(https://www.darshan.ac.in/)

Python Programming - 2101CS405

Lab - 12

OOP

01) Write a Program to create a class by name Students, and initialize attributes like name, age, and grade while creating an object.

In [3]:
```
class Students:
    def __init__(self,name,age,grade):
        self.name = name;
        self.age = age;
        self.grade = grade;
s1 = Students("Devika",20,"A+");
print(f"Name:{s1.name}\nAge:{s1.age}\nGrade:{s1.grade}");
```

```
Name:Devika
Age:20
Grade:A+
Name:Devika
Age:20
Grade:A+
```

02) Create a class named Bank_Account with Account_No, User_Name, Email,Account_Type and Account_Balance data members. Also create a method GetAccountDetails() and DisplayAccountDetails(). Create main method to demonstrate the Bank_Account class.

In [11]:
```
acno = input("Enter Account No:")
uname = input("Enter Name:")
actype = input("Enter Account Type:")
acbalance = input("Enter Balance:")
class Bank_Account:
    def getAccountDetails(self,acno,uname,actype,acbalance):
        self.acno = acno;
        self.uname = uname;
        self.actype = actype;
        self.acbalance = acbalance;

    def displayAccountDetails(self):
        print(f"Account NO:{self.acno}\nUser Name:{self.uname}\nAccount Type:{

b = Bank_Account();
b.getAccountDetails(acno,uname,actype,acbalance);
b.displayAccountDetails();
```

```
Enter Account No:123456789
Enter Name:Devika
Enter Account Type:Saving
Enter Balance:1500000
Account NO:123456789
User Name:Devika
Account Type:Saving
Account Balance:1500000
```

03) WAP to create Circle class with area and perimeter function to find area and perimeter of circle.

In [17]:
```
r = float(input("Enter Radius:"))
import math
class Circle:
    def __init__(self,r):
        self.r = r;
    def perimeter(self):
        return 2*math.pi*self.r;
    def area(self):
        return math.pi*self.r*self.r;
c = Circle(r);
print(f"Perimeter:{c.perimeter()}\nArea:{c.area()}")
```

```
Enter Radius:5
Perimeter:31.41592653589793
Area:78.53981633974483
```

04) Create a class for employees that includes attributes such as name, age, salary, and methods to update and display employee information.

In [26]:
```
name = input("Enter Name:")
age = int(input("Enter Age:"))
salary = input("Enter Salary:")
class Employee:
    name=""
    age=0
    salary=""
    def __init__(self,name,age,salary):
        self.name = name;
        self.age = age;
        self.salary = salary;
    def updateEmployee(self,name,age,salary):
        self.name = name;
        self.age = age;
        self.salary = salary;
    def displayEmployee(self):
        print(f"Name:{self.name}\nAge:{self.age}\nSalary:{self.salary}")

e = Employee(name,age,salary);
e.displayEmployee()
name1 = input("Enter Name:")
age1 = int(input("Enter Age:"))
salary1 = input("Enter Salary:")
e.updateEmployee(name1,age1,salary1);
e.displayEmployee()
```

```
Enter Name:de
Enter Age:52
Enter Salary:5632
Name:de
Age:52
Salary:5632
Enter Name:devika
Enter Age:54
Enter Salary:896574
Name:devika
Age:54
Salary:896574
```

Out[26]:
```
'devika'
```

05) Create a bank account class with methods to deposit, withdraw, and check balance.

In [46]:

```
class Bank:
        amount =0
        flag = 0

        def deposite(self,amount):
            self.amount = self.amount+amount;
            Bank.flag = 1
        def withdraw(self,amount):
            if self.amount - amount >=0:
                self.amount = self.amount - amount;
                Bank.flag= 1;
            else :
                Bank.flag= 0;
        def checlBalance(self):
                    print(f"Amount:{self.amount}")

b = Bank();
while True:
    print("1.deposite\n2.withdraw\n3.check balance\n4.Exit")
    choice = int(input("Enter Chioice:"))
    if choice == 1:
        amt = int(input("Enter Amount:"));
        b.deposite(amt);
    elif choice == 2:
        if b.flag == 0:
            print("------------------------\nERROR:You Can't Withdraw\n------
        else:
            amt = int(input("Enter Amount:"));
            if b.amount - amt <0:
                print("------------------------\nERROR:You Can't Withdraw\n--
            else:
                b.withdraw(amt);
    elif choice == 3:
        print("------------------------------");
        b.checlBalance()
        print("------------------------------");
    else:
        break;
```

```
1.deposite
2.withdraw
3.check balance
4.Exit
Enter Chioice:3
------------------------------
Amount:0
------------------------------
1.deposite
2.withdraw
3.check balance
4.Exit
Enter Chioice:2
-------------------------
ERROR:You Can't Withdraw
-------------------------
1.deposite
2.withdraw
3.check balance
4.Exit
Enter Chioice:1
Enter Amount:100
1.deposite
2.withdraw
3.check balance
4.Exit
Enter Chioice:3
------------------------------
Amount:100
------------------------------
1.deposite
2.withdraw
3.check balance
4.Exit
Enter Chioice:2
Enter Amount:101
-------------------------
ERROR:You Can't Withdraw
-------------------------
1.deposite
2.withdraw
3.check balance
4.Exit
Enter Chioice:3
------------------------------
Amount:100
------------------------------
1.deposite
2.withdraw
3.check balance
4.Exit
Enter Chioice:4
```

06) Create a class for managing inventory that includes attributes such as item name, price, quantity, and methods to add, remove, and update items.

In [62]:
```
l1 = []
class Items:
    ID = 0
    def post(self,item,price,qunatity):
        self.ID = self.ID + 1
        self.item = item;
        self.price = price;
        self.qunatity = qunatity;
        new = {
            'item':self.item,
            'price':self.price,
            'qunatity':self.qunatity,
            'id':self.ID
        }
        l1.append(new)

    def put(self,item,price,qunatity):
        self.item = item;
        self.price = price;
        self.qunatity = qunatity;
        new = {
            'item':self.item,
            'price':self.price,
            'qunatity':self.qunatity
        }

i = Items()
i.post("mobile","25000","7")
i.post("mobile","25000","7")
print(l1)
```

```
[{'item': 'mobile', 'price': '25000', 'qunatity': '7', 'id': 1}, {'item': 'm
obile', 'price': '25000', 'qunatity': '7', 'id': 2}]
```

09) Create a Class with instance attributes

In [ ]:
```
```

07)

Create one class student_kit
Within the student_kit class create one class attribute principal name (Mr ABC)
Create one attendance method and take input as number of days.

While creating student take input their name .
Create one certificate for each student by taking input of number of days present in class.

In [ ]:

08) Define Time class with hour and minute as data member. Also define addition method to add two time objects.

In [ ]:

09) WAP to demonstrate inheritance in python.

In [ ]:

10) Create a child class Bus that will inherit all of the variables and methods of the Vehicle class

class Vehicle:

```
def __init__(self, name, max_speed, mileage):
    self.name = name
    self.max_speed = max_speed
    self.mileage = mileage
```

Create a Bus object that will inherit all of the variables and methods of the parent Vehicle class and display it.

In [ ]:

11) Create a class hierarchy for different types of animals, with a parent Animal class and child classes for specific animals like Cat, Dog, and Bird.

In [ ]:

12) Create a class hierarchy for different types of vehicles, with a parent Vehicle class and child classes for specific vehicles like Car, Truck, and Motorcycle.

In [ ]:

13) Create a class hierarchy for different types of bank accounts, with a parent Account class and child classes for specific account types like Checking, Savings, and Credit.

In [ ]:

14) Create a Shape class with a draw method that is not implemented. Create three child classes Rectangle, Circle, and Triangle that implement the draw method with their respective drawing behaviors. Create a list of Shape objects that includes one instance of each child class, and then iterate through the list and call the draw method on each object.

In [ ]:

15) Create a Person class with a constructor that takes two arguments name and age. Create a child class Employee that inherits from Person and adds a new attribute salary. Override the init method in Employee to call the parent class's init method using the super keyword, and then initialize the salary attribute.

In [ ]: