Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 13

Charmi Bhalodiya

23010101020

4B 448 8th batch

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

student1 = Students("charmi", 15, "10th Grade")
student2 = Students("mahek", 14, "9th Grade")

print("Student 1:", student1.name, student1.age, student1.grade)
print("Student 2:", student2.name, student2.age, student2.grade)
```

```
Student 1: charmi 15 10th Grade
Student 2: mahek 14 9th Grade
```

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 [2]:
```
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 = input("Enter Account Number: ")
        self.user_name = input("Enter User Name: ")
        self.email = input("Enter Email: ")
        self.account_type = input("Enter Account Type: ")
        self.account_balance = float(input("Enter Account Balance: "))

    def DisplayAccountDetails(self):
        print("\nAccount Details:")
        print("Account Number:", 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)
```

```
account = Bank_Account("", "", "", "", 0)
account.GetAccountDetails()
account.DisplayAccountDetails()
```

```
Account Details:
Account Number: 23010101020
User Name: charmi
Email: charmi@gmail.com
Account Type: savings
Account Balance: 230000.0
```

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

In [3]:
```
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

radius = float(input("Enter the radius of the circle: "))
circle = Circle(radius)

print("Area of Circle:", circle.area())
print("Perimeter of Circle:", circle.perimeter())
```

```
Area of Circle: 1661.9025137490005
Perimeter of Circle: 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 [4]:
```
class Employee:
    def __init__(self, name, age, salary):
        self.name = name
        self.age = age
        self.salary = salary

    def update_info(self, name=None, age=None, salary=None):
        if name:
            self.name = name
        if age:
            self.age = age
        if salary:
            self.salary = salary

    def display_info(self):
        print("\nEmployee Details:")
        print("Name:", self.name)
        print("Age:", self.age)
        print("Salary:", self.salary)

emp = Employee("John Doe", 30, 50000)
emp.display_info()

emp.update_info(age=32, salary=55000)
emp.display_info()
```

```
Employee Details:
Name: John Doe
Age: 30
Salary: 50000

Employee Details:
Name: John Doe
Age: 32
Salary: 55000
```

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

In [5]:
```
class BankAccount:
    def __init__(self, name, balance=0):
        self.name = name
        self.balance = balance
```

```
    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"{amount} deposited successfully. New balance: {self.balance}")
        else:
            print("Invalid deposit amount.")

    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            print(f"{amount} withdrawn successfully. New balance: {self.balance}")
        else:
            print("Insufficient balance or invalid amount.")

    def check_balance(self):
        print(f"Account holder: {self.name}, Current balance: {self.balance}")

account = BankAccount("Charmi", 1000)
account.check_balance()
account.deposit(500)
account.withdraw(300)
account.check_balance()
```

```
Account holder: Charmi, Current balance: 1000
500 deposited successfully. New balance: 1500
300 withdrawn successfully. New balance: 1200
Account holder: Charmi, Current balance: 1200
```

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 [6]:

```
class Inventory:
    def __init__(self):
        self.items = {}

    def add_item(self, name, price, quantity):
        if name in self.items:
            self.items[name]['quantity'] += quantity
        else:
            self.items[name] = {'price': price, 'quantity': quantity}
        print(f"{quantity} units of {name} added successfully.")

    def remove_item(self, name, quantity):
        if name in self.items and self.items[name]['quantity'] >= quantity:
            self.items[name]['quantity'] -= quantity
            if self.items[name]['quantity'] == 0:
                del self.items[name]
            print(f"{quantity} units of {name} removed successfully.")
        else:
            print("Item not found or insufficient quantity.")

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

    def display_inventory(self):
        if self.items:
            print("\nInventory Details:")
            for name, details in self.items.items():
                print(f"Item: {name}, Price: {details['price']}, Quantity: {details['quantity']}")
        else:
            print("Inventory is empty.")

inventory = Inventory()
inventory.add_item("Laptop", 50000, 10)
inventory.add_item("Mouse", 500, 50)
inventory.display_inventory()
inventory.remove_item("Mouse", 10)
inventory.update_item("Laptop", price=48000)
inventory.display_inventory()
```

```
10 units of Laptop added successfully.
50 units of Mouse added successfully.

Inventory Details:
Item: Laptop, Price: 50000, Quantity: 10
Item: Mouse, Price: 500, Quantity: 50
10 units of Mouse removed successfully.
Laptop updated successfully.

Inventory Details:
Item: Laptop, Price: 48000, Quantity: 10
Item: Mouse, Price: 500, Quantity: 40
```

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

In [7]:
```python
class Car:
    def __init__(self, brand, model, year, price):
        self.brand = brand
        self.model = model
        self.year = year
        self.price = price

    def display_details(self):
        print(f"Car Details: {self.year} {self.brand} {self.model}, Price: {self.price}")

car1 = Car("Toyota", "Corolla", 2022, 20000)
car2 = Car("Honda", "Civic", 2023, 25000)

car1.display_details()
car2.display_details()
```

```
Car Details: 2022 Toyota Corolla, Price: 20000
Car Details: 2023 Honda Civic, Price: 25000
```

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 [9]:
```python
class StudentKit:
    principal_name = "Mrs.Sanghani"

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

    def attendance(self):
        self.days_present = int(input(f"Enter number of days {self.student_name} was present: "))

    def generate_certificate(self):
        print("\n--- Attendance Certificate ---")
        print(f"Principal: {StudentKit.principal_name}")
        print(f"Student Name: {self.student_name}")
        print(f"Days Present: {self.days_present}")
        print("Congratulations on your attendance!\n")

student_name = input("Enter student name: ")
student = StudentKit(student_name)
student.attendance()
student.generate_certificate()
```

```
--- Attendance Certificate ---
Principal: Mrs.Sanghani
Student Name: charmi
Days Present: 20
Congratulations on your attendance!
```

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

In [10]:
```python
class Time:
    def __init__(self, hour, minute):
        self.hour = hour
        self.minute = minute

    def add_time(self, other):
```

```
        total_minutes = self.minute + other.minute
        extra_hours = total_minutes // 60
        new_hour = self.hour + other.hour + extra_hours
        new_minute = total_minutes % 60
        return Time(new_hour, new_minute)

    def display(self):
        print(f"{self.hour} hours {self.minute} minutes")

time1 = Time(2, 50)
time2 = Time(1, 30)

result = time1.add_time(time2)
result.display()
```

```
4 hours 20 minutes
```

In [ ]: