Python Programming - 2301CS404

Lab - 13

Dhol Namra

Enroll:23010101407

03-03-2025

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

stu = Students("Yash",19,"A++")

print("Name =",stu.name)
print("Age =",stu.age)
print("Grade =",stu.grade)
```

```
Name = Yash
Age = 19
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 [6]:
```
class Bank_Account:
    def __init__(self, Account_No, User_Name, Email, Account_Type, Account_Balan
        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):
        return {
            "Account_No" : self.Account_No,
            "User_Name" : self.User_Name,
            "Email" : self.Email,
            "Account_Type" : self.Account_Type,
            "Account_Balance" : self.Account_Balance
        }

    def DisplayAccountDetails(self):
        Account_Details = self.GetAccountDetails()
        for key, value in Account_Details.items():
            print(f"{key} : {value}")

Account = Bank_Account(23010101169, "Yash", "yash@gmail.com", "Saving",3005000)
Account.DisplayAccountDetails()
```

```
Account_No : 23010101169
User_Name : Yash
Email : yash@gmail.com
Account_Type : Saving
Account_Balance : 3005000
```

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

In [15]:
```
import math

class circle:
    def __init__(self,radius):
        self.radius = radius

    def area(self):
        return math.pi * (self.radius ** 2)

    def perimeter(self):
        return 2 * math.pi * self.radius

circle = circle(5)
print("Area :",circle.area())
print("Perimeter :",circle.perimeter())
```

```
Area : 78.53981633974483
Perimeter : 31.41592653589793
```

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

In [21]:
```
class Employee:
    def __init__(self, name, age, salary):
        self.name = name
        self.age = age
        self.salary = salary

    def update_salary(self,new_salary):
        self.salary = new_salary

    def update_age(self, new_age):
        self.age = new_age

    def display_information(self):
        print("Name =",self.name)
        print("Age =",self.age)
        print("Salary =",self.salary)

employee = Employee("Yash",19,250000)
employee.display_information()
print("----------------------")

employee.update_salary(150000)
employee.update_age(22)
employee.display_information()
```

```
Name = Yash
Age = 19
Salary = 250000
----------------------
Name = Yash
Age = 22
Salary = 150000
```

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

In [25]:
```
class BankAccount:
    def __init__(self, initial_balance=0):
        self.balance = initial_balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited {amount}. New balance is {self.balance}.")
        else:
            print("Enter Positive Deposit Amount.")

    def withdraw(self, amount):
        if amount > 0 and amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew {amount}. New balance is {self.balance}.")
        else:
            print("Invalid withdrawal amount or insufficient balance.")
```

```
    def check_balance(self):
        print(f"Your current balance is {self.balance}.")
```

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 [26]:
```
class InventoryItem:
    def __init__(self, name, price, quantity):
        self.name = name
        self.price = price
        self.quantity = quantity

    def add(self, amount):
        self.quantity += amount

    def remove(self, amount):
        if amount <= self.quantity:
            self.quantity -= amount
        else:
            print("Not enough items to remove.")

    def update_price(self, new_price):
        self.price = new_price

    def __str__(self):
        return f"Item: {self.name}, Price: {self.price}, Quantity: {self.quantit
```

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

In [27]:
```
class Dog:
    def __init__(self, name, breed, age):
        self.name = name
        self.breed = breed
        self.age = age

my_dog = Dog("Lucky", "Golden Retriever", 3)

print(my_dog.name)
print(my_dog.breed)
print(my_dog.age)
```

```
Lucky
Golden Retriever
3
```

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 [2]:
```
class StudentKit:
    # Class attribute
    principal_name = "Mr ABC"

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

    def attendance(self, days_present):
        self.days_present = days_present
        print(f"Attendance recorded: {self.student_name} attended {days_present}

    def generate_certificate(self):
        certificate = f"""

        Certificate of Attendance

        This is to certify that {self.student_name} has attended {self.days_pres

        Principal: {StudentKit.principal_name}

        """
        print(certificate)

# Taking input for student name
student_name = input("Enter student name: ")
student = StudentKit(student_name)

# Taking input for attendance days
days_present = int(input("Enter number of days present: "))
student.attendance(days_present)

# Generating certificate
student.generate_certificate()
```

```
Attendance recorded: demo attended 50 days.


        Certificate of Attendance

        This is to certify that demo has attended 50 days of classes.

        Principal: Mr ABC
```

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

In [4]:
```
class Time:
    def __init__(self, hour, minute):
        self.hour = hour
        self.minute = minute

    def add(self, other):
        total_minutes = self.minute + other.minute
```

```
        total_hours = self.hour + other.hour + (total_minutes // 60)
        total_minutes %= 60
        total_hours %= 24

        return Time(total_hours, total_minutes)

    def display(self):
        print(f"{self.hour:02d}:{self.minute:02d}")

# Example usage
t1 = Time(10, 45)
t2 = Time(2, 30)

result = t1.add(t2)
print("Resultant Time:", end=" ")
result.display()
```

```
Resultant Time: 13:15
```

In [ ]: