Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 13

Name: Jadeja Rudrarajsinh

Enrollment No: 23010101411

Roll No: 487

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 [2]:
```
class Students:
    def __init__(self,name,age,grade):
        self.name=name
        self.age=age
        self.grade=grade
name = input("Enter name of Student : ")
age = int(input("Enter age of Student : "))
grade = input("Enter grade of Student : ")
s = Students(name,age,grade)
print(f"Name = {s.name},Age = {s.age},Grade = {s.grade}")
```

```
Name = Jadeja Rudrarajsinh,Age = 20,Grade = 8.7
```

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 [4]:
```
class Bank_Account:
    def __init__(self,Account_No,User_Name,Email,Account_Type,Account_Balance):
        self.Account_No=Account_No
        self.User_Name=User_Name
        self.Email=Email
        self.Account_Type=Account_Type
        self.Account_Balance=Account_Balance
    def GetAccountDetails(self):
        self.Account_No = int(input("Enter Account Number"))
        self.User_Name = input("Enter UserName")
        self.Email = input("Enter Email")
        self.Account_Type = input("Enter Account Type")
        self.Account_Balance = float(input("Enter Account Balance"))

    def DisplayAccountDetails(self):
        print(f"Account_No:{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}")
def main():
    account = Bank_Account("","","","",0.0)
    account.GetAccountDetails()
    account.DisplayAccountDetails()
if __name__ == "__main__":
    main()
```

```
Account_No:23010101411
User_Name:Rudrarajsinh
Email:rudra@gmail.com
Account_Type:savings
Account_Balance:9999999999.0
```

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

In [6]:
```
import math
class Circle:
    def __init__(self,r):
        self.r=r
    def area(self):
        return math.pi*self.r*self.r
    def perimeter(self):
        return 2*math.pi*self.r
r = float(input("Enter radius of circle : "))
c = Circle(r)
print(f"Area = {c.area()}")
print(f"Perimeter = {c.perimeter()}")
```

```
Area = 1661.9025137490005
Perimeter = 144.51326206513048
```

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

In [8]:
```
class Employee:
    def __init__(self,name,age,salary):
```

```
        self.name=name
        self.age=age
        self.salary=salary
    def update(self):
        self.name=input("Enter name : ")
        self.age=int(input("Enter age : "))
        self.salary=float(input("Enter salary : "))
    def display(self):
        print(f"Name = {self.name}")
        print(f"Age = {self.age}")
        print(f"Salary = {self.salary}")
name = input("Enter name : ")
age = int(input("Enter age : "))
salary = float(input("Enter salary"))
e = Employee(name,age,salary)
n = int(input("Enter 1 to update and 0 to not update"))
if n==1:
    e.update()
    e.display()
else:
    e.display()
```

```
Name = Rudrarajsinh Jadeja
Age = 19
Salary = 99999.0
```

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

In [10]:

```
class BankAccount:
    account=0
    def __init__(self,balance):
        self.balance=balance
    def deposit(self,amount):
        self.balance+=amount
    def withdraw(self,amount):
        if balance>=amount:
            self.balance-=amount
        else:
            print("Insufficient Balance")
    def checkBalance(self):
        print("Current Balance =",self.balance)
balance = int(input("Enter balance"))
b = BankAccount(balance)
choice=0
while choice!=-1:
    print("Enter 1 to deposit")
    print("Enter 2 to withdraw")
    print("Enter 3 to display Balance")
    print("Enter -1 to exit")
    choice = int(input())
    match(choice):
        case 1:
            amount = int(input("Enter amount to deposit : "))
            b.deposit(amount)

        case 2:
            amount = int(input("Enter amount to withdraw : "))
            b.withdraw(amount)
```

```
        case 3:
            b.checkBalance()

        case -1:
            break
```

```
Enter 1 to deposit
Enter 2 to withdraw
Enter 3 to display Balance
Enter -1 to exit
Enter 1 to deposit
Enter 2 to withdraw
Enter 3 to display Balance
Enter -1 to exit
```

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 [12]:
```
class Inventory:
    def __init__(self):
        # Initialize an empty list to hold inventory items
        self.items = []

    def add_item(self, item_name, price, quantity):
        # Add a new item or update the quantity if it already exists
        for item in self.items:
            if item[0] == item_name:  # Check if item already exists
                item[2] += quantity  # Update quantity
                print(f"Added {quantity} more of '{item_name}' to the inventory.
                return
        self.items.append([item_name, price, quantity])  # Add new item
        print(f"Added '{item_name}' to the inventory.")

    def remove_item(self, item_name, quantity):
        # Remove a specific quantity of an item or the item entirely
        for item in self.items:
            if item[0] == item_name:
                if quantity >= item[2]:
                    self.items.remove(item)  # Remove the entire item
                    print(f"Removed all of '{item_name}' from the inventory.")
                else:
                    item[2] -= quantity  # Update quantity
                    print(f"Removed {quantity} of '{item_name}' from the invento
                return
        print(f"'{item_name}' is not in the inventory.")

    def update_item(self, item_name, price=None, quantity=None):
        # Update the price and/or quantity of an item
        for item in self.items:
            if item[0] == item_name:
                if price is not None:
                    item[1] = price  # Update price
                if quantity is not None:
                    item[2] = quantity  # Update quantity
                print(f"Updated '{item_name}' in the inventory.")
                return
        print(f"'{item_name}' is not in the inventory.")
```

```
    def display_inventory(self):
        # Display the entire inventory with details
        if not self.items:
            print("The inventory is empty.")
        else:
            print("Current Inventory:")
            for item in self.items:
                print(f"{item[0]} - Price: {item[1]}, Quantity: {item[2]}")
inventory = Inventory()

# Add items
inventory.add_item("Apple", 10, 50)
inventory.add_item("Banana", 5, 100)
inventory.add_item("Orange", 8, 30)

# Display the inventory
inventory.display_inventory()

# Remove some items
inventory.remove_item("Apple", 20)

# Update the price of an item
inventory.update_item("Banana", price=6)

# Update the quantity of an item
inventory.update_item("Orange", quantity=50)

# Display the updated inventory
inventory.display_inventory()
```

```
Added 'Apple' to the inventory.
Added 'Banana' to the inventory.
Added 'Orange' to the inventory.
Current Inventory:
Apple - Price: 10, Quantity: 50
Banana - Price: 5, Quantity: 100
Orange - Price: 8, Quantity: 30
Removed 20 of 'Apple' from the inventory.
Updated 'Banana' in the inventory.
Updated 'Orange' in the inventory.
Current Inventory:
Apple - Price: 10, Quantity: 30
Banana - Price: 6, Quantity: 100
Orange - Price: 8, Quantity: 50
```

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

In [16]:

```
class Book:
    def __init__(self, title, author, year_published, genre):
        # Instance attributes
        self.title = title
        self.author = author
        self.year_published = year_published
        self.genre = genre

    def display_info(self):
        # Method to display book details
        print(f"'{self.title}' by {self.author}, published in {self.year_publish
```

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 [20]:
```
class student_kit:
    principal_name="Mr ABC"
    def __init__(self,student_name):
        self.student_name = student_name
        self.attendance_days=0
    def attendance(self,days_present):
        self.attendance_days+=days_present
        print(f"Marked {days_present} days of attendance for {self.student_name}
    def generate_certificate(self):
        print(f"\n--- Certificate of Attendance ---")
        print(f"Principal: {student_kit.principal_name}")
        print(f"Student Name: {self.student_name}")
        print(f"Days Present: {self.attendance_days}")
        print("----------------------------------\n")
s = student_kit("Rudrarajsinh Jadeja")
s.attendance(10)
s.generate_certificate()
```

```
Marked 10 days of attendance for Rudrarajsinh Jadeja

--- Certificate of Attendance ---
Principal: Mr ABC
Student Name: Rudrarajsinh Jadeja
Days Present: 10
----------------------------------
```

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

In [22]:
```
class Time:
    def __init__(self,hour,minute):
        self.hour=hour
        self.minute=minute
    def add(self,t):
        total_minute = self.minute+t.minute
        total_hour = self.hour+t.hour+(total_minute//60)
        total_minute%=60
        return Time(total_hour,total_minute)
    def display_time(self):
        print(f"{self.hour}:{self.minute}")
h1 = int(input("Enter hour1 : "))
m1 = int(input("Enter minute1 : "))
h2 = int(input("Enter hour1 : "))
```

```
m2 = int(input("Enter minute1 : "))
t1 = Time(h1,m1)
t2 = Time(h2,m2)
result = t1.add(t2)
result.display_time()
```

```
19:6
```

In [ ]: