In [1]:

```
Agenda = '''
1. Module
a) How to import ?
b) Built in math
c) Create User Define Module
d) os
e) sys
f) pickle / unpickle ( Object Serialisation )
g) glob
h) json

2. Object Oriented Programming
a) What is class and Object ?
b) Create Class with pass
c) Method
d) difference between Method and Function
e) use of self ? What it is ? Why ?
f) Object variable / attribute
g) Class Variable
h) Understand difference between Object and
Class attribute
i) Constructor ( Class Initialisation )
Use of __init__ method ( built in method )
j) Inheritance
   Subclass / Superclass
k) Overriding ( Polymorphism )
l) Difference between Overriding and Overloading
m) user of super keyword
n) Demo Class Application
o) Group Assignment for Developing Three Object
oriented Application
p)Group Discussion and Quiz

'''

```

In [2]:

```
print(Agenda)
```

```
1. Module
a) How to import ?
b) Built in math
c) Create User Define Module
d) os
e) sys
f) pickle / unpickle ( Object Serialisation )
g) glob

2. Object Oriented Programming
a) What is class and Object ?
b) Create Class with pass
c) Method
d) difference between Method and Function
e) use of self ? What it is ? Why ?
f) Object variable / attribute
g) Class Variable
h) Understand difference between Object and
Class attribute
i) Constructor ( Class Initialisation )
Use of __init__ method ( built in method )
j) Inheritance
   Subclass / Superclass
k) Overriding ( Polymorphism )
l) Difference between Overriding and Overloading
m) user of super keyword
n) Demo Class Application
o) Group Assignment for Developing Three Object
oriented Application
p)Group Discussion and Quiz
```

In [3]:

```
import numpy
```

In [4]:

```
import socket
```

In [6]:

```
#dir(socket)
```

In [8]:

```
import threading
#dir(threading)
```

In [9]:

```
import math
```

In [10]:

```
dir(math)
```

Out[10]:

```
['__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'acos',
 'acosh',
 'asin',
 'asinh',
 'atan',
 'atan2',
 'atanh',
 'ceil',
 'copysign',
 'cos',
 'cosh',
 'degrees',
 'e',
 'erf',
 'erfc',
 'exp',
 'expm1',
 'fabs',
 'factorial',
 'floor',
 'fmod',
 'frexp',
 'fsum',
 'gamma',
```

```
'gcd',
'hypot',
'inf',
'isclose',
'isfinite',
'isinf',
'isnan',
'ldexp',
'lgamma',
'log',
'log10',
'log1p',
'log2',
'modf',
'nan',
'pi',
'pow',
'radians',
'remainder',
'sin',
'sinh',
'sqrt',
'tan',
'tanh',
'tau',
'trunc']
```

In [12]:

```
1  #help(math)
```

In [13]:

```
1  math.pi
```

Out[13]:

```
3.141592653589793
```

In [14]:

```
1  math.sin(90)
```

Out[14]:

```
0.8939966636005579
```

In [15]:

```
1 math.factorial(5)
```

Out[15]:

```
120
```

In [16]:

```
1 from math import sin, cos, pi
```

In [17]:

```
1 sin(90)
```

Out[17]:

```
0.8939966636005579
```

In [18]:

```
1 cos(90)
```

Out[18]:

```
-0.4480736161291701
```

In [19]:

```
1 pi
```

Out[19]:

```
3.141592653589793
```

In [20]:

```
1 from math import *
```

In [21]:

```
1 e
```

Out[21]:

```
2.718281828459045
```

In [22]:

```
1 log(9)
```

Out[22]:

```
2.1972245773362196
```

In [23]:

```
1 pow(2,5)
```

Out[23]:

```
32.0
```

In [25]:

```
1 factorial(5)
```

Out[25]:

```
120
```

In [26]:

```
1 help(cos)
```

```
Help on built-in function cos in module math:

cos(x, /)
    Return the cosine of x (measured in radians).
```

In [28]:

```
help(log)
```

```
Help on built-in function log in module math:

log(...)
    log(x, [base=math.e])
    Return the logarithm of x to the given base.

    If the base not specified, returns the natural logarithm (base e
) of x.
```

In [29]:

```
import math as M
```

In [30]:

```
M.sin(90)
```

Out[30]:

```
0.8939966636005579
```

In [31]:

```
help(M.fsum)
```

```
Help on built-in function fsum in module math:

fsum(seq, /)
    Return an accurate floating point sum of values in the iterable
seq.

    Assumes IEEE-754 floating point arithmetic.
```

In [32]:

```
M.fsum([3.2, 6.9, 4.0])
```

Out[32]:

```
14.100000000000001
```

In [34]:

```
del math
```

In [36]:

```
SUM1 = M.fsum([3.2, 6.9, 4.0])
```

In [37]:

```
SUM1
```

Out[37]:

```
14.100000000000001
```

In [38]:

```
%%writefile mathacc.py

'''mathacc module
This is a math accenture module for basic mathemical
function and data operation.
'''
VERSION = 1.0
module_name = 'mathacc module'


def addition(num1, num2):
    '''Addition of two numbers'''
    return num1 + num2


def substraction(num1, num2):
    '''Substraction of two numbers'''
    return num1 - num2


def multiplication(num1, num2):
    '''Multiplication of two numbers'''
    return num1 * num2


def division(num1, num2):
    '''Division of two numbers'''
    return num1 / num2


def modulus(num1, num2):
    '''modulus of two numbers'''
    return num1 % num2


def average(num1, num2):
    '''Average of two numbers'''
    return ( num1 + num2 ) / 2


```

```
Writing mathacc.py
```

In [39]:

```
import mathacc
```

In [40]:

```
1 dir(mathacc)
```

Out[40]:

```
['VERSION',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'addition',
 'average',
 'division',
 'module_name',
 'modulus',
 'multiplication',
 'substraction']
```

In [41]:

```
1 help(mathacc)
```

```
Help on module mathacc:

NAME
    mathacc

DESCRIPTION
    mathacc module
    This is a math accenture module for basic mathemical
    function and data operation.

FUNCTIONS
    addition(num1, num2)
        Addition of two numbers

    average(num1, num2)
        Average of two numbers

    division(num1, num2)
        Division of two numbers

    modulus(num1, num2)
        modulus of two numbers

    multiplication(num1, num2)
        Multiplication of two numbers

    substraction(num1, num2)
        Substraction of two numbers

DATA
    VERSION = 1.0
    module_name = 'mathacc module'

FILE
    /Users/surendra/mathacc.py
```

In [42]:

```
1 mathacc.addition(45,90)
```

Out[42]:

```
135
```

In [43]:

```
mathacc.average(34,64)
```

Out[43]:

```
49.0
```

In [44]:

```
help(mathacc.division)
```

```
Help on function division in module mathacc:

division(num1, num2)
    Division of two numbers
```

In [45]:

```
from mathacc import addition
```

In [46]:

```
addition(43,34)
```

Out[46]:

```
77
```

In [47]:

```
mathacc.VERSION
```

Out[47]:

```
1.0
```

In [48]:

```
from mathacc import *
```

In [50]:

```
multiplication(5,2)
```

Out[50]:

```
10
```

In [51]:

```
import os
```

In [53]:

```
# dir(os)
```

In [54]:

```
os.getcwd()
```

Out[54]:

```
'/Users/surendra'
```

In [55]:

```
os.system('ls -l')
```

Out[55]:

```
0
```

In [58]:

```
import glob
```

In [61]:

```
glob.glob('m*.py')
```

Out[61]:

```
['mymath.py', 'my_docs.py', 'mathacc.py', 'mydoc.py']
```

In [70]:

```
import pickle

Data = [{'apple': 150, 'banana' : 50, 'grape' : 90 }]

print(type(Data))

pickle_data = pickle.dumps(Data)

print(pickle_data)
print(type(pickle_data))

```

```
<class 'list'>
b'\x80\x03]q\x00}q\x01(X\x05\x00\x00\x00appleq\x02K\x96X\x06\x00\x00
\x00bananaq\x03K2X\x05\x00\x00\x00grapeq\x04KZua.'
<class 'bytes'>
```

In [83]:

```python
import pickle

Data = [{'apple': 150, 'banana' : 50, 'grape' : 90 }]

print(type(Data))

pickle_data = pickle.dumps(Data)

print(pickle_data)
print(type(pickle_data))

unpickle_data = pickle.loads(pickle_data)

print(unpickle_data)
print(type(unpickle_data))

print("Original Data: ", Data)

print("Unpickle Data: ",unpickle_data)

print("Check wheather data is Equal")

print("Is it equal: ", Data == unpickle_data )

print("Original Data id: ",id(Data))

print("Unpickle Data id: ",id(unpickle_data))

print("Is it same: ", Data is unpickle_data )
```

```
<class 'list'>
b'\x80\x03]q\x00}q\x01(X\x05\x00\x00\x00appleq\x02K\x96X\x06\x00\x00
\x00bananaq\x03K2X\x05\x00\x00\x00grapeq\x04KZua.'
<class 'bytes'>
[{'apple': 150, 'banana': 50, 'grape': 90}]
<class 'list'>
Original Data:  [{'apple': 150, 'banana': 50, 'grape': 90}]
Unpickle Data:  [{'apple': 150, 'banana': 50, 'grape': 90}]
Check wheather data is Equal
Is it equal:  True
Original Data id:  4482819464
Unpickle Data id:  4482819272
Is it same:  False
```

In [84]:

```
x = 200
print(id(x))
y = 200
print(id(y))

x is y
```

```
4427095328
4427095328
```

Out[84]:

```
True
```

In [85]:

```
if Data is unpickle_data:
    print("Yes")
else:
    print("No")
```

```
No
```

In [87]:

```
import json

Data = [{'apple': (150, 40), 'banana' : [50, 90],
         'grape' : 90 }]

print(type(Data))

pickle_data = json.dumps(Data)

print(pickle_data)
print(type(pickle_data))

unpickle_data = json.loads(pickle_data)

print(unpickle_data)
print(type(unpickle_data))

print("Original Data: ", Data)

print("Unpickle Data: ",unpickle_data)

print("Check wheather data is Equal")

print("Is it equal: ", Data == unpickle_data )

print("Original Data id: ",id(Data))

print("Unpickle Data id: ",id(unpickle_data))

print("Is it same: ", Data is unpickle_data )
```

```
<class 'list'>
[{"apple": [150, 40], "banana": [50, 90], "grape": 90}]
<class 'str'>
[{'apple': [150, 40], 'banana': [50, 90], 'grape': 90}]
<class 'list'>
Original Data:  [{'apple': (150, 40), 'banana': [50, 90], 'grape': 9
0}]
Unpickle Data:  [{'apple': [150, 40], 'banana': [50, 90], 'grape': 9
0}]
Check wheather data is Equal
Is it equal:  False
Original Data id:  4482819464
Unpickle Data id:  4463545608
Is it same:  False
```

In [106]:

```python
import pickle

Data = [{'apple': 150, 'banana' : 50, 'grape' : 90 }]

with open('pickle_file','wb') as  File_Create:
    pickle_data = pickle.dump(Data, File_Create)


with  open('pickle_file','rb') as File_Load:
    unpickle_data = pickle.load(File_Load)


print("Original Data: ", Data)

print("Unpickle Data: ",unpickle_data)

print("Check wheather data is Equal")

print("Is it equal: ", Data == unpickle_data )

print("Original Data id: ",id(Data))

print("Unpickle Data id: ",id(unpickle_data))

print("Is it same: ", Data is unpickle_data )
```

```
Original Data:  [{'apple': 150, 'banana': 50, 'grape': 90}]
Unpickle Data:  [{'apple': 150, 'banana': 50, 'grape': 90}]
Check wheather data is Equal
Is it equal:  True
Original Data id:  4633118920
Unpickle Data id:  4632957192
Is it same:  False
```

In [99]:

```python
import pickle

# An arbitrary collection of objects supported by pickle.
data = {
    'a': [1, 2.0, 3, 4+6j],
    'b': ("character string", b"byte string"),
    'c': {None, True, False}
}

with open('data.pickle', 'wb') as f:
    # Pickle the 'data' dictionary using the
    # highest protocol available.
    pickle.dump(data, f, pickle.HIGHEST_PROTOCOL)
```

In [101]:

```
import pickle

with open('data.pickle', 'rb') as f:
    # The protocol version used is detected automatically, so we do not
    # have to specify it.
    data = pickle.load(f)
    print(data)
```

```
{'a': [1, 2.0, 3, (4+6j)], 'b': ('character string', b'byte string')
, 'c': {False, None, True}}
```

In [107]:

```
print('''2. Object Oriented Programming
a) What is class and Object ?
b) Create Class with pass
c) Method
d) difference between Method and Function
e) use of self ? What it is ? Why ?
f) Object variable / attribute
g) Class Variable
h) Understand difference between Object and
Class attribute
i) Constructor ( Class Initialisation )
Use of __init__ method ( built in method )
j) Inheritance
   Subclass / Superclass
k) Overriding ( Polymorphism )
l) Difference between Overriding and Overloading
m) user of super keyword
n) Demo Class Application
o) Group Assignment for Developing Three Object
oriented Application
p)Group Discussion and Quiz

''')
```

```
2. Object Oriented Programming
a) What is class and Object ?
b) Create Class with pass
c) Method
d) difference between Method and Function
e) use of self ? What it is ? Why ?
f) Object variable / attribute
g) Class Variable
h) Understand difference between Object and
Class attribute
i) Constructor ( Class Initialisation )
Use of __init__ method ( built in method )
j) Inheritance
   Subclass / Superclass
k) Overriding ( Polymorphism )
l) Difference between Overriding and Overloading
m) user of super keyword
n) Demo Class Application
o) Group Assignment for Developing Three Object
oriented Application
p)Group Discussion and Quiz
```

In [108]:

```
'surendra'.upper()
```

Out[108]:

```
'SURENDRA'
```

In [109]:

```
L = []
L.append('data1')
print(L)
```

```
['data1']
```

In [112]:

```
class MyClass:  # Camel Case for Class Name
    pass        # pass means do nothing
```

In [111]:

```
Obj = MyClass()  # Create Object using MyClass
print(Obj)
```

```
<__main__.MyClass object at 0x1142a1940>
```

In [148]:

```python
class MyClass:  # Camel Case for Class Name
    '''This is a MyClass'''

    def hello(self, data):
        '''Hello Method'''
        self.localdata = data  # Object Data
        data1 = 90
        print("Hello Method of MyClass")
        print("Object data:",self.localdata)
        print("Local Data",data1)

    def good_day(self, wish):
        '''Good Day Method'''
        self.w = wish  # Object Variable
        print("Its %s Day"%self.w)


Obj = MyClass()  # Create Object using MyClass
Obj.hello(344)
Obj.good_day('Excellent')
print(Obj.localdata)
print(Obj.data1)
```

```
Hello Method of MyClass
Object data: 344
Local Data 90
Its Excellent Day
344


---------------------------------------------------------------------
-------
AttributeError                            Traceback (most recent cal
l last)
<ipython-input-148-23171161695c> in <module>()
     20 Obj.good_day('Excellent')
     21 print(Obj.localdata)
---> 22 print(Obj.data1)

AttributeError: 'MyClass' object has no attribute 'data1'
```

In [116]:

```python
def hello():
    '''Hello Method'''
    print("Hello Method of MyClass")
```

In [117]:

```
hello()
```

```
Hello Method of MyClass
```

In [132]:

```
'''Data Center Application for Accenture.

Objective : To know inventory of Data Center.

We will have 3 class for DataCenter Inventory.

1. DataCenter ( Base Class )
    1) Constructor Method
    2) dc_info
        CenterName
        location
        State
        Country


2. ServerClass ( SubClass )
    1) Constructor Method __init__(self)
    2) server_info
        a) server_type
    3) vendor_info

3. StorageClass ( SubClass )

    1) Constructor Method __init__(self)
    2) storage_info
        a) storage_type
    3) vendor_info

'''

class DataCenter:  # Base Class
    '''This is a DataCenter Class '''

    no_of_dc = 0  # class variable

    def __init__(self, center_name, c_location,\
                 c_state, c_country):
        '''Initialise DataCenter Class'''

        DataCenter.no_of_dc += 1

        self.center_name = center_name  # Object Variable
        self.c_location = c_location
        self.c_state = c_state
        self.c_country = c_country
```

```
46
47          print("Number of Data Centers are %d"\
48               %DataCenter.no_of_dc)
49
50      def dc_info(self):
51          '''Data Center Information'''
52
53          print("Data Center Name: %s " %self.center_name)
54          print("Data Center Location: %s " %self.c_location)
55          print("Data Center State: %s " %self.c_state)
56          print("Data Center Country: %s " %self.c_country)
57
58
59  dc1 = DataCenter('PDC', 'Pune',\
60                   'Maharashtra','India')
61
62  dc1.dc_info()
63
64
65  dc2 = DataCenter('BDC', 'Bangalore',\
66                   'Karnataka','India')
67
68  dc2.dc_info()
```

```
Number of Data Centers are 1
Data Center Name: PDC
Data Center Location: Pune
Data Center State: Maharashtra
Data Center Country: India
Number of Data Centers are 2
Data Center Name: BDC
Data Center Location: Bangalore
Data Center State: Karnataka
Data Center Country: India
```

In [139]:

```
1  '''Data Center Application for Accenture.
2
3  Objective : To know inventory of Data Center.
4
5  We will have 3 class for DataCenter Inventory.
6
7  1. DataCenter ( Base Class )
8      1) Constructor Method
9      2) dc_info
10         CenterName
11         location
12         State
13         Country
14
15
16 2. ServerClass ( SubClass )
17     1) Constructor Method __init__(self)
18     2) server_info
```

```
        a) server_type
    3) vendor_info

3. StorageClass ( SubClass )

    1) Constructor Method __init__(self)
    2) storage_info
        a) storage_type
    3) vendor_info

'''

class DataCenter:  # Base Class
    '''This is a DataCenter Class '''

    no_of_dc = 0  # class variable

    def __init__(self, center_name, c_location,\
                 c_state, c_country):
        '''Initialise DataCenter Class'''

        DataCenter.no_of_dc += 1

        self.center_name = center_name  # Object Variable
        self.c_location = c_location
        self.c_state = c_state
        self.c_country = c_country

        print("Number of Data Centers are %d"\
              %DataCenter.no_of_dc)

    def dc_info(self):
        '''Data Center Information'''

        print("Data Center Name: %s " %self.center_name)
        print("Data Center Location: %s " %self.c_location)
        print("Data Center State: %s " %self.c_state)
        print("Data Center Country: %s " %self.c_country)



class DataCenterServer(DataCenter):   # Inheritance of DataCenter class
    '''DataCenter Server Information'''

    def server_info(self):
        '''Data Center Server Information '''
        print("Data Center Server Information")

    def vendor_info(self, vname, vcontact):
        '''Data Center Vendor Information '''

        self.vname = vname
        self.vcontact = vcontact
        print("Data Center Vendor Information")
        print("Vendor Name: %s Vendor Contact: %d "\
              %(self.vname,self.vcontact ))

```

```
dcs1 = DataCenterServer('PDC', 'Pune',\
                'Maharashtra','India')

dcs1.dc_info()
dcs1.server_info()
dcs1.vendor_info('eServe',374837423)

print("\n"*2)

dcs2 = DataCenterServer('BDC', 'Bangalore',\
                'Karnataka','India')

dcs2.dc_info()
dcs2.server_info()
dcs2.vendor_info('Cisco',2323236)


```

```
Number of Data Centers are 1
Data Center Name: PDC
Data Center Location: Pune
Data Center State: Maharashtra
Data Center Country: India
Data Center Server Information
Data Center Vendor Information
Vendor Name: eServe Vendor Contact:374837423
```

```
Number of Data Centers are 2
Data Center Name: BDC
Data Center Location: Bangalore
Data Center State: Karnataka
Data Center Country: India
Data Center Server Information
Data Center Vendor Information
Vendor Name: Cisco Vendor Contact:2323236
```

In [142]:

```
'''Data Center Application for Accenture.

Objective : To know inventory of Data Center.

We will have 3 class for DataCenter Inventory.

1. DataCenter ( Base Class )
    1) Constructor Method
    2) dc_info
        CenterName
        location
        State
        Country

```

```
2. ServerClass ( SubClass )
    1) Constructor Method __init__(self)
    2) server_info
        a) server_type
    3) vendor_info

3. StorageClass ( SubClass )

    1) Constructor Method __init__(self)
    2) storage_info
        a) storage_type
    3) vendor_info

'''

class DataCenter:  # Base Class
    '''This is a DataCenter Class '''

    no_of_dc = 0  # class variable

    def __init__(self, center_name, c_location,\
                 c_state, c_country):
        '''Initialise DataCenter Class'''

        DataCenter.no_of_dc += 1

        self.center_name = center_name  # Object Variable
        self.c_location = c_location
        self.c_state = c_state
        self.c_country = c_country

        print("Number of Data Centers are %d"\
             %DataCenter.no_of_dc)

    def dc_info(self):
        '''Data Center Information'''

        print("Data Center Name: %s " %self.center_name)
        print("Data Center Location: %s " %self.c_location)
        print("Data Center State: %s " %self.c_state)
        print("Data Center Country: %s " %self.c_country)



class DataCenterServer(DataCenter):   # Inheritance of DataCenter class
    '''DataCenter Server Information'''

    def server_info(self):
        '''Data Center Server Information '''
        print("Data Center Server Information")

    def vendor_info(self, vname, vcontact):
        '''Data Center Vendor Information '''

        self.vname = vname
        self.vcontact = vcontact
```

```
        print("Data Center Vendor Information")
        print("Vendor Name: %s Vendor Contact: %d "\
              %(self.vname,self.vcontact ))



class DataCenterStorage(DataCenter):   # Inheritance of DataCenter class
    '''DataCenter Storage Information'''

    def storage_info(self):
        '''Data Center Storage Information '''
        print("Data Center Storage Information")

    def vendor_info(self, vname, vcontact):
        '''Data Center Storage Vendor Information '''

        self.vname = vname
        self.vcontact = vcontact
        print("Data Center Storage Vendor Information")
        print("Vendor Name: %s Vendor Contact: %d "\
              %(self.vname,self.vcontact ))


dcs1 = DataCenterServer('PDC', 'Pune',\
                'Maharashtra','India')

dcs1.dc_info()
dcs1.server_info()
dcs1.vendor_info('eServe',66666666)


print("\n"*2)

dcs2 = DataCenterServer('BDC', 'Bangalore',\
                'Karnataka','India')

dcs2.dc_info()
dcs2.server_info()
dcs2.vendor_info('Cisco',77777777)


print("\n"*2)

dcst1 = DataCenterStorage('CDC', 'Chennai',\
                'TamilNadu','India')

dcst1.dc_info()
dcst1.storage_info()
dcst1.vendor_info('eServe',8888888)

print("\n"*2)

dcst2 = DataCenterStorage('NDC', 'Noida',\
                'Delhi','India')

dcst2.dc_info()
dcst2.storage_info()
dcst2.vendor_info('Cisco',9999999)
```

```
130
```

```
Number of Data Centers are 1
Data Center Name: PDC
Data Center Location: Pune
Data Center State: Maharashtra
Data Center Country: India
Data Center Server Information
Data Center Vendor Information
Vendor Name: eServe Vendor Contact: 66666666



Number of Data Centers are 2
Data Center Name: BDC
Data Center Location: Bangalore
Data Center State: Karnataka
Data Center Country: India
Data Center Server Information
Data Center Vendor Information
Vendor Name: Cisco Vendor Contact: 77777777



Number of Data Centers are 3
Data Center Name: CDC
Data Center Location: Chennai
Data Center State: TamilNadu
Data Center Country: India
Data Center Storage Information
Data Center Storage Vendor Information
Vendor Name: eServe Vendor Contact: 8888888



Number of Data Centers are 4
Data Center Name: NDC
Data Center Location: Noida
Data Center State: Delhi
Data Center Country: India
Data Center Storage Information
Data Center Storage Vendor Information
Vendor Name: Cisco Vendor Contact: 9999999
```

In [146]:

```
'''Data Center Application for Accenture.

Objective : To know inventory of Data Center.

We will have 3 class for DataCenter Inventory.

1. DataCenter ( Base Class )
    1) Constructor Method
    2) dc_info
```

```
        CenterName
        location
        State
        Country


2. ServerClass ( SubClass )
    1) Constructor Method __init__(self)
    2) server_info
        a) server_type
    3) vendor_info

3. StorageClass ( SubClass )

    1) Constructor Method __init__(self)
    2) storage_info
        a) storage_type
    3) vendor_info

'''

class DataCenter:  # Base Class
    '''This is a DataCenter Class '''

    no_of_dc = 0  # class variable

    def __init__(self, center_name, c_location,\
                 c_state, c_country):
        '''Initialise DataCenter Class'''

        DataCenter.no_of_dc += 1

        self.center_name = center_name  # Object Variable
        self.c_location = c_location
        self.c_state = c_state
        self.c_country = c_country

        print("Number of Data Centers are %d"\
             %DataCenter.no_of_dc)

    def dc_info(self):
        '''Data Center Information'''

        print("Data Center Name: %s " %self.center_name)
        print("Data Center Location: %s " %self.c_location)
        print("Data Center State: %s " %self.c_state)
        print("Data Center Country: %s " %self.c_country)



class DataCenterServer(DataCenter):   # Inheritance of DataCenter class
    '''DataCenter Server Information'''

    def server_info(self):
        '''Data Center Server Information '''
        print("Data Center Server Information")

```

```
    def vendor_info(self, vname, vcontact):
        '''Data Center Vendor Information '''

        self.vname = vname
        self.vcontact = vcontact
        print("Data Center Vendor Information")
        print("Vendor Name: %s Vendor Contact: %d "\
              %(self.vname,self.vcontact ))


# Inherit Data Center Server

class DataCenterStorage(DataCenterServer):
    '''DataCenter Storage Information'''

    def storage_info(self):
        '''Data Center Storage Information '''
        print("Data Center Storage Information")

    def vendor_info(self, vname, vcontact, svname, svcontact):
        '''Data Center Storage Vendor Information '''

        DataCenterServer.vendor_info(self, vname, vcontact)

        self.svname = svname
        self.svcontact = svcontact
        print("Data Center Storage Vendor Information")
        print("Storage Vendor Name: %s Vendor Contact: %d "\
              %(self.svname,self.svcontact ))



dcst1 = DataCenterStorage('CDC', 'Chennai',\
                'TamilNadu','India')

dcst1.dc_info()
dcst1.server_info()
dcst1.storage_info()
dcst1.vendor_info('eServe',8888888, 'Cisco', 23928748)


print("\n"*2)

dcst2 = DataCenterStorage('NDC', 'Noida',\
                'Delhi','India')

dcst2.dc_info()
dcst2.server_info()
dcst2.storage_info()
dcst2.vendor_info('IBM',67867, 'Juniper', 7657857)
```

```
Number of Data Centers are 1
Data Center Name: CDC
Data Center Location: Chennai
Data Center State: TamilNadu
Data Center Country: India
Data Center Server Information
```

```
Data Center Storage Information
Data Center Vendor Information
Vendor Name: eServe Vendor Contact: 8888888
Data Center Storage Vendor Information
Storage Vendor Name: Cisco Vendor Contact: 23928748



Number of Data Centers are 2
Data Center Name: NDC
Data Center Location: Noida
Data Center State: Delhi
Data Center Country: India
Data Center Server Information
Data Center Storage Information
Data Center Vendor Information
Vendor Name: IBM Vendor Contact: 67867
Data Center Storage Vendor Information
Storage Vendor Name: Juniper Vendor Contact: 7657857
```

In [149]:

```
'''Data Center Application for Accenture.

Objective : To know inventory of Data Center.

We will have 3 class for DataCenter Inventory.


1. DataCenter ( Base Class )
    1) Constructor Method
    2) dc_info
       CenterName
       location
       State
       Country


2. ServerClass ( SubClass )
    1) Constructor Method __init__(self)
    2) server_info
       a) server_type
    3) vendor_info

3. StorageClass ( SubClass )

    1) Constructor Method __init__(self)
    2) storage_info
       a) storage_type
    3) vendor_info

'''

class DataCenter:  # Base Class
    '''This is a DataCenter Class '''

    no_of_dc = 0  # class variable
```

```
    no_of_dc = 0  # Class Variable

    def __init__(self, center_name, c_location,\
                 c_state, c_country):
        '''Initialise DataCenter Class'''

        DataCenter.no_of_dc += 1

        self.center_name = center_name  # Object Variable
        self.c_location = c_location
        self.c_state = c_state
        self.c_country = c_country

        print("Number of Data Centers are %d"\
              %DataCenter.no_of_dc)

    def dc_info(self):
        '''Data Center Information'''

        print("Data Center Name: %s " %self.center_name)
        print("Data Center Location: %s " %self.c_location)
        print("Data Center State: %s " %self.c_state)
        print("Data Center Country: %s " %self.c_country)



class DataCenterServer(DataCenter):   # Inheritance of DataCenter class
    '''DataCenter Server Information'''

    no_of_server = 0 # Class Variable

    def __init__(self, center_name, c_location,\
                 c_state, c_country, dc_server_type):
        '''Initialise DataCenterServer Class'''
        DataCenter.__init__(self, center_name, c_location,\
                 c_state, c_country)
        self.dc_server_type = dc_server_type # Object Variable

        DataCenterServer.no_of_server += 1

        print("Number of DataCenterServer: %d"%self.no_of_server)

    def server_info(self):
        '''Data Center Server Information '''
        print("Data Center Server Information")
        print("Data Center Server Types: %s " %self.dc_server_type)

    def vendor_info(self, vname, vcontact):
        '''Data Center Vendor Information '''

        self.vname = vname
        self.vcontact = vcontact
        print("Data Center Vendor Information")
        print("Vendor Name: %s Vendor Contact: %d "\
              %(self.vname,self.vcontact ))


# Inherit Data Center Server
```

```
class DataCenterStorage(DataCenterServer):
    '''DataCenter Storage Information'''

    def storage_info(self):
        '''Data Center Storage Information '''
        print("Data Center Storage Information")

    def vendor_info(self, vname, vcontact, svname, svcontact):
        '''Data Center Storage Vendor Information '''

        DataCenterServer.vendor_info(self, vname, vcontact)

        self.svname = svname
        self.svcontact = svcontact
        print("Data Center Storage Vendor Information")
        print("Storage Vendor Name: %s Vendor Contact: %d "\
              %(self.svname,self.svcontact ))



dcst1 = DataCenterStorage('CDC', 'Chennai',\
            'TamilNadu','India','WebServer')

dcst1.dc_info()
dcst1.server_info()
dcst1.storage_info()
dcst1.vendor_info('eServe',8888888, 'Cisco', 23928748)

print('\n'*2)

dcst2 = DataCenterStorage('NDC', 'Noida',\
            'Delhi','India','WAP Server')

dcst2.dc_info()
dcst2.server_info()
dcst2.storage_info()
dcst2.vendor_info('IBM',67867, 'Juniper', 7657857)
```

```
Number of Data Centers are 1
Number of DataCenterServer: 1
Data Center Name: CDC
Data Center Location: Chennai
Data Center State: TamilNadu
Data Center Country: India
Data Center Server Information
Data Center Server Types:WebServer
Data Center Storage Information
Data Center Vendor Information
Vendor Name: eServe Vendor Contact: 8888888
Data Center Storage Vendor Information
Storage Vendor Name: Cisco Vendor Contact: 23928748



Number of Data Centers are 2
```

```
Number of DataCenterServer: 2
Data Center Name: NDC
Data Center Location: Noida
Data Center State: Delhi
Data Center Country: India
Data Center Server Information
Data Center Server Types:WAP Server
Data Center Storage Information
Data Center Vendor Information
Vendor Name: IBM Vendor Contact: 67867
Data Center Storage Vendor Information
Storage Vendor Name: Juniper Vendor Contact: 7657857
```

In [150]:

```
'''Data Center Application for Accenture.

Objective : To know inventory of Data Center.

We will have 3 class for DataCenter Inventory.

1. DataCenter ( Base Class )
    1) Constructor Method
    2) dc_info
        CenterName
        location
        State
        Country


2. ServerClass ( SubClass )
    1) Constructor Method __init__(self)
    2) server_info
        a) server_type
    3) vendor_info

3. StorageClass ( SubClass )

    1) Constructor Method __init__(self)
    2) storage_info
        a) storage_type
    3) vendor_info

'''

class DataCenter:  # Base Class
    '''This is a DataCenter Class '''

    no_of_dc = 0  # class variable

    def __init__(self, center_name, c_location,\
                 c_state, c_country):
        '''Initialise DataCenter Class'''

        DataCenter.no_of_dc += 1
```

```
        self.center_name = center_name  # Object Variable
        self.c_location = c_location
        self.c_state = c_state
        self.c_country = c_country

        print("Number of Data Centers are %d"\
              %DataCenter.no_of_dc)

    def dc_info(self):
        '''Data Center Information'''

        print("Data Center Name: %s " %self.center_name)
        print("Data Center Location: %s " %self.c_location)
        print("Data Center State: %s " %self.c_state)
        print("Data Center Country: %s " %self.c_country)



class DataCenterServer(DataCenter):   # Inheritance of DataCenter class
    '''DataCenter Server Information'''

    no_of_server = 0 # Class Variable

    def __init__(self, center_name, c_location,\
                 c_state, c_country, dc_server_type):
        '''Initialise DataCenterServer Class'''
        DataCenter.__init__(self, center_name, c_location,\
                 c_state, c_country)
        self.dc_server_type = dc_server_type # Object Variable

        DataCenterServer.no_of_server += 1

        print("Number of DataCenterServer: %d"%self.no_of_server)

    def server_info(self):
        '''Data Center Server Information '''
        print("Data Center Server Information")
        print("Data Center Server Types: %s " %self.dc_server_type)

    def vendor_info(self, vname, vcontact):
        '''Data Center Vendor Information '''

        self.vname = vname
        self.vcontact = vcontact
        print("Data Center Vendor Information")
        print("Vendor Name: %s Vendor Contact: %d "\
              %(self.vname,self.vcontact ))


# Inherit Data Center Server

class DataCenterStorage(DataCenterServer):
    '''DataCenter Storage Information'''

    def storage_info(self):
        '''Data Center Storage Information '''
        print("Data Center Storage Information")
```

```
    def vendor_info(self, vname, vcontact, svname, svcontact):
        '''Data Center Storage Vendor Information '''

        #DataCenterServer.vendor_info(self, vname, vcontact)

        super(DataCenterStorage,self).vendor_info(vname, vcontact)

        self.svname = svname
        self.svcontact = svcontact
        print("Data Center Storage Vendor Information")
        print("Storage Vendor Name: %s Vendor Contact: %d "\
              %(self.svname,self.svcontact ))



dcst1 = DataCenterStorage('CDC', 'Chennai',\
                'TamilNadu','India','WebServer')

dcst1.dc_info()
dcst1.server_info()
dcst1.storage_info()
dcst1.vendor_info('eServe',8888888, 'Cisco', 23928748)


print("\n"*2)

dcst2 = DataCenterStorage('NDC', 'Noida',\
                'Delhi','India','WAP Server')

dcst2.dc_info()
dcst2.server_info()
dcst2.storage_info()
dcst2.vendor_info('IBM',67867, 'Juniper', 7657857)
```

```
Number of Data Centers are 1
Number of DataCenterServer: 1
Data Center Name: CDC
Data Center Location: Chennai
Data Center State: TamilNadu
Data Center Country: India
Data Center Server Information
Data Center Server Types: WebServer
Data Center Storage Information
Data Center Vendor Information
Vendor Name: eServe Vendor Contact: 8888888
Data Center Storage Vendor Information
Storage Vendor Name: Cisco Vendor Contact: 23928748



Number of Data Centers are 2
Number of DataCenterServer: 2
Data Center Name: NDC
Data Center Location: Noida
Data Center State: Delhi
Data Center Country: India
```

Data Center Server Information
Data Center Server Types: WAP Server
Data Center Storage Information
Data Center Vendor Information
Vendor Name: IBM Vendor Contact: 67867
Data Center Storage Vendor Information
Storage Vendor Name: Juniper Vendor Contact: 7657857

In [151]:

```
1 dir(dcst1)
```

Out[151]:

```
['__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 'c_country',
 'c_location',
 'c_state',
 'center_name',
 'dc_info',
 'dc_server_type',
 'no_of_dc',
 'no_of_server',
 'server_info',
 'storage_info',
 'svcontact',
 'svname',
 'vcontact',
 'vendor_info',
 'vname']
```

In [152]:

```
1 help(dcst1)
```

```
Help on DataCenterStorage in module __main__ object:
```

```
class DataCenterStorage(DataCenterServer)
 |  DataCenterStorage(center_name, c_location, c_state, c_country, d
c_server_type)
 |
 |  DataCenter Storage Information
 |
 |  Method resolution order:
 |      DataCenterStorage
 |      DataCenterServer
 |      DataCenter
 |      builtins.object
 |
 |  Methods defined here:
 |
 |  storage_info(self)
 |      Data Center Storage Information
 |
 |  vendor_info(self, vname, vcontact, svname, svcontact)
 |      Data Center Storage Vendor Information
 |
 |  ----------------------------------------------------------------
------
 |  Methods inherited from DataCenterServer:
 |
 |  __init__(self, center_name, c_location, c_state, c_country, dc_s
erver_type)
 |      Initialise DataCenterServer Class
 |
 |  server_info(self)
 |      Data Center Server Information
 |
 |  ----------------------------------------------------------------
------
 |  Data and other attributes inherited from DataCenterServer:
 |
 |  no_of_server = 2
 |
 |  ----------------------------------------------------------------
------
 |  Methods inherited from DataCenter:
 |
 |  dc_info(self)
 |      Data Center Information
 |
 |  ----------------------------------------------------------------
------
 |  Data descriptors inherited from DataCenter:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------
------
```

```
|  Data and other attributes inherited from DataCenter:
|
|  no_of_dc = 2
```

In [ ]:

```
1
```