import csv 🡨 (Is used to import the library used to read and write data to a CSV file.)

import os 🡨 (Is used to check if the contcts.csv file exists before trying to access it.)

CONTACT\_FILE = 'contacts.csv' 🡨( Defines the name of the file that stores the contacts.)

def create\_contact\_file():🡨(Creates a new CSV file to store contacts.)

    with open(CONTACT\_FILE, 'w', newline='') as file:

        writer = csv.writer(file)

        writer.writerow(['Name', 'Phone', 'Email'])🡨(Writes the column headers to the file.)

    print("\nContact file created successfully!") 🡨(Prints a success message at output for user to read.)

def add\_contact():🡨(Adds a new contact to the CSV file.)

    print("\nEnter the contact name: ", end="")🡨(Gives user option to input contact name.)

    name = input().strip()🡨(Asks user to input name, and strip removes extra spaces from user input)

    print("Enter contact phone number: ", end="")🡨(Asks user to input for contact #.)

    phone = input().strip()

    print("Enter contact email address: ", end="")🡨(Shows in message in output.)

    email = input().strip() 🡨(Asks user to input email and strip removes extra spaces from user input)

.)

    with open(CONTACT\_FILE, 'a', newline='') as file:🡨(Opens files in append mode also adds new daa without overwriting existing contacts.)

        writer = csv.writer(file)🡨(

        writer.writerow([name, phone, email])🡨(Writes the new contact as a row in the CSV file.)

    print("Contact added successfully!")🡨(Prints a success message.)

def view\_contacts():🡨(Displays all contacts in a neat table format.)

    if not os.path.exists(CONTACT\_FILE):

        print("No contacts file found. Please create a file first.")🡨(Prints message.)

        return

    with open(CONTACT\_FILE, 'r') as file:🡨(Opens file contacts.csv in read mode.)

        reader = csv.reader(file)🡨(Reads a CSV file line by line, converts each row into a list of strings, provides an easy way to work with structured data in CSV files.)

        contacts = list(reader)🡨(Converts the CSV files rows into a list of lists.)

        if len(contacts) <= 1:🡨(Checks if the contacts list has 1 or fewer rows.)

            print("No contacts found.")🡨(Displays a message informing the user that there are no contacts to show.)

            return 🡨(Exits current function immediately.)

        print("\nContacts Information:")🡨(Displays a heading.\n adds a blank line before the text to improve readability.)

        print("-" \* 40)🡨(Prints a line of 40 dashes separating the heading from contact details.)

    print(f"{'Name':<20} {'Phone':<15} {'Email'}")🡨(Displays the column header Name, Phone, and Email in formatted manner.)

    for row in contacts[1:]:🡨(Loops through each contact row excluding the header.)

        print(f"{row[0]:<20} {row[1]:<15} {row[2]}")🡨(Displays the details of each contact in a formatted way.)

def modify\_contact():🡨(Defines a function named modify\_contact().)

    if not os.path.exists(CONTACT\_FILE):🡨(Checks if the file contacts.csv exists on the system.)

        print("No contacts file found. Please create a file first.")🡨(Displays a message to inform the user that the file contacts.csv does not exist.)

        return🡨(Exits current function immediately.)

    with open(CONTACT\_FILE, 'r') as file:🡨(Opens contacts.csv for reading and assigns the file object to file.)

        reader = csv.reader(file)🡨(Creates an iterator that reads the filr as rows of CSV data.)

        contacts = list(reader)🡨(Converts the CSV into a list of ists for easier manipulation.)

    if len(contacts) <= 1:🡨(Checks if there are no contact rows but only in the header or an empty file.)

        print("No contacts available to modify.")🡨(Displays a message informing the user that there are no contacts to mofify.)

        return🡨(Exits current function immediately.)

    print("\nContacts Information:") :")🡨(Displays a heading.\n adds a blank line before the text to improve readability.)

    print("-" \* 40) )🡨(Prints a line of 40 dashes separating the heading from contact details.)

    print(f"{'Name':<20} {'Phone':<15} {'Email'}")}")🡨(Displays the column header Name, Phone, and Email in formatted manner.)

    for row in contacts[1:]:🡨(Loops through each contact row excluding the header.)

        print(f"{row[0]:<20} {row[1]:<15} {row[2]}") 🡨(Displays the details of each contact in a formatted way.)

    print("\nEnter the name of the contact you want to modify: ", end="")🡨(Displays a user-friendly prompt for input, keeping the cursor on the same line for cleaner input.)

    contact\_name = input().strip()🡨(Captures the users input, removes unnecessary spaces, and stores the result in the variable contact\_name.)

    contact\_found = False🡨(Initializes a flag to track whether a matching contact is found.)

    for i, row in enumerate(contacts[1:], start=1):🡨(Loops through all rows in the contacts list, skipping the header row.)

        if row[0].lower() == contact\_name.lower():🡨(Checks if the name in the current row matches the users input.)

            contact\_found = True🡨(Updates the flag to indicate that the contact has been found.)

            print(f"\nCurrent details: Name: {row[0]}, Phone: {row[1]}, Email: {row[2]}")🡨(Displays the current details of the contact being modified.)

            print("\nEnter new phone number (leave blank to keep current): ", end="")🡨(Gives the user to enter a new phone number for the contact. Informs the user that leaving the input blank will keep the current phone number. And \n adds a blank line before the prompt.)

            new\_phone = input().strip()🡨(Captures the users input for the new phone number. Removing any leading or trailing spaces from the input.)

            print("Enter new email address (leave blank to keep current): ", end="")🡨(Prompts the user to enter a new email for the contact. Informs the user that leaving the inpu blank will retain the current email.

            new\_email = input().strip()🡨(Captures the users input for the new email address. Removes any leading or trailing spaces from the input.)

            contacts[i][1] = new\_phone if new\_phone else row[1]🡨(Updates the phone number of the contact in the contacts list.)

            contacts[i][2] = new\_email if new\_email else row[2]🡨(Updates the email address of the contact in the contacts list.)

            with open(CONTACT\_FILE, 'w', newline='') as file:🡨(Opens contacts.csv in write mode, making sure no extra blank lines in the file.)

                writer = csv.writer(file)🡨(Creates a writer object for writing CSV rows.)

                writer.writerows(contacts)🡨(Writes al rows from the contacts list to the file.)

            print("Contact updated successfully!") 🡨(Prints a success message.)

            break🡨(Exits the loop after updating the contact.)

    if not contact\_found:🡨(Checks if the contact was not found during the loop.)

        print("Contact not found.")<(Displays an error message if no matching contact was found.)

def main():🡨(Entry part of the program.)

    print("Welcome to the Contact Manager App")🡨(Displays a welcome message at the begging of the program to the user.)

    print("-" \* 40) )🡨(Prints a line of 40 dashes separating the heading from contact details.)

    while True:🡨(Creates an infinite loop to continuously display the menu until the user choose to exit.)

        print("\nPush the following options to perform the corresponding action:")🡨(\adds a blank line before the heading for readability, displays the menu heading to the user.

        print("1 - Create new contact file")<(Prints each option on a new line, informing the user what each selection does.)----|

        print("2 - Add new contact") |

        print("3 - View all contacts") |

        print("4 - Modify an existing contact") |

        print("5 - Save and exit") |

        ------------------------------------------

        try:<(Ensures the program doesn’t crash if user uses enters invalid input.)

            print("Enter your selection: ", end="")<(Prompts the user to enter their menu choice. End keeps the cursor on the same line after the prompt, so the user cabn input immediately after the text.)

            selection = int(input().strip())<(Captures and validates the users input as an integer representing their menu choice.)

            if selection in [1, 2, 3, 4]: :<(Checks if the users selection ids one of the valid menu options.)

                print("-" \* 40) )🡨(Prints a line of 40 dashes separating the heading from contact details.)

            if selection == 1 <(Checks if the user selected option 1 create new contact file.)

                create\_contact\_file()<(Calls the function that creates a new contact file.)

            elif selection == 2: <(Checks if the user selected option 2 add new contact.)

                add\_contact()()<(Executes the add\_contact() function which prompts the user for contact details such as name, phone and email and appends them to the contacts.csv file.

            elif selection == 3: <(Checks if the user selected option 3 view all contacs.)

                view\_contacts()()<(Executes view\_contacts() function, which reads the contacts.csv file and displays all contacts in a formatted table.

            elif selection == 4: <(Checks if the user selected option 4 modify an existing contact.)

                modify\_contact()🡨(Executes the modify\_contact function which promps the user to select a contact by name. Allows the user to update the phone number and/or email address for the contact. Saves the updated contact information back to the contacts.csv file.)

            elif selection == 5: <(Checks if the user selected option 5 exit the program.)

                print("\nCompleted by, {Dajia-Rae Moreno}")

                break

            else:

                print("Invalid option. Please try again.")

        except ValueError:<(Catches invalid input.)

            print("Invalid input. Please enter a valid number.")

if \_\_name\_\_ == "\_\_main\_\_": <(Ensures the main function runs only when the script is executed directly.)

    main()<Starts the program.)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated