpha()
>>> print(b)
True
>>> s="Rossum123"
>>> b=s.isalpha()
>>> print(b)
False
>>> s="1234"
>>> b=s.isalpha()
>>> print(b)
False
>>> s=" Rossum "
>>> b=s.isalpha()
>>> print(b)
False
--------------------------------------------------------------------------
----
6) isdigit():
------------------
=> =>This function returns True provided str data must contain purely
digits.
   otherwise we get False
=>Syntax:-     varname=strobj.isdigit()
--------------------
Examples:
--------------------
>>> s="1223"
>>> b=s.isdigit()
>>> print(b)
True
>>> s="a1223b"
>>> b=s.isdigit()
>>> print(b)
False
>>> s="1_2_2_3"
>>> b=s.isdigit()
>>> print(b)
False
>>> s="python"
>>> b=s.isdigit()
>>> print(b)
False
--------------------------------------------------------------------------
---------------------------
7) isspace():
----------------------
=>This Function returns True provided str object must contains purely
spaces otherwise we get False.
```

```
Syntax:-     varname=strobj.isspace()
---------------------
Examples:
--------------------
>>> s=" Rossum "
>>> b=s.isspace()
>>> print(b)
False
>>> s="      "
>>> b=s.isspace()
>>> print(b)
True
>>> s="1234"
>>> b=s.isspace()
>>> print(b)
False
-------------------------------------------------------------------------
----------------
8) islower()   -->Returns True provided str object data is purely lower
case                                  data otherwise it returns False
9) isupper() -->Returns True provided str object data is purely upper case
data otherwise it returns False

Examples:
-----------------
>>> s="PYTHON"
>>> b=s.islower()
>>> print(b)
False
>>> s="Python"
>>> b=s.islower()
>>> print(b)
False
>>> s="python"
>>> b=s.islower()
>>> print(b)
True
>>> s="PYTHON"
>>> b=s.isupper()
>>> print(b)
True
>>> s="PYThon"
>>> b=s.isupper()
>>> print(b)
False
>>> s="1234"
>>> b=s.isupper()
>>> print(b)
False
>>> s="1234"
>>> b=s.islower()
>>> print(b)
False
```

```
------------------------------------------------------------------------
-------------------
10) lower()-> This Function converts Upper Case data into lower case
11) upper()->This Function converts lower Case data into upper case

Syntax:      strobj1=strobj1.lower()
              strobj1=strobj1.upper()
-------------------
Examples:
-------------------
>>> s="PYTHON PROGRAMMING"
>>> lc=s.lower()
>>> print(lc)---------python programming
>>> uc=lc.upper()
>>> print(uc)----PYTHON PROGRAMMING
>>> s1="PyThOn PROGlang"
>>> uc=s1.upper()
>>> lc=s1.lower()
>>> print(uc)-----------PYTHON PROGLANG
>>> print(lc)-----------python proglang
------------------------------------------------------------------------
------------
12) join():
------------------------------------------------------------------------
=>This is used for concatenating different str values of any iterable
objects / Collection types.
Examples:
------------------
>>> l=["Python", " is ", " an ", " oop" " lang"]
>>> print(l,type(l))---['Python', ' is ', ' an ', ' oop lang'] <class
'list'>
>>> k=""
>>> k1=k.join(l)
>>> print(k1)-----Python is  an  oop lang
>>> print(k)
>>> l=["Python", " is ", " an ", 123, " lang"]
>>> print(l,type(l))---------['Python', ' is ', ' an ', 123, ' lang']
<class 'list'>
>>> k=""
>>> k1=k.join(l)---TypeError: sequence item 3: expected str instance, int
found
>>> l=["Python", " is ", " an ", "123", " lang"]
>>> k=""
>>> k1=k.join(l)
>>> print(k1)--------Python is  an 123 lang
------------------------------------------------------------------------
-------------------------------------
13) split()
-----------------
=>This function is used for splitting the given str data into different
Tokens based on delimeter. The default delimeter space.
=>We can also specify the programmer-defined delemiter.
Syntax:    listobj=strobj.split( str delimeter)
```

```
Examples:
--------------------
>>> s="Python is an oop lang"
>>> print(s)-------Python is an oop lang
>>> l=s.split()  # default delimeter is space
>>> print(l, type(l))----['Python', 'is', 'an', 'oop', 'lang'] <class
'list'>

>>> s="11-12-2021"
>>> print(s)
11-12-2021
>>> lst=s.split("-")
>>> print(lst)---------['11', '12', '2021']
>>> s="Mango#apple#kiwi/waterMillon"
>>> lst=s.split("#")
>>> print(lst)---['Mango', 'apple', 'kiwi/waterMillon']
>>> lst=s.split("/")
>>> print(lst)----['Mango#apple#kiwi', 'waterMillon']
--------------------------------------------------------------------------
--------
Note:
-----------
>>> x=65
>>> c=chr(x)
>>> print(c)
A
>>> x=97
>>> c=chr(x)
>>> print(c)
a
>>> x="A"
>>> v=ord(x)
>>> print(v)
65
>>> x="a"
>>> v=ord(x)
>>> print(v)
97
>>> x="d"
>>> v=ord(x)
>>> print(v)
100
>>> x="AB"
>>> print(x)
AB
>>> v=ord(x)---TypeError: ord() expected a character, but string of length
2                                    found
>>> x="a-A"
>>> v=ord(x)-------TypeError: ord() expected a character, but string of
length 3 found
====================================X==================
```

Introduction to Multi Threading

```
                        ========================================
                              Introduction to Multi Threading
                        ========================================
=>The purpose of multi threading is that "To provide Concurrent /
Simultaneous
    execution / Paralllel Execution".
=>Concurrent  Execution is nothing but executing the operations all at
once.
=>The advantage of Concurrent  execution is that to get less execution
time.
=>If a Python Program contains multiple threads then it is called Multi
Threading program.
-------------------------
=>Def. of thread:
-------------------------
=>A flow of Control is called thread.
=>The purpose of thread is that "To Perform certain operation whose logic
    developed in Functions / Methods  concurently."
-------------------------------------------------------------------------
-------------------------
=>By default Every Python contains Single Thread and whose name is
"MainThread" and It provides Sequential Execution.
=>Programtaically, In a Python Program we can create multiple sub  / Child
   threads  and whose purpose is that "To execute operations whose logic
is written in Functions / Methods Concurrently ".
=>Hnece Programatically a Python Program two types of Threads. They are
              a) MainThread
              b) Sub / Child Threads
=>MainThread is created / Initiated  by PVM when program exeuction starts
and the role of mainThread is to execute main program statements and
Monitor the exeuction status of Sub threads.
=>The Sub / Child Threads always executes operations whose logic is
written in Functions / Methods Concurrently ".
==========================X==================================
```

Module Name for thread based application development(partially discussed)

Number of approaches to develop thread based applications(partially discussed)

```
                    ======================================================
                     Module Name for thread based application development
                    ======================================================
=>In Python Programming, to develop thread based applications, we use a
pre-defined module called "threading".
=>We know that a module contains Variable Names, Functions and Classes.
=>Here "threading" module contains Variables , Functions and Classes
-------------------------------------------------------------------------
-------------------------------------
Functions in "threading" module
-------------------------------------------------------------------------
-------------------------------------
1) current_thread():
------------------------------
=>This Function is used for obtaining current thread which is executing by
default.
```

```
Syntax:-      varname=threading.current_thread()
-------------------------------
2) active_count():
-------------------------------
=>This function is used for obtaing the number threads which are under
execution. By default there exists only one thread as active thread and
whose name is "MainThread".
=>Syntax:-       varname=threading.active_count()
-------------------------------------------------------------------------
-------------------------------------
Class Name in "threading" module
-------------------------------------------------------------------------
-------------------------------------
=>threading module contains a two classes.  They are 1) Thread class

        2) Lock class
**********************************
Thread class Details
**********************************
=>The purpose of Thread class is that "To create sub thread(s) and whose
purpose is to execute the logic of python program which is written in
function / method "
=>Creating a sub thread is nothing but creating an object of Thread class.
-----------------------
Constructor:
-----------------------
1) Thread(target,args):
---------------------------------
=>This Constructor is used for creating a sub / child thread by specifying
the function name  for executing the logic as target attribute and passing
the values to specified function as args attribute(optional).
Syntax:
      subthreadname=threading.Thread(target=function name,
args=(arg1,arg2.... ) )


-----------------------------
Instance Methods:
-----------------------------
1) setName(str)
2) getName()
3) is_alive()
4) start()
5) join()
```

```
      ============================================================
       Number of approaches to develop thread based applications
      ============================================================
=>We can thread based applications in 3 ways. They are
1) By Using Functional Approach
2) By using sub class of Thread class ( with OOPs by using Inheritance)
3) By using non-sub class of Thread Class ( with OOPs without using
Inheritance)
-------------------------------------------------------------------------
-------------------------------------
```

1) By Using Functional Approach:

Steps:
---

1) import threading module
2) Define a Function where it contains Logic
3) Create a sub / child thread
4) Dispatch / start the sub / child thread to execute specified Function name, where it contains logic.

```
#ApproachEx1.py
import threading

def   greet():
       subthname=threading.current_thread().name
       print("Default Name of Sub thread={}".format(subthname))
       #set the programmer-defined name
       st1.name="hyd"
       subthname=threading.current_thread().name
       print("Programmer defined Name of Sub thread={}".format(subthname))
       print("Good Morning")

#main program
mtn=threading.current_thread().name
print("Name of main thread={}".format(mtn))
print("-"*40)
st1=threading.Thread(target=greet)
st1.start()
```

```
#Numgenex1.py
import threading,time
def generate(n):
       print("\nName of sub
thread={}".format(threading.current_thread().name))
       if(n<=0):
              print("{} is invalid input".format(n))
       else:
              print("-"*50)
              print("Numbers within :{}".format(n))
              print("-"*50)
              for i in range(1,n+1):
                     print("\tVal of i={}".format(i))
                     time.sleep(2)
              else:
                     print("-"*50)
#main program
print("Program execution started")
st1=threading.Thread(target=generate, args=(10,) )
print("\nExecution status of sub thread before start:", st1.is_alive())#
False
print("Line-20->Number thread active=",threading.active_count())#1
st1.name="Generater"
st1.start()
print("\nLine-23->Number thread active=",threading.active_count())#2
```

```
print("Line-24-Execution status of sub thread after start=",
st1.is_alive())# True
st1.join()
print("\nLine-26-->Execution status of sub thread after completion=",
st1.is_alive())# False
print("\nLine-23->Number thread active=",threading.active_count()) # 1
```

```
#threadcount.py
import threading,time

def   greet():
       subthname=threading.current_thread().name
       print("Name of Sub thread={}".format(subthname))
       print("Good Morning")

def  hello(sname):
       subthname=threading.current_thread().name
       print(" Name of Sub thread={}".format(subthname))
       print("Hi {}, How are u".format(sname))


#main program
mtn=threading.current_thread().name
print("Number of threads in this program=",threading.active_count())
print("Name of main thread={}".format(mtn))
print("-"*40)
st1=threading.Thread(target=greet)
st2=threading.Thread(target=hello,args=("Omprakash",) )
st1.start()
st2.start()
print("Number of threads in this program=",threading.active_count())
```

```
#This program makes us to understand obtaing current thread which is
executing by default
#threadex1.py
import threading
th=threading.current_thread().name
print("Program execution started and executed by default thread name=",th)
"""  OR
th=threading.current_thread()
tname=th.name
print("Program execution started and executed by default thread
name=",tname)
"""
nt=threading.active_count()
print("Number of threads active by default=",nt)
a=10
b=20
c=a+b
print("sum({},{})={}".format(a,b,c))
```

Number of approaches to develop thread based applications

Module Name for thread based application development

programs

```
       ============================================================
       Number of approaches to develop thread based applications
       ============================================================
=>We can thread based applications in 3 ways. They are
1) By Using Functional Approach
2) By using sub class of Thread class ( with OOPs by using Inheritance)
3) By using non-sub class of Thread Class ( with OOPs without using
Inheritance)
-------------------------------------------------------------------------
-----------------------------------
1) By Using Functional Approach:
-------------------------------------------------------------
Steps:
-------------------------------------------------------------
       1) import threading module
       2) Define a Function where it contains Logic
       3) Create a sub / child thread
       4) Dispatch / start the sub / child thread to execute specified
Function name,        where it contains logic.
Examples:
#ApproachEx1.py
import threading   # step-1

def   greet():   # step-2
       subthname=threading.current_thread().name
       print("Default Name of Sub thread={}".format(subthname))
       #set the programmer-defined name
       st1.name="hyd"
       subthname=threading.current_thread().name
       print("Programmer defined Name of Sub thread={}".format(subthname))
       print("Good Morning")

#main program
mtn=threading.current_thread().name
print("Name of main thread={}".format(mtn))
print("-"*40)
st1=threading.Thread(target=greet)  # Step-3
st1.start()  # Step-4
-------------------------------------------------------------------------
-------------------------------
2) By using sub class of Thread class ( with OOPs by using Inheritance):
-------------------------------------------------------------------------
-----------------------------------
step-1:     import  threading
step-2:    Choose the Programmer-defined class
Step-3:   The Programmer-defined class must inherit from Thread Class
Step-4:    Override    run(self) in programmer-defined class, which was
Inherited
               from Thread Class
```

```
Step-5: create an object of Programmer-defined sub class of Thread ( is
nothing an
             object of Thread class)
Step-6:  start / dispatch the sub thread where sub thread enters into
overridden
              run() of Programmer-defined class, which was inherited from
Thread class
---------------------------
Examples:
---------------------------
              #Approach2.py
              import threading  # step-1
                             #step- 2   step-3
              class Sample(threading.Thread):
                     def   run(self):  # step-4
                             print("i am from run:")

              #main program
              st1=Sample()  # step-5
              st1.start()    # step-6
-------------------------------------------------------------------------
---------------------------------------
3) By using non-sub class of Thread Class ( with OOPs without using
Inheritance)
-------------------------------------------------------------------------
-----------------------------------------
step-1:     import  threading
Step-2:   Choose the programmer-defined  class
Step-3:   define a method in a programmer defined class where a method
contains
              logic , which is executed by sub thread
Step-4:  Create an object of Programmer-defined class for calling Instance
Method
             by the sub thread
Step-5:  create sub thread for executing logic of python program, which is
              available in the method defined in Programmer-defined class.
Step-6:  Dispatch / start the execution of sub thread.

Examples:
--------------------
#Approach3.py
import threading  # step-1
class Test:  # Ste-2
       def   welcome(self,sname):   #step-3
              print("\nHi:{}, Good Morning:".format(sname))


# main program
t=Test()  # step-4
st1=threading.Thread(target=t.welcome, args=("Rossum",)  ) # step-5
st1.start() # step-6

==============================X================================
```

```
            ============================================================
                 Module Name for thread based application development
             ============================================================
=>In Python Programming, to develop thread based applications, we use a
pre-defined module called "threading".
=>We know that a module contains Variable Names, Functions and Classes.
=>Here "threading" module contains Variables , Functions and Classes
-------------------------------------------------------------------------
-------------------------------------
Functions in "threading" module
-------------------------------------------------------------------------
-------------------------------------
1) current_thread():
------------------------------
=>This Function is used for obtaining current thread which is executing by
default.
Syntax:-      varname=threading.current_thread()
--------------------------------
2) active_count():
------------------------------
=>This function is used for obtaing the number threads which are under
execution. By default there exists only one thread as active thread and
whose name is "MainThread".
=>Syntax:-       varname=threading.active_count()
-------------------------------------------------------------------------
-------------------------------------
Class Name in "threading" module
-------------------------------------------------------------------------
-------------------------------------
=>threading module contains a two classes.  They are 1) Thread class

        2) Lock class
**********************************
Thread class Details
**********************************
=>The purpose of Thread class is that "To create sub thread(s) and whose
purpose is to execute the logic of python program which is written in
function / method "
=>Creating a sub thread is nothing but creating an object of Thread class.
----------------------
Constructor:
----------------------
1) Thread(target,args):
---------------------------------
=>This Constructor is used for creating a sub / child thread by specifying
the function name  for executing the logic as target attribute and passing
the values to specified function as args attribute(optional).
Syntax:
      subthreadname=threading.Thread(target=function name,
args=(arg1,arg2.... ) )
------------------------------
Instance Methods:
------------------------------
1) setName(str):
```

```
-------------------------------------------------
=>This Function is used for setting Programmer-defined name to the thread.

=>Syntax:-     threadobjname.setName("Programmer-defined name")

=>Since setName() is deprecated to an attribute "name" and in the program
it is recommened to use "name" attribute.

=>New Syntax:-       threadobjname.name="Programmer-defined name"
-------------------------------------------------------------------------
---------------------------------------
2) getName():
-------------------------
=>This Function is used for getting the thread name
=>Syntax:-      varname=threadobjname.getName()
=>Since getName() is deprecated to an attribute "name" and in the program
it is recommened to use "name" attribute.
=>New Syntax:-      varname=threadobjname.name
=>here "varname" contains programmer-defined thread name.
-------------------------------------------------------------------------
---------------------------------------
3) is_alive():
-------------------------------------
=>This Function returns True provided the sub thread / child thread is
under exec ution.
=>This Function returns False  in the case sub thread not yet started or
after completion sub thread execution.
=>Syntax:-      varname=threadobjname.is_alive()
=>here "varname" contains boolean value ( True or False)
-------------------------------------------------------------------------
---------------------------------------
4) start():
-----------------------------------
=>This function is used for dispatching / starting the sub thread to
target function where thread execution  started.
=>Syntax:-        threadobjname.start()
-------------------------------------------------------------------------
---------------------------------------
5) join():
---------------------------------
=>This function is used for making the sub thread(s) to join as single
unit and ensures that main thread join them as single unit and at a time
hand over to GC.

Syntax:-               threadobjname1.join()
                          threadobjname2.join()
                          ---------------------------------
                          threadobjname-n.join()
-------------------------------------------------------------------------
---------------------------------
6) run(self):
-----------------------------------
```

=>This function is used for defining logic of python program , which is executed by sub thread provided thread based application developed with OOPs based approach(With Inheritance).

==========X==========

```
#Approach2.py
import threading  # step-1
              #step- 2   step-3
class Sample(threading.Thread):
       def   run(self):  # step-4
              print("i am from run:")

#main program
st1=Sample()  # step-5
st1.start()    # step-6
```

```
#Approach21.py
from threading import Thread,current_thread
import time
class NumGen(Thread):
       def   run(self):
              print("\nName of sub thread un
run()=",current_thread().name)
              n=int(input("Enter how many number u want to generate:"))
              if(n<=0):
                     print("{} is invalid number:".format(n))
              else:
                     print("-"*50)
                     print("Numbers within {}".format(n))
                     print("-"*50)
                     for i in range(1,n+1):
                            print("\tVal of i={}".format(i))
                            time.sleep(1)
                     print("-"*50)



#main program
print("Current Thread Name in main program=",current_thread().name)
no=NumGen()  # creating an object NumGen class is nothing creating an
object of
                                                                  #Thread
class
no.start()
```

```
#Approach3.py
import threading  # step-1
class Test:  # Ste-2
       def   welcome(self,sname):   #step-3
              print("\nHi:{}, Good Morning:".format(sname))


# main program
t=Test()  # step-4
st1=threading.Thread(target=t.welcome, args=("Rossum",)  ) # step-5
st1.start() # step-6
```

```
#Approach31.py
import threading
class Test:
       def   welcome(self,sname):
              print("\nHi:{}, Good Morning:".format(sname))

class Sample:
       def   hello(self):
              print("Wel come to Python Class")

# main program
t=Test()
st1=threading.Thread(target=t.welcome, args=("Rossum",)  )
st1.start()

st2=threading.Thread(target=Sample().hello)
st2.start()
```

```
#This Program accept line of text and display every character after one
second
#chargen.py
import threading,time
def   charactergen(line):
       print("-"*40)
       print("\nGiven Line:{}".format(line))
       print("-"*40)
       for char in line:
              print("\t{}".format(char))
              time.sleep(1)
       print("-"*40)

#main program
t1=threading.Thread(target=charactergen,args=( (input("Enter a line of
text:"),) ) )
t1.start()
```

```
#This Program accept line of text and display every character after one
second
#chargen2.py---with oops inheritance
import threading,time
class Character(threading.Thread):
       def   setvalue(self):
              self.line=input("Enter a line of text:")
       def  run(self):
              self.setvalue()
              print("-"*40)
              print("\nGiven Line:{}".format(self.line))
              print("-"*40)
              for char in self.line:
                     print("\t{}".format(char))
                     time.sleep(1)
              print("-"*40)

#main program
c=Character()
c.start()
```

```
#This Program accept line of text and display every character after one
second
#chargen3.py---with oops  without inheritance
import threading,time
class Character:
       def   setvalue(self):
              self.line=input("Enter a line of text:")
       def  chargen(self):
              self.setvalue()
              print("-"*40)
              print("\nGiven Line:{}".format(self.line))
              print("-"*40)
              for char in self.line:
                     print("\t{}".format(char))
                     time.sleep(1)
              print("-"*40)

#main program
t=threading.Thread(target=Character().chargen)
t.start()
```

```
#threadcountex.py
import threading,time

def   greet():
       subthname=threading.current_thread().name
       print("Name of Sub thread={}".format(subthname))
       print("Good Morning")
       time.sleep(5)

def  hello(sname):
       subthname=threading.current_thread().name
       print(" Name of Sub thread={}".format(subthname))
       print("Hi {}, How are u".format(sname))
       time.sleep(5)

#main program
mtn=threading.current_thread().name
print("Number of threads in this program=",threading.active_count())# 1
print("Name of main thread={}".format(mtn))
print("-"*40)
st1=threading.Thread(target=greet)  # sub thread1----thread-1
st2=threading.Thread(target=hello,args=("Omprakash",) )# sub thread2---
thread-2
st1.start()
st2.start()
print("Number of threads in this program=",threading.active_count())# 3
st1.join()
st2.join()
print("Number of threads after completion of program execution in this
program=",threading.active_count())# 1
```

Synchronization in Multi Threading

or dead locks in Python

```
                ================================================
                        Synchronization in Multi Threading
                ================================================
=>When multiple threads are operating / working on the same
resource(function / method) then by default we get dead lock result / race
condition / wrong result.
=>To overcome this dead lock problems, we must apply the concept
Synchronization conncept.
=>The advantage of synchronization concept is that to avoid dead lock
result and provides Thread Safety Result.
=>In Python Programming, we can obtain synchronization concept by using
locking and un-locking concept.
-------------------------------------------------------------------------
------------
=>Steps for implementing Synchronization Concept:
-------------------------------------------------------------------------
------------
1) obtain / create an object of Lock class, which is present in threading
module.
       Stntax:-
       ----------------
                              lockobj=threading.Lock()
2) To obtain the lock on the sharable resource, we must use acquire()
               Syntax:
               --------------
                              lockobj.acquire()
       Once current object acquire the lock, other objects are made wait
until curent object releases the lock.
3) To un-lock the sharable resource/current object, we must use release()
       Syntax:
       -------------
                              lockobj.release()

       Once current object releases the lock, other objects are permitted
into shrable resource. This process of aquiring the releasing the lock
will be continued until all the objects completed their execution.
```

```
#lockfunex1.py
import time,threading
def   multable(n):
       #get the lock object
       l1.acquire()
       print("Sub thread Name=",threading.current_thread().name)
       if(n<=0):
              print("\n{} is invalid input".format(n))
       else:
              print("-"*50)
              print("\nMul Table for :{}".format(n))
              print("-"*50)
              for i in range(1,11):
                     print("\t{} x {}={}".format(n,i,n*i))
                     time.sleep(1)
```

```
                else:
                        print("-"*50)
        #release the lock
        l1.release()
#main program
#create an object of Lock class
l1=threading.Lock()
#create sub threads
t1=threading.Thread(target=multable,args=(10,))
t2=threading.Thread(target=multable,args=(12,))
t3=threading.Thread(target=multable,args=(19,))
t4=threading.Thread(target=multable,args=(-5,))
#dispatch the threads
t1.start()
t2.start()
t3.start()
t4.start()
```

```
#lockoopex1.py
import threading,time
class MulTable(threading.Thread):
        def setvalue(self,n):
                self.n=n
        def  run(self):
                #get the lock
                L.acquire()
                print("Sub thread Name=",threading.current_thread().name)
                if(self.n<=0):
                        print("\n{} is invalid input".format(self.n))
                else:
                        print("-"*50)
                        print("\nMul Table for :{}".format(self.n))
                        print("-"*50)
                        for i in range(1,11):
                                print("\t{} x
{}={}".format(self.n,i,self.n*i))
                                time.sleep(1)
                        else:
                                print("-"*50)
                #release the lock
                L.release()
#main program
#create a lock object
L=threading.Lock()
#create multiple sub threads
mt1=MulTable()
mt2=MulTable()
mt3=MulTable()
mt4=MulTable()
#set the values
mt1.setvalue(10)
mt2.setvalue(12)
mt3.setvalue(19)
mt4.setvalue(-10)
#dispatch the threads
```

```
mt1.start()
mt2.start()
mt3.start()
mt4.start()
```

```
#lockoopex2.py
import threading,time
class MulTable:
	def  __init__(self):
		self.l=threading.Lock()  # create an object of Lock class

	def  table(self,n):
		#get the lock
		self.l.acquire()
		print("Sub thread Name=",threading.current_thread().name)
		if(n<=0):
			print("\n{} is invalid input".format(n))
		else:
			print("-"*50)
			print("\nMul Table for :{}".format(n))
			print("-"*50)
			for i in range(1,11):
				print("\t{} x {}={}".format(n,i,n*i))
				time.sleep(1)
			else:
				print("-"*50)
		#release the lock
		self.l.release()
#main program
#create multiple sub threads
t=MulTable()
mt1=threading.Thread(target=t.table,args=(10,))
mt2=threading.Thread(target=t.table,args=(12,))
mt3=threading.Thread(target=t.table,args=(19,))
mt4=threading.Thread(target=t.table,args=(-5,))
#dispatch the threads
mt1.start()
mt2.start()
mt3.start()
mt4.start()
```

```
#lockreservationex1.py
import threading,time
class Reservation:
	def __init__(self):
		self.nos=10
		self.l=threading.Lock()
	def  booktickets(self,nos):
		#get the lock
		self.l.acquire()
		if(nos>self.nos):
			print("Hi {},U dont have {}
Seats:".format(threading.current_thread().name,nos))
			time.sleep(1)
		else:
			self.nos=self.nos-nos
```

```
                        print("\nHi {}, {} Seats
Reserved:".format(threading.current_thread().name,nos))
                        print("Visit again-Happy Journey!")
                        time.sleep(1)
                #release the lock
                self.l.release()
#main program
R=Reservation()
p=threading.Thread(target=R.booktickets, args=(1,))
p1=threading.Thread(target=R.booktickets, args=(20,))
p1.name="Ramu"
p2=threading.Thread(target=R.booktickets, args=(2,))
p2.name="Raju"
p3=threading.Thread(target=R.booktickets, args=(6,))
p3.name="Rakesh"
p4=threading.Thread(target=R.booktickets, args=(2,))
p4.name="Rossum"
p5=threading.Thread(target=R.booktickets, args=(1,))
p5.name="Ramesh"
#dispatch the threads
p.start()
p1.start()
p2.start()
p3.start()
p4.start()
p5.start()
```

```
#nonlockfunex1.py
import time,threading
def   multable(n):
       print("Sub thread Name=",threading.current_thread().name)
       if(n<=0):
              print("\n{} is invalid input".format(n))
       else:
              print("-"*50)
              print("\nMul Table for :{}".format(n))
              print("-"*50)
              for i in range(1,11):
                     print("\t{} x {}={}".format(n,i,n*i))
                     time.sleep(1)
              else:
                     print("-"*50)

#main program
t1=threading.Thread(target=multable,args=(10,))
t2=threading.Thread(target=multable,args=(12,))
t3=threading.Thread(target=multable,args=(19,))
t4=threading.Thread(target=multable,args=(-5,))
#dispatch the threads
t1.start()
t2.start()
t3.start()
t4.start()
```

```
#nonlockoopex1.py
import threading,time
class MulTable(threading.Thread):
        def setvalue(self,n):
                self.n=n
        def  run(self):
                print("Sub thread Name=",threading.current_thread().name)
                if(self.n<=0):
                        print("\n{} is invalid input".format(self.n))
                else:
                        print("-"*50)
                        print("\nMul Table for :{}".format(self.n))
                        print("-"*50)
                        for i in range(1,11):
                                print("\t{} x
{}={}".format(self.n,i,self.n*i))
                                time.sleep(1)
                        else:
                                print("-"*50)
#main program
#create multiple sub threads
mt1=MulTable()
mt2=MulTable()
mt3=MulTable()
mt4=MulTable()
#set the values
mt1.setvalue(10)
mt2.setvalue(12)
mt3.setvalue(19)
mt4.setvalue(-10)
#dispatch the threads
mt1.start()
mt2.start()
mt3.start()
mt4.start()
```

```
#nonlockoopex2.py
import threading,time
class MulTable:
        def  table(self,n):
                print("Sub thread Name=",threading.current_thread().name)
                if(n<=0):
                        print("\n{} is invalid input".format(n))
                else:
                        print("-"*50)
                        print("\nMul Table for :{}".format(n))
                        print("-"*50)
                        for i in range(1,11):
                                print("\t{} x {}={}".format(n,i,n*i))
                                time.sleep(1)
                        else:
                                print("-"*50)
#main program
#create multiple sub threads
t=MulTable()
```

```
mt1=threading.Thread(target=t.table,args=(10,))
mt2=threading.Thread(target=t.table,args=(12,))
mt3=threading.Thread(target=t.table,args=(19,))
mt4=threading.Thread(target=t.table,args=(-5,))
#dispatch the threads
mt1.start()
mt2.start()
mt3.start()
mt4.start()
```

```
#nonlockreservationex1.py
import threading,time
class Reservation:
       def __init__(self):
              self.nos=1

       def  booktickets(self,nos):
              if(nos>self.nos):
                     print("Hi {},U dont have {}
Seats:".format(threading.current_thread().name,nos))
                     time.sleep(1)
              else:
                     self.nos=self.nos-nos
                     print("\nHi {}, {} Seats
Reserved:".format(threading.current_thread().name,nos))
                     print("Visit again-Happy Journey!")
                     time.sleep(1)

#main program
R=Reservation()
p1=threading.Thread(target=R.booktickets, args=(20,))
p1.name="Ramu"
p2=threading.Thread(target=R.booktickets, args=(2,))
p2.name="Raju"
p3=threading.Thread(target=R.booktickets, args=(6,))
p3.name="Rakesh"
p4=threading.Thread(target=R.booktickets, args=(2,))
p4.name="Rossum"
p5=threading.Thread(target=R.booktickets, args=(1,))
p5.name="Ramesh"
#dispatch the threads
p1.start()
p2.start()
p3.start()
p4.start()
p5.start()
```

---

random module

```
              ========================================
                     random module
              ========================================
=>random one of pre-defined module present in python
=>The purpose of random is that "To generate random values in various
contexts".
=>random module contains the follwoing essential functions.
              a) randrange()
```

```
              b) randint()
              ---------------------------------
              c) random()
              d) uniform()
              ---------------------------------
              e) choice()
              f)  shuffle()
              g) sample()
              ---------------------------------------
============================================================
a) randrange()
  -------------------------
=>This function is used for generating random integer values between
specified limits.
Syntax1:-         random.randrang(Value)
              This syntax generates any random value between 0 to Value-1

Syntax-2:             random.rangerange(start,stop)
              This syntax generates any random value between start to
stop-1

Examples:
---------------
>>> import random
>>> print(random.randrange(100,150))----133
>>> print(random.randrange(100,150))----121
>>> print(random.randrange(100,150))----139
>>> print(random.randrange(100,150))----143
>>> print(random.randrange(100,150))---106
>>> print(random.randrange(100,150))---133
>>> print(random.randrange(10))----5
>>> print(random.randrange(10))----9
-----------------------------------------------------
#randrangeex.py
import random
for i in range(1,6):
       print(random.randrange(10))
print("---------------------------------------")
for i in range(1,6):
       print(random.randrange(1000,1100))
print("---------------------------------------")
================================X============================
b) randint():
--------------------
=>Syntax:-     random.radint(start,stop)
=>This syntax generates any random value between start to stop. Here start
and stop are inclusive.
Examples:
----------------
>>> print(random.randint(10,15))------10
>>> print(random.randint(10,15))-----13
>>> print(random.randint(10,15))----14
>>> print(random.randint(10,15))----11
>>> print(random.randint(10,15))----15
```

```
-----------------------------------------------------------
#randintex.py
import random
for i in range(1,6):
       print(random.randint(10,20))
print("----------------------------------------")
==============================X================================
c) random()
-----------------------
=>Syntax:-      random.random()
=>This syntax generates floating point random values between 0.0 and 1.0
(Exlusive))
Examples:
----------------
>>> import random
>>> print(random.random())----------0.1623906138450063
>>> print(random.random())--------0.15382209709271966
>>> print(random.random())-------0.09542283007844476
>>> print(random.random())-----0.6134301633766425
-------------------------
#randomex.py
import random
lst=[]
for i in range(1,6):
       lst.append("%0.2f" %random.random())
print("----------------------------------------")
print("Content of lst={}".format(lst))
============================X============================
d) uniform()
----------------------
Syntax:-     random.uniform(start,stop)
=>This generates random floting point values from start to stop-1 values
=>The values of start and stop can both Integer or floating point values.
Examples:
----------------
>>> import random
>>> print(random.uniform(10,15))----------14.416746067678286
>>> print(random.uniform(10,15))----13.2420406264978
>>> print(random.uniform(10,15))-----11.716110933506432
>>> print(random.uniform(10,15))--------10.703499588966528
>>> print(random.uniform(10,15))-----11.306226559323017
>>> print(random.uniform(10.75,15.75))--------13.939787347170148
>>> print(random.uniform(10.75,15.75))----10.760428232717597
-----------------------------------------------------------------------------
-----------------
#uniformex.py
import random
lst=[]
for i in range(1,6):
       lst.append(float("%0.2f" %random.uniform(10,15.5)))
print("----------------------------------------")
print("Content of lst={}".format(lst))
============================X================================
e) choice():
```

```
-------------------
Syntax:-     random.choice(Iterable_object)
=>This function obtains random values from Iterable_object.
---------------------
EXAMPLES:
---------------------
>>>
print(random.choice([10,20,30,40,50]),random.choice("PYTHON"),random.choic
e(range(10,15)))---40 T 11
>>>
print(random.choice([10,20,30,40,50]),random.choice("PYTHON"),random.choic
e(range(10,15)))----------30 P 12
>>>
print(random.choice([10,20,30,40,50]),random.choice("PYTHON"),random.choic
e(range(10,15)))-----------40 N 12
-------------------------------------
#choiceex.py
import random
s="AaBRe#^%@8YuQLPau*&"
for i in range(1,6):

print(random.choice(s),random.choice(s),random.choice(s),random.choice(s))
==========================X================================
f)  shuffle():
--------------------
=>This Function is used for re-organizing the elements of any mutable
object.

Syntax:-    random.shuffle(list)
=>We can shuffle the data of list but not other objects of Data Types
Examples:
-------------------
>>> d={10:"cadburry",20:"kitkat",30:"malkybar", 40:"dairymilk"}
>>> print(d)---{10: 'cadburry', 20: 'kitkat', 30: 'malkybar', 40:
'dairymilk'}
>>> for k,v in d.items():
...     print(k,"--",v)
...
       10 -- cadburry
       20 -- kitkat
       30 -- malkybar
       40 -- dairymilk
>>> import random
>>> print(random.shuffle(d))----Traceback (most recent call last):
                                                    File "<stdin>", line
1, in <module>
                                                    File
"C:\Users\nareshit\AppData\Local\Programs\Python\Python310\lib\random.py",
line 394, in shuffle
                                                      x[i], x[j] = x[j],
x[i]
                                                  KeyError: 3
>>> s={10,20,30,40,50}
>>> print(random.shuffle(s))
```

```
                                                Traceback (most recent call
last):
                                                  File "<stdin>", line 1, in
<module>
                                                  File
"C:\Users\nareshit\AppData\Local\Programs\Python\Python310\lib\random.py",
line 394, in shuffle
                                                    x[i], x[j] = x[j], x[i]
                                                TypeError: 'set' object is not
subscriptable

>>> t=(10,20,30,40,50)
>>> print(random.shuffle(t))
                                                Traceback (most recent call
last):
                                                  File "<stdin>", line 1, in
<module>
                                                  File
"C:\Users\nareshit\AppData\Local\Programs\Python\Python310\lib\random.py",
line 394, in shuffle
                                                    x[i], x[j] = x[j], x[i]
                                                TypeError: 'tuple' object does
not support item assignment
>>> l=[10,20,30,40,50]
>>> print(random.shuffle(l))-----None
>>> print(l)-------------[30, 40, 50, 10, 20]
>>> random.shuffle(l)
>>> print(l)------------[40, 30, 10, 20, 50]
>>> random.shuffle(l)
>>> print(l)---------[40, 10, 50, 20, 30]
>>> random.shuffle(l)
>>> print(l)------------[30, 50, 20, 40, 10]

#shuffleex.py
import random as r
l=[10,"Python","Rossum",34.56,True]
for i in range(1,6):
       r.shuffle(l)
       print("content of l=",l)
==================================X============================
g) sample()
------------------
=>This Function is used for selecting random samples from any Iterable
object based on number of samples(+ve)
Syntax:-       random.sample(iterable_object, k)
=>Here 'k' can be number of samples.

Examples:
------------------
>>> import random
>>> s="ABCabcERTYUertyu$%^&*#@!%^&ghjkiyl"
>>> print(random.sample(s,5))----------['A', '*', '^', 'j', 't']
>>> print(random.sample(s,5))---------['%', 'l', 'b', 'C', 'y']
>>> print(random.sample(s,5))----------['%', 'e', 'Y', 'j', 'u']
```

```
>>> print(random.sample(s,5))------['y', 'E', '&', '$', '#']
>>> print(random.sample(s,5))----------['j', '*', 't', '$', 'u']
-----------------------------------------------------------------
#sampleex.py
import random
lst=[10,"Rossum","Python",34.56,True]
for i in range(1,6):
       print(random.sample(lst,2))
==============================X=========================
#choiceex.py
import random
s="AaBRe#^%@8YuQLPau*&"
for i in range(1,6):

print(random.choice(s),random.choice(s),random.choice(s),random.choice(s))
```

```
#randintex.py
import random as r
for i in range(1,6):
       print(r.randint(100,105))
```

```
#randomex.py
import random
lst=[]
for i in range(1,6):
       lst.append("%0.2f" %random.random())
print("---------------------------------------")
print("Content of lst={}".format(lst))
```

```
#randrangeex.py
import random as r
for i in range(1,10):
       print(r.randrange(100,150))
```

```
#sampleex.py
import random
lst=[10,"Rossum","Python",34.56,True]
for i in range(1,6):
       print(random.sample(lst,2))
```

```
#shuffleex.py
import random as r
l=[10,"Python","Rossum",34.56,True]
for i in range(1,6):
       r.shuffle(l)
       print("content of l=",l)
```

```
#uniformex.py
import random
lst=[]
for i in range(1,6):
       lst.append(float("%0.2f" %random.uniform(10,15.5)))
print("---------------------------------------")
print("Content of lst={}".format(lst))
-------------------------------------------------------------------------
```