Program: 01

July 15, 2022

Problem Statement:

Write a program to create a class Account which contains the attributes Name, Age, and Address. Also define the display function to print all the attributes

Solution:

```
class Account:
    def  __init__(self, name, age,  address):
        self.Name=name
        self.Age=age
        self.Address=address

    def display(self):
        print("Your Nsme Is :",self.Name)
        print("Your Nsme Is :",self.Age)
        print("Your Nsme Is :",self.Address)

object = Account("Dikshant Bhatiya","19","Lakhisarai")
object.display()
```

Program: 02

July 19, 2022

Problem Statement:

Write a program to create a class with name Student whose data members are name, enrollment number, age, branch and semester, and member function are putData() and gatData(). Create two object, take data for objects and print details.

Solution:

```
class Student:
    def GetData(self):
        self.Name = input("Enter The Name  :")
        self.Enroll = input("Enter The Enroll  :")
        self.Age = input("Enter The Age  :")
        self.Branch = input("Enter The Branch  :")
        self.Sem = input("Enter The Sem  :")

    def PutData(self):
        print("Your Name Is  :",  self.Name)
        print("Your Enroll Is  :",  self.Enroll)
        print("Your Age Is  :",  self.Age)
        print("Your Branch Is  :",  self.Branch)
        print("Your Sem Is  :",  self.Sem)

Obj = Student()
Obj.GetData()
Obj.PutData()
Obj2 = Student()
Obj2.GetData()
Obj2.PutData()
```

Program: 03

July 22, 2022

Problem Statement:

Define a class employee with instance variable: employee id , name, salary. Define constructor to initialze member variables. Define a fuction to show employee data

Solution:

```
class Employee:
    def __init__(self, Id, Name, Sal):
        self.EmpId = Id
        self.EmpName = Name
        self.EmpSal = Sal

    def Show(self):
        print("Employee Id  :", self.EmpId)
        print("Employee Name  :", self.EmpName)
        print("Employee Salary  :", self.EmpSal)

iD = int(input("Enter Employee Id :"))
name = input("Enter Employee Name :")
sal = float(input("Enter Employee Salary :"))

EMP = Employee(iD, name, sal)
EMP.Show()
```

Program: 04

July 22, 2022

Problem Statement:

Using class ‘employee’ of above question create a list of employee and display a list of employee in sorted order according to their names. Also define a fuction to sort list of employee according to their salary in decreasing order.

Solution:

```
L1 = []
L2 = []
class Employee:
    def __init__(self, Id, Name, Sal):
        self.EmpId = Id
        self.EmpName = Name
        self.Empsalary = Sal
        L1.append(self.Empsalary)
        L2.append(self.EmpName)
    def EmpList(self):
        print("\n Employee List Sorted Order :\n")
        print(sorted(L2))
    def EmpListSal(self):
        print("\nEmployee List Salary List Sorted Decending Order :\n")
        print(sorted(L2,reverse = True))
N = int(input("Enter No Of Employees  :"))
print("Enter Employee Id , Name And Salary :")
for i in range(N):
    Id = int(input())
    Name = input()
    Sal = float(input())
    EMP = Employee(Id, Name, Sal)
EMP.EmpList()
```

Program: 05

Aug 12, 2022

Problem Statement:

Define a class BOOK with the following specifications :

Members of the class BOOK areBook_no, Book_title, Price. total_cost():A function to calculate the total cost for N number of copies where N is passed to the function as argument. Input(): function to read Book_no, Book_title, Price Purchase() function to ask the user to input the number of copies to be purchased. It invokes total_cost() and prints the total cost to be paid by the user.

Solution:

Solution:

```
class Book:
    def __init__(self):
        self.Input()
        self.Purchase()
    def Input(self):
        self.Book_no=input("Enter Book Number: ")
        self.Book_title=input("Enter Book Title: ")
        self.Price=input("Enter Price: ")
    def Purchase(self):
        self.N=int(input("Number of copies to be purchased: "))
        self.total_cost()
    def total_cost(self):
        print("Total cost to be paid:",float(self.Price)*int(self.N))

b1=Book()
```

Program: 06

Aug 18, 2022

Problem Statement:

Write a Python class to find validity of a string of parentheses, '(', ')', '{', '}', '[' and ']. These brackets must be close in the correct order.

Solution:

Solution:

```
class validity:
    def f(str):
        a = ['()', '{}', '[]']
        while any(i in str for i in a):
            for j in a:
                str = str.replace(j, '')
        return not str
s = input("enter : ")
print(s,"is valid" if validity.f(s) else "is Not valid")
```

Program: 07

Aug 18, 2022

Problem Statement:

Imagine a tollbooth at a bridge. Cars passing by the booth are expected to pay Rs. 50/- toll. Mostly they do, but sometimes a car goes by without paying. The tollbooth keeps track of the number of cars that have gone by, and of the total amount of money collected. Model this tollbooth with a class called TollBooth. The two data items are a type int to hold the total number of cars, and a type double to hold the total amount of money collected. A constructor initializes both of these to 0. A member function called payingCar() increments the car total and adds 0.50 to the cash total. Another function, called nopayCar(), increments the car total but adds nothing to the cash total. Finally, a member function called display() displays the two totals. Include a program to test this class. This program should allow the user to push one key to count a paying car, and another to count a nonpaying car. Pushing the $ key should cause the program to print out the total cars and total cash and then exit.

Solution:

```
class ToolBooth:
    def __init__(self):
        ToolBooth.cars=0
        ToolBooth.amnt=0
    def payingCar(self):
        self.cars=self.cars+1
        self.amnt=self.amnt+50
    def nopayCar(self):
        self.cars=self.cars+1
    def display(self):
        print('\nNumber of cars passed: ',self.cars)
        print('total cash collected: ',self.amnt)
toll=ToolBooth()
a=0
while(a!='end') :
    a=input("Enter 'p' for paying car, 'n' for non-paying car,'$' to display and 'end' to stop:")
    if a=='p':
        toll.payingCar()
    elif a=='n':
        toll.nopayCar()
    elif a=='$':
        toll.display()
    else:
        pass
```

Program: 08

Aug 25, 2022

Problem Statement:

Create a class called employee that contains a name and an employee number. Include a member function called getdata() to get data from the user for insertion into the object, and another function called putdata() to display the data. Assume the name has no embedded blanks. Write a main() program to exercise this class. It should create an array of type employee, and then invite the user to input data for up to 100 employees.
Finally, it should print out the data for all the employees.

Solution:

```
l1 = []
class Employee:
    def getData(self):
        self.Ename = input("Enter Employee Name : ")
        self.Eno   = int(input("Enter Employee Number : "))
        t = self.Ename ,self.Eno
        l1.append(t)
    def putData(self):
        for i in l1:
            for j in i:
                print(j,end=' ')
            print()
n = int(input("Enter number of Emloyees : "))
e = Employee()
for i in range(n):
    e.getData()
    e.putData()
```

Program: 09

Aug 25, 2022

Problem Statement:

Write a program to maintain the record of movies, one record of movie contains movie name, actor or actress, movie rating, production house, more than one record can be inserted by a operator and all records should be displayed to user only.

Solution:

```
class Movies:
    def __init__(self):
        self.name = ""
        self.actor = ""
        self.rating = ""
        self.prohouse= ""

    def insert(self, name, actor, rating, prohouse):
        self.name = name
        self.actor = actor
        self.rating = rating
        self.prohouse = prohouse

    def display(self):
        print("name : ", self.name)
        print("actor and actress : ",self.actor)
        print("rating : ",self.rating)
        print("production house : ", self.prohouse)

a = Movies()
a.insert("Avengers : Endgame ","Robert Downey Jr","10", "Marval Studios")
a.display()
```

Program: 10

Oct 07, 2022

Problem Statement:

Create a class that includes a data member that holds a serial number for each object created from a class. That is the first object will be numbered 1, the second 2 and so on. When each object is creating, its constructor can examine this count member variable to determine the appropriate serial number for the new object. Add a member function that permits an object to report its own serial number. Then the main () function that creates three objects and queries each one about its serial number. they should respond i am object 2, and so on.

Solution:

```
sn = 1
class SerialNumber():
    def __init__(self,name):
        global sn
        self.name = name
        self.sn = sn
        sn += 1
    def display(self):
        print("name : ",self.name)
        print("Serial number : ",self.sn)

a = SerialNumber("Deepak")
b = SerialNumber("Ravi")
c = SerialNumber("Rampal")

a.display()
b.display()
```

Program: 11

Oct 07, 2022

Problem Statement:

Imagine a publishing company that markets both book and compact disk version of its work. Create a class publication that stores the title and price of a publication. From this class derive two classes book which adds a page count and CD which adds a playing timein minutes. Each of these three classes should have a getData() function to get its data from the userat the keyboard and a putData() function to display the data.Write a main program to test the book and CD classes by creating instances of them.Asking the user to fill in thedata with getData() and displaying the data with putData().

Solution:

```
class Publication:
    def getData(self):
        self.title=input("Enter Title: ")
        self.price=input("Enter Price: ")
    def putData(self):
        print("\nTitle:",self.title)
        print("Price:",self.price)
class Book(Publication):
    def getData(self):
        super().getData()
        self.pg=input("Enter page count: ")
    def putData(self):
        super().putData()
        print("Page Count:",self.pg)
class CD(Publication):
    def getData(self):
        super().getData()
        self.playT=input("Enter playing time: ")
    def putData(self):
        super().putData()
        print("Playing Time:",self.playT,"min")


def main():
    cd=CD()
    b=Book()
    cd.getData()
    cd.putData()
    b.getData()
    b.putData()
main()
```

Program: 12

Oct 11, 2022

Problem Statement:

Start with the publication, book, and CD classes of previous exercise. Add a base class sale that holds an array of three floats so that it can record the dollar sales of a particular publication for the last three months. Include a getdata() function to get three sales amounts from the user, and a putdata() function to display the salesfigures. Alter the book and CD classes so they are derived from both publication and sales. An object of class book or CD should input and output sales data along withits other data. Write a main() function to create a book object and a tape object and exercise their input/output capabilities.

Solution:

```
class Sale:
    l1=list()
class Publication(Sale):
    def getData(self):
        self.title=input("Enter Title: ")
        self.price=input("Enter Price: ")
    def putData(self):
        print("\nTitle:",self.title)
        print("Price:",self.price)
class Book(Publication):
    def getData(self):
        super().getData()
        self.pg=input("Enter page count: ")
    def putData(self):
        super().putData()
        print("Page Count:",self.pg)
class CD(Publication):
    def getData(self):
        super().getData()
        self.playT=input("Enter playing time: ")
    def putData(self):
        super().putData()
        print("Playing Time:",self.playT,"min")
def main():
    obj1=Book()
    obj1.get_book_data()
    obj1.put_book_data()
    obj2=CD()
    obj2.get_cd_data()
    obj2.put_cd_data()
    print("The sale",Sale.l1)
main()
```

Program: 13

Oct 11, 2022

Problem Statement:

Write A Program to create a table in oracle

Solution:

```
import cx_Oracle
try:
   conn=cx_Oracle.connect("scott/sistec")
   cursor=conn.cursor()
   print(conn.version)
   sql="create table student(eno number(10),ename varchar2(30))"
   cursor.execute(sql)
   print("Table created successfully")
except conn.DatabaseError as e:
   print("Not connected due to msg=",e)
finally:
   cursor.close()
   conn.close()
```

```
11.2.0.2.0
Table created successfully
```

```
import cx_Oracle
try:
   con=cx_Oracle.connect("scott/sistec")
   cursor=con.cursor()
   sql="insert into students values(:eno,:ename)"
   records=[(1,"Deepak"),(2,"Amit")]
   cursor.executemany(sql,records)
   con.commit()
   print("record inserted successfully")
except cx_Oracle.DatabaseError as e:
   if con:
      con.rollback()
      print("There is a problem with sql",e)
finally:
   if cursor:
      cursor.close()
   if con:
      con.close()
```

```
record inserted successfully
```

Program: 14

Nov 01, 2022

Problem Statement:

Write a program for update query in mysqlite.

Solution:

```
import sqlite3
try:
   conn=sqlite3.connect("emp.db")
   cursor=conn.cursor() #print(conn.version)
   sql="CREATE TABLE IF NOT EXISTS emp(eno INTEGER, ename TEXT, esal INTEGER,
eaddr TEXT)"
   cursor.execute(sql)
   print("Table created successfully")
except conn.DatabaseError as e:
   print("Not connected due to msg=",e)
finally:
   if cursor:
      cursor.close()
   if conn:
      conn.close()
Table created successfully

Enter the number of records:4
Enter Employee Number: 1
Enter Employee Name:Ankit
Enter Employee Salary:11000
Enter Employee Address:Bhopal
record inserted successfully

Enter Employee Number: 2
Enter Employee Name:Amit
Enter Employee Salary:12000
Enter Employee Address:Matasi
record inserted successfully

Enter Employee Number: 3
Enter Employee Name:Sumit
Enter Employee Salary:15000
Enter Employee Address:Jamui
record inserted successfully

Enter Employee Number: 4
Enter Employee Name:Sunil
Enter Employee Salary:21000
Enter Employee Address:Gaya
record inserted successfully
```

```
import sqlite3
try:
    con=sqlite3.connect("emp.db")
    cursor=con.cursor()
    cursor.execute("select * from emp")
    data=cursor.fetchall()
    for row in data:
        print(row)
        print('#'*31)
except sqlite3.DatabaseError as e:
    if con:
        con.rollback()
        print("There is a problem with sql:",e)
finally:
    if cursor:
        cursor.close()
    if con:
        con.close()
(1, 'Ankit', 11000, 'Bhopal')
###############################
(2, 'Amit', 12000, 'Matasi')
###############################
(3, 'Sumit', 15000, 'Jamui')
###############################
(4, 'Sunil', 21000, 'Gaya')
###############################

# Update Query
import sqlite3
try:
    conn=sqlite3.connect("emp.db")
    cursor=conn.cursor()
    increment=float(input("Enter increment salary:"))
    salrange=float(input("enter salary range:"))
    sql="update emp set esal=esal+{} where esal>{}".format(increment,salrange)
    cursor.execute(sql)
    print("record updated successfully")
    conn.commit()
except sqlite3.DatabaseError as e:
    if conn:
        conn.rollback()
        print("There is a problem with sql",e)
finally:
    if cursor:
        cursor.close()
    if conn:
        conn.close()
Enter increment salary:22000
enter salary range:11000
record updated successfully
```

```
import sqlite3
try:
    con=sqlite3.connect("emp.db")
    cursor=con.cursor()
    cursor.execute("select * from emp")
    data=cursor.fetchall()
    for row in data:
        print(row)
        print('#'*31)
except sqlite3.DatabaseError as e:
    if con:
        con.rollback()
        print("There is a problem with sql:",e)
finally:
    if cursor:
        cursor.close()
    if con:
        con.close()
```

```
(1, 'Ankit', 11000, 'Bhopal')
###############################
(2, 'Amit', 34000, 'Matasi')
###############################
(3, 'Sumit', 37000, 'Jamui')
###############################
(4, 'Sunil', 43000, 'Gaya')
###############################
```

Program: 15

Nov 01, 2022

Problem Statement:

Write a Program to insert fake data in database.

Solution:

```
import sqlite3
conn=sqlite3.connect("books.db")
cur=conn.cursor()
cur.execute("CREATE TABLE IF NOT EXISTS emp(eno INTEGER, ename TEXT, esal
INTEGER, eaddr TEXT)")
conn.commit()
conn.close()
from faker import Faker
import random
fake=Faker()
import sqlite3
con=sqlite3.connect("books.db")
cursor=con.cursor()
def populate(n):
    for i in range(n):
        eno=fake.random_int(min=1,max=999)
        ename=fake.name()
        esal=round(random.uniform(5000,25000),2)
        eaddr=fake.city()
        sql="insert into emp values(%d,'%s', %.2f, '%s' )"%(eno,ename,esal,eaddr)
        cursor.execute(sql)
        con.commit()
n=int(input("Enter the number of records:"))
populate(n)
print("record inserted successfully")
cursor.close()
con.close()
```

```
Enter the number of records:11
record inserted successfully
```

```
import sqlite3
try:
    con=sqlite3.connect("books.db")
    cursor=con.cursor()
    cursor.execute("select * from emp")
    n=int(input("Enter the number of required rows:"))
    data=cursor.fetchmany(n)
    for row in data:
        print(row)
        print('#'*51)
except cx_Oracle.DatabaseError as e:
    if con:
        con.rollback()
        print("There is a problem with sql:",e)
finally:
    if cursor:
        cursor.close()
    if con:
        con.close()
```

```
Enter the number of required rows:5
(504, 'Aaron Mullins', 13800.96, 'Burnettshire')
###################################################
(168, 'Andrew Reed', 5875.94, 'South Sherry')
###################################################
(254, 'Daniel Mcmahon', 14753.8, 'North Jeannebury')
###################################################
(377, 'Mary Morrison', 13773.5, 'Lake Tyler')
###################################################
(380, 'Jennifer Fisher', 11479.44, 'Lindaton')
###################################################
```