

In [1]:

```
1 Agenda = '''
2 1. Module
3 a) How to import ?
4 b) Built in math
5 c) Create User Define Module
6 d) os
7 e) sys
8 f) pickle / unpickle ( Object Serialisation )
9 g) glob
10 h) json
11
12 2. Object Oriented Programming
13 a) What is class and Object ?
14 b) Create Class with pass
15 c) Method
16 d) difference between Method and Function
17 e) use of self ? What it is ? Why ?
18 f) Object variable / attribute
19 g) Class Variable
20 h) Understand difference between Object and
21 Class attribute
22 i) Constructor ( Class Initialisation )
23 Use of __init__ method ( built in method )
24 j) Inheritance
25     Subclass / Superclass
26 k) Overriding ( Polymorphism )
27 l) Difference between Overriding and Overloading
28 m) user of super keyword
29 n) Demo Class Application
30 o) Group Assignment for Developing Three Object
31 oriented Application
32 p)Group Discussion and Quiz
33
34 '''
35
```

In [2]:

```
1 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]:

```
1 import numpy
```

In [4]:

```
1 import socket
```

In [6]:

```
1 #dir(socket)
```

In [8]:

```
1 import threading
2 #dir(threading)
```

In [9]:

```
1 import math
```

In [10]:

```
1 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]:

```
1 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]:

```
1 import math as M
```

In [30]:

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

Out[30]:

```
0.8939966636005579
```

In [31]:

```
1 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]:

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

Out[32]:

```
14.1000000000000001
```

In [34]:

```
1 del math
```

In [36]:

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

In [37]:

```
1 SUM1
```

Out[37]:

```
14.1000000000000001
```


In [38]:

```
1 %%writefile mathacc.py
2
3 '''mathacc module
4 This is a math accenture module for basic mathematical
5 function and data operation.
6 '''
7 VERSION = 1.0
8 module_name = 'mathacc module'
9
10
11 def addition(num1, num2):
12     '''Addition of two numbers'''
13     return num1 + num2
14
15
16 def subtraction(num1, num2):
17     '''Substraction of two numbers'''
18     return num1 - num2
19
20
21 def multiplication(num1, num2):
22     '''Multiplication of two numbers'''
23     return num1 * num2
24
25
26 def division(num1, num2):
27     '''Division of two numbers'''
28     return num1 / num2
29
30
31 def modulus(num1, num2):
32     '''modulus of two numbers'''
33     return num1 % num2
34
35
36 def average(num1, num2):
37     '''Average of two numbers'''
38     return ( num1 + num2 ) / 2
39
40
```

Writing mathacc.py

In [39]:

```
1 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',  
 'subtraction']
```

In [41]:

```
1 help(mathacc)
```

Help on module mathacc:

```
NAME
  mathacc

DESCRIPTION
  mathacc module
  This is a math accenture module for basic mathematical
  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

  subtraction(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]:

In [43]:

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

Out[43]:

49.0

In [44]:

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

Help on function division in module mathacc:

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

In [45]:

```
1 from mathacc import addition
```

In [46]:

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

Out[46]:

77

In [47]:

```
1 mathacc.VERSION
```

Out[47]:

1.0

In [48]:

```
1 from mathacc import *
```

In [50]:

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

Out[50]:

10

In [51]:

```
1 import os
```

In [53]:

```
1 # dir(os)
```

In [54]:

```
1 os.getcwd()
```

Out[54]:

'/Users/surendra'

In [55]:

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

Out[55]:

0

In [58]:

```
1 import glob
```

In [61]:

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

Out[61]:

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

In [70]:

```
1 import pickle
2
3 Data = [{'apple': 150, 'banana' : 50, 'grape' : 90 }]
4
5 print(type(Data))
6
7 pickle_data = pickle.dumps(Data)
8
9 print(pickle_data)
10 print(type(pickle_data))
11
```

```
<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]:

```
1 import pickle
2
3 Data = [{'apple': 150, 'banana' : 50, 'grape' : 90 }]
4
5 print(type(Data))
6
7 pickle_data = pickle.dumps(Data)
8
9 print(pickle_data)
10 print(type(pickle_data))
11
12 unpickle_data = pickle.loads(pickle_data)
13
14 print(unpickle_data)
15 print(type(unpickle_data))
16
17 print("Original Data: ", Data)
18
19 print("Unpickle Data: ",unpickle_data)
20
21 print("Check wheather data is Equal")
22
23 print("Is it equal: ", Data == unpickle_data )
24
25 print("Original Data id: ",id(Data))
26
27 print("Unpickle Data id: ",id(unpickle_data))
28
29 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]:

```
1 x = 200
2 print(id(x))
3 y = 200
4 print(id(y))
5
6 x is y
```

4427095328

4427095328

Out[84]:

True

In [85]:

```
1 if Data is unpickle_data:
2     print("Yes")
3 else:
4     print("No")
```

No

In [87]:

```
1 import json
2
3 Data = [{'apple': (150, 40), 'banana' : [50, 90],
4         'grape' : 90 }]
5
6 print(type(Data))
7
8 pickle_data = json.dumps(Data)
9
10 print(pickle_data)
11 print(type(pickle_data))
12
13 unpickle_data = json.loads(pickle_data)
14
15 print(unpickle_data)
16 print(type(unpickle_data))
17
18 print("Original Data: ", Data)
19
20 print("Unpickle Data: ",unpickle_data)
21
22 print("Check wheather data is Equal")
23
24 print("Is it equal: ", Data == unpickle_data )
25
26 print("Original Data id: ",id(Data))
27
28 print("Unpickle Data id: ",id(unpickle_data))
29
30 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': 90}]
Unpickle Data:  [{'apple': [150, 40], 'banana': [50, 90], 'grape': 90}]
Check wheather data is Equal
Is it equal:  False
Original Data id:  4482819464
Unpickle Data id:  4463545608
Is it same:  False
```

In [106]:

```
1 import pickle
2
3 Data = [{'apple': 150, 'banana' : 50, 'grape' : 90 }]
4
5 with open('pickle_file','wb') as File_Create:
6     pickle_data = pickle.dump(Data, File_Create)
7
8
9 with open('pickle_file','rb') as File_Load:
10    unpickle_data = pickle.load(File_Load)
11
12
13 print("Original Data: ", Data)
14
15 print("Unpickle Data: ",unpickle_data)
16
17 print("Check wheather data is Equal")
18
19 print("Is it equal: ", Data == unpickle_data )
20
21 print("Original Data id: ",id(Data))
22
23 print("Unpickle Data id: ",id(unpickle_data))
24
25 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]:

```
1 import pickle
2
3 # An arbitrary collection of objects supported by pickle.
4 data = {
5     'a': [1, 2.0, 3, 4+6j],
6     'b': ("character string", b"byte string"),
7     'c': {None, True, False}
8 }
9
10 with open('data.pickle', 'wb') as f:
11     # Pickle the 'data' dictionary using the
12     # highest protocol available.
13     pickle.dump(data, f, pickle.HIGHEST_PROTOCOL)
```

In [101]:

```
1 import pickle
2
3 with open('data.pickle', 'rb') as f:
4     # The protocol version used is detected automatically, so we do not
5     # have to specify it.
6     data = pickle.load(f)
7     print(data)
```

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

In [107]:

```
1 print('''2. Object Oriented Programming
2 a) What is class and Object ?
3 b) Create Class with pass
4 c) Method
5 d) difference between Method and Function
6 e) use of self ? What it is ? Why ?
7 f) Object variable / attribute
8 g) Class Variable
9 h) Understand difference between Object and
10 Class attribute
11 i) Constructor ( Class Initialisation )
12 Use of __init__ method ( built in method )
13 j) Inheritance
14     Subclass / Superclass
15 k) Overriding ( Polymorphism )
16 l) Difference between Overriding and Overloading
17 m) user of super keyword
18 n) Demo Class Application
19 o) Group Assignment for Developing Three Object
20 oriented Application
21 p)Group Discussion and Quiz
22
23 ''')
```

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]:

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

Out[108]:

```
'SURENDRA'
```

In [109]:

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

```
['data1']
```

In [112]:

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

In [111]:

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

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

In [148]:

```
1 class MyClass: # Camel Case for Class Name
2     '''This is a MyClass'''
3
4     def hello(self, data):
5         '''Hello Method'''
6         self.localdata = data # Object Data
7         data1 = 90
8         print("Hello Method of MyClass")
9         print("Object data:",self.localdata)
10        print("Local Data",data1)
11
12       def good_day(self, wish):
13           '''Good Day Method'''
14           self.w = wish # Object Variable
15           print("Its %s Day"%self.w)
16
17
18 Obj = MyClass() # Create Object using MyClass
19 Obj.hello(344)
20 Obj.good_day('Excellent')
21 print(Obj.localdata)
22 print(Obj.data1)
```

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

```
-----
-----
AttributeError                                Traceback (most recent call
1 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]:

```
1 def hello():
2     '''Hello Method'''
3     print("Hello Method of MyClass")
```

In [117]:

```
1 hello()
```

Hello Method of MyClass

In [132]:

```
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
19        a) server_type
20     3) vendor_info
21
22 3. StorageClass ( SubClass )
23
24     1) Constructor Method __init__(self)
25     2) storage_info
26        a) storage_type
27     3) vendor_info
28
29 '''
30
31 class DataCenter: # Base Class
32     '''This is a DataCenter Class '''
33
34     no_of_dc = 0 # class variable
35
36     def __init__(self, center_name, c_location,\
37                 c_state, c_country):
38         '''Initialise DataCenter Class'''
39
40         DataCenter.no_of_dc += 1
41
42         self.center_name = center_name # Object Variable
43         self.c_location = c_location
44         self.c_state = c_state
45         self.c_country = c_country
```

```

46         print("Number of Data Centers are %d"\
47               %DataCenter.no_of_dc)
48
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

```



```

19     a) server_type
20     3) vendor_info
21
22 3. StorageClass ( SubClass )
23
24     1) Constructor Method __init__(self)
25     2) storage_info
26         a) storage_type
27     3) vendor_info
28
29 '''
30
31 class DataCenter: # Base Class
32     '''This is a DataCenter Class '''
33
34     no_of_dc = 0 # class variable
35
36     def __init__(self, center_name, c_location,\
37                 c_state, c_country):
38         '''Initialise DataCenter Class'''
39
40         DataCenter.no_of_dc += 1
41
42         self.center_name = center_name # Object Variable
43         self.c_location = c_location
44         self.c_state = c_state
45         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
60 class DataCenterServer(DataCenter): # Inheritance of DataCenter class
61     '''DataCenter Server Information'''
62
63     def server_info(self):
64         '''Data Center Server Information '''
65         print("Data Center Server Information")
66
67     def vendor_info(self, vname, vcontact):
68         '''Data Center Vendor Information '''
69
70         self.vname = vname
71         self.vcontact = vcontact
72         print("Data Center Vendor Information")
73         print("Vendor Name: %s Vendor Contact: %d "\
74               %(self.vname,self.vcontact ))
75
76

```

```

76
77
78 dcs1 = DataCenterServer('PDC', 'Pune',\
79                          'Maharashtra', 'India')
80
81 dcs1.dc_info()
82 dcs1.server_info()
83 dcs1.vendor_info('eServe', 374837423)
84
85 print("\n"*2)
86
87 dcs2 = DataCenterServer('BDC', 'Bangalore',\
88                          'Karnataka', 'India')
89
90 dcs2.dc_info()
91 dcs2.server_info()
92 dcs2.vendor_info('Cisco', 2323236)
93
94

```

```

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]:

```

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
19         a) server_type
20     3) vendor_info
21
22 3. StorageClass ( SubClass )
23
24     1) Constructor Method __init__(self)
25     2) storage_info
26         a) storage_type
27     3) vendor_info
28
29 '''
30
31 class DataCenter: # Base Class
32     '''This is a DataCenter Class '''
33
34     no_of_dc = 0 # class variable
35
36     def __init__(self, center_name, c_location,\
37                 c_state, c_country):
38         '''Initialise DataCenter Class'''
39
40         DataCenter.no_of_dc += 1
41
42         self.center_name = center_name # Object Variable
43         self.c_location = c_location
44         self.c_state = c_state
45         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
60 class DataCenterServer(DataCenter): # Inheritance of DataCenter class
61     '''DataCenter Server Information'''
62
63     def server_info(self):
64         '''Data Center Server Information '''
65         print("Data Center Server Information")
66
67     def vendor_info(self, vname, vcontact):
68         '''Data Center Vendor Information '''
69
70         self.vname = vname
71         self.vcontact = vcontact

```

```

72     print("Data Center Vendor Information")
73     print("Vendor Name: %s Vendor Contact: %d "\
74           %(self.vname,self.vcontact ))
75
76
77
78 class DataCenterStorage(DataCenter):    # Inheritance of DataCenter class
79     '''DataCenter Storage Information'''
80
81     def storage_info(self):
82         '''Data Center Storage Information '''
83         print("Data Center Storage Information")
84
85     def vendor_info(self, vname, vcontact):
86         '''Data Center Storage Vendor Information '''
87
88         self.vname = vname
89         self.vcontact = vcontact
90         print("Data Center Storage Vendor Information")
91         print("Vendor Name: %s Vendor Contact: %d "\
92               %(self.vname,self.vcontact ))
93
94
95 dcs1 = DataCenterServer('PDC', 'Pune',\
96                        'Maharashtra','India')
97
98 dcs1.dc_info()
99 dcs1.server_info()
100 dcs1.vendor_info('eServe',66666666)
101
102 print("\n"*2)
103
104 dcs2 = DataCenterServer('BDC', 'Bangalore',\
105                        'Karnataka','India')
106
107 dcs2.dc_info()
108 dcs2.server_info()
109 dcs2.vendor_info('Cisco',77777777)
110
111
112 print("\n"*2)
113
114 dcst1 = DataCenterStorage('CDC', 'Chennai',\
115                          'TamilNadu','India')
116
117 dcst1.dc_info()
118 dcst1.storage_info()
119 dcst1.vendor_info('eServe',88888888)
120
121 print("\n"*2)
122
123 dcst2 = DataCenterStorage('NDC', 'Noida',\
124                          'Delhi','India')
125
126 dcst2.dc_info()
127 dcst2.storage_info()
128 dcst2.vendor_info('Cisco',99999999)
129

```

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: 88888888

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: 99999999

In [146]:

```
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
19         a) server_type
20     3) vendor_info
21
22 3. StorageClass ( SubClass )
23
24     1) Constructor Method __init__(self)
25     2) storage_info
26         a) storage_type
27     3) vendor_info
28
29 '''
30
31 class DataCenter: # Base Class
32     '''This is a DataCenter Class '''
33
34     no_of_dc = 0 # class variable
35
36     def __init__(self, center_name, c_location,\
37                 c_state, c_country):
38         '''Initialise DataCenter Class'''
39
40         DataCenter.no_of_dc += 1
41
42         self.center_name = center_name # Object Variable
43         self.c_location = c_location
44         self.c_state = c_state
45         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
60 class DataCenterServer(DataCenter): # Inheritance of DataCenter class
61     '''DataCenter Server Information'''
62
63     def server_info(self):
64         '''Data Center Server Information '''
65         print("Data Center Server Information")
66

```

```

67     def vendor_info(self, vname, vcontact):
68         '''Data Center Vendor Information '''
69
70         self.vname = vname
71         self.vcontact = vcontact
72         print("Data Center Vendor Information")
73         print("Vendor Name: %s Vendor Contact: %d "\
74               %(self.vname,self.vcontact ))
75
76
77 # Inherit Data Center Server
78
79 class DataCenterStorage(DataCenterServer):
80     '''DataCenter Storage Information'''
81
82     def storage_info(self):
83         '''Data Center Storage Information '''
84         print("Data Center Storage Information")
85
86     def vendor_info(self, vname, vcontact, svname, svcontact):
87         '''Data Center Storage Vendor Information '''
88
89         DataCenterServer.vendor_info(self, vname, vcontact)
90
91         self.svname = svname
92         self.svcontact = svcontact
93         print("Data Center Storage Vendor Information")
94         print("Storage Vendor Name: %s Vendor Contact: %d "\
95               %(self.svname,self.svcontact ))
96
97
98
99 dcst1 = DataCenterStorage('CDC', 'Chennai',\
100                          'TamilNadu', 'India')
101
102 dcst1.dc_info()
103 dcst1.server_info()
104 dcst1.storage_info()
105 dcst1.vendor_info('eServe',8888888, 'Cisco', 23928748)
106
107
108 print("\n"*2)
109
110 dcst2 = DataCenterStorage('NDC', 'Noida',\
111                          'Delhi', 'India')
112
113 dcst2.dc_info()
114 dcst2.server_info()
115 dcst2.storage_info()
116 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.
2
Objective : To know inventory of Data Center.
3
4
We will have 3 class for DataCenter Inventory.
5
6
1. DataCenter ( Base Class )
7
8 1) Constructor Method
9 2) dc_info
10 CenterName
11 location
12 State
13 Country
14
15
2. ServerClass ( SubClass )
16
17 1) Constructor Method __init__(self)
18 2) server_info
19 a) server_type
20 3) vendor_info
21
22
3. StorageClass ( SubClass )
23
24 1) Constructor Method __init__(self)
25 2) storage_info
26 a) storage_type
27 3) vendor_info
28
'''
29
30
class DataCenter: # Base Class
31
32 '''This is a DataCenter Class '''
33
34 no_of_dc = 0 # class variable
```



```

no_of_dc = 0 # Class Variable
35
def __init__(self, center_name, c_location,\
36         c_state, c_country):
37     '''Initialise DataCenter Class'''
38
39
40 DataCenter.no_of_dc += 1
41
42 self.center_name = center_name # Object Variable
43 self.c_location = c_location
44 self.c_state = c_state
45 self.c_country = c_country
46
47 print("Number of Data Centers are %d"\
48       %DataCenter.no_of_dc)
49
def dc_info(self):
50
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
class DataCenterServer(DataCenter): # Inheritance of DataCenter class
60
61 '''DataCenter Server Information'''
62
no_of_server = 0 # Class Variable
63
64
def __init__(self, center_name, c_location,\
65         c_state, c_country, dc_server_type):
66     '''Initialise DataCenterServer Class'''
67
68 DataCenter.__init__(self, center_name, c_location,\
69                    c_state, c_country)
70 self.dc_server_type = dc_server_type # Object Variable
71
72 DataCenterServer.no_of_server += 1
73
74 print("Number of DataCenterServer: %d"%self.no_of_server)
75
def server_info(self):
76
77 '''Data Center Server Information '''
78 print("Data Center Server Information")
79 print("Data Center Server Types: %s " %self.dc_server_type)
80
def vendor_info(self, vname, vcontact):
81
82 '''Data Center Vendor Information '''
83
84 self.vname = vname
85 self.vcontact = vcontact
86 print("Data Center Vendor Information")
87 print("Vendor Name: %s Vendor Contact: %d "\
88       %(self.vname,self.vcontact ))
89
90
# Inherit Data Center Server

```

```

92
93 class DataCenterStorage(DataCenterServer):
94     '''DataCenter Storage Information'''
95
96     def storage_info(self):
97         '''Data Center Storage Information '''
98         print("Data Center Storage Information")
99
100     def vendor_info(self, vname, vcontact, svname, svcontact):
101         '''Data Center Storage Vendor Information '''
102
103         DataCenterServer.vendor_info(self, vname, vcontact)
104
105         self.svname = svname
106         self.svcontact = svcontact
107         print("Data Center Storage Vendor Information")
108         print("Storage Vendor Name: %s Vendor Contact: %d "\
109             %(self.svname,self.svcontact ))
110
111
112
113 dcst1 = DataCenterStorage('CDC', 'Chennai',\
114                          'TamilNadu', 'India', 'WebServer')
115
116 dcst1.dc_info()
117 dcst1.server_info()
118 dcst1.storage_info()
119 dcst1.vendor_info('eServe',8888888, 'Cisco', 23928748)
120
121
122 print("\n"*2)
123
124 dcst2 = DataCenterStorage('NDC', 'Noida',\
125                          'Delhi', 'India', 'WAP Server')
126
127 dcst2.dc_info()
128 dcst2.server_info()
129 dcst2.storage_info()
130 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]:

```
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
19         a) server_type
20      3) vendor_info
21
22  3. StorageClass ( SubClass )
23
24      1) Constructor Method __init__(self)
25      2) storage_info
26         a) storage_type
27      3) vendor_info
28
29  '''
30
31  class DataCenter: # Base Class
32      '''This is a DataCenter Class '''
33
34      no_of_dc = 0 # class variable
35
36      def __init__(self, center_name, c_location,\
37                  c_state, c_country):
38          '''Initialise DataCenter Class'''
39
40          DataCenter.no_of_dc += 1
41
```

```

42     self.center_name = center_name # Object Variable
43     self.c_location = c_location
44     self.c_state = c_state
45     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
60 class DataCenterServer(DataCenter): # Inheritance of DataCenter class
61     '''DataCenter Server Information'''
62
63     no_of_server = 0 # Class Variable
64
65     def __init__(self, center_name, c_location,\
66                 c_state, c_country, dc_server_type):
67         '''Initialise DataCenterServer Class'''
68         DataCenter.__init__(self, center_name, c_location,\
69                             c_state, c_country)
70         self.dc_server_type = dc_server_type # Object Variable
71
72         DataCenterServer.no_of_server += 1
73
74         print("Number of DataCenterServer: %d"%self.no_of_server)
75
76     def server_info(self):
77         '''Data Center Server Information '''
78         print("Data Center Server Information")
79         print("Data Center Server Types: %s " %self.dc_server_type)
80
81     def vendor_info(self, vname, vcontact):
82         '''Data Center Vendor Information '''
83
84         self.vname = vname
85         self.vcontact = vcontact
86         print("Data Center Vendor Information")
87         print("Vendor Name: %s Vendor Contact: %d "\
88               %(self.vname,self.vcontact ))
89
90
91 # Inherit Data Center Server
92
93 class DataCenterStorage(DataCenterServer):
94     '''DataCenter Storage Information'''
95
96     def storage_info(self):
97         '''Data Center Storage Information '''
98         print("Data Center Storage Information")

```

```

99
100     def vendor_info(self, vname, vcontact, svname, svcontact):
101         '''Data Center Storage Vendor Information '''
102
103         #DataCenterServer.vendor_info(self, vname, vcontact)
104
105         super(DataCenterStorage,self).vendor_info(vname, vcontact)
106
107         self.svname = svname
108         self.svcontact = svcontact
109         print("Data Center Storage Vendor Information")
110         print("Storage Vendor Name: %s Vendor Contact: %d "\
111               %(self.svname,self.svcontact ))
112
113
114
115     dcst1 = DataCenterStorage('CDC', 'Chennai',\
116                               'TamilNadu', 'India', 'WebServer')
117
118     dcst1.dc_info()
119     dcst1.server_info()
120     dcst1.storage_info()
121     dcst1.vendor_info('eServe',8888888, 'Cisco', 23928748)
122
123
124     print("\n"*2)
125
126     dcst2 = DataCenterStorage('NDC', 'Noida',\
127                               'Delhi', 'India', 'WAP Server')
128
129     dcst2.dc_info()
130     dcst2.server_info()
131     dcst2.storage_info()
132     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__',  
'__getattr__',  
'__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	
---	--