GANPAT UNIVERSITY
U. V. PATEL COLLEGE OF ENGINEERING
B.Tech. CE/IT Semester IV
2CEIT404: Python Programming

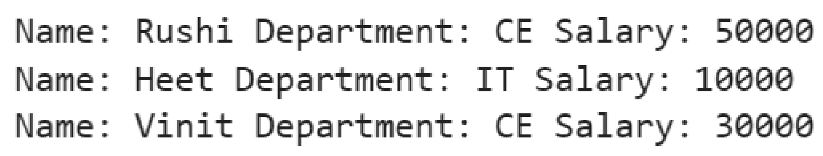
Practical-8: Object oriented programming with python

1) Create a class Employee with data members: name, department and salary. Use constructor to initialize values and display() method for printing information of three employees.

INPUT:

```
# Question1
class Employee:
    def __init__(self,name,department,salary):
        self.nm=name
        self.dpt=department
        self.sal=salary
    def display(self):
        print("Name:",self.nm,"Department:",self.dpt,"Salary:",self.sal)
e1=Employee("Rushi","CE",50000)
e2=Employee("Heet","IT",10000)
e3=Employee("Vinit","CE",30000)
e1.display()
e2.display()
e3.display()
```

OUTPUT:



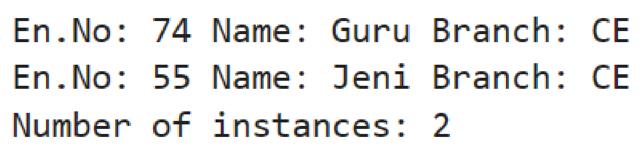
```
Name: Rushi Department: CE Salary: 50000
Name: Heet Department: IT Salary: 10000
Name: Vinit Department: CE Salary: 30000
```

2) Write a program to create class Student with following attributes: instance variables enrollment_no, name and branch; instance methods get_value() and print_value(); class variable cnt; static method show(). Variable cnt counts number of instances created and show() method displays value of cnt.

INPUT:

```
# Question2
class Student:
    cnt=0
    def __init__(self,):
        Student.cnt +=1
    def get_values(self,en_no,name,branch):
        self.en=en_no
        self.nm=name
        self.br=branch
    def print_values(self):
        print("En.No:",self.en,"Name:",self.nm,"Branch:",self.br)

    @staticmethod
    def show():
        print("Number of instances:",Student.cnt)

s1=Student()
s1.get_values(74,"Guru","CE")
s2=Student()
s2.get_values(55,"Jeni","CE")
s1.print_values()
s2.print_values()
Student.show()
```

OUTPUT:



```
En.No: 74 Name: Guru Branch: CE
En.No: 55 Name: Jeni Branch: CE
Number of instances: 2
```

3) Write a program to overload ** (exponential) operator.

INPUT:

```
# Question3
class exp:
    def __init__(self,a):
        self.a=a
    def __pow__(self,o):
        return self.a**o.a
ob1=exp(1)
ob2=exp(2)
print(ob1 ** ob2)
```

OUTPUT:

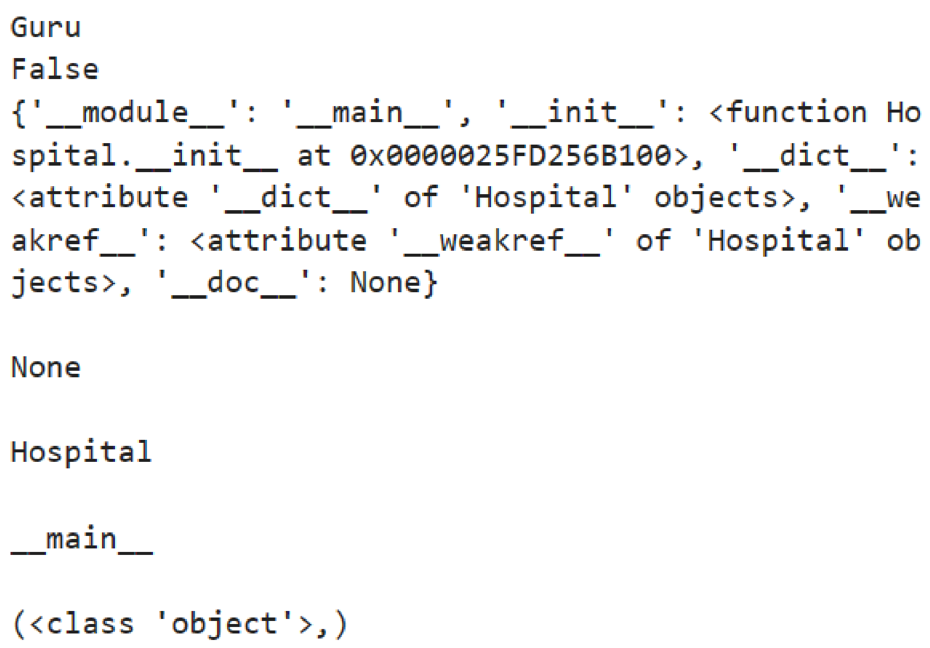
```
1
```

4. Create class Hospital having attributes patient_no, patient_name and disease_name and an instance p1. Show use of methods getattr(), setattr(), delattr(), and hasattr() for p1. Display values of attributes __dict__, __doc__, __name__, __module__, __bases__ with respect to class Hospital. Delete instance p1 in the end.

INPUT:

```
# Question4
class Hospital:
    def __init__(self,pat_no,pat_name,dis_name):
        self.pat_no=pat_no
        self.pat_name=pat_name
        self.dis_name=dis_name
p1=Hospital(74,"Guru","Corona")
print(getattr(p1,'pat_name'))
setattr(p1,'pat_no',72)
delattr(p1,'dis_name')
print(hasattr(p1,'dis_name'))

print(Hospital.__dict__,end="\n\n")
print(Hospital.__doc__,end="\n\n")

print(Hospital.__name__,end="\n\n")

print(Hospital.__module__,end="\n\n")

print(Hospital.__bases__,end="\n\n")
```

OUTPUT:



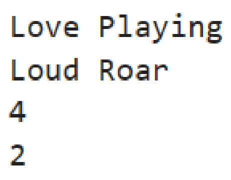
```
Guru
False
{'__module__': '__main__', '__init__': <function Ho
spital.__init__ at 0x0000025FD256B100>, '__dict__':
<attribute '__dict__' of 'Hospital' objects>, '__we
akref__': <attribute '__weakref__' of 'Hospital' ob
jects>, '__doc__': None}

None

Hospital

__main__

(<class 'object'>,)
```

5) Design a class Lion having method roar() and a class Cub having method play() which inherits class Lion. Use instance of Cub called- simba to access methods roar() and play(). Define public attribute legs, protected attribute ears and private attribute mane of class Lion. Show accessibility of these variables according to their scope.

INPUT:

```
# Question5

class Lion:

    def __init__(self,legs,ears,name):

        self.legs=legs

        self._ears=ears

        self.__name=name

    def roar(self):

        print("Loud Roar")

class cub(Lion):

    def __init__(self,legs,ears,name):
        super().__init__(legs,ears,name)
    def play(self):
        print("Love Playing")


c=cub(4,2,"Simba")
c.play()
c.roar()
print(c.legs)
print(c._ears)
```

OUTPUT:



```
Love Playing
Loud Roar
4
2
```

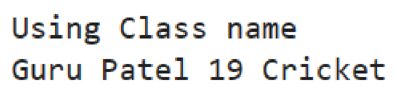
6) Class Person with attributes- name and age is inherited by class SportPerson with attribute sport_name. Use appropriate __init__() method for both classes. Call parent __init__() method from child __init__() method with the help of (A) super() method (B) parent class name.

INPUT:

```
# Question6
class Person:
    def __init__(self,name,age):
        self.nm=name
        self.age=age
class SportPerson(Person):
    def __init__(self,name,age,sport_name):


        super().__init__(name,age)
        self.sn=sport_name


    def Print(self):
        print(self.nm,self.age,self.sn)
```

```
x=SportPerson("Guru Patel",19,"Cricket")

print("Using Class name")

x.Print()
```

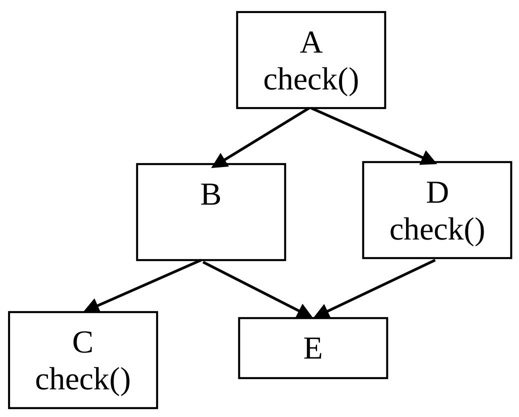
OUTPUT:



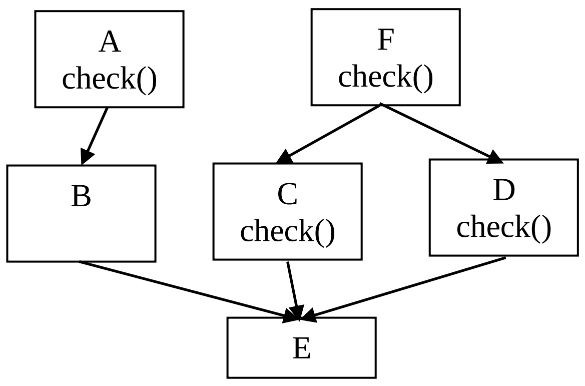
```
Using Class name
Guru Patel 19 Cricket
```

7) Write programs to implement following scenarios where A, B, C, D, E and F are classes and check() is a method. In both scenarios, which check() method is called, when we execute statement- E()**.**check()

Scenario-1



Scenario-2



INPUT:

```
# Question 7(a)

class A:

   def check():

      print('class A')

class B(A):

   pass

class C(B):

   def check():

      print('class C')

class D(A):

   def check():

      print('class D')

class E(B):

      pass
```

```
E.check()


# Question 7(b)
class A:
    def check():
        print('class A')
class B(A):
    pass
class C(F):
    def check():
        print('class C')
class D(F):
    def check():
        print('class D')
class E(B,C,D):
    pass
class F:
    def check():
        print('class F')


E.check()
```

OUTPUT:

```
class A
```

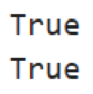
```
class A
```

8) Write a program in which Python and Snake sub classes implement abstract methods- crawl() and sting() of the super class Reptile. What is the output of following statements?
i) **issubclass**(Python, Reptile) ii) **isinstance**(Snake(), Reptile)

INPUT:

```
# Question8
class Reptile:
    def crawl():
        pass
    def sting():
        pass

class Python(Reptile):
    def crawl():
        pass
    def sting():
        pass
class snake(Reptile):
    def crawl():
        pass
    def sting():
        pass

print(issubclass(Python,Reptile))
print(isinstance(snake(),Reptile))
```

OUTPUT:



```
True
True
```