UNIT...1

1.Write a program to swap two numbers without taking a temporary variable.

```
a=10
b=20
print ("Before swap")
print (a)
print (b)
c=a+b
a=c-a
b=c-a
print ("After swap")
print (a)
print (b)
```

2.Write a program to display sum of two complex numbers.

```
a=10
b=3
c=20
d=-5
print(complex(a,b)+complex(c,d))
```

```
c1=10+2j
c2=30-1j
c3=c1+c2
print(c3)
```

3.Write a program to create byte type array, read, modify and display the elements of the array.

```
elements =[10,20,30,45,67]
x=bytearray(elements)
print("Original byte array")
for i in x: print(i)
x[0]=33
x[1]=55

print("Modified byte array")
for i in x: print(i)
```

4.Create a sequence of numbers using range datatype to display 1 to 30 , with an increment of 2

```
r=range(30)
print("Original")

for i in r: print(i)
print("Incremented by 2")
r=range(1,30,2)
for i in r: print(i)
```

5..Write a program to find out and display common and non common elements in the list using membership operators

```
lst=[10,11,12,12,13,14,10,10]
```

```
s={}
print("All elements of the list are")
for ele in lst:
   print(ele)



for ele in lst:
   cnt=0
   for ele1 in lst:
      if(ele==ele1):
         cnt=cnt+1


   if(cnt>=2):
      s[cnt]=ele
      # print("Common element is %i"  %ele)


print("Common element is",s.values(),"is repeated",s.keys(),"times")
print(s)
```

6..Create a program to display memory locations of two variables using id() function, and then use identity operators to compare whether two objects are same or not.

```
a=25
b=25
if(a is b):
      print("Both are same")
else:
   print("Both are not same")
```

```
print("Ids of a and b are")

print("id of a is %i"%id(a))
print("id of b is %i"%id(b))

lst1=[1,2,3,4]
lst2=[1,2,3,4]

if(lst1 is lst2):
    print("Both list are same")
else:
   print("Both list are not same")

print("Ids of list1 and list2 are")
print("id of List1 is %i"%id(lst1))
print("id of List2 is %i"%id(lst2))
```

7..write a program that evaluate an expression given by the user at run time using eval() function.

```
x=eval(input("Enter your expression"))
print("Result is",x)
```

8..Write program to find sum of even numbers using command line arguments

```
import sys

sum=0
```

```
args=sys.argv
print("All arguments one by one")
for x in args:
   print(x)

n=len(sys.argv)
j=1
while(j<=n-1):
   y=int(sys.argv[j])
   if(y%2==0):
      sum=sum+y
   j=j+1

print("Sum is ",sum)
```

9..Write menu driven program that performs following operations :-

find area of circle

find area of triangle

find area of square

find area of rectangle

find simple interest

```
while(True):
   print("Select any one option from following: ")
   print("\n 1. Area of circle")
   print("\n 2. Area of triangle")
   print("\n 3. Area of Square")
   print("\n 4. Area of Rectangle")
```

```
print("\n 5. Simple Interest")
print("\n 6. Exit prom the program")
ch=int(input("Press number between 1 to 5"))

if(ch==1):
    print("Find area of circle")
    r=float(input("Enter Radius"))
    a=3.14*r*r
    print("The area of circle is ",a)
elif(ch==2):
    print("Find area of triangle")
    h=float(input("Enter height"))
    b=float(input("Enter base"))
    a=(b*h)/2
    print("The area of triangle is ",a)
elif(ch==3):
    print("Find area of square")
    l=float(input("Enter length"))
    a=l*l
    print("The area of square is  ",a)
elif(ch==4):
    print("Find area of rectangle")
    l=float(input("Enter length"))
    b=float(input("Enter  base"))
    a=l*b
    print("The area of rectangle is  ",a)
elif(ch==5):
    print("Find area of simple interest")
```

```
        p=float(input("Enter principle amount"))
        r=float(input("Enter rate of ineterest"))
        n=int(input("Enter number of year"))
        si=(p*r*n)/100
        print("The simple interest is  ",si)
    elif(ch==6):
        break
    else:
        print("Wrong choice entered")
        break
```

10...Write a program to assert to enter number greater then zero

```
no=int(input("Enter no greater then zero"))

assert no>0, "Number is not greater then zero"
print(no , "is greater then zero")
```

11..Write a program to search an element in the list using for loop and also demonstrate the use of else with for loop

```
lst=[1,2,3,10,11,20,30]

x=int(input("Enter element you want to search"))
flag=0

for i in lst:
```

```
    if(i==x):
        flag=1
        print(x, "is present in the list")

else:
    if(flag==0)      print(x,"is not present in the list")
```

12..Write a python program that asks the user to enter a length in centimeter. if the user enters a negative length, the program should tell the user that the entry is invalid. otherwise the program should convert the length to inches and print out the result.(2.54=1 inch)

```
x=int(input("Enter the length in centimeter"))

try:
    assert x>0

    inch=x/2.54
    print("The length in inch is  ",inch)

except AssertionError:
    print("The length isEmail This In valid")
```

UNIT....2

1..Write a program to create one array from another array.

```
#Initialize array
arr1 = [1, 2, 3, 4, 5];

#Create another array arr2 with size of arr1
arr2 = [None] * len(arr1);

#Copying all elements of one array into another
for i in range(0, len(arr1)):
   arr2[i] = arr1[i];

#Displaying elements of array arr1
print("Elements of original array: ");
for i in range(0, len(arr1)):
  print(arr1[i]),

print();

#Displaying elements of array arr2
print("Elements of new array: ");
for i in range(0, len(arr2)):
  print(arr2[i]),
```

2..Create a program to retrieve, display and update only a range of elements from an array using indexing and slicing in arrays.

```
from array import*
x=array('i',[1,2,3,4,5,6,7,8,9,10])
```

```
y=x[2:3]
print(y)

y=x[:3]
print(y)

y=x[:3]
print(y)

y=x[-4:]
print(y)

y=x[-4:-2]
print(y)

y=x[-4:-1]
print(y)

y=x[-5:-1]
print(y)

y=x[1:8:2]
print(y)
```

3..Write a program to understand various methods of array class mentioned: append, insert, remove, pop, index, tolist and count.

```
from array import*
```

```
a=array('i',[1,2,3,4,5])
print("Original array",a)

#append
a.append(10)
a.append(20)
print("afet append value is 10 and 20 the array is",a)

#insert
a.insert(30,40)
print("insert 30 and 40 at the position in array ",a)

#remove
a.remove(4)
print("Value remove in index 4 in array",a)

#pop is use last index value delete
a.pop()
print("The array afetr pop function",a)

#index()
print("the index position of 5 in array",a.index(5))

#tolist
list=a.tolist()
print("the array convert to list",list)

#count
```

```
x=int(input("Enter no 4 want to count from array"))
print("the occurance of ",x,"is",a.count(x))
```

4..Write a program to sort the array elements using bubble sort technique.

```
def bubbleSort(nlist):
    for passnum in range(len(nlist)-1,0,-1):
        for i in range(passnum):
            if nlist[i]>nlist[i+1]:
                temp = nlist[i]
                nlist[i] = nlist[i+1]
                nlist[i+1] = temp

nlist = [14,46,43,27,57,41,45,21,70]
bubbleSort(nlist)
print(nlist)
```

5..Create a program to search the position of an element in an array using index() method of array class.

```
from array import*
x=array('i',[])
n=int(input("Enter The no of Element"))
for i in range(n):
    x.append(int(input("Enter the No")))
```

```
print(x)
key=int(input("Enter The element you want to serch"))

#index method
try:
    pos=x.index(key)
    print('found at psotion',pos+1)

except ValueError:
    print('not found in arry')
```

6..Write a program to generate prime numbers with the help of a function to test prime or not.

```
num = int(input("Enter a number: "))

if num > 1:
  for i in range(2,num):
    if (num % i) == 0:
      print(num,"is not a prime number")
      #print(i,"times",num//i,"is",num)
      break
  else:
    print(num,"is a prime number")

else:
  print(num,"is not a prime number")
```

7..Write a python program that removes any repeated items from a list so that each item appears at most once. For instance, the list [1,1,2,3,4,3,0,0] would become [1,2,3,4,0].

```
def Remove(duplicate):
    final_list = []
    for num in duplicate:
        if num not in final_list:
            final_list.append(num)
    return final_list

duplicate = [1,1,2,3,4,3,0,0]
print(Remove(duplicate))
```

8..Write a program to pass a list to a function and display it.

```
def display(a):
    for x in a:
        print(x)

list1 = [1,2,3,4,5,6]

display(list1)
```

9..Write a program to demonstrate the use of Positional argument, keyword argument and default arguments.

```
# Positional Arguments
```

```
def attach(s1,s2):
        print(s1+s2)

attach("New","Delhi")

# Keywprd Arguments

def grocery(item,price):
        print("Item: ",item)
        print("Price: ",price)

grocery(item = 'Sugar',price = 40)
grocery(price = 80,item = 'Oil')


# Default Arguments

def grocery1(item,price=40):
        print("Item: ",item)
        print("Price: ",price)

grocery1(item = 'Sugar',price = 70)
grocery1(item = 'Oil')
# Take price as 40 By deafult
```

10.Write a program to show variable length argument and its use.

```
def add(farg, *args):
```

```
        print('Formal arguments: ',farg)
        sum = 0
        for i in args:
                sum += i
        print("Sum of all numbers: ",(farg+sum))

add(5,10)
add(5,10,20,30,40,50)
```

11..# Write a lambda/Anonymous function to find bigger number in two given numbers.

```
max = lambda x,y: x if x>y else y
a,b = [int(n) for n in input("Enter two numbers in [n1,n2]: ").split(',')]
print("bigger number :- ",max(a,b))

#value enter 12,36
```

12..Create a decorator function to increase the value of a function by 3.

```
def decor(fun): # this is decorator function
        def inner():
                value = fun()
                return value+3
        return inner

@decor
```

```
# result_fun = decor(num)


def num():
        return 10


print(num())
```

14...# Write a program to create a list using range functions

 # and perform append, update and delete elements operations in it.

```
list1 = list(range(10))


print(list1) # Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]


list1.append(10)
print(list1) # Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]


list1.insert(9,8.5) # 9 represents Position , 8.5 represents Value


print(list1)  # Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 8.5, 9, 10]


del list1[9]


print(list1) # Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

15..Write a program to combine two List, perform repetition of lists and create cloning of lists.

```
list1 = [10,20,30,40,50]

list2 = [1,2,3,4,5]


# Combining two list


list3 = list1+list2

print("Combined List: ",list3) # Output:  [10, 20, 30, 40, 50, 1, 2, 3, 4, 5]



# Repetition of list


print("Repetition of List: ",list1 * 3) # Output:  [10, 20, 30, 40, 50, 10, 20, 30, 40, 50, 10, 20, 30, 40, 50]


# Cloning List

clonedList = list2[:]


list2[1] = 99


print("Updated List: ",list2) # Output: [1, 99, 2, 3, 4, 5]

print("Cloned List: ",clonedList) # Output: [1, 2, 3, 4, 5]
```

16..Create a sample list of 7 elements and implement the List methods mentioned: append, insert, copy, extend, count, remove, pop, sort, reverse and clear.

```
sampleList = [10,12,11,13,9,14,8]
```

```
sampleList.append(6)
print("Append List: " ,sampleList) # Output: [10, 12, 11, 13, 9, 14, 8, 6]

sampleList.insert(6,15) # 6 represents Position , 15 represents Value
print("Insert List: ",sampleList) # Output: [10, 12, 11, 13, 9, 14, 15, 8, 6]

copyList = sampleList.copy()
print("Copy List: ",copyList)

list2 = [11,22,33,44]
list2.extend(sampleList) # Appends sampleList to list2
print("Extend List: ",list2) # Output:  [11, 22, 33, 44, 10, 12, 11, 13, 9, 14, 15, 8, 6]

print("Occurance time of 11 in list2: ",list2.count(11)) # Output: 2

list2.remove(11)
print("list2 removes first occurance of 11: ",list2) # Output: [22, 33, 44, 10, 12, 11, 13, 9, 14, 15, 8, 6]

list2.pop()
print("Pop List: ",list2) # Output: [22, 33, 44, 10, 12, 11, 13, 9, 14, 15, 8]

list2.sort()
print("Sorted List: ",list2) # Output: [8, 9, 10, 11, 12, 13, 14, 15, 22, 33, 44]

list2.reverse()
print("Reversed List: ",list2) # Output: [44, 33, 22, 15, 14, 13, 12, 11, 10, 9, 8]

list2.clear()
```

```
print("Clear list: ",list2)
```

17..# Write a program to create nested list and display its elements.

```
nestedList = [10, 20, 30, 40, [1, 2, 3, 4, 5]]

for i in nestedList:
        print(i)
```

18..# Write a program to accept elements in the form of a tuple

 # and display its minimum, maximum, sum and average.

```
num = eval(input("Enter tupel in (): "))
print(num)

sum = 0
for i in num:
        sum += i

print("Sum: ",sum)
print("Minimum: ",min(num))
print("Maximum: ",max(num))
print("Average: ",sum/len(num))  #example value enter 2,3
```

19.. Create a program to sort tuple with nested tuples.

```
emp = ((10,"Ajay",4000),(20,"Vijay",10000),(30,"Sanjay",12000))
```

```
print(emp)
print("Sort by default: ",sorted(emp)) #Sorts on default id
print("Sort by default in reverse: ",sorted(emp,reverse=True))
print("Sort by Name: ",sorted(emp,key = lambda x:x[1])) # sort on name
print("Sort by Salary: ",sorted(emp,key = lambda x:x[2])) # sort on salary
```

20..# Write a program to create a dictionary from the user and

 # display the elements.

```
x = {} # Empty Dictionary

n = int(input("How many elements you want in dictionary.."))

for i in range(n):
        k = input("Enter key: ")
        v = int(input("Enter Values: "))
        x.update({k:v})

print(x)
```

21..# Create a dictionary that will accept cricket players name

and scores in a match. Also we are retrieving runs by entering the player's name.

```
x = {} # Empty Dictionary

n = int(input("How many players.."))

for i in range(n):
```

```
        k = input("Enter Player Name: ")
        v = int(input("Enter Runs: "))
        x.update({k:v})

print(x)

name = input("Enter player name: ")
runs = x.get(name,-1)

if(runs == -1):
        print("Player not found")
else:
        print("{} made runs {}".format(name,runs))
```

22.. Write a program to convert the elements of two lists into key-value pairs of a dictionary.

```
student = ["Akshay","Satish","Nishi"]
marks = [55,78,89]

# make dictionary

z = zip(student,marks)
d = dict(z)

print("{:15s} -- {:15s}".format("STUDENT","MARKS"))
for i in d:
        print("{} -- {}".format(i,d[i]))
```

```
        # print(i," --- ",d[i])
print(d)
```

23..# Create a python function to accept python function as a

 # dictionary and display its elements.

```
def fun(dictionary):
        for i,j in dictionary.items():
                print(i,"--",j)

d = {"a":"Apple",
         "b":"Book",
         "c":"Cat"}

fun(d)
```

UNIT...3

1..Write a program to create a Student class with name, age and marks as data members. Also create a method named display() to view the student details. Create an object to Student class and call the method using the object.

```
class Student:
   def __init__(self,name1='abc',age1=18,marks1=35):
      self.name=name1
      self.age=age1
      self.marks=marks1
   def display(self):
      print("student name is",self.name)
      print("student age is",self.age)
      print("Student marks is",self.marks)
```

```
s1=Student()

s1.display()

s2=Student('def',20,80)

s2.display()
```

2.. Write a program to create Student class with a constructor having more than one parameter.

```
class Student:
     def __init__(self,id1='1',name1='abc'):
       self.id=id1
       self.name=name1
     def printdata(self):
       print("Student id is",self.id)
       print("Student name is",self.name)

s1=Student()
s1.printdata()
s2=Student(2,"def");
s2.printdata()
```

3..Write a program to demonstrate the use of instance and class/static variables.

```
class test:
   x=5 #class variable
   def __init__(self):
     self.y=5 #instance variable

   @classmethod
   def printdata(cls):
      cls.x=cls.x+1
      print("Class variable-X value is",cls.x)

   def printdata1(self):
       self.y=self.y+1
       print("Instance variable-y value is",self.y)
```

```
print("First object")
t1=test()
t1.printdata()
t1.printdata1()

print("Second object")
t2=test()
t2.printdata()
t2.printdata1()
```

4..Write a program to store data into instances using mutator methods and to retrieve data from the instances using accessor methods.'

There are two types of instance methods
1. accessor methods /getter methods
2. mutator methods / setter methods

```
class book:
    def setid(self,id):
        self.id=id
    def setname(self,name):
        self.name=name
    def getid(self):
        return self.id
    def getname(self):
        return self.name

b1=book()
x=int(input("Enter id"))
y=input("Enter name")

b1.setid(x)
b1.setname(y)

print("The book id is ",b1.getid(),"and book name is ", b1.getname())
```

5..'"**Write a program to use class method to handle the common features of all the instance of Student class.'"**

```
class Student:
   subject='python'
   @classmethod
   def setsubjectname(cls,sem):
      print("your semester is ",sem," Subject is ",cls.subject)


s1=Student()
x=int(input("Enter semester"))

Student.setsubjectname(x)
```

6..'''**Write a program to create a static method that counts the number of instances created for a class.'''**
'''static method-does not requires class or instance, it is used to set environment variable'''

```
class countobj:
   stvar=0
   def __init__(self):
      countobj.stvar=countobj.stvar+1

   @staticmethod
   def findtotobj():
      print("The total number of objects are:  ", countobj.stvar)


obj1=countobj()
obj2=countobj()
obj3=countobj()
countobj.findtotobj()
```

'7..**''Create a Bank class with two variables name and balance. Implement a constructor to initialize the variables. Also implement deposit and withdrawals using instance methods.'''**

```
class Bank:
    def __init__(self,name='abc',balance=5000):
        self.name=name
        self.balance=balance

    def deposit(self, amt):
        self.balance=self.balance+amt
```

```
        self.display()


    def withdrawals(self,amt):
        self.balance=self.balance-amt
        self.display()

    def display(self):
        print(self.name," ",self.balance)

b1= Bank('def',10000)
amt=int(input("Enter amount to deposite"))
b1.deposit(amt)

amt=int(input("Enter amount to withdraw"))
b1.withdrawals(amt)

b2= Bank()
amt=int(input("Enter amount to deposite"))
b2.deposit(amt)

amt=int(input("Enter amount to withdraw"))
b2.withdrawals(amt)
```

8..'"Write a program to create a Emp class and make all the members of the Emp class available to another class (Myclass). [By passing members of one class to another]"'

```
class Emp:
    def __init__(self):
        self.id=1
        self.name='abc'
    def printdetails(self):
        print("Id is",self.id,"Name is",self.name)

class Myclass:
    def printobj(self,e1):
        e1.id=2
        e1.name='def'
        e1.printdetails()

e1=Emp()
e1.printdetails()

e2=Myclass()
```

```
e2.printobj(e1)

e1.printdetails()
```

'9.."Create a Student class to with the methods set_id, get_id, set_name, get_name, set_marks and get_marks where the method name starting with set are used to assign the values and method name starting with get are returning the values. Save the program by student.py. Create another program to use the Student class which is already available in student.py."'

```
from Student import Student

s1=Student()
s1.setid(1)
s1.setname('abc')
s1.setmarks(50)

print("Students Details is: ")
print(s1.getid()," ",s1.getname()," ",s1.getmarks())
```

```
class Student:
    def setid(self,id):
        self.id=id

    def getid(self):
        return self.id

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

    def getname(self):
        return self.name

    def setmarks(self,marks):
        self.marks=marks

    def getmarks(self):
        return self.marks
```

10.. Write a program to access the base class constructor from a sub class by using *super()* method and also without using *super()* method.

```
class base:
        def __init__(self,v):
                self.value=v
        def display(self):
                print("valie is ",self.value)
class sub5(base):

        def __init__(self,v,v1):
                self.value1=v1
                super(). __init__(v)
        def display1(self):
                print("value is ",self.value1)

s1=sub(5,2)
s1.display()
s1.display1()
```

11.Write a program to override super class constructor and method in sub class.

```
class Teacher:

        def __init__(self):

                self.name = "Hari"

        def display(self):

                print("Name: ",self.name)


class Student(Teacher):

        def __init__(self):
```

```
            self.name = "Niti"
        def display(self):
            print("Name: ",self.name)

s = Student()
s.display()
```

(12)..'''12.Write a program to implement single inheritance in which two sub classes are derived from a single base class.'''

```
class car(c1):
   wheels=4
   @classmethod
   def bmw(cls):
      print("",cls.wheels)
class honda(car):
   wheels=4
   @classmethod
   def hondacity(cls):
      print("",cls.wheels)

class ford(car):
   wheels=4
   @classmethod
   def xuv(cls):
```

```
        print(cls.wheels+car.wheels)
h=honda()
h.xuv()
f=ford()
f.xuv()
```

(13)..'''13.Write a program to implement multiple inheritance using two base classes.'''

```
class a:
    def __init__(self):
        self.name="Amit"
        self.age=20
    def getname(self):
        return self.name
class b:
    def __init__(self):
        self.name="Ansh"
        self.age=22
    def getname(self):
        return self.name
class c(a,b):
    def __init__(self):
        a.__init__(self)
        b.__init__(self)
    def getname(self):
```

```
        return self.name
c1=c()
print("Name is ",c1.getname())
```

(14)..'''14.. Write a program to understand the order of execution of methods in several base classes according to method resolution order (MRO).'''

```
class a:
    def __init__(self):
        super().__init__()
        self.name="Amit"
        self.age=20
    def getname(self):
        return self.name
class b:
    def __init__(self):
        super().__init__()
        self.name="Ansh"
        self.age=22
    def getname(self):
        return self.name
class c(a,b):
    def __init__(self):
        super().__init__()
    def getname(self):
```

```
        return self.name
c1=c()
print("Name is ",c1.getname())
```

(16)..'''16.Write a program to overload the addition operator (+) to make it act on the class objects.'''

```
class book:
    def __init__(self,page):
        self.page=page
    def __add__(self,other):
        return(self.page+other.page)
class book1:
    def __init__(self,page):
        self.page=page
b1=book(100)
b2=book1(300)
print("total page :",(b1+b2))
```

(17)..'''17.Write a program to show method overloading to find sum of two or three numbers.'''

```
'''class mathod:
    def add(self,a,b):
        print(a+b)
    def add(self,a,b,c):
        print(a+b+c)
obj = mathod()
```

```
obj.add(7,8,10)'''


########
class math:
   def add(self,p, *args):
    sum = 0
    for a in args:
      sum = sum + a
    print(sum+p)
m=math()
m.add(1,2,3)
m.add(5,6)
```

(18)..''' 18.Write a program to override the super class method in

```
subclass.'''
class comp:
   def __init__(self,computer,ram):
      self.computer=computer
      self.ram=ram
class laptop(comp):
   def __init__(self,computer,ram):
      super().__init__(computer,ram)
```

```
l=laptop('lenovo',512)

print("This computer is",l.computer)

print("This computer ram is ",l.ram)
```

UNIT…4

1..Write a program to handle some built in exceptions like *ZeroDivisionError*.

```
a,b=[int(x)for x in input ("Enter Two Value").split()]

try:

   z=a/b

except ZeroDivisionError:

   print("Not Allowed")
```

2.. Write a program to handle multiple exceptions like *SyntaxError* and *TypeError*

```
try:
    def fun():
        a="Hello"+4
        print(a)
    fun()

except TypeError:
    print("Not Allowed")
except SyntaxError:
    print("Invalid")
```

3..Write a program to import "*os*" module and to print the current working directory and returns a list of all module functions

```
try:
    def fun():
        a="Hello"+4
        print(a)
    fun()

except TypeError:
    print("Not Allowed")
except SyntaxError:
    print("Invalid")
```

4.. Write a program to provide a function for making file lists from directory wildcard searches.

```
import os

os.getcwd()
```

5. Write a program to import *datetime* module and format the date as required. Also use the same module to calculate the difference between your birthday and today in days.

```
from datetime import date
now=date.today()
print(now)#current date
date1 = date.today()
x = int(input("Enter year"))
y = int(input("Enter month"))
z = int(input("Enter date"))
date2 = date(x,y,z)
print(date1-date2)
```

6. Write a program to create a database named “Sample_DB” in MySQL(). [First ensure connection is made or not and then check if the database *Sample_DB* already exists or not, if yes then print appropriate message]

```
import mysql.connector
try:
con =
mysql.connector.connect(host="localhost",user="root",passwd="",database="my
db")
print("Database Connected")
cursor = con.cursor()
cursor.execute("CREATE DATABASE Sample_DB")
print("database created")
except Exception as e:
print("Database already created")
```

7. Write a program to retrieve and display all the rows in the employee table. [First create an *employee* table in the Sample_DB with the fields as eid, name, sal . Also enter some valid records]

```
import mysql.connector
try:
con =
mysql.connector.connect(host="localhost",user="root",password="",database="S
ample_DB")
cursor = con.cursor()
cursor.execute("create table employee(eid int primary key auto_increment,name
text,sal int)")
```

```
print("Table created")
except Exception as e:
pass
finally:
def insert_data(name,salary):
try:
cursor.execute("insert into employee(name,sal)
values('{}',{})".format(name,salary))
except Exception as e:
print(e)
def show_data():
cursor.execute("select * from employee")
row = cursor.fetchall()
for i in row:
print(i)
no = int(input("enyet no of employees details u want to add"))
for x in range(no):
name = input("Enter employeee name")
salary = int(input("Enter employee salary"))
insert_data(name,salary)
show_data()
```

8. Write a program to insert several rows into *employee* table from the keyboard.

```
from mysql.connector import *
con =
connect(host="localhost",user="root",password="",database="sample_db")
def insert_emp(name,salary):
try:
cursor = con.cursor()
cursor.execute("insert into employee(name,sal)
values('{}',{})".format(name,salary))
con.commit()
print("data submitted")
except Exception as e:
print(e)
def show_data():
cursor = con.cursor()
cursor.execute("select * from employee")
row = cursor.fetchall()
for i in row:
print(i)
no = int(input("enyet no of employees details u want to add"))
for x in range(no):
```

```
name = input("Enter employeee name")
salary = int(input("Enter employee salary"))
insert_emp(name,salary)
show_data()
```

9. Write a program to delete a row from an *employee* table by accepting the employee identity number (eid) from the user.

```
from mysql.connector import *
def insert_emp(eid):
try:
con =
connect(host="localhost",user="root",password="",database="sample_db")
cursor = con.cursor()
cursor.execute("delete from employee where eid={}".format(eid))
con.commit()
print("data deleted")
except Exception as e:
print(e)
no = int(input("Enter id of employee u want to delete"))
insert_emp(no)
```

10. Write a program to increase the salary (sal) of an employee in the *employee* table by accepting the employee identity number (eid) from the user.

```
from mysql.connector import *
def insert_emp(eid,salary):
try:
con = connect(host="localhost",user="root",password="",database="sample_db")
cursor = con.cursor()
cursor.execute("update employee set sal={1} where eid={0}".format(eid,salary))
con.commit()
print("data updated")
except Exception as e:
print(e)
no = int(input("Enter id of employee u want to change salary"))
salary = int(input("Enter updated salary"))
insert_emp(no,salary)
```

11. Write a program to create a table named *new_employee_tbl* with the fields eno , ename , gender and salary in Sample_DB database. The datatypes of the fields are eno-int, ename-char(30), gender-char(1) and salary-float.

```
import mysql.connector as c
try:
con =
c.connect(host="localhost",user="root",password="",database="sample_db")
cursor = con.cursor()
cursor.execute("create table new_employee_tbl(eno int(3) primary key
auto_increment,ename varchar(30),gender char(1) ,salary float)")
print("table created")
except Exception as e:
print(e)
```