Program: Python program to create and list of employees using Employee class.It should also print total number of employees.

```
class Employee:
   empcount=0 #class variable
   def _init_(self,eid): #constructer to set id
      self.id=eid
   #instance methods
   def set_name(self,ename):
      self.name=ename
   def get_name(self):
      print(f"The Employee name is {self.name}")
   def get_id(self):
      print(f"The Employee id is {self.id}")
   #class method
   @classmethod
   def set_emp_count(cls,s):
      cls.empcount +=1


e1=Employee(156)
e1.set_name('Niyati')
e1.set_emp_count(e1)


e2=Employee(157)
e2.set_name('Kaveri')
e2.set_emp_count(e2)


e3=Employee(158)
e3.set_name('Soweda')
e3.set_emp_count(e3)
print("Employee details are:")
```

```
e1.get_name()
e1.get_id()
e2.get_name()
e2.get_id()
e3.get_name()
e3.get_id()
print(f"Total number of emloyees are {Employee.empcount}")
```

OUTPUT:

```
Employee details are:
The Employee name is Niyati
The Employee id is 156
The Employee name is Kaveri
The Employee id is 157
The Employee name is Soweda
The Employee id is 158
Total number of emloyees are 3
```

Program to demonstrate multiple inheritance with three classes, Employ Student and Intern, where Intern inherits from Employee and Student.

```
class Employee:
    def __init__(self, eid):
        self.id = eid

    def set_name(self, name):
        self.name = name

    def get_name(self):
        print(f"The Employee name is {self.name}")

    def get_id(self):
```

```
        print(f"The Employee id is {self.id}")


class Student:
    def __init__(self, name, college):
        self.college = college


    def get_college(self):
        print(f"The College name is {self.college}")


class Intern(Employee, Student):
    def __init__(self, eid, college, period):
        super().__init__(eid)
        self.college = college
        self.period = period


    def set_details(self, name):
        self.name = name


    def get_details(self):
        return self.name


i1 = Intern("105", "Thadomal", 6)
i1.set_details("Niyati")
i1.get_id()
i1.get_college()
print(f“The name is {i1.get_details()}”)
```

Output:

```
The Employee id is 105
The College name is Thadomal
The name is Niyati
```

Python program to overload greater than (>) operator to make it act on user defined class objects

```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def __gt__(self, other):
        return self.area() > other.area()

r1 = Rectangle(5, 10)
r2 = Rectangle(3, 8)

if r1 > r2:
    print("r1 has a greater area than r2")
else:
    print("r2 has a greater area than r1")
```

Output:

```
r1 has a greater area than r2
```

Python program to demonstrate concept of Interfaces.

```
from abc import *
class Printer(ABC):
    @abstractmethod
    def printit(self,text):
        pass
    @abstractmethod
```

```
    def disconnect(self):
        pass
class IBM(Printer):
    def printit(self,text):
        print(text)
    def disconnect(self):
        print("Printed on IBM")
class HP(Printer):
    def printit(self,text):
        print(text)
    def disconnect(self):
        print("Printed on HP")


user_choice=input("Enter name of printer: ").upper()
classname=globals()[user_choice]
x=classname()
x.printit("Request sent for Printing")
x.disconnect()
```

OUTPUT:

```
Enter name of printer: ibm
Request sent for Printing
Printed on IBM

Enter name of printer: Hp
Request sent for Printing
Printed on HP
```