Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 13

223 | Vishal Baraiya | 23010101014

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 [4]:
```
class Students:
    def __init__(self,name,age,grade):
        self.name = name
        self.age = age
        self.grade = grade

s1 = Students("Karan Aahir",55,'F')
print(f" name = {s1.name} \n Age = {s1.age} \n Grade = {s1.grade}")
```

```
name = Karan Aahir
Age = 55
Grade = F
```

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 [8]:
```
class Bank_Account:
    Account_No = 0
    User_Name = ""
    Email = ""
    Account_Type = ""
    Account_Balance = 0.0

    def GetAccountDetails(self):
        self.Account_No = int(input("Enter the Account Number : "))
        self.User_Name = input("Enter the User Name : ")
        self.Email = input("Enter the Email : ")
        self.Account_Type = input("Enter the Account Type : ")
        self.Account_Balance = float(input("Enter the Account Blance : "))

    def DisplayAccountDetails(self):
        print(f'Account Number : {self.Account_No}')
        print(f'User Name : {self.User_Name}')
        print(f'Email : {self.Email}')
        print(f'Account Type : {self.Account_Type}')
        print(f'Account Balance : {self.Account_Balance}')

b = Bank_Account()
b.GetAccountDetails()
b.DisplayAccountDetails()
```

```
Account Number : 123456789
User Name : Karan
Email : karan1234@gmail.com
Account Type : demate
Account Balance : 150.0
```

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

In [15]:
```
class Circle:

    def __init__(self,r):
        self.r = r

    def findPerimeter(self):
        return 2 * math.pi * self.r

    def findArea(self):
        return math.pi * self.r * self.r

c = Circle(14)
print(f"Area = {c.findArea()}")
print(f"Perimeter = {c.findPerimeter()}")
```

```
Area = 615.7521601035994
Perimeter = 87.96459430051421
```

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

In [16]:
```
class Employees:

    def __init__(self,name,age,salary):
        self.name = name
        self.age = age
        self.salary = salary
    def UpdateEmployeeDetails(self):
        self.name = input("Enter the Name : ")
        self.age = int(input("Enter the Age : "))
        self.salary = int(input("Enter the Salary : "))

    def DisplayEmplooyeeDetails(self):
        print(f"Name = {self.name}")
        print(f"age = {self.age}")
        print(f"Salry = {self.salary}")

e = Employees('Karan Aahir',18,120000)
e.UpdateEmployeeDetails()
e.DisplayEmplooyeeDetails()
```

```
Name = Rishil
age = 12
Salry = 7800
```

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

In [21]:

```
class Bank_Account:

    def __init__(self,accountno,name,balance):
        self.Account_No = accountno
        self.Name = name
        self.Balance = balance

    def deposit(self,Amount):
        self.Balance += Amount
        print(f"Hello! {self.Name} your {Amount} is Suuccessfully Deposited.")

    def withdraw(self,Amount):
        if (Amount <= self.Balance) and (Amount >= 0):
            self.Balance -= Amount
            print(f"Hello! {self.Name}, your {Amount} is successfully withdrawn.
        else:
            print(f"Insufficient Balance!")
            print(f"Your Withdraw Ammount = {Amount} and your Current Balance =

    def display(self):
        print(f"Account Number = {self.Account_No}")
        print(f"Name = {self.Name}")
        print(f"Current Balance = {self.Balance}")

b = Bank_Account(102321,'Karan Aahir',200)
b.display()
b.deposit(50)
b.display()
b.withdraw(100)
b.display()
```

```
Account Number = 102321
Name = Karan Aahir
Current Balance = 200
Hello! Karan Aahir your 50 is Suuccessfully Deposited.
Account Number = 102321
Name = Karan Aahir
Current Balance = 250
Insufficient Balance!
Your Withdraw Ammount = 1000 and your Current Balance = 250.
Account Number = 102321
Name = Karan Aahir
Current Balance = 250
```

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 [1]:
```python
class Inventory:
    def __init__(self):
        self.inventory = {}

    def add_item(self, item_name, price, quantity):
        if item_name in self.inventory:
            print(f"Item '{item_name}' already exists.")
        else:
            self.inventory[item_name] = {'price': price, 'quantity': quantity}
            print(f"Item '{item_name}' added to inventory.")

    def remove_item(self, item_name):
        if item_name in self.inventory:
            del self.inventory[item_name]
            print(f"Item '{item_name}' removed from inventory.")
        else:
            print(f"Item '{item_name}' not found in inventory.")

    def update_item(self, item_name, price=None, quantity=None):
        if item_name in self.inventory:
            self.inventory[item_name]['price'] = price
            self.inventory[item_name]['quantity'] = quantity
            print(f"Item '{item_name}' updated.")
        else:
            print(f"Item '{item_name}' not found in inventory.")

    def view_inventory(self):
        if not self.inventory:
            print("Inventory is empty.")
        else:
            for item_name, details in self.inventory.items():
                print(f"{item_name} - ${details['price']} x {details['quantity']

# Example usage:
i = Inventory()

# Add items
i.add_item("Laptop", 1200, 10)
i.add_item("Smartphone", 800, 20)

# View current inventory
i.view_inventory()
```

```
# Update item
i.update_item("Laptop", quantity=15)

# Remove item
i.remove_item("Smartphone")

# View updated inventory
i.view_inventory()
```

```
Item 'Laptop' added to inventory.
Item 'Smartphone' added to inventory.
Laptop - $1200 x 10 in stock
Smartphone - $800 x 20 in stock
Item 'Laptop' updated.
Item 'Smartphone' removed from inventory.
Laptop - $1200 x 15 in stock
```

07) Create a Class with instance attributes of your choice.

In [5]:
```
class Student:
    def __init__(self, Name, Age, StudentID, Grade):
        self.Name = Name
        self.Age = Age
        self.StudentID = StudentID
        self.Grade = Grade

    def display(self):
        return f"Student ID: {self.StudentID}, Name: {self.Name}, Age: {self.Age

s = Student("Karan", 20, "101", "A")
print(s.display())
```

```
Student ID: 101, Name: Karan, Age: 20, Grade: A
```

08) 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 [4]:
```
class StudentKit:
    PrincipalName = "Mr. ABC"

    def __init__(self, name):
        self.StudentName = name
        self.AttendanceDays = 0

    def Attendance(self, days):
        self.AttendanceDays = days

    # Method to generate a certificate
    def GetCertificate(self):
        print(f"Certificate of Attendance")
        print(f"This is to certify that {self.StudentName}")
        print(f"has attended {self.AttendanceDays} days of class.")
        print(f"Principal: {StudentKit.PrincipalName}")

name = input("Enter the student's name: ")
student = StudentKit(name)

days = int(input("Enter the number of days present in class: "))
student.Attendance(days)
student.GetCertificate()
```

```
Certificate of Attendance
This is to certify that Vishal
has attended 12 days of class.
Principal: Mr. ABC
```

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

In [7]:

```
class Time :
    hour = 0
    minute = 0

    def __init__(self,h,m):
        self.hour = h
        self.minute = m

    def addTime(self,t1,t2):
        self.hour = t1.hour + t2.hour
        self.minute = t1.minute + t2.minute

        if (self.minute >= 60):
            self.hour += self.minute // 60
            self.minute = self.minute % 60

    def display(self):
        print(f"Hour : {self.hour} Minute: {self.minute}")

t1 = Time(12,34)
t2 = Time(3,45)
t3 = Time(0,0)

t3.addTime(t1,t2)
t1.display()
t2.display()
t3.display()
```

```
Hour : 12 Minute: 34
Hour : 3 Minute: 45
Hour : 16 Minute: 19
```

In [ ]: