#1c

class Customer:

  def \_\_init\_\_(self,cust\_id,name,email,contact):

   self.cust\_id = cust\_id

   self.name = name

   self.email = email

   self.contact = contact

   def input\_details(self):

    print("Enter Customer Details :")

    self.cust\_id = input("Customer ID :")

    self.name = input("Name :")

    self.email = input("Email :")

    self.contact = input("Contact :")

  def display\_details(self):

    print(f"Customer ID : {self.cust\_id}")

    print(f"Name : {self.name}")

    print(f"Email : {self.email}")

    print(f"Contact : {self.contact}")

  def \_\_del\_\_(self):

    print("Destructor is called")

c1 = Customer(101,"ROHAN","abc@xyz.com","123456789")

c1.display\_details()

c2 = Customer(100,"JOHN","abc@xyz.com","234567891")

c2.display\_details()

#2a

#Method to add,subtract,multiply and divide two variables

class Calculator:

    def \_\_init\_\_(self, a, b):

        self.a = a  # Instance variable

        self.b = b  # Instance variable

    def add(self):

        return self.a + self.b  # Adding instance variables

    def sub(self):

        return self.a - self.b

# Create an object

obj = Calculator(5, 10)

print(obj.add())

print(obj.sub())

#2c

class StaticDemo:

    @staticmethod

    def is\_positive(number):

        """Check if a number is positive or negative."""

        if number > 0:

            return f"{number} is Positive"

        elif number < 0:

            return f"{number} is Negative"

        else:

            return "The number is Zero"

    @staticmethod

    def is\_even(number):

        """Check if a number is even or odd."""

        if number % 2 == 0:

            return f"{number} is Even"

        else:

            return f"{number} is Odd"

    @staticmethod

    def is\_valid\_email(email):

        """Check if an email is valid."""

        if "@" in email and "." in email:

            return f"{email} is a Valid Email"

        else:

            return f"{email} is an Invalid Email"

# Using the static methods

print(StaticDemo.is\_positive(10))    # Output: 10 is Positive

print(StaticDemo.is\_positive(-5))    # Output: -5 is Negative

print(StaticDemo.is\_positive(0))     # Output: The number is Zero

print(StaticDemo.is\_even(4))         # Output: 4 is Even

print(StaticDemo.is\_even(7))         # Output: 7 is Odd

print(StaticDemo.is\_valid\_email("test@example.com"))  # Output: test@example.com is a Valid Email

print(StaticDemo.is\_valid\_email("invalid-email"))     # Output: invalid-email is an Invalid Email

#8

import sqlite3

import os

#

conn = sqlite3.connect("my\_database.db")  # Creates a new database file if it doesn’t exist

cursor = conn.cursor()

#

cursor.execute("""

CREATE TABLE IF NOT EXISTS users (

    id INTEGER PRIMARY KEY AUTOINCREMENT,

    name TEXT,

    age INTEGER

)

""")

conn.commit()  # Save changes

#

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Alice", 25))

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Rohit", 26))

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("BOb", 27))

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Rahul", 28))

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("John", 29))

conn.commit()

#

cursor.execute("SELECT \* FROM users")

rows = cursor.fetchall()

for row in rows:

    print(row)

#

##to update records

cursor.execute("UPDATE users SET age = ? WHERE name = ?", (30, "Alice"))

conn.commit()

print("Record updated successfully!")

#

cursor.execute("SELECT \* FROM users")

rows = cursor.fetchall()

for row in rows:

    print(row)

#

##to update records

cursor.execute("UPDATE users SET age = 30 WHERE age > 25")

conn.commit()

print("Record updated successfully!")

#

cursor.execute("SELECT \* FROM users")

rows = cursor.fetchall()

for row in rows:

    print(row)

#

##to delete record

cursor.execute("DELETE FROM users WHERE name = ?", ("Rohit",))

conn.commit()

print("Record deleted successfully!")

#

cursor.execute("SELECT \* FROM users")

rows = cursor.fetchall()

for row in rows:

    print(row)

#

##to delete all records

cursor.execute("DELETE FROM users")

conn.commit()

print("All records deleted successfully!")

#

cursor.execute("SELECT \* FROM users")

rows = cursor.fetchall()

for row in rows:

    print(row)

#

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Alice", 25))

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Rohit", 26))

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("BOb", 27))

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Rahul", 28))

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("John", 29))

conn.commit()

#

cursor.execute("SELECT \* FROM users")

rows = cursor.fetchall()

for row in rows:

    print(row)

#

from sqlite3 import dbapi2

db\_name = "my\_database.db"

if os.path.exists(db\_name):

  os.remove(db\_name)

  print(f"Database '{db\_name}' has been deleted.")

else:

  print(f"Database '{db\_name}' does not exist.")

#9

#a

import ipywidgets as widgets

from IPython.display import display

# Create input box

text\_input = widgets.Text(description="Enter text:")

# Create output label

output\_label = widgets.Output()

# Create a button

button = widgets.Button(description="Submit")

# Function to handle button click

def on\_button\_click(b):

    with output\_label:

        output\_label.clear\_output()

        print("You entered:", text\_input.value)

# Attach function to button

button.on\_click(on\_button\_click)

# Display widgets

display(text\_input, button, output\_label)

#b

import ipywidgets as widgets

from IPython.display import display

# Text input

text\_input = widgets.Text(description="Enter text:")

# Radio buttons

radio = widgets.RadioButtons(

    options=["Option 1", "Option 2", "Option 3"],

    description="Choose one:"

)

# Checkboxes

checkbox1 = widgets.Checkbox(value=False, description="Enable feature A")

checkbox2 = widgets.Checkbox(value=False, description="Enable feature B")

# Output area

output\_label = widgets.Output()

# Button click function

def on\_button\_click(b):

    with output\_label:

        output\_label.clear\_output()

        print("You entered:", text\_input.value)

        print("Selected option:", radio.value)

        print("Feature A enabled:", checkbox1.value)

        print("Feature B enabled:", checkbox2.value)

# Submit button

button = widgets.Button(description="Submit")

button.on\_click(on\_button\_click)

# Display GUI elements

display(text\_input, radio, checkbox1, checkbox2, button, output\_label)

#c

import ipywidgets as widgets

from IPython.display import display

# Text input

text\_input = widgets.Text(description="Name:")

# Dropdown menu

dropdown = widgets.Dropdown(

    options=["Option A", "Option B", "Option C"],

    description="Dropdown:"

)

# List selection (multi-select enabled)

list\_select = widgets.SelectMultiple(

    options=["Item 1", "Item 2", "Item 3", "Item 4"],

    description="Select:"

)

# Text area

text\_area = widgets.Textarea(

    description="Remarks:",

    placeholder="Enter your comments here..."

)

# Output display

output\_label = widgets.Output()

# Submit button action

def on\_button\_click(b):

    with output\_label:

        output\_label.clear\_output()

        print("Name:", text\_input.value)

        print("Dropdown Selected:", dropdown.value)

        print("List Selection:", list(list\_select.value))

        print("Remarks:", text\_area.value)

# Submit button

button = widgets.Button(description="Submit")

button.on\_click(on\_button\_click)

# Display all widgets

display(text\_input, dropdown, list\_select, text\_area, button, output\_label)