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

Python Programming - 2101CS405

Lab - 12

Name : Sachin patadiya

Enrollment no : 22010101142

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 [1]:
```
class Students:
    def __init__(self,name,age,grade):
        self.name=name
        self.age=age
        self.grade=grade
    def display(self):
        print("name ",self.name)
        print("age ",self.age)
        print("grade ",self.grade)


name=input("Enter name : ")
age=input("Enter age : ")
grade=input("Enter grade : ")
ob1=Students(name,age,grade)
ob1.display()
```

```
Enter name : sachin
Enter age : 18
Enter grade : a
name  sachin
age  18
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 [5]:
```
class Bank_Account:
    def getAccountDetails(self,account_no,user_name,email,account_type,acc
        self.account_no=account_no
        self.user_name=user_name
        self.email=email
        self.account_type=account_type
        self.account_balance=account_balance
    def displayAccountDetails(self):
        print("account no : ",self.account_no)
        print("user name : ",self.user_name)
        print("email : ",self.email)
        print("account type : ",self.account_type)
        print("account balance : ",self.account_balance)

ob1=Bank_Account()
ob1.getAccountDetails("123456789","SACHIN PATADIYA","sachinpatadiya@mail.c
ob1.displayAccountDetails()
```

```
account no :  123456789
user name :  SACHIN PATADIYA
email :  sachinpatadiya@mail.com
account type :  SAVING
account balance :  45360
```

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

In [11]:
```
import math
class Circle:
    def __init__(self,redius):
        self.redius=redius
    def area(self):
        print("area : ",math.pi*self.redius*self.redius)
    def perimeter(self):
        print("perimeter : ",2*math.pi*self.redius)

ob1=Circle(10)
ob1.area()
ob1.perimeter()
```

```
area :  314.1592653589793
perimeter :  62.83185307179586
```

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

In [6]:
```
class Employees:
    def __init__(self,name,age,salary):
        self.name=name
        self.age=age
        self.salary=salary
    def update(self):
        print("name : ",self.name)
        print("age : ",self.age)
        print("salary : ",self.salary)

ob1=Employees("sachin",18,50000)
ob1.update()
```

```
name :  sachin
age :  18
salary :  50000
```

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

In [7]:
```
class Bank_Account:
    def getAccountDetails(self,account_no,user_name,email,account_type,acc
        self.account_no=account_no
        self.user_name=user_name
        self.email=email
        self.account_type=account_type
        self.account_balance=account_balance

    def displayAccountDetails(self):
        print("account no : ",self.account_no)
        print("user name : ",self.user_name)
        print("email : ",self.email)
        print("account type : ",self.account_type)
        print("account balance : ",self.account_balance)

    def deposit(self,ammount):
        self.account_balance+=ammount

    def withdraw(self,ammount):
        if ammount < self.account_balance:
            self.account_balance-=ammount
        else:
            print("NOt balance present")

    def checkBalance(self):
        print("balance is : ",self.account_balance)

ob1=Bank_Account()
ob1.getAccountDetails("123456789","SACHIN PATADIYA","sachinpatadiya@mail.co
ob1.displayAccountDetails()
ob1.deposit(50000)
print("----------------------------")
ob1.checkBalance()
ob1.withdraw(10231)
print("----------------------------")
ob1.checkBalance()
```

```
account no :  123456789
user name :  SACHIN PATADIYA
email :  sachinpatadiya@mail.com
account type :  SAVING
account balance :  45360
----------------------------
balance is :  95360
----------------------------
balance is :  85129
```

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 [23]:
```
class Inventory:
    def __init__(self,name,price,quantity):
        self.name=name
        self.price=price
        self.quantity=quantity
    def add(self,quantity):
        self.quantity+=quantity
    def remove(self,quantity):
        if quantity < self.quantity:
            self.quantity-=quantity
        else:
            print("not quantity")
    def update(self,quantity):
        self.quantity=quantity
    def display(self):
        print("quantity : ",self.quantity,end="\n-----------------------------
ob1=Inventory("cake",500,50)
ob1.display()
ob1.add(25)
ob1.display()
ob1.update(50)
ob1.display()
ob1.remove(10)
ob1.display()
```

```
quantity :  50
-------------------------------
quantity :  75
-------------------------------
quantity :  50
-------------------------------
quantity :  40
-------------------------------
```

09) Create a Class with instance attributes

In [8]:
```
class Demo:
    def __init__(self,name):
        self.name=name
    def printName(self):
        print("Name : ",self.name)

ob1=Demo("sachin")
ob1.printName()
```

```
Name :  sachin
```

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 [27]:
```
class Student_Kit:
    principal_name="Mr ABC"
    def __init__(self,student_name):
        self.student_name=student_name
    def attendance(self,day):
        self.day=day
    def certificate(self):
        if self.day > 20:
            print("Wowwwwwwwwww....")
        else:
            print("Meet parants to ",self.principal_name)
ob1=Student_Kit("Jash")
ob2=Student_Kit("Jay")
ob1.attendance(26)
ob2.attendance(10)
ob1.certificate()
print("---------------------------------")
ob2.certificate()
```

```
Wowwwwwwwwww....
---------------------------------
Meet parants to  Mr ABC
```

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

In [31]:
```
class Time:
    def __init__(self, hour, mint):
        self.hour = hour
        self.mint = mint

    def add(self, ob):
        h = self.hour + ob.hour
        m = self.mint + ob.mint
        if m > 59:
            m -= 60
            h += 1
        if h > 12:
            h -= 12

        print(h, ":", m)

o1 = Time(10, 45)
o2 = Time(5, 15)
o1.add(o2)
```

```
4 : 0
```

09) WAP to demonstrate inheritance in python.

In [32]:
```
class P:
    def printP(self):
        print("Parants")
class C(P):
    def printC(self):
        print("Child")

o=C()
o.printP()
o.printC()
```

```
Parants
Child
```

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

In [35]:
```
class Vehicle:

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

class Bus(Vehicle):
    def __init__(self,name,max_speed,mileage):
        super().__init__(name,max_speed,mileage)
    def display(self):
        print(self.name," ",self.max_speed," ",self.mileage)
o=Bus("B",150,50)
o.display()
```

```
B   150   50
```

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 [39]:
```
class Animal:
    def __init__(self, leg):
        self.leg = leg

    def display(self):
        print("Legs:", self.leg)

class Cat(Animal):
    def __init__(self, leg):
        super().__init__(leg)

    def display(self):
        super().display()

class Crow(Animal):
    def __init__(self, leg):
        super().__init__(leg)

    def display(self):
        super().display()

o = Cat(4)
o.display()

o2 = Crow(2)
o2.display()
```

```
Legs: 4
Legs: 2
```

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 [42]:
```
class Vehical:
    def __init__(self, wheel):
        self.wheel = wheel

    def display(self):
        print("wheel :", self.wheel)

class Car(Vehical):
    def __init__(self, wheel):
        super().__init__(wheel)

    def display(self):
        super().display()

class Truck(Vehical):
    def __init__(self, wheel):
        super().__init__(wheel)

    def display(self):
        super().display()

class MotorCycle(Vehical):
    def __init__(self, wheel):
        super().__init__(wheel)

    def display(self):
        super().display()

o = Car(4)
o.display()

o2 = Truck(4)
o2.display()

o3 = MotorCycle(2)
o3.display()
```

```
wheel : 4
wheel : 4
wheel : 2
```

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 [43]:
```
class Account:
    def __init__(self, account_number, balance=0):
        self.account_number = account_number
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print("Deposited ",amount," New balance: ",self.balance)

    def withdraw(self, amount):
        if self.balance >= amount:
            self.balance -= amount
            print(self.balance)
        else:
            print("Insufficient funds")

    def display_balance(self):
        print("Account Number: ",self.account_number," Balance: ",self.bala


class Checking(Account):
    def __init__(self, account_number, balance=0, overdraft_limit=100):
        super().__init__(account_number, balance)
        self.overdraft_limit = overdraft_limit

    def withdraw(self, amount):
        if self.balance + self.overdraft_limit >= amount:
            self.balance -= amount
            print("balance : ",self.balance)
        else:
            print("Overdraft limit exceeded")


class Savings(Account):
    def __init__(self, account_number, balance=0, interest_rate=0.01):
        super().__init__(account_number, balance)
        self.interest_rate = interest_rate

    def add_interest(self):
        interest = self.balance * self.interest_rate
        self.balance += interest
        print("Balance : ",self.balance)


class Credit(Account):
    def __init__(self, account_number, balance=0, credit_limit=1000):
        super().__init__(account_number, balance)
        self.credit_limit = credit_limit

    def withdraw(self, amount):
        if self.balance + self.credit_limit >= amount:
            self.balance -= amount
            print("Balance : ",self.balance)
        else:
            print("Credit limit exceeded")`
```

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 [ ]:
```
class Shape:
    def __init__(self):
class Rectangle:
class Cirlce:
class Triangel:
```

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 [ ]:
```

```