Contact Book Terminal Application

Emily Mehegan

FUNCTIONS OVERVIEW

The application will have the following functions.

1. Contact Menu

2. View contact book

3. Add contact

4. Delete contact

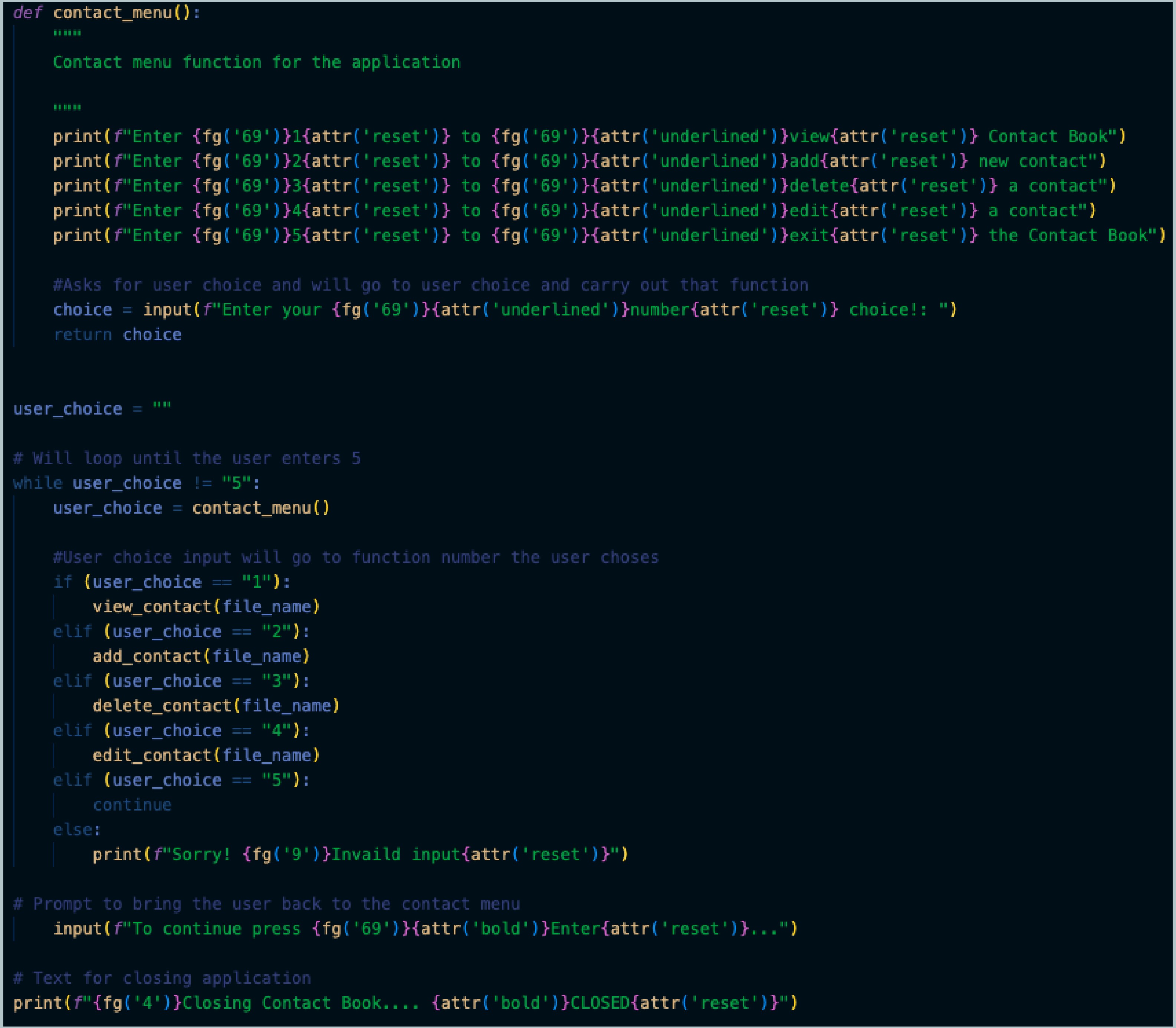
5. Edit a contact

6. Exit contact book

HOW TO USE

The application was built so that it would be easy to use and follow for any user. Since the application is intended to be a resource that is used for the user, this was kept in mind when creating and designing the terminal application.

1. The user is prompted to enter a number for the function that they wish to use
2. This will then carry out the chosen function
3. Once the function has run it will prompt the user the press enter
4. User is returned to the contact menu
5. Functions will carry out until the user chooses the 'exit the contact book' by pressing 5
6. This will close the application



```python
def contact_menu():
    """
    Contact menu function for the application

    """
    print(f"Enter {fg('69')}1{attr('reset')} to {fg('69')}{attr('underlined')}view{attr('reset')} Contact Book")
    print(f"Enter {fg('69')}2{attr('reset')} to {fg('69')}{attr('underlined')}add{attr('reset')} new contact")
    print(f"Enter {fg('69')}3{attr('reset')} to {fg('69')}{attr('underlined')}delete{attr('reset')} a contact")
    print(f"Enter {fg('69')}4{attr('reset')} to {fg('69')}{attr('underlined')}edit{attr('reset')} a contact")
    print(f"Enter {fg('69')}5{attr('reset')} to {fg('69')}{attr('underlined')}exit{attr('reset')} the Contact Book")

    #Asks for user choice and will go to user choice and carry out that function
    choice = input(f"Enter your {fg('69')}{attr('underlined')}number{attr('reset')} choice!: ")
    return choice


user_choice = ""

# Will loop until the user enters 5
while user_choice != "5":
    user_choice = contact_menu()

    #User choice input will go to function number the user choses
    if (user_choice == "1"):
        view_contact(file_name)
    elif (user_choice == "2"):
        add_contact(file_name)
    elif (user_choice == "3"):
        delete_contact(file_name)
    elif (user_choice == "4"):
        edit_contact(file_name)
    elif (user_choice == "5"):
        continue
    else:
        print(f"Sorry! {fg('9')}Invaild input{attr('reset')}")

# Prompt to bring the user back to the contact menu
    input(f"To continue press {fg('69')}{attr('bold')}Enter{attr('reset')}...")

# Text for closing application
print(f"{fg('4')}Closing Contact Book.... {attr('bold')}CLOSED{attr('reset')}")
```

MENU FUNCTION

HOW THE FUNCTION WORKS

- Welcomes user to the application
- Prompts user to enter user choice offering various functions
- Once choice entered will carry out the function
- When chosen function finish, will return to Contact Menu
- Will loop through function choices until user exits application by pressing '5'

User is prompted to enter a number from the number options.

Return statement will return the user's input choice, which is then stored in the user choice variable.

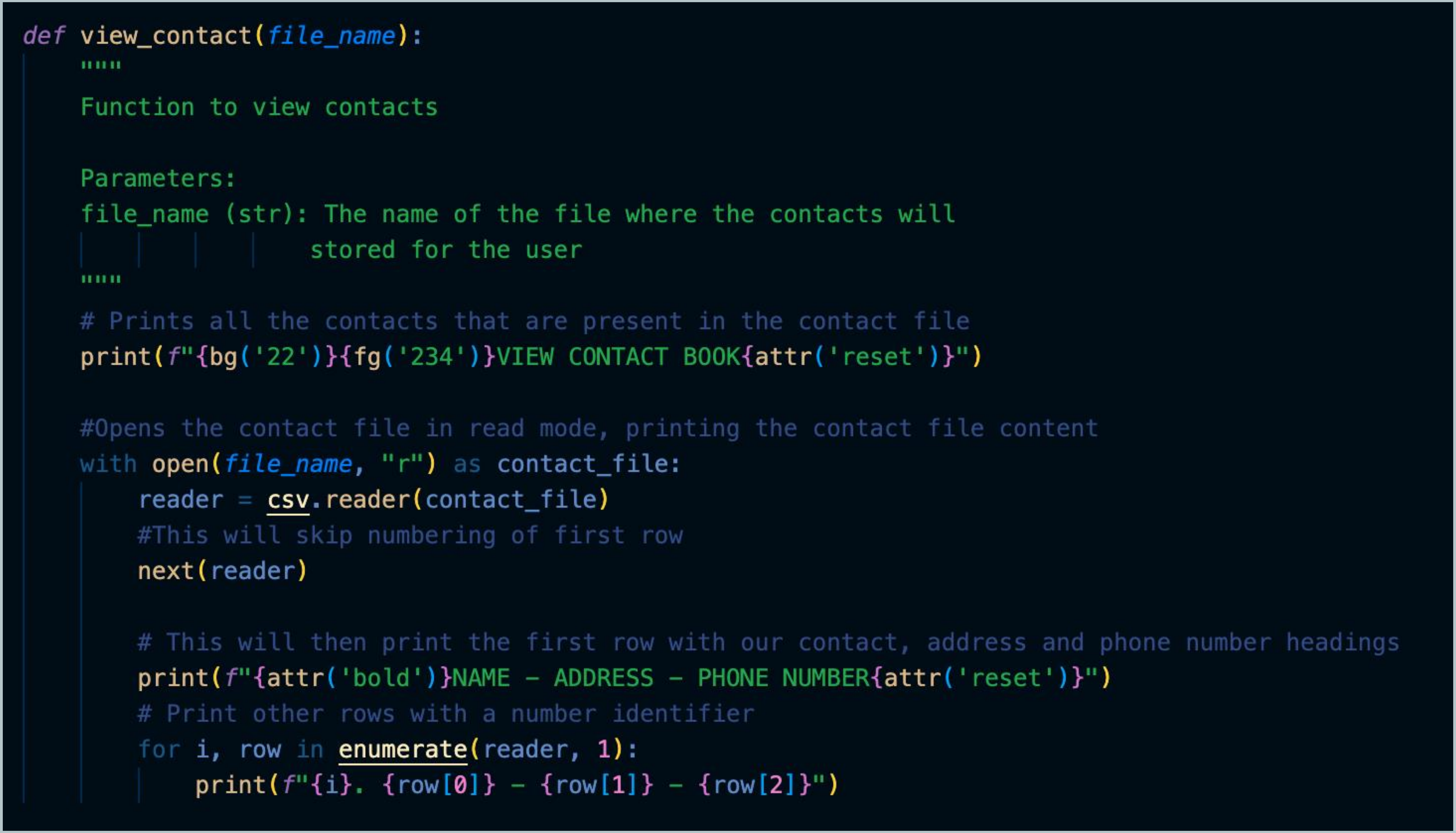
This local variable is assigned in the while loop, used to determine whether the application will continue to loop or exit, depending on the user input.

Determine which function the application will execute based on the user's input number choice. Bring user to input choice function

While loop will continue to bring the user back to the contact menu.

This will only stop until user chooses to exit the program by choosing '5'.

CONTACT MENU



```python
def view_contact(file_name):
    """
    Function to view contacts

    Parameters:
    file_name (str): The name of the file where the contacts will
                    stored for the user
    """
    # Prints all the contacts that are present in the contact file
    print(f"{bg('22')}{fg('234')}VIEW CONTACT BOOK{attr('reset')}")

    #Opens the contact file in read mode, printing the contact file content
    with open(file_name, "r") as contact_file:
        reader = csv.reader(contact_file)
        #This will skip numbering of first row
        next(reader)

        # This will then print the first row with our contact, address and phone number headings
        print(f"{attr('bold')}NAME - ADDRESS - PHONE NUMBER{attr('reset')}")
        # Print other rows with a number identifier
        for i, row in enumerate(reader, 1):
            print(f"{i}. {row[0]} - {row[1]} - {row[2]}")
```

FUNCTION 1

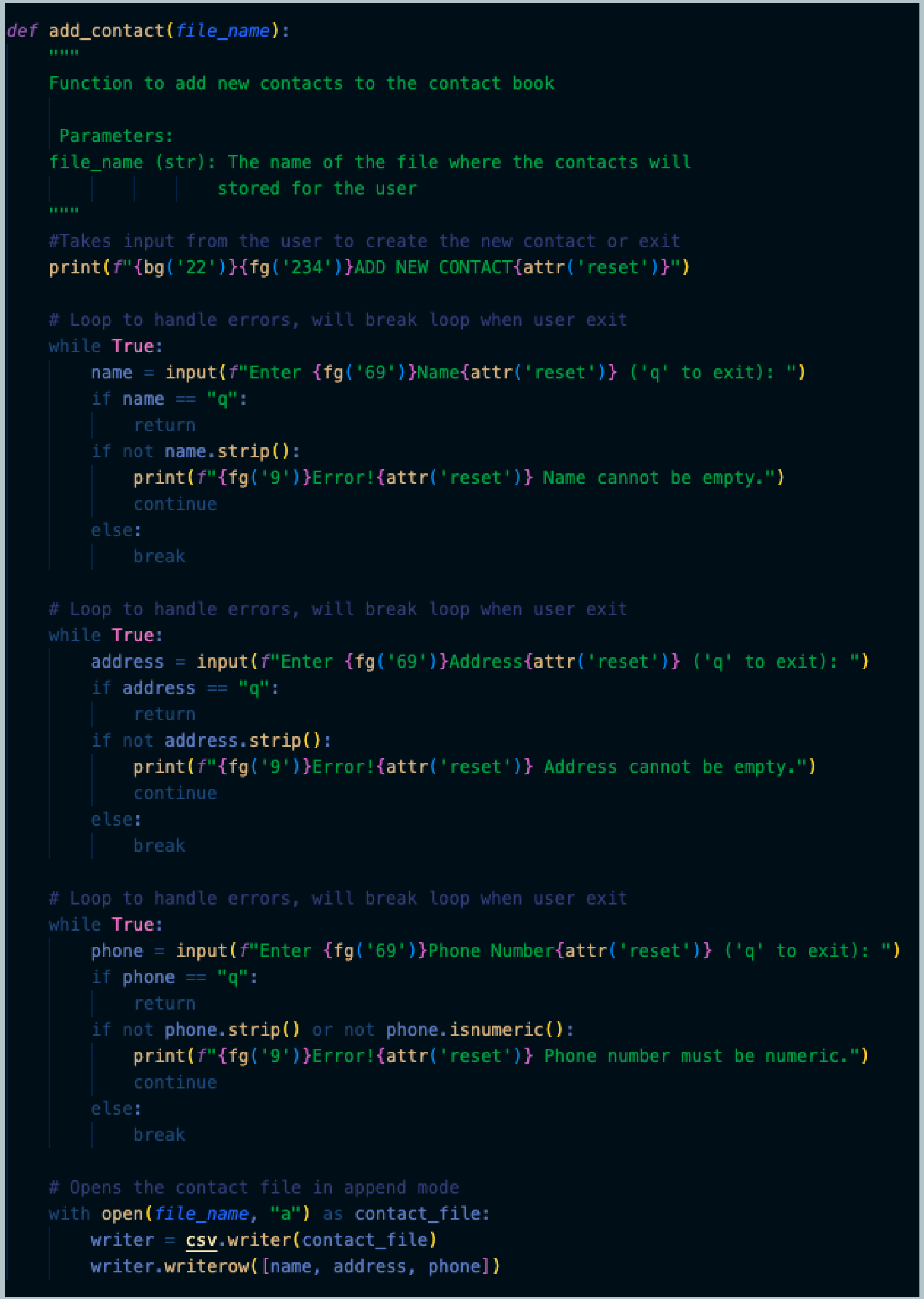
VIEW CONTACT BOOK

HOW THE FUNCTION WORKS

- User will press option '1' from contact menu
- Shows the contacts list from contact file in read mode
- Prompted to press 'Enter' once viewed
- Returned to the contact menu

The 'View Contact' function was made so that the user will have quick and easy access to their contacts.

Using the enumerate function also aids in organising the contact file information so that is more readable to the viewer.



```python
def add_contact(file_name):
    """
    Function to add new contacts to the contact book

     Parameters:
    file_name (str): The name of the file where the contacts will
                    stored for the user
    """
    #Takes input from the user to create the new contact or exit
    print(f"{bg('22')}{fg('234')}ADD NEW CONTACT{attr('reset')}")

    # Loop to handle errors, will break loop when user exit
    while True:
        name = input(f"Enter {fg('69')}Name{attr('reset')} ('q' to exit): ")
        if name == "q":
            return
        if not name.strip():
            print(f"{fg('9')}Error!{attr('reset')} Name cannot be empty.")
            continue
        else:
            break

    # Loop to handle errors, will break loop when user exit
    while True:
        address = input(f"Enter {fg('69')}Address{attr('reset')} ('q' to exit): ")
        if address == "q":
            return
        if not address.strip():
            print(f"{fg('9')}Error!{attr('reset')} Address cannot be empty.")
            continue
        else:
            break

    # Loop to handle errors, will break loop when user exit
    while True:
        phone = input(f"Enter {fg('69')}Phone Number{attr('reset')} ('q' to exit): ")
        if phone == "q":
            return
        if not phone.strip() or not phone.isnumeric():
            print(f"{fg('9')}Error!{attr('reset')} Phone number must be numeric.")
            continue
        else:
            break

    # Opens the contact file in append mode
    with open(file_name, "a") as contact_file:
        writer = csv.writer(contact_file)
        writer.writerow([name, address, phone])
```

FUNCTION 2

HOW THE FUNCTION WORKS

- User chooses option '2' from contact menu
- Ask user the name of new contact
- Ask user address of contact
- Ask user phone number contact
- Will add new contact to list
- Will prompt user to press 'Enter'
- User returned to contact menu

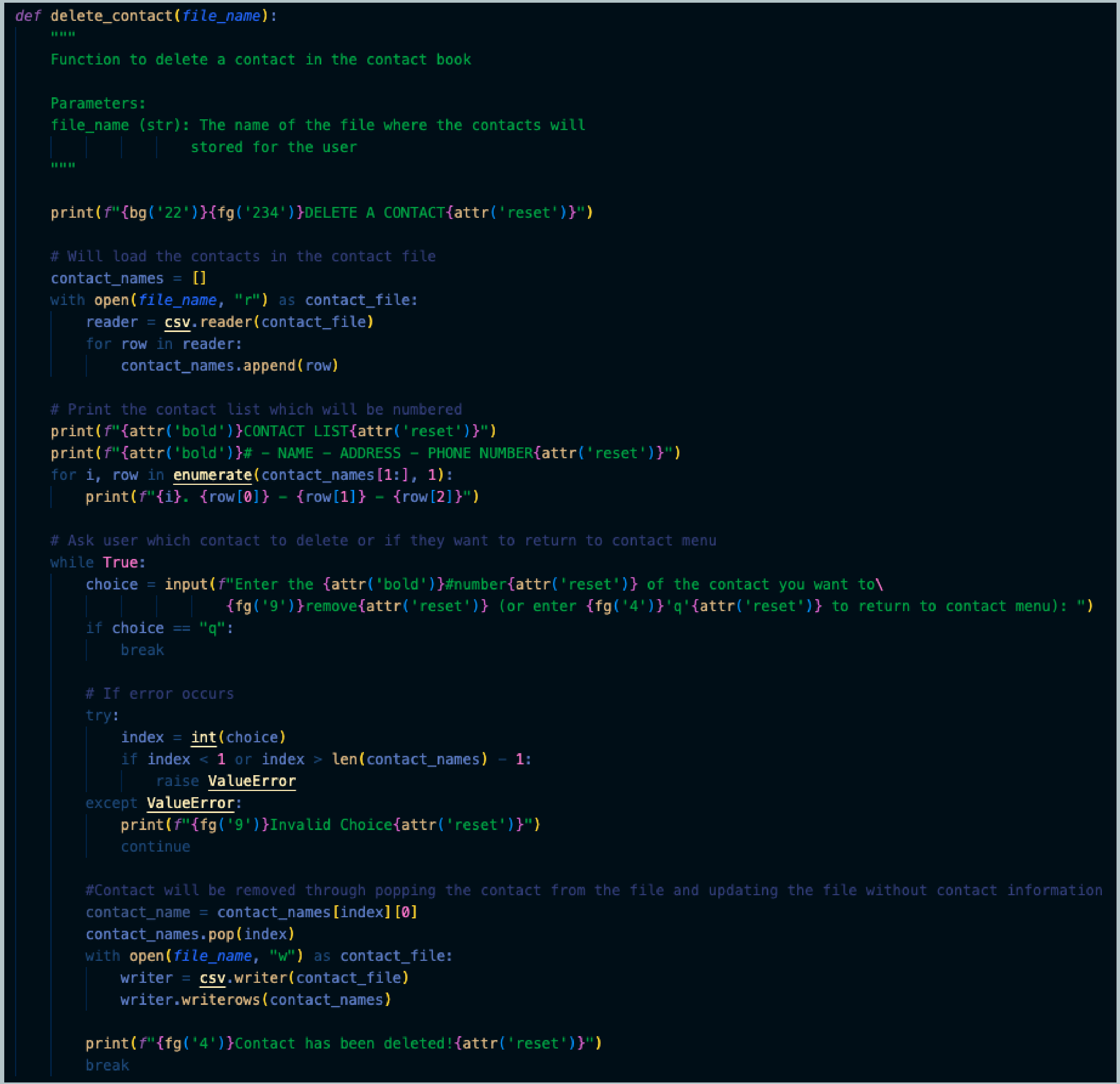
The user is prompted to provide the name, address and phone number of the contact.

User input added to contact list, will be able to be able to viewed in the 'view contacts' function

The contacts file is opened in append mode to be able to update the data in the file

An option to quit the function is offered to for each input.

ADD NEW CONTACT



```python
def delete_contact(file_name):
    """
    Function to delete a contact in the contact book

    Parameters:
    file_name (str): The name of the file where the contacts will
                    stored for the user
    """

    print(f"{bg('22')}{fg('234')}DELETE A CONTACT{attr('reset')}")

    # Will load the contacts in the contact file
    contact_names = []
    with open(file_name, "r") as contact_file:
        reader = csv.reader(contact_file)
        for row in reader:
            contact_names.append(row)

    # Print the contact list which will be numbered
    print(f"{attr('bold')}CONTACT LIST{attr('reset')}")
    print(f"{attr('bold')}# - NAME - ADDRESS - PHONE NUMBER{attr('reset')}")
    for i, row in enumerate(contact_names[1:], 1):
        print(f"{i}. {row[0]} - {row[1]} - {row[2]}")

    # Ask user which contact to delete or if they want to return to contact menu
    while True:
        choice = input(f"Enter the {attr('bold')}#number{attr('reset')} of the contact you want to\
                        {fg('9')}remove{attr('reset')} (or enter {fg('4')}'q'{attr('reset')} to return to contact menu): ")
        if choice == "q":
            break

        # If error occurs
        try:
            index = int(choice)
            if index < 1 or index > len(contact_names) - 1:
                raise ValueError
        except ValueError:
            print(f"{fg('9')}Invalid Choice{attr('reset')}")
            continue

        #Contact will be removed through popping the contact from the file and updating the file without contact information
        contact_name = contact_names[index][0]
        contact_names.pop(index)
        with open(file_name, "w") as contact_file:
            writer = csv.writer(contact_file)
            writer.writerows(contact_names)

        print(f"{fg('4')}Contact has been deleted!{attr('reset')}")
        break
```

FUNCTION 3

HOW THE FUNCTION WORKS

- User chooses option '3' from the contact menu
- Will display the contacts in a numbered list
- Ask user to input number of contact they wish to delete
- Read and save data except for deleted contact
- Makes sure that user input is case insensitive
- Writes updated contact list information
- Will prompt user to press 'Enter'
- User returned to contact menu

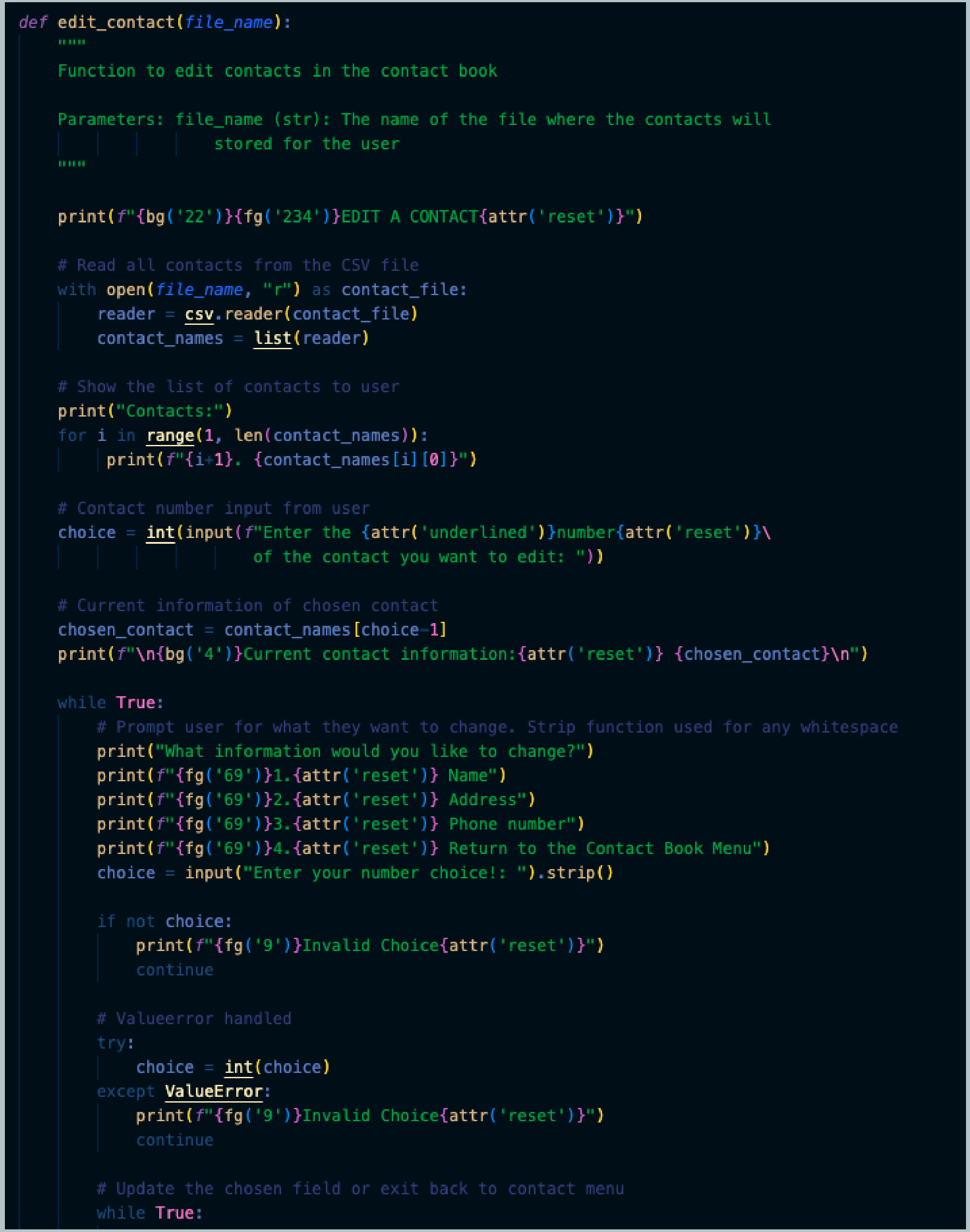
'Delete contact' function allows the user to delete a contact by inputting the number of the contact they wish to delete.

Will show a list of the contacts in the contact file with a number.

The contact list will read and saves all contacts except for deleted contact.

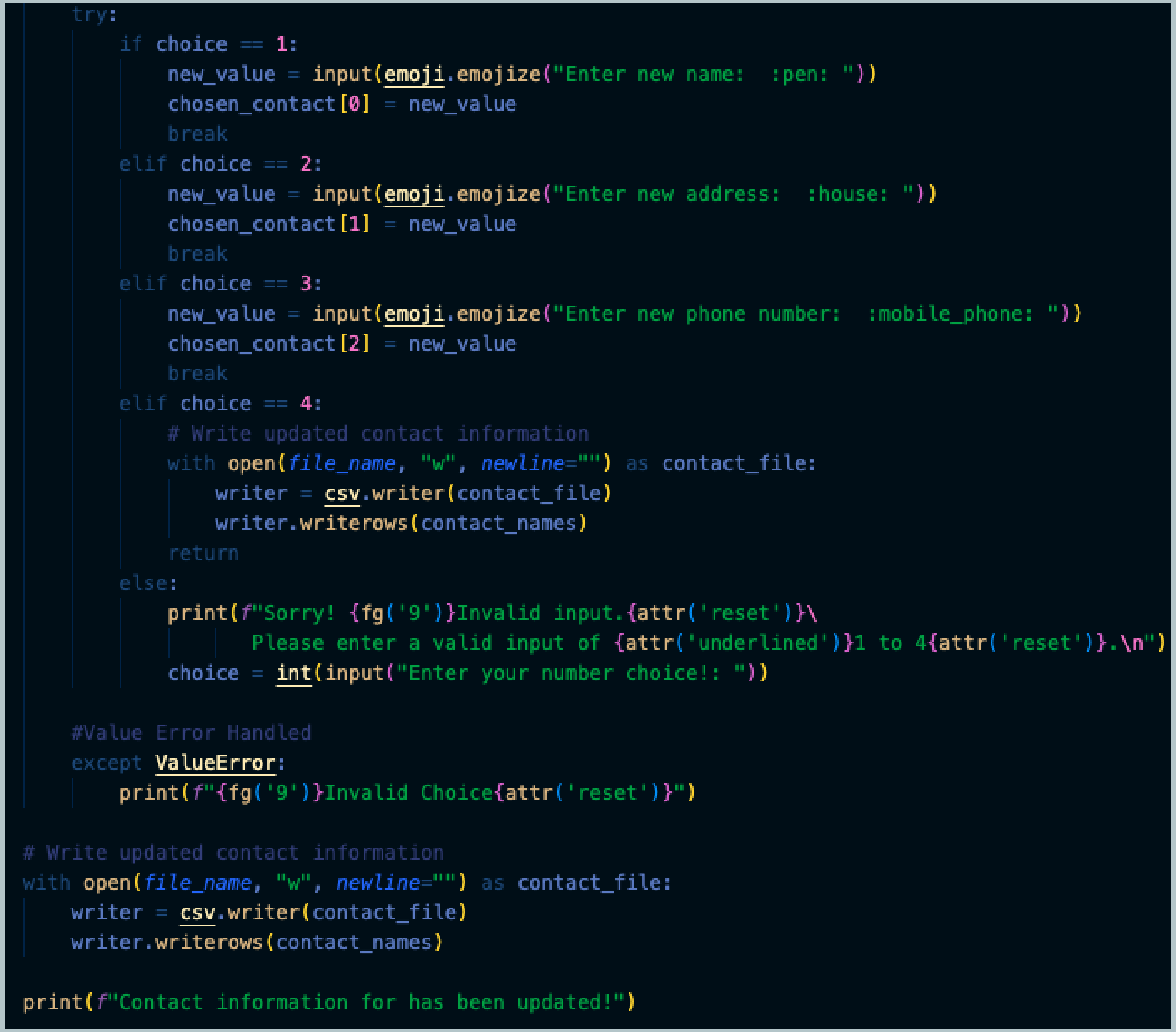
DELETE A CONTACT

1.



```python
def edit_contact(file_name):
    """
    Function to edit contacts in the contact book

    Parameters: file_name (str): The name of the file where the contacts will
                    stored for the user
    """

    print(f"{bg('22')}{fg('234')}EDIT A CONTACT{attr('reset')}")

    # Read all contacts from the CSV file
    with open(file_name, "r") as contact_file:
        reader = csv.reader(contact_file)
        contact_names = list(reader)

    # Show the list of contacts to user
    print("Contacts:")
    for i in range(1, len(contact_names)):
        print(f"{i+1}. {contact_names[i][0]}")

    # Contact number input from user
    choice = int(input(f"Enter the {attr('underlined')}number{attr('reset')}\
                    of the contact you want to edit: "))

    # Current information of chosen contact
    chosen_contact = contact_names[choice-1]
    print(f"\n{bg('4')}Current contact information:{attr('reset')} {chosen_contact}\n")

    while True:
        # Prompt user for what they want to change. Strip function used for any whitespace
        print("What information would you like to change?")
        print(f"{fg('69')}1.{attr('reset')} Name")
        print(f"{fg('69')}2.{attr('reset')} Address")
        print(f"{fg('69')}3.{attr('reset')} Phone number")
        print(f"{fg('69')}4.{attr('reset')} Return to the Contact Book Menu")
        choice = input("Enter your number choice!: ").strip()

        if not choice:
            print(f"{fg('9')}Invalid Choice{attr('reset')}")
            continue

        # Valueerror handled
        try:
            choice = int(choice)
        except ValueError:
            print(f"{fg('9')}Invalid Choice{attr('reset')}")
            continue

        # Update the chosen field or exit back to contact menu
        while True:
```

2.



```python
            try:
                if choice == 1:
                    new_value = input(emoji.emojize("Enter new name:  :pen: "))
                    chosen_contact[0] = new_value
                    break
                elif choice == 2:
                    new_value = input(emoji.emojize("Enter new address:  :house: "))
                    chosen_contact[1] = new_value
                    break
                elif choice == 3:
                    new_value = input(emoji.emojize("Enter new phone number:  :mobile_phone: "))
                    chosen_contact[2] = new_value
                    break
                elif choice == 4:
                    # Write updated contact information
                    with open(file_name, "w", newline="") as contact_file:
                        writer = csv.writer(contact_file)
                        writer.writerows(contact_names)
                    return
                else:
                    print(f"Sorry! {fg('9')}Invalid input.{attr('reset')}\
                          Please enter a valid input of {attr('underlined')}1 to 4{attr('reset')}.\n")
                    choice = int(input("Enter your number choice!: "))

            #Value Error Handled
            except ValueError:
                print(f"{fg('9')}Invalid Choice{attr('reset')}")

        # Write updated contact information
        with open(file_name, "w", newline="") as contact_file:
            writer = csv.writer(contact_file)
            writer.writerows(contact_names)

        print(f"Contact information for has been updated!")
```

FUNCTION 4

EDIT A CONTACT

HOW THE FUNCTION WORKS

•User chooses option '4' from contact menu
•Contact file is read in read mode
•Displays the numbered contacts
•User will input number of the contact they want to change
•User is shown current information for that contact and given options on what information they want to change (Name, address, phone) they are also given option to exit
•User gives input on the change to either the name, address or phone number
•This is then appended in the contact list file
•User loops back to information change menu until they decide the return to the contacts menu

It will read all the contacts in the contact list using the 'csv.reader()' method.

Then a list of the contact names with corresponding number are given so the user can choose which contact they wish to edit.

The function will use a 'for' loop which will start at 1 so it skips the heading line in the CSV file.

An 'input() function is used so the user can input the number of the contact they wish to edit. The information on the contact is stored in the 'chosen_contact' list.

A loop is entered so the user can choose what information (name, address, phone number or exit to contact book menu) they wish to change. With the 'strip()' function is used to aid in removing whitespace for the input. This helps handle some errors.

If the user has entered a valid choice, another loop will be entered to choose a new input (name, address, phone number) for a contact. This new contact information is updated and stored in the 'new_value' variable.

The updated information will be updated to the correct area of the 'chosen_contact' list. Which will then write the updated contact information in the CSV file.

It will continue to loop until the user decides to exit the edit function. Once the user decides to exit it will make sure that all the updated information is written into the file, save it and return to the contact menu.

FUNCTION 5

EXIT CONTACT BOOK

HOW THE FUNCTION WORKS

- User chooses option '5' from contact menu
- Will Exit the application

'Exit Contact Book' feature allows the user to just enter the number '5' which will then close the application.

Done through a looping function and the continue statement.

Continue statement makes sure to exit the current loop in the contact menu and move to the next loop. Since 'user_choice != 5' is not fulfilled, the program will break the loop and exit.

This is confirmed through the print statement "Closing Contact Book... CLOSED" to confirm.

If the user inputs an incorrect input choice, to avoid any errors, a 'Sorry! Invalid input' statement will be given to the user.

This will prompt them to press enter which will bring then back to the contact menu function.

DEVELOPMENT AND BUILD PROCESS

- Development process was broken down in Trello to aid with planning.
- Help breaks down the bigger tasks
- Start with the main.py file to build main components and functions for the application
- Separate larger functions into other files for organisation and readability
- Continue to check if functions work
- Create loops where needed
- Error handling in the functions
- Importing packages to elevate application

CHALLENGES

- Contact removal not deleting the contact
- A line that was added before the contact that wasn't meant to be there.
- Finding python packages that would work well with the application
- Running into problem after finishing a function
- Having to change the code to better suit function aspects or possible errors.

ETHICAL ISSUES

- The application does include options for people with disabilities.

- Issues with personal data
  - People's private information is stored in an application that the wrong people can have access too.
  - Do they want their information saved
  - Is it done with contact's consent

FAVOURITE PARTS

- I really enjoyed the styling of the application through colours/bolding/underline/emojis/highlighting

- Having the application work properly

- That the functions are practical