CHAPTER - 2
OBJECT ORIENTED PROGRAMMING LANGUAGE

Question 1:
How is a static method different from an instance method?

Answer:
Static method has to be defined outside a class. It can be called without an object. Instance method is defined within a class and has to be invoked on an object.

Question 2:
Explain Data Hiding with respect to OOP.

Answer:
Data hiding can be defined as the mechanism of hiding the data of a class from the outside world or to be precise, from other classes. Data hiding is achieved by making the members of the class private. Access to private members is restricted and is only available to the member functions of the same class. However the public part of the object is accessible outside the class.

Question 3:
Fill in the blanks:

1. Act of representing essential features without background detail is called _____ .
2. Wrapping up of data and associated functions into a single unit is called _____ .
3. _____ is called the instance of a class.

Answer:

1. Data Abstraction
2. Encapsulation
3. Object

Question 4:
What is Object Oriented Programming? List some of its advantages.

Answer:
OOP allows decomposition of a problem into a number of entities called objects and then builds data and functions around these objects. Advantages:

- Simplicity
- Modifiability

- Extensibility and Maintainability
- Reusability
- Security

Question 5:
Differentiate between an object and a class.

Answer:
A class is a collection of objects of similar type.
For example, mango, apple and orange are members of the class fruit. Classes are user-defined data types and behave like the built-in types of a programming language. The syntax used to create an object is not different than the syntax used to create an integer object in C. If fruit has been defined as a class, then the statement fruit mango; will create an object mango belonging to the class fruit.

Question 6:
Explain polymorphism with an example.

Answer:
Polymorphism is the ability for a message or data to be processed in more than one form. An operation may exhibit different behaviors in different instances. For example, consider the operation of addition of two numbers, the operation will generate a sum. If the operands are strings, then the operation would produce a third string by concatenation.

Question 7:
List three features that make an important charac-teristic of OOP.

Answer:

- Capability to express closeness with the real- world models.
- Reusability-allows addition of new features to an existing one.
- Transitivity-changes in one class get automatically reflected across.

Question 8:
How do we implement abstract method in python? Give an example for the same.

Answer:
Abstract method : An unimplement method is called an abstract method. When an abstract medhod is declared in a base class the derived class has to define the method or raise “Notimplemented Error”

Or

Abstract Method can be used to enable parent class method execution.
Class Shape (object):
def findArea (self): pass
Class Square (Shape):
def init (self, side):
self, side = side def findArea self.side*self.side

Question 9:
List few disadvantages of OOP.

Answer:

- Classes tend to be overly generalized.
- Relationship among classes might become artificial.
- Program design is tricky and complicated.
- More skills and thinking in terms of objects is required.

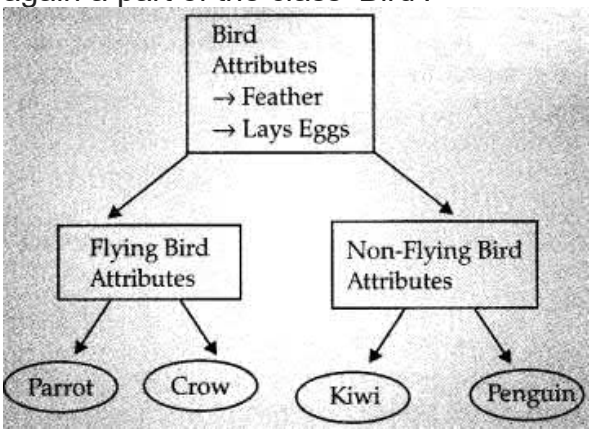
Question 10:
Explain Function overloading with an example.

Answer:
When several function declarations are specified for a single function name in the same scope, the function is said to be overloaded. In other languages, the same function name can be used to define multiple functions with different number and type of arguments, def test(): #function 1 print “hello”
def test(a, b): #function 2
return a+b
def test(a, b, c): #function 3
return a+b+c 2

Question 11:
What is inheritance? Explain with an example.

Answer:
Inheritance is the capability of one class to acquire the properties or capabilities of another class. For example, the bird ‘Parrot’ is a part of the class ‘Flying Bird’ which

again a part of the class 'Bird'.



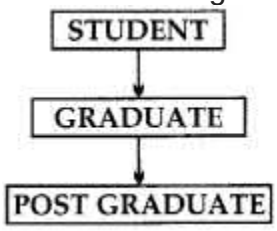
Question 12:
How data encapsulation and data abstraction are implemented in Python, explain with an example.

Or

How do abstraction and encapsulation complement each other?

Answer:
Abstraction and Encapsulation are complementary concepts. Through encapsulation only we are able to enclose the components of the object into a single unit and separate the private and public members. It is through abstraction that only the essential behaviors of the objects are made visible to the outside world.
So, we can say that encapsulation is the way to implement data abstraction. For example in class Student, only the essential information like roll no, name, date_of_birth, course, etc. of the student will be visible. The secret information like calculation of grades, allotment of examiners etc. will be hidden.

Question 13:
Consider the figure given below and answer the questions that follows:



1. Name the base class and the derived class.
2. Which concept of OOP is implemented in the figure given above?

Answer:

1. Base class – STUDENT
   Derived classes – GRADUATE & POST GRADUATE
2. Inheritance

Question 14:
What is abstract method? Give a suitable example to illustrate the same.

Answer:
Abstract Method: An abstract method is a method defined in the base class, but does not require code for the implementaion. e.g.,
Class teacher: def entry (self):
teach#=int(input("Enter No"))
def display ( ):
print (self, teach #)

Question 15:
What is the concept of overriding method? Give an example for the same.

Answer:
Overriding Methods : It is a method to access, the paramterized constructer with same name but having different parameters. For example,
Class emp:
def_init_(self, n):
self, a = n def_init_(self):
self, a = 100

Question 16:
Is function overloading supported by Python? Give reasons.

Answer:
A given name can only be associated with one function at a time, so cannot overload a function with multiple definitions. If you define two or more functions with the same name, the last one defined is used.
However, it is possible to overload a function, or otherwise genericized it. You simply need to create a dispatcher function that then dispatches to your set of corresponding functions. Another way to genericized a function is to make use of the simple generic module which lets you define simple single-dispatch generic functions.
def test(): #function 1
print "hello"
def test(a, b): #function 2
return a+b
def test(a, b, c): #function 3
return a+b+c

If you run the code of three test functions, the second test() definition will overwrite the first one. Subsequently the third test() definition will overwrite the second one. That means if you give the function call test (20,20), it will flash an error stating, “Type Error: add() takes exactly 3 arguments (2 given)”. This is because, Python understands the latest definition of the function test() which takes three arguments.

Question 17:
Predict the output of the following program. Also state which concept of OOP is being implemented?

```
def sum(x,y,z):
print “sum= ”, x+y+z
def sum(a,b):
print “sum= ”, a+b
sum(10,20)
sum(10,20,30)
```

Answer:
Type Error: sum() takes exactly 3 arguments (2 given)
Concept: Polymorphism [Function Overloading]

Question 18:
Define binding. Differentiate between static and dynamic binding.

Answer:
Binding is the process of linking the function call to the function definition. The body of the function is executed when the function call is made. Binding can be of two types:
Static Binding: In this type of binding, the linking of function call to the function definition is done during compilation of the program.
Dynamic Binding: In this type of binding, linking of a function call to the function definition is done at run time. That means the code of the function that is to be linked with function call is unknown until it is executed.

Long Answer Type Questions (4 marks)

Question 1:
Write a program that uses an area() function for the calculation of area of a triangle or a rectangle or a square. Number of sides (3, 2 or 1) suggest the shape for which the area is to be calculated.

Answer:

```
from functools import wraps
import math
def overloaded(func):
@wraps(func)
def overloaded_func(*args, **kwargs):
```

```
for f in overloaded_func.overloads:
try:
return f(*args, **kwargs)
except TÿpeError:
pass
else:
# it will be nice if the error message prints a list of
# possible signatures here
raise TÿpeError(“No compatible signatures”)
def overload_with(func):
overloaded_func.overloads.append(func)
return overloaded_func
overloadedjunc.overloads = [func]
overloaded_func.overload_with = overload_with
return overloaded_func


##############
©overloaded
def area():
print ‘Area’
pass
@area.overload_with
def _(a):
# note that, like property(), the function’s name in
# the “def _(n):” line can be arbitrary, the important
# name is in the “@overloads(a)” line
print ‘Area of square=’,a*a
pass


@area.overload_with
def _(a,b):
# note that, like property(), the function’s name in
# the “def _(nl,n2):” line can be arbitrary, the important t
# name is in the “@overloads(a)” line
print Area of rectangle=’,a*b
pass
@area.overload_with
def _(a.b,c):
s= (a+b+c)/2
print s
print Area of triangle=’, math.sqrt(s*(s-a)* (s-b)*(s-c))
pass
choice=input(“Enter 1-square 2-rectangle 3- tr-iangle”)
if choice==1:
side = input(“Enter side”) area(side)
elif choice ==2:
```

length = input(“Enter length”)
breadth = input(“Enter breadth”)
area(length,breadth) elif choice==3:
a = inputfEnter sidel”)
b = inputfEnter side2”)
c = inputfEnter side3”) area(a,b,c)
else:
print “Invalid choice”

Question 2:
Write a program to find out volume of a cube, cuboid and cylinder using function overloading.

Answer:
from functools import wraps import math def overloaded(func):
@wraps(func)
def overloaded_funcfargs, **kwargs):
for f in overloaded_func.overloads:
try:
return ff args, **kwargs)
except TypeError:
pass 1
else:
# it will be nice if the error message prints a list of
# possible signatures here raise
TypeError(“No compatible signatures”)
def overload_with(func):
overloaded_func.overloads.append(func)
return overloaded_func
overloaded_func.overloads = [func]
overloaded_func.overload_with = overload_with
return overloaded_func ##############
@overloaded
def volume():
print ‘Volume’
pass
@volume.overload_with def _(a):
print ‘Volume of cube=’,a*a*a, ‘cubic units’
pass
@volume.overload_with def _(a,b):
print ‘Volume of cylinder, 3.14*a*a*b,‘cubic units’
pass
@volume.overload_with def _(a,b,c):
print ‘Volume of cuboid=’, a*b*c, ‘cubic units’
pass
choice = input(“Enter 1-cube 2-cylinder 3-cuboid”)

```
if choice==1:
side = input(“Enter side”)
volume (side)
elif choice ==2:
radius = input(“Enter radius”)
height = input( “Enter height”)
volume(radius,height)
elif choice==3:
length = input(“Enter length”)
breadth = input(“Enter breadth”)
height = input(“Enter height”)
volume (length, breadth, height)
else:
print “Invalid choice”
```

Question 3:

Write a class CITY in Python with following specification :

- Code # Numberic value
- Name # String value
- Pop # Numberic value for Population
- KM # Numberic value
- Density # Numberic value for Population Density
  Methods:
- CalDen ( ) # Method to calculate Density as Pop /KM
- Record ( ) # Method to allow user to enter values Code, Name, Pop, KM and call CalDen ( ) method
- See ( ) # Method to display all the date members also display a message “Highly Populated Area” is the Density is more than 12000.

Answer:

```
class CITY:
def _init_ (self):
self. Code = 0
self. Name =” ”
self. Pop = 0
self. KM = 0
self. Density = 0
def CalDen (self):
self. Density = self. Pop/self. KM
def Record (self)
self: Code = input (“Enter Code”)
self.Name=raw_input(“Enter Name”)
self. Pop = input (“Enter population”)
self. KM = input (“Enter KM”)
CalDen (self) // or self.CalDen ( )
```

def See (self):
print Code, name, Pop, KM, Density
if self. Density > 12000:
print (“Highly Populated Area”)
OR print (“Highly populated Area”)
Note : Accept self. _Cose to indicate private members

Question 4:
Give a suitable example using Python code to illustrate single level inheritance considering COUNTRY to be BASE class and STATE to be derived class.

Answer:
Class COUNTRY: statejist = [ ]
def_init_(self, name):
self.name = name class state (COUNTRY):
def_init_(self, name, capital): super ( )._init_(name)
self.capital = capital

Question 5:
Write a class DISTRICT in Python with following specification:
Instance Attributes
– Dcode # Numeric value
– DName # String value
– People # Numeric value for Population
– Area # Numeric value
– Density # Numeric value for Population Density

Answer:

Class DISTRICT ( ) :
def_init_(self, Dcode, DName, People, Area, Density):
self.Dcode = Dcode
self.DName = DName
self.People = People
self. Area = Area
self.Density = Density
Def display (self):
print (“District Code”, self.Dcode)
printf (“District Name”, self.Dname)
printf (“Population”, self.people)
printf (“Area”, self.Area)
printf (“Density”, self.Denstity)

Question 6:
Answer the question (i) to (iv) based on the following:

```
Class Shop (object):
Class shop_(self) :
self . no_of _employees = 0
self. no_of _brands= 0
def getSdate (self) :
self. no_of_employees=input(“Number of employees”)
self.no_of_brands=input(”Number of brands”)
def showSdate (self) :
Print self.no of employees
Print self.no of brands class Brand (object) :
def init_(self) :
self.name = ” ”
self.category=(“Mens”, “Womens”, “Kids”)
self.avgprice=0, 0
def getdate (self) :
self.name = raw_input(“Enter Brand Name”)
self.avgprice = input(”Enter Average Price”)
def showdate (self) :
Print self, name
Print self, category
Print self, avgprice
Class Mall (Brand, Shop) :
def showdate (self) :
self.no of shops = 0
def getdate (self) :
super (mall, self).getSdate ( ) # Statement
super (mall, self).getdate ( ) # Statement 2
self.no_of_shops=input (“Enter number of shops”)
def showdata(self)
print self.no_of_shops
print self.no of brands # Blank 1
```

1. Which type of inheritance is demonstrated in the above code?
2. Explain Statement 1 and 2.
3. Name the methods that are overridden along with their class name.
4. Fill Blank 1 with a statement to display variable category of class Brand.

Answer:

1. Multiple Inheritance
2. Statement 1 and 2 invoke the getSdate() function of class shop and getData() function of class Brand respectively.
3. getdata() method of class Brand is overridden. When object of class Mall is created.
   M=Mall ( )

k.getdata ( )
getdate( ) methodof class Mall is invoked and not of class Brand is called.

4. print Brand ( ). category

TOPIC-2
Classes
Short Answer Type Questions (2 marks)

Question 1:
Give one word for the following:
a. A sort of constructor in Python _____
b. A region of a Python program where a namespace is directly accessible._____
c. It returns the docstring of a class._____
d. It returns the string representation of the object. _____
e. A method used to delete an attribute. _____

Answer:

a. _init_
b. scope
c. _doc_
d. _str_
e. _delattr_ ()

Question 2:
Define a namespace. Give examples of namespaces with respect to Python.

Answer:
Namespace is a mapping from names to objects. Examples of namespaces are built-in names, global names in a module and local names in function invocation.

Question 3:
What is the significance of super method? Give on example of the same.

Answer:
super ( ) function is used to call base class methods which has been extended in derived class.
Example :
class GradStudent (Student):
def_ init _(self) :

```
super (GradStudent, self). _init _( )
self. subject = ” ”
self. working = ” ”
def readGrad (self) :
# Call readStudent method of parent class super (GradStudent, self). readStudent ( )
```

Question 4:
Explain LEGB rule.

Answer:

LEGB rule: when a name is encountered during the execution of the program , it searches for that name in the following order:
L. Local – It first makes a local search, i.e. in current def statement.
E. Enclosing functions – It searches in all enclosing functions, form inner to outer.
G. Global (module) – It searches for global modules or for names declared global
B. Built-in (Python) – Finally it checks for any built in functions in Python.

Question 5:
Is object of a class mutable? Why/why not?

Answer:
User classes are considered mutable. Python doesn’t have (absolutely) private attributes, so you can always change a class.

Question 6:
Explain the usage of keyword ‘pass’ in class definition.

Answer:
When a class doesn’t define any methods or attributes, but syntactically, there needs to be something in the definition, so we use pass. It is a statement that does nothing, and is a good placeholder when you are stubbing out functions or classes.

Question 7:
What is the use of _init_ ? When is it called? Explain with an example.

Answer:

1. _init_ help to create objects and instances to the parent class.
2. It reserve the memory to the members of the class.

Question 8:
Explain the importance of self in Python classes.

Answer:
self is an object reference to the object itself, therefore, they are same. Python methods are not called in the context of the object itself, self in Python may be used to deal with custom object models.

Question 9:
Differentiate between class attributes and instance attributes.

Answer:
The difference is that the attribute on the class is shared by all instances. The attribute on an instance is unique to that instance.

Question 10:
Explain _str_ with an example.

Answer:
str , returns a string representation of a Point object. If a class provides a method named _str_ , it overrides the default behavior of the Python built-in _str_ function.
>>> p = Point(3, 4)
>>> str(p)
'(3,4)'
Printing a Point object implicitly invokes _str_ on the object, so*- defining _str_ also changes the behavior of print:
>>> p = Point(3,4)
>>> print p(3,4)

Question 11:
What do you mean by name mangling? Support your answer with relevant example.

Answer:
Name mangling of the double underscore makes the most sense for achieving "private-ness". Now when a function is called from the 'self' instance
and it notices that it starts with '_', it just performs the name mangling right there. Name mangling is helpful for letting sub-classes override methods without breaking intra class method calls.

Question 12:
Differentiate between reference counting and automatic garbage collection with respect to Python.

Answer:
Reference counting works by counting the number of times an object is referenced by other objects in the system. Python's garbage collector runs during program execution and is triggered when an object's reference count reaches zero. An object's reference count changes as the number of aliases that point to it change. An object's reference

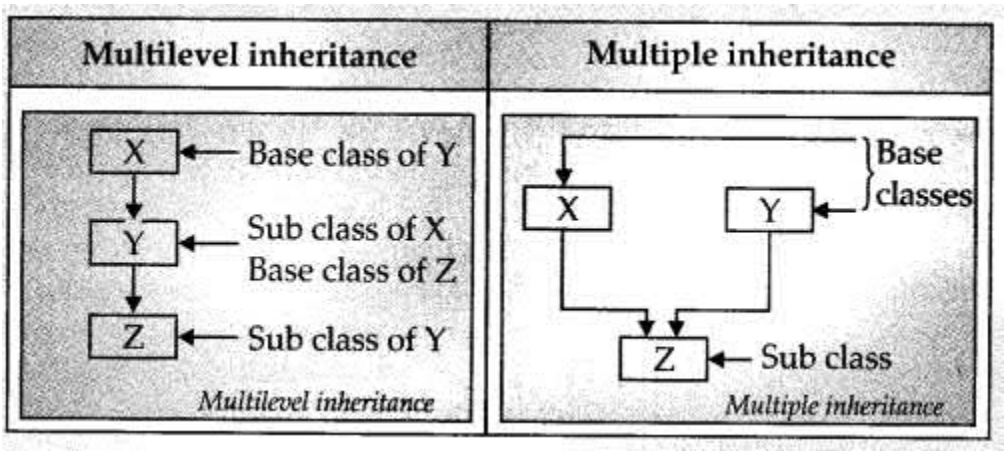
count increases when it is assigned a new name or placed in a container (list, tuple or dictionary). The object's reference count decreases when it is deleted with del, its reference is reassigned, or its reference goes out of scope. When an object's reference count reaches zero, Python collects it automatically.
Automatic garbage collectiomPython deletes the objects which are not required, may it be built-in types or class instances, through the process named garbage collection. When the number of allocations minus the number of de-allocations are greater than the threshold number, the garbage collector is run and the unused block of memory is reclaimed.
One can inspect the threshold for new objects by loading the garbage collector (gc) module and asking for garbage collection thresholds.

Question 13:
Predict the output of the following code snippet:

```
ptr=40
def result():
print ptr
ptr=90
def func(var):
if var<=60:
ptr=30
print ptr
result()
func(60)
func(70)
```

Answer:
UnboundLocalError: local variable 'ptr' referenced before assignment.

Question 14:
What is the difference between Multilevel inheritance and multiple inheritance? Give suitable examples to illustrate.
Answer:



Question 15:
How do you implement abstract method in Python? Give an example for the same.

Answer:
Abstract method : An unimplemented method is an abstract method. When an abstract method is declared in a base class, the drived class has to either difine the method or raise" NothImplemented Error"
class Shape (object):
def findArea (self):
pass
class Square (Shape):
def_ init_(self, side):
self, side = side
def find Area (self):
return self.side*self.side
Note: We can use @ abstract method to enable parent class method to be executed.

Question 16:
Predict the output of the following code snippet:
ptr=50
def result():
global ptr
ptr=ptr+1
print ptr result()
print ptr

Answer:
51
51

Question 17:
Name the methods that can be used to:

1. access attribute of an object
2. delete an attribute of an object

Answer:

1. getattr(obj, name[, default])
2. delattr(obj, name)

Question 18:
Give the statement to:

1. Check whether the attribute str exists in the class Test whose object is T
2. Assign a value “Hello” to the attribute str of class Test and object Tl.

Answer:

1. hasattr (Tl,str)
2. setattr (Tl, str, “Hello”)

Question 19:
Predict the output of the following code:

```
class Match:
#“Runs and Wickets”
runs=281
wickets=5
def init (self,runs, wickets) :
self.runs=runs
self.wickets=wickets
print “Runs scored are :” ,runs
print “Wickets taken are :” .wickets
print “Test. do :”, Match. doc
print “Test._name_ , Match. _name_
print “Test_module_ , Match.__module_
print “Test. _bases_ , Match._bases_
print “Test._dict_ , Match. _diet_
```

Answer:

```
Runs scored are : 281
Wickets taken are : 5
Test._ do_ : None
Test._name_ : Match
Test._module_ : main
Test._bases_ : ()
Test. _diet_ : {‘ _module _’: ‘_ main_ ’, ‘_doc _’: None, ‘runs’: 281, ‘_init_ ’:
<function _init_ at 0x0000000002DDFBA8>, ‘wickets’: 5}
```

Long Answer Type Questions (4 marks)

Question 1:
Write a class customer in Python Containing Deduct % Mark to be deducted if caldiscount () is following specifications. not invoked properly

- Instance attributes: inside input( ) function
  customernumber – numeric value No mark to be deducted if member function
  customemame – string value definitions are written inside the class

- price, qty discount, totalprice, netprice – numeric value
  methods :
- init()-To assign initial values of customernumber as 111, customemame as “Leena”, qty as 0 and price, discount & netprice as 0.
- caldiscount ( ) – To calculate discount, totalprice and netprice
  totalprice = price * qty
- discount is 25% of totalprice, if totalprice >=50000
- discount 15% of totalprice, if totalprice >=25000 and totalprice <50000
- discount 10% of totalprice, if totalprice <250000
  netprice= totalprice – discount
- input()-to read data members customer- name, customernumber, price, qty and call caldiscount() to calculate discount, totalprice and netprice.
- show( ) – to display Customer details.

Answer:

```
class customer:
def _init_(self):
self.customemumber=111
self.customemame=‘Leena’
self.qty=0
self.price=0
self.discount=0
self.netprice=0
def caldiscount(self):
totalprice = self.price*self.qty
if totalprice >= 50000:
self.discount=totalprice * 0.25
elif totalprice >= 25000:
self.discount = totalprice * 0.15 else:
self.discount = totalprice * 0.10
self.netprice = totalprice – self.discount
def input(self):
self.customernumber=input(“Enter Customer Number”)
self.customemame = raw_input(“Enter Customer Name”)
self.qty = input(“Enter Quantity”)
self.price = input(“Enter Price”)
self.caldiscount()
def show(self):
print “Customer Number”,
self.customernumber
print “Customer Name”,
self.customemame
print “Quantity”,self.quanti-ty
print “Price”,self.price
```

```
print “Discount”,self.discount
print “Net price”,self.netprice
c = customer()
c.inputO c.show()
```

Question 2:

Create the class SOCIETY with following information:
society_name,house_no,no_of_members,flat, income

Methods :

- An _init_ method to assign initial values of society_name as “Surya Apartments”, flat as “AType”, house_.no as 20, no_of_ members as 3, income as 25000.
- Inputdata()-To read data members (society,house_no,no_of members & income) and call allocate_flat().
- allocate_flat( )-To allocate flat according to income

Income	Flat
>=25000	A Type
>=20000 and <25000	B Type
<15000	C Type

ShowData() to display the details of the entire class.

Answer:

```
class SOCIETY:
# constructor to create an object
def init (self):
#constructor
self. society_name=‘Surya Apartments’
self.house_no=20
self.no_of_members=3
self.flat=AType’
self.income=25000
def Inputdata(self):
self. society_name = raw_input (“Enter Society Name”)
self.house_no=input(“Enter House Number”)
self.no_of_members = input (“Enter No. of members”)
self.income = float(raw_input (“Enter income”))
def Allocate_Flat(self):
if self.income >= 25000:
self.flat = AType’
elif self.income >= 20000 and self.income < 25000 :
self.flat = ‘BTÿpe’ else:
```

```
self.flat = 'CType'
def Showdata(self):
print "Society Name",
self.society_name
print "House_No",
self.house_no print "No.of members",
self.no_of mem-bers
print "Flat Type",
self.flat print "Income",
self.income
S = SOCIETY))
S.InputdataO
S.Allocate_Flat()
S.ShowdataO
```

Question 3:

Define a class ITEMINFO in Python with the following description:
ICode (Item Code), Item (Item Name), Price (Price of each item), Qty (quantity in stock)
Discount (Discount percentage on the item), Netprice (Final Price)
Methods

- A member function FindDisc( ) to calculate discount as per the following rules:
  If Qty< = 10
  Discount is 0
  If Qty (11 to 20)
  Discount is 15
  If Qty > =20
  Discount is 20
- A constructor init method) to assign the value with 0 for ICode, Price, Qty, Netprice and Discountand null for Item respectively
- A function Buy( ) to allow user to enter values for ICode, Item, Price, Qty and call function FindDisc( )to calculate the discount and N etprice(Price * Qty-Discount).
- A Function ShowAll( ) to allow user to view the content of all the data members.

Answer:

```
class ITEMINFO:
#constructor to create an .object
def_init_(self):
#constructor
self.ICode=0
self.Item=' '
self.Price=0.0
self.Qty=' '
self.Discount=0
self.Netprice=0.0
def Buy(self):
```

```
self.ICode=input("Enter Item Code")
self. Item=raw_input("Enter Item Name")
self.Price=float(raw_input("Enter Price"))
self.Qty=input("Enter Quantity")
def FindDisc(self):
if self.Qty <= 10:
self.Discount = 0
elif self.Qty >= 11 and self.Qty < 20 :
self.Discount = 15
else:
self.Discount = 20
self.Netprice= (self.Price*self.Qty)
self.Discount
def ShowAll(self):
print "Item Code'',self.ICode
print "Item Name",self.Item
print "Price",self.Price
print "Quantity",self.Qty
print "NetPrice",self.Netprice
I = ITEMINFO()
I.Buy()
I.FindDisc()
I. Show All ()
```

Question 4:

Define a class PRODUCT in Python with the following specifications :

Data members:

Pid – A string to store product.

Pname -A string to store tha name of the product. Peostprice – A decimal to store the cost price of the product

Psellingprice – A decimal to store Selling Price Margin- A decimal to be calculated as Psellingprice- Pcostprice

Remarks- To store "Profit" if Margin is positive else "Loss' if Margin is negative

Member Functions :

- A constructor to intialize All the data members with valid default values.
- A method SetRemarks() that assigns Margin as Psellingprice – Peostprice and sets Remarks as mentioned below :

Margin	Remarks
<0 (negative)	Loss
>0 (positive)	Profit

- A method Getdetails() to accept values for Pid. Pname,Psellingprice and invokes SetRemarks() method.
- A method Setdetails () that displays all the data members.

Answer:

```
class PRODUCT:
def init (self):
self. Pid = self. Pname = self. Peostprice = 0.0 self. Psellingprice = 0.0 self. Margin =
0.0 self. Remarks = def SetRemarks (self) :
self . Margin = self.Psellinrprice-self. Peostprice
if (self.Margin < 0) :
self. Ramarks = “Loss”
else:
self. Remarks = “Profit” defGetdetails (self):
self.Pid = rawjnput (“Enter Product Id”)
self.Pname = rawjnput (“Enter Product Name”)
self.Peostprice = input (“Enter Cost Price”)
self.Psellingprice = input (“Enter Selling Price”)
self. SetRemarks ( ) def Setdateils (self) :
print “Product Id” ,
self.Pid print “Product Name”,
self.Pname print “Cost Price”,
self.Pcostprice print “Selling Price”,
self.Esellingprice print “Margin : ” ,
self.Margin print “Incurred :” ,
self.Remarks
```

Question 5:

Write a Python program using classes and objects to simulate result preparation system for 20 students. The data available for each student includes: Name, Rollno, and Marks in 3 subjects. The percentage marks and grade are to be calculated from the following information:

Percentage of marks	Grade
80 to 100	A
60 to 80	B
45 to 60	C
Less than 45	D

GRADE FORMULA

Also demonstrate constructor overloading.

Answer:

```
# constructor to create an object
def init (self,s=None):
#non-copy constructor if s==None:
self.rollno = 0 self.name = ‘ ’
self.marks = [0,0,0]
self.avg = 0.0 self.grade = ‘ ’
```

```
# copy constructor else:
self.rollno = s.rollno self.name = s.name self.marks = s.marks self.avg = s.avg
self.grade = s.grade
def read (self):
# This function gets the details of a student from the user
selfrollno=iaw_input(“Enter roll number.-’)
self.name = raw_input(“Enter name:-”) s=0
for i in range(0,3):
self.marks [i] = int(r a w_input (“Enter the marks ?”)) s = s + self.marksfi] self.avg = s/3
if(self.avg>60):
self.grade=‘A’
elif self.avg>40:
self.grade=‘B’
else:
self.grade=‘C’
def display (self):
# This function prints each student’s details
print self.rollno,“\t”, self, name, “\t\ t”,
self.grade s = StudentO studlist = []
num = int(raw_input(“Enter no. of students:-’’))
for i in range(O.num):
s.read()
studlist. append(Student(s))
print “ STUDENT REPORT \n”
print “**********\n”
print “RollNo \t Name \t\t Grade”
print “**********\n”
for i in range(0,num):
studlist[i] .displayO
#Initialize avg as the first student’s avg maxavg = studlist[0].avg totavg = 0
for i in range(l.num):
totavg = totavg + studlist[i].avg if studlist[i].avg > maxavg: maxavg = studlist[i].avg
topper = studlist[i].name totavg = totavg/num
print “Class topper is”,studlist[i],name,“with average”, studlist [i]. avg
print “Class average is”,totavg
class Student:
```

Question 6:

Define a class SUPPLY in Python with the following description:
Private Members Code of type int FoodName of type string FoodType of type string Sticker of type string
A member function GetType() to assign the following values for Food Type as per the given Sticker

- A function FoodIn() to allow user to enter values for Code, EoodName, Sticker and call function GetType() to assign respective FoodType.

Sticker	Food Type
GREEN	Vegetarian
YELLOW	Contains Egg
RED	Non-Vegetarian

PUBLIC MEMBERS

- A function FoodOut() to allow user to view the contents of all the data members.

Answer:

```
class SUPPLY:
• constructor to create an object
def init (self) :
#constructor
self.FoodCode=()
self.FoodName=‘ ’
self.FoodType=‘ ’
self.Sticker=‘ ’
def FoodIn(self):
self.FoodCode=input(“Enter Food Code”)
self.FoodName=raw_input(“Enter Food Name”)
self.Sticker=raw_input(“Enter Sticker Colour”)
def GetType(self):
if self.Sticker==‘GREEN’:
self.FoodType = ‘Vegetarian’
elif self.Sticker==’YELLOW’:
self.FoodType = ‘Contains Egg’
elif self.Sticker==‘RED’:
self.FoodType = ‘Non-Vegetarian’ else:
self.FoodType = ‘Not Known’
def FoodOut(self):
print “FoodCode’’,self.FoodCode
print “FoodName”,self.FoodName
print “FoodType”,self.FoodType
print “Sticker”,self.Sticker
S = SUPPLY()
S.FoodIn()
S.GetTÿpe()
S.FoodOut()
```

TOPIC-3
Inheritance
Very Short Answer Type Questions (1 mark)

Question 1:
Give one example for an abstract method.

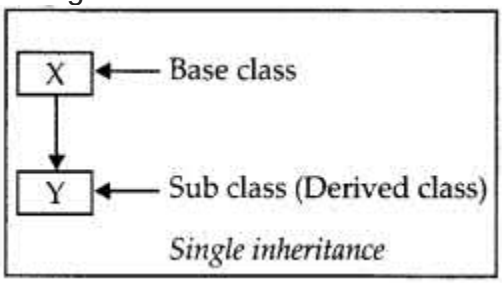
Answer:
An abstract method is a method declared in a parent class, but not implemented in it. The implementation of such a method can be given in the derived class, class circle(object):
def getradius(self):

Question 2:
Define the term inheritance.

Answer:
Inheritance is a mechanism in which a new class is derived from an already defined class. The derived class is known as a subclass or a child class. The pre-existing class is known as base class or a parent class or a super class. The mechanism of inheritance gives rise to hierarchy in classes. The major purpose of inheriting a base class into one or more derived class is code reuse. The subclass inherits all the methods and properties of the super class.

Question 3:
What is single inheritance?

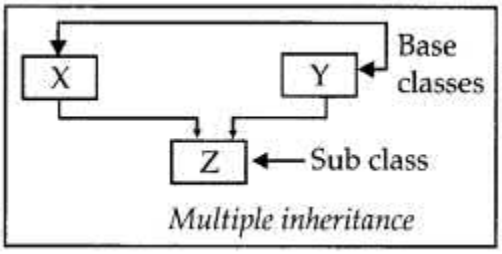
Answer:
In single inheritance a subclass is derived from a single base class.



Question 4:
What is multiple inheritance? Explain with an example.

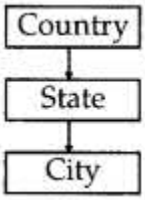
Answer:
In this type of inheritance, the derived class inherits from one or more base classes. In

the figure below, X and Y are the base classes while Z is the derived class.



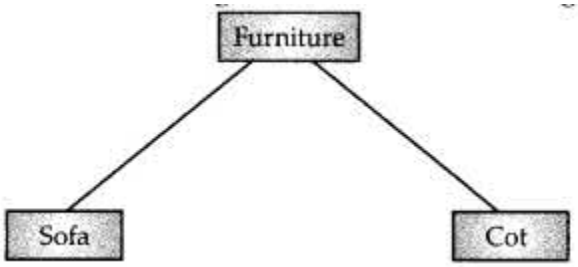
Question 5:
Give one example of multilevel inheritance.

Answer:
In multilevel inheritance, the derived class becomes the base of another class.
For example, Country is the base class of State and City is the derived class of State.



Short Answer Type Questions (2 marks)

Question 1:
Based on the diagram, answer the following:



1. Write the name of the base class and the derived classes.
2. Write the type of inheritance depicted in the above diagram.

Answer:

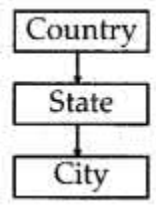
1. Base class – Furniture; Derived classes – Sofa & Cot
2. Hierarchical inheritance

Question 2:
Explain the concept of overriding methods.

Answer:
Overriding is a very important part of OOB since it is the feature that makes inheritance exploit its full power. Through method overriding, a class may “copy” another class, avoiding duplicated code, and at the same time enhance or customize part of it.

Question 3:
Based on the diagram, answer the following:

1. Write the name of the base class and derived class of state.
2. Write the type of inheritance depicted in the diagram.



Answer:

1. Base class of state – Country
   Derived class of state – City
2. Multilevel inheritance

Question 4:
What are the different ways of overriding function call in derived class of python? Illustrate with example.

Answer:
Overriding enables the programmer to provide specific implementation to a method in the derived class. So, the method invoked depends on the object used to invoke it. If base class object is used then base class version of the method is called else the derived class method is called, class Parent:
def func(self):
print ‘Parent method’ class child:
def func(self):
print ‘child method’
C = child()
c.func() # child calls overridden method

Question 5:
How do we implement abstract method in Python?
Support your answer with an example.

Answer:
An abstract method is a method defined in a base class, but that may not provide any implementation. It is done by using the abc module in Python, import abc
class Shape(object):
metaclass = abc.ABCMeta
@abc.abstractmethod
def method_to_implement(self, input):
“’’’Method documentation
return

Question 6:
Find the output of the following code and write the type of inheritance:
p=Gstudent(‘Mona’, 20,12, $9, ‘computer’)
def_init_(self, name, age, roll no, marks, stream):
super (Gstudent, self) _init_(name, age, roll no, marks)
self stream=stream
def display 2 (self):
self display()
print “stream:”,
self.stream P=Gstudent (‘Mona’, 20,12, 99, ‘Computer’)
p.display2()

Answer:
TVpe of inheritance: Multilevel Output:
Name : Mona
Age : 20
Roll No: 12
Marks : 99
stream: computer

Question 7:
Rewrite the following code after removing errors. Underline each correction and write the output after correcting the code:
class First():
def _init_ (self):
print “first”:
class Second(object):
def _init_ (self):
print “second” class Third(First, Second):
def _init_ (self):
First. _init_ (self):
Second. _init_ (self):
print “that’s it”
t=Third()
t=Third()

Answer:

```
class First():
def _init_ (self):
print “first”
class Second(object):
def_ init_(self):
print “second” class Third(First, Second):
def _init_ (self):
First._init_ (self)
Second._init_(self)
print “that’s it”
t=Third()
t=Third()
```

Question 8:

Complete the following code:

```
class employee(object):
def _init_(self,no,name,age):
self.no=no
self.name=___ #complete the statement
self.age= ___ #complete the statement
def printval(self):
print “Number:”,
self.no print “Name :”,
self.name print “Age :”,
self.age class pay(object):
def _init_ (self,dept,salary): #complete
the definition
_________
_________
def display(self): #complete the definition
_______ # call printval()
_______ # print dept
_______ # print salary
```

Answer:

```
class employee (object):
def init (self,no,name,age):
self.no=no
self.name=name
self.age=age
def printval(self):
print “Number:”,
self.no print “Name :”,
self.name
print “Age :’’,
```

```
self.age
class pay(object):
def _init_ (self,dept,salary):
self.dept =dept
self.salary=salary
def display(self)
self.printval()
print self.dept
print self.salary
```

Long Answer Type Questions (4 marks)

Question 1:

Write a Python program to demonstrate multiple inheritance. Consider 3 classes with the following description.

Student class has 3 protected data members roll number, markl and mark2 of type integer. It has a get() function to get these details from the user. Sports class has a protected data member sports marks of type integer and a function getsm() to get the sports mark.

Statement class uses the marks from Student class and the sports marks from the Sports class to calculate the total and average and displays the final result

Answer:

```
class student(object):
# constructor to create an object def init (self):
self.mo=0
self.m1=0
self.m2=0
def get(self):
self.mo=int(raw_input(“Enter the Roll no:”))
print “Enter 2 marks”
self.m1=int(raw_input(“Markl?”))
self.m2=int(raw_input(“Mark2 ?”))
classsports(object):
# Sports mark
def_init_(self):
self.sm=0
def getsm(self):
self.sm=int(raw_input(“Enter the sports mark:”))
class statement(student,sports):
def_init_(self):
super(statement,self)._init_()
def display (self):
tot=(self.ml+self.m2+self.sm); avg=tot/3;
print“\n\n\tRoll No. : ”,
self.mo,“\n\tTotal :”,tot print“\tAverage : ”,avg
```

```
obj=statement()
obj.get()
obj.getsm()
obj. display()
```

Question 2:

SKP Hotel offers accommodation, meals facilities.
Create a class Accommodation with Room Number, type of room, and rent, etc..
Create a class meals services includes: meals code, name, price, etc..
Create a class customer with customer number, name, address, etc.
Customer class is derived by using Accommodation and meals classes.

Answer:

```
class Accommodation(object):
# constructor to create an object
def_init_(self):
self.roomno=0
self.roomtype=”
self.roomrent=0.0
defgetroom(self):
self.roomno=input(“Enter the Room no:”)
self.roomtype=rawjnput(“Enter the Room type:’)
self.roomrent=float(raw_input(“Enter Room Rent”))
class Meals(object):
# Meals class
def_init_(self):
self.mealcode=0
self.mealname=“”
self.mealprice=0.0
def getmeals(self):
self.mealcode=input(“Enter the meal code:”)
self.mealname=raw_input(“Enter the meal name:”)
self.mealprice=float(raw_input(“Enter the meal price”))
class Customer(Accommodation,Meals):
def_init_(self):
super(Customer,self)._init_()
self.custnum=0
self.custname=‘ ’
self.address=‘ ’
def getCustomer(self):
self.custnum=input(“Enter the customer number”)
self.custname=raw_input(“Enter customer name”)
self. address=raw _input(“Enter customer address”)
def displaybill(self):
print“Customer Name:”,
self.custname,“Address:”, self, address
```

```
print“Room No:”,
self.roomno,“Room Type:”,
self.roomtype,“Room Rent: Rs’jself.roomrent
print“Meal Name:”,
self.mealname,“Price:”,
self.mealprice
print“Total Amount Due:”,
self.roomrent+ self.mealprice
obj=Customer()
obj.getCustomer()
obj.getroom()
obj.getmeals()
obj.displaybill()
```

Question 3:

Pay roll information system:

Declare the base class ’employee’ with employee’s number, name, designation, address, phone number. Define and declare the function getdata() and putdata() to get the employee’s details and print employee’s details. Declare the derived class salary with basic pay, DA, HRA, Gross pay, PF, Income tax and Net pay. Declare and define the function getdatal() to call getdata() and get the basic pay. Define the function calculate() to find the net pay. Define the function display() to call putdata() and display salary details .

Create the derived class object. Read the number of employees. Call the function getdatal)) and calculate() to each employees. Call the display() function.

Answer:

```
class employee (object):
# constructor to create an object
def init (self):
self.eno=0
self.name=‘ ’
self.designation=‘ ’
self.address = ‘ ’
self.phonenumber=0
def getdata(self):
self.eno=input(“Enter Employee Number”)
self.name=raw_input(“Enter Employee Name’)
self.designation=raw_input(“Enter Employee Designation”)
self.address = raw_input(“Enter Employee Address”)
self.phonenumber=input(“Enter Employee Phone Number”)
def putdata (self):
print “Employee Number”,self.eno
print “Employee Name”,self.name
print “Employee Designation”,self.designation
print “Employee Address”,self.address
```

```
print “Employee Phone Number”,self phonenumber
class salary (employee):
# Salary details
def_init_(self):
super(salaryself)._init_()
self.basic=0
self.DA=0
self.HRA=0
self.Gross=0
self.PF=0
self.tax=0
self.netpay=0
def getdatal(self): self.getdata()
self.basic=float(raw_input(“Enter the basic pay’’))
self.DA=float(raw_input(“Enter the DA”))
self.HRA=float(raw_input(“Enter the HRA))
self.PF = float(raw_input(“Enter the PF”))
def calculate(self):
self.Gross= self.basic + self.FîRA + self.DA
if .self. Gross < 100000:
self.tax=0
elif self.Gross < 500000:
self.tax=self.Gross*0.10
elif self.Gross < 1000000:
self.tax=self.Gross*0.15
else:
self.tax=self.Gross*0.20
self.netpay= self.Gross – self.PF – self.tax
def display(self):
self.putdataO
print “Gross Income”,self.Gross
print “Tax “,self.tax
print “Net Income”,self.netpay
salaryobj =salary()
num = int(raw_input(“Enter no. of employees:-”))
for i in range(0,num):
salaryobj.getdatal()
salaryobj.calculate()
salaryobj.display()
```

Question 4:

Define a class employee in Python with the given specifications:

Instance variables:

Employee number, name Methods:

Getdata()- To input employee number and name

Printdata()- To display employee number and name

Define another class payroll, which is derived from employee
Instance variable Methods:
Inputdata() – To call Getdata() and input salary.
Outdata() – To call Printdata() and to display salary.
Define another class leave, which is derived from payroll.
Instance variable No of days Methods:
acceptdata() – To call Inputdata() and input no of days.
showdata() – To call Outdata() and to display no of days.
Implement the above program in python.

Answer:

```
class Employee(object):
# constructor to create an object
def_init_(self):
self.eno=0
self.name=“”
def Getdata(self):
self.eno=input(“Enter the Employee no:”)
self.name= raw_input(“Enter the Employee Name:”)
def Printdata(self):
print “Employee Number”,
self.eno print “Employee Name”,
self.name
class payroll(Employee):
def_init_(self):
super(payroll,self). _init_()
self.salary=0
def Inputdata(self): self.GetdataQ
self.salary=float(raw_input(“Enter the salary:”))
def Outdata(self): self.Printdata()
print “Salary is”,self.salary
class leave(payroll):
def_init_(self):
super(leave,self)._init_()
self.Noofdays=0 def acceptdata(self):
self.Inputdata()
self.Noofdays =input(“Enter leave days”)
def showdata(self): self.Outdata()
print “No. of leave days”,
self.Noofdays leaveobj = leave()
leaveobj. acceptdataO
leaveobj.showdata()
```

Question 5:

What output will be generated when the following Python code is executed?

```
def changeList () :
L = []
LI = [ ]
L2 = []
for i in range (10,1, -2) :
LI append (i)
for i in range (len ( LI)) :
L2. append (LI [i] + L [i] )
print L2
ChangeList ( )
```

Answer:

[11, 10,9, 8, 7,4]