THE FOUNDATION SCHOOL, GUNJUR BENGALURU

COMPUTER SCIENCE

Topic: AIY Note - Abstract Information Y-type Note

Name: Yash Prasad

Grade: XII

Academic Year: 2021-22

CBSE Roll no.: 18611806

THE FOUNDATION SCHOOL, GUNJUR

CERTIFICATE

This is to certify that Yash Prasad of class XII has successfully completed his computer project titled “AIY Note - Abstract Information Y-type Note”, under the guidance of Ms. Swati Choudhary, as prescribed by the Central Board of Secondary Education, towards partial fulfilment of AISSCE course during the academic year 2021-22. The project is certified to be bonafide work of the student.

Date:

_____ _____

External Examiner Internal Examiner

_____

Principal

Declaration

I hereby declare that the project entitled, “AIY Note or Abstract information Y-type Note”, submitted to Department of Computer Science, The Foundation School, Sarjapura Hubli, Anekal Taluk, Bangalore is prepared by my team and I . All the coding is a result of our personal efforts.

YASH PRASAD

GRADE 12 A

Acknowledgement

I would like to express a deep sense of gratitude and thankfulness to our computer teacher, Mrs. Swati Choudhary giving us this wonderful opportunity to perform and present this project. She through this project has taught us a great deal of programming. Her constructive advice and consistent motivation have pushed us forward to successfully complete this project.

I would also like to thank our principal, Mrs.Anita Jayaram for her coordination in extending every possible support for the completion of this project.

I would also like to thank our respective parents for their motivation and support. I must also thank my classmates for the help they provided towards the completion of this project and especially my team mates for their support and determination.

Finally, I would like to thank all those people who helped with this project directly or indirectly.

Table of Content

Program Synopsis

AIY Note or Abstract Information Y-Note is a note taking application which uses python as the core programming language and SQL database and text files for the storing information data.

The application is a Y-type system where all the three components are connected with each other. The program uses python to connect the application to SQL and text files. The SQL database works as servers through which information is accessed. And the text files are the form in which the application data are stored.

The program is open source and user friendly in its design and working.

Configuration

The program can be run on python idle version: 3.9.1 of python version: 3.9.1 and Tk version: 8.6.8 .

The program uses external modules and applications. If the user does not have them installed, it is required that they install mysql.connector module and the MySQL version: 8.0.23 application.

Program

"""

AIY Note

AIY Note or Abstract Information Y-Note is a note taking application which uses python as the core programming language and SQL database and text files for the storing information data.

The application is a Y-type system where all the three components are connected with each other.

The program uses python to connect the application to SQL and text files.

The SQL database works as servers through which information is accessed.

And the text files are the form in which the application data are stored.

The program is open source and user friendly in its design and working.

"This is a computer project undertaken by a few students of The Foundation School, Gunjur,Bangalore."

"Our aim is to provide a note taking application using python, SQL database and text files."

"""

print("AIY Note")

print(

"""

AIY Note or Abstract Information Y-Note is a note taking application which uses python as the core programming language and SQL database and text files for the storing information data.

The application is a Y-type system where all the three components are connected with each other.

The program uses python to connect the application to SQL and text files.

The SQL database works as servers through which information is accessed.

And the text files are the form in which the application data are stored.

The program is open source and user friendly in its design and working.

"This is a computer project undertaken by a few students of The Foundation School, Gunjur,Bangalore."

"Our aim is to provide a note taking application using python, SQL database and text files."

"""

)

print()

```
# imports
import mysql.connector
from tkinter import *
from PIL import Image, ImageTk
from tkinter import ttk
from tkinter import messagebox
import random
import time
import smtplib
import os

# display window
displaywindow = Tk()
displaywindow.title('AIY Note')
displaywindow.geometry('500x500')
displaywindow.resizable(False, False)
displaywindow["bg"] = '#000000'

# Connection to MYSQL
```

```
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd=""
)

# create a cursor and initializing sql terminal
my_c = mydb.cursor()

# global variable
no_section = 2
no_note = 2
dict_section_note = {"section_1": ["note_1", "note_2"], "section_2": ["note_1", "note_2"]}
activelog_section_note_list = [['section_1', 'note_1']]

entrylog_section = list()
entrylog_note = list()
deleted_section_list = list()
deleted_notes_section_dict = dict()
directory_path_entry_log = list()
directory_section_note_dict = dict()
active_button_feature_panel = list()
count_maindisplay_function_activated = 0
account_no_user = None
active_feature_panel = None
location_AIY_Logo = "AIY LOGO_2.png"

# for trial purposes
account_no_user = "AIY0001"
```

```
pincode = 0
username_ = "XAY"
login_id_entry_allow = "AIY_login"
login_password_entry_allow = "AIY_password"


# accessing sql server for checking if AIY_notes_database exist and if user_accounts exist in that database
AIY_notes_database = False
while AIY_notes_database is False:
    my_c.execute("Show databases;")
    list_databases = list()
    for db in my_c:
        list_databases.append(db)
    databases = list()
    for ld in list_databases:
        for d in ld:
            databases.append(d)


    if "AIY_note_database" in databases:
        AIY_notes_database = True
        break
    else:
        my_c.execute("create database AIY_note_database")


# After creation of AIY_notes_database we verifying and updating the accounts in the user_AIY_notes_database table
if AIY_notes_database is True:
    user_accounts = False
    while user_accounts is False:
        my_c.execute("use AIY_note_database")
```

```
        my_c.execute("show tables")
        list_tables = list()
        for t in my_c:
            list_tables.append(t)
        tables = list()
        for lt in list_tables:
            for t_ in lt:
                tables.append(t_)

        if "User_accounts" in tables:
            user_accounts = True
        else:
            my_c.execute("use AIY_note_database")
                        execute_code_1 = "create table User_accounts(S_Name VarChar(100),DoB
DATE,Password VarChar(100)," \
                           "Account_no VarChar (18),ph_no VarChar(20),email_id VarChar(100)) "
            my_c.execute(execute_code_1)



def sql_part(purpose_sql, list_info_sql=None):
    # consist of coding part to help access sql server for required purpose and use
    global account_no_user, username_
    if purpose_sql == 'account_creation':
        print("account_creation activate")
        if list_info_sql is not None:
            my_c.execute('use AIY_note_database')
            my_c.execute("SELECT * FROM User_accounts")
            myresult_1 = my_c.fetchall()
            sql_username_list = list()
```

```
        sql_account_no_list = list()
        if myresult_1 is not None:
            for sql_data in myresult_1:
                sql_username_list.append(sql_data[0])
                sql_account_no_list.append(sql_data[3])
            if list_info_sql[0] in sql_username_list:
                messagebox.showerror("AIY Note", "The username already exist, enter another
username.")
            else:
                new_account_create = False
                account_no_no = 0
                while new_account_create is False:
                    account_testing = 'AIY000' + str(account_no_no)
                    if account_testing not in sql_account_no_list:
                        account_no_account_creation = account_testing
                        list_info_sql.insert(3, account_no_account_creation)
                        break
                    else:
                        account_no_no += 1
            sql_creation_code = "insert into User_accounts values(%s,%s,%s,%s,%s,%s)"
            value_creation = tuple()
            for sqlinfo in list_info_sql:
                value_creation = value_creation + (sqlinfo,)
            print("125 value creation", value_creation)
            my_c.execute(sql_creation_code, value_creation)
            my_c.execute("use AIY_note_database")
            print("creating account table")
                        execute_code_2 = "create table " + list_info_sql[3] + "(Section_no
INT,Section_name varchar(150)," \
                    "Note_no INt,Note_name Varchar(150),Delete_note varchar(10))"
```

```
            my_c.execute(execute_code_2)

            username_ = list_info_sql[0]
            if list_info_sql[3] is not None:
                account_no_user = list_info_sql[3]
            else:
                print("account creation of sql part not working")

            maindisplay()

        else:
            print('list_info_sql not present')

    elif purpose_sql == 'forget_account':
        print("sql forget activate")
        print("list_info_sql: ", list_info_sql)
        my_c.execute('use AIY_note_database')
        my_c.execute("SELECT * FROM User_accounts")
        myresult_forget = my_c.fetchall()
        print("myresult_forget: ", myresult_forget)
        correct_account_info = None
        if myresult_forget is not None:
            if list_info_sql is not None:
                if correct_account_info is None:
                    for info in myresult_forget:
                        print("info: ", info)
                                print('str(info[0]), str(info[4]), str(info[5]): ', str(info[0]), str(info[4]),
str(info[5]))
                        print(
```

```
                    "str(list_info_sql[0]), str(list_info_sql[4]), str(list_info_sql[5]) :",
                    str(list_info_sql[0]), str(list_info_sql[4]), str(list_info_sql[5])
                )
                if str(info[0]) == str(list_info_sql[0]):
                    if str(info[5]) == str(list_info_sql[5]):
                        correct_account_info = info
                        break
                    else:
                        pass
                else:
                    continue
            else:
                correct_account_info = "not found"
        else:
            print('list_info_sql not present')
    else:
        print('Not working')

    print("correct_account_info", correct_account_info)
    if correct_account_info is not None:
        return correct_account_info
    elif correct_account_info == 'not found':
        return 'not found'
    else:
        print('correct_account not working')

elif purpose_sql == 'loginwindow':
    print("sql loginwindow activate")
    if list_info_sql is not None:
```

```
if list_info_sql[0] == "AIY_note" and list_info_sql[1] == "AIY_password":
    username_ = "XAY"
    account_no_user = "AIY0001"
    return True
else:
    print("seaching through SQL database")
    my_c.execute('use AIY_note_database')
    my_c.execute("SELECT S_name,Password FROM User_accounts")
    myresult = my_c.fetchall()
    print("myresult: ", myresult)
    print("list_info_sql:", list_info_sql)
    id_password_verify = None
    if myresult is not None:
        if list_info_sql is not None:
            for info in myresult:
                if list_info_sql[0] == info[0]:
                    if list_info_sql[1] == list_info_sql[1]:
                        id_password_verify = True
                    else:
                        break
                else:
                    continue
        else:
            id_password_verify = False
    else:
        print(" login sql not working")

print("id_password_verify", id_password_verify)
if id_password_verify is not None:
```

```
            if list_info_sql is not None:
                print("Finding account no")
                username_ = list_info_sql[0]
                print("username_, account_no_user : ", username_, account_no_user)
                my_c.execute('use AIY_note_database')
                    login_sql_code_1 = "SELECT Account_no FROM User_accounts WHERE
S_Name = '%s'" % username_
                my_c.execute(login_sql_code_1)
                myresult_2 = my_c.fetchall()
                print(myresult_2)
                account_no_user = str(myresult_2[0][0])
                print("account_no_user : ", account_no_user)
            else:
                pass
            return id_password_verify
        elif id_password_verify is False:
            return False
        else:
            print(" login sql not working")
    else:
        print(" login sql not working")


elif purpose_sql == "maindisplay_entry":
    if count_maindisplay_function_activated == 0:
        print("maindisplay entry sql part activate")
        global no_section, no_note, dict_section_note, activelog_section_note_list
        print("account_no_user", account_no_user)


        my_c.execute("use AIY_note_database")
```

```
        my_c.execute("show tables")
        list_tables_2 = list()
        for t_22 in my_c:
            list_tables_2.append(t_22)
        tables_34 = list()
        for lt_ in list_tables_2:
            for t_4 in lt_:
                tables_34.append(t_4)

        print("tables: ", tables_34)
        account_no_table_found = False
        if account_no_user in tables_34:
            account_no_table_found = True
        else:
            my_c.execute("use AIY_note_database")
                        execute_code_3 = "create table " + account_no_user + "(Section_no
INT,Section_name varchar(150)," \
                                                                    "Note_no INt,Note_name
Varchar(150),Delete_note" \
                                                " varchar(10)) "
            my_c.execute(execute_code_3)
            account_no_table_found = True

        if account_no_table_found is True:
            my_c.execute('use AIY_note_database')
            my_c.execute("SELECT * FROM "+account_no_user+"")
            myresult_maindiaplay_entry = my_c.fetchall()
            print("myresult_forget: ", myresult_maindiaplay_entry)

            if len(myresult_maindiaplay_entry) > 0:
```

```
print("len(myresult_maindiaplay_entry) > 0")
global deleted_section_list, deleted_notes_section_dict
deleted_section_sql_target = list()
deleted_section_list = list()
deleted_notes_section_dict = dict()
dict_section_note = dict()
sql_section_list = list()

for sql_data_me in myresult_maindiaplay_entry:
    if sql_data_me[0] not in sql_section_list:
        sql_section_list.append(sql_data_me[0])
        if sql_data_me[0] == sql_section_list[-1]:
            section_target_sql = "section_"+str(sql_data_me[0])
            list_note_section_sql = list()
            section_note_target_sql = "note_"+str(sql_data_me[2])
            list_note_section_sql.append(section_note_target_sql)
            dict_section_note[section_target_sql] = list_note_section_sql

        else:
            pass

    elif sql_data_me[0] in sql_section_list:
        if sql_data_me[0] == sql_section_list[-1]:
            section_target_sql_2 = "section_" + str(sql_data_me[0])
            list_note_section_sql_2 = dict_section_note[section_target_sql_2]
            section_note_target_sql_2 = "note_" + str(sql_data_me[2])
            list_note_section_sql_2.append(section_note_target_sql_2)
            dict_section_note[section_target_sql_2] = list_note_section_sql_2
        else:
```

```
                pass
            else:
                pass

            # deleted section notes
            if sql_data_me[4] == "Y":
                if sql_data_me[0] not in deleted_section_sql_target:
                    deleted_section_sql_target.append(sql_data_me[0])
                    if sql_data_me[0] == deleted_section_sql_target[-1]:
                        section_target_sql_3 = "section_" + str(sql_data_me[0])
                        list_note_section_sql_3 = list()
                        section_note_target_sql_3 = "note_" + str(sql_data_me[2])
                        list_note_section_sql_3.append(section_note_target_sql_3)
                                        deleted_notes_section_dict[section_target_sql_3] =
list_note_section_sql_3

                    else:
                        pass

                elif sql_data_me[0] in deleted_section_sql_target:
                    deleted_section_sql_target.append(sql_data_me[0])
                    if sql_data_me[0] == deleted_section_sql_target[-1]:
                        section_target_sql_4 = "section_" + str(sql_data_me[0])
                        list_note_section_sql_4 = dict_section_note[section_target_sql_4]
                        section_note_target_sql_4 = "note_" + str(sql_data_me[2])
                        list_note_section_sql_4.append(section_note_target_sql_4)
                                        deleted_notes_section_dict[section_target_sql_4] =
list_note_section_sql_4
            else:
                pass
```

```
                for section_sql_target_don in sql_section_list:
                    section_target_name = "section_"+str(section_sql_target_don)
                    print("section_target_name: ", section_target_name)
                                                    print("dict_section_note[section_target_name]:  ",
dict_section_note[section_target_name])

                    deleted_section_list_append = False
                    if len(dict_section_note[section_target_name]) > 0:
                        if section_target_name in deleted_notes_section_dict:
                            if len(deleted_notes_section_dict[section_target_name]) > 0:
                                print("deleted_notes_section_dict[section_target_name]: ",
                                    deleted_notes_section_dict[section_target_name])
                                for note_targeted in dict_section_note[section_target_name]:
                                                                        if note_targeted in
deleted_notes_section_dict[section_target_name]:
                                        deleted_section_list_append = True
                                    else:
                                        deleted_section_list_append = False
                                        break
                            else:
                                deleted_section_list_append = False
                                break
                        else:
                            deleted_section_list_append = False
                            break
                    else:
                        continue

                    if deleted_section_list_append is True:
```

```
                    deleted_section_list.append(section_target_name)
                else:
                    continue

            no_section = len(sql_section_list)
            activelog_section_note_list = list()
            first_section_name_sql = "section_" + str(sql_section_list[0])
            sql_section_note_tagret_list = dict_section_note[first_section_name_sql]
                                    activelog_section_note_list = [[first_section_name_sql, sql_section_note_tagret_list[0]]]

        else:
            print("sql account table is empty hence everything remains the same.")
            no_section = 2
            no_note = 2
              dict_section_note = {"section_1": ["note_1", "note_2"], "section_2": ["note_1", "note_2"]}
            activelog_section_note_list = [['section_1', 'note_1']]
            deleted_section_list = list()
            deleted_notes_section_dict = dict()
      else:
          print("Account was found but cannot be accessed")
    else:
      pass

  elif purpose_sql == "exit_program":
    print("exit program activate")
    my_c.execute("use AIY_note_database")
    my_c.execute("SELECT * FROM " + account_no_user + "")
    myresult_maindiaplay_entry = my_c.fetchall()
```

```
list_ep_entry = list()
print("dict_section_note: ", dict_section_note)
print("deleted_notes_section_dict: ", deleted_notes_section_dict)

for section_notes_ep in dict_section_note:
    if section_notes_ep in deleted_section_list:
        note_list_ep = dict_section_note[section_notes_ep]
        for note_ep in note_list_ep:
            list_ep_entry_tuple = (
                int(section_notes_ep[8:]), section_notes_ep,
                int(note_ep[5:]), note_ep,
                "Y"
            )
            list_ep_entry.append(list_ep_entry_tuple)
    elif section_notes_ep not in deleted_section_list:
        note_list_ep_2 = dict_section_note[section_notes_ep]
        note_list_ep_2_deleted = deleted_notes_section_dict[section_notes_ep]
        for note_ep_2 in note_list_ep_2:
            if note_ep_2 not in note_list_ep_2_deleted:
                list_ep_entry_tuple_2 = (
                    int(section_notes_ep[8:]), section_notes_ep,
                    int(note_ep_2[5:]), note_ep_2,
                    "N"
                )
            elif note_ep_2 in note_list_ep_2_deleted:
                list_ep_entry_tuple_2 = (
                    int(section_notes_ep[8:]), section_notes_ep,
                    int(note_ep_2[5:]), note_ep_2,
```

```
                "Y"
            )
        list_ep_entry.append(list_ep_entry_tuple_2)
    else:
        pass

if len(myresult_maindiaplay_entry) == 0:
    sql_ep_entry_code = "insert into " + account_no_user + " values(%s,%s,%s,%s,%s)"
    print(sql_ep_entry_code)
    for tuple_data_entry_ep in list_ep_entry:
        value_data_ep = tuple()
        for tuple_data_write in tuple_data_entry_ep:
            value_data_ep = value_data_ep + (tuple_data_write,)
        print("writing into database", account_no_user)
        print(tuple_data_entry_ep, value_data_ep)
        my_c.execute(sql_ep_entry_code, value_data_ep)

elif len(myresult_maindiaplay_entry) != 0:
    sql_ep_entry_code_1_ = "drop table " + account_no_user +""
    my_c.execute(sql_ep_entry_code_1_)
            execute_code_2_ep = "create table " + account_no_user + "(Section_no
INT,Section_name varchar(150)," \
                                "Note_no INt,Note_name Varchar(150)," \
                                "Delete_note varchar(10)) "
    my_c.execute(execute_code_2_ep)

        sql_ep_entry_code_2 = "insert into " + account_no_user + " values(%s,%s,%s,%s,
%s)"
    print(sql_ep_entry_code_2)
    for tuple_data_entry_ep in list_ep_entry:
```

```
            value_data_ep_2 = tuple()
            for tuple_data_write_2 in tuple_data_entry_ep:
                value_data_ep_2 = value_data_ep_2 + (tuple_data_write_2,)
            print("writing into database", account_no_user)
            print(tuple_data_entry_ep, value_data_ep_2)
            my_c.execute(sql_ep_entry_code_2, value_data_ep_2)

        print("End of AIY note program.")


def donothing():
    # function for trial work
    filewin = Toplevel(displaywindow)
    button = Button(filewin, text="Do nothing button", pady=4)
    button.place(x=50, y=50)


def widgets_destroy(widget_list):
    # destroys widgets through a given list for a window
    for widget_1 in widget_list:
        widget_1.destroy()


def restart(window, list_widgets):
    # destroys widgets through a given list for a window and restarts the maindisplay window
    if window == "maindisplay":
        for widget_2 in list_widgets:
            widget_2.destroy()
        maindisplay()
```

```
    else:
        pass


def functions_panel_features(purpose_, list_widgets=None, direct_not=None):
    # program to add, recovery, about, remove within this
    # program for the feature panel
    def widget_shown_destroy(widget=None):
        # to check if a particular widget is shown or not
        if widget is not None:
            if widget.IsShown():
                print(widget + " is shown")
            else:
                print(widget + " is not shown")
        else:
            pass

    widget_shown_destroy()

    purpose_title = ["+ Add", "Recovery", "About", "- Remove"]
    if purpose_ in purpose_title:
        # feature_panel
        global feature_panel
                feature_panel = PanedWindow(displaywindow, orient="vertical", height=800,
width=150, bd=2, bg="#999999")
        feature_panel.pack(side=RIGHT, expand="False")
        feature_panel.configure(sashrelief=FLAT)

        if list_widgets is not None:
```

```
        global list_widgets_next
        list_widgets.append(feature_panel)
        list_widgets_next = list_widgets
    else:
        pass

    _Title_ = Label(feature_panel, text=purpose_, font="calbri 16", bg="#999999")
    _Title_.pack(side=TOP)
    feature_panel.add(_Title_)

    if purpose_ == "+ Add":
        print("462", activelog_section_note_list)
        section_current_ = activelog_section_note_list[-1][0]
            current_section_ = Label(feature_panel, text="Current Section:", font="calbri 16",
pady=4, bg="#999999")
                    _current_section_displayed = Label(feature_panel, text=section_current_,
font="calbri 16", pady=4,
                                bg="#999999")

        current_section_.pack(side='top')
        feature_panel.add(current_section_)
        _current_section_displayed.pack(side="top")
        feature_panel.add(_current_section_displayed)

        section_add = Button(feature_panel, text="+ Section", pady=4,
                                                                command=lambda:
[text_frame_configure('data_textframe_into_notefile'),
                                text_frame_configure('clear_textframe'),
                                feature_panel_activity(False),
                                modify_panel('section', list_widgets_next),
```

```
                                text_frame_configure('clear_textframe'),
                                text_frame_configure('data_notefile_into_textframe')])

    note_add = Button(feature_panel, text="+ Note", pady=4,
                      command=lambda: [text_frame_configure('data_textframe_into_notefile'),
                                       text_frame_configure('clear_textframe'),
                                       feature_panel_activity(False),
                                       modify_panel('note', list_widgets_next),
                                       text_frame_configure('clear_textframe'),
                                       text_frame_configure('data_notefile_into_textframe')])

    section_add.pack(side="top")
    feature_panel.add(section_add)
    note_add.pack(side="top")
    feature_panel.add(note_add)

elif purpose_ == '- Remove':
    def delete_section_note(purpose_delete_section_note):
        # program to delete or remove a note or section from the screen
        global deleted_section_list, deleted_notes_section_dict

        def delete(purpose_2, list_widgets_2):
            # program to delete or remove a note or section from the screen
            if combobox_.get() != "":
                if purpose_2 == "section":
                    deleted_section_list.append(combobox_.get())

                    # return to previous section note
                    active_section_note = ''
```

```
                active_section_sec = ''
                if active_section_note == '':
                    for sec in entrylog_section:
                        if sec not in deleted_section_list:
                            active_section_sec = sec
                            break
                        else:
                            continue
                    note_list_sec = dict_section_note[active_section_sec]
                    deleted_note_list = deleted_notes_section_dict[active_section_sec]
                    for ns in note_list_sec:
                        if ns not in deleted_note_list:
                            active_section_note = [active_section_sec, ns]
                            break
                        else:
                            continue
                activelog_section_note_list.append(active_section_note)

            elif purpose_2 == "note":
                present_section_delete = present('find_section')
                list_notes = deleted_notes_section_dict[present_section_delete]
                if len(list_notes) >= 0:
                    if len(list_notes) == 0:
                                        deleted_notes_section_dict[present_section_delete] =
[combobox_.get()]
                    elif len(list_notes) > 0:
                        list_notes.append(combobox_.get())
                        deleted_notes_section_dict[present_section_delete] = list_notes
```

```
            # return to previous section note
            active_section_note_2 = ''
            if active_section_note_2 == '':
                active_sec = ''
                if present_section_delete not in deleted_section_list:
                    active_sec = present_section_delete
                elif present_section_delete in deleted_section_list:
                    for es in entrylog_section:
                        if es not in deleted_section_list:
                            active_sec = es
                            break
                        else:
                            continue
                note_list_sec = dict_section_note[active_sec]
                deleted_note_list = deleted_notes_section_dict[active_sec]

                for ns_2 in note_list_sec:
                    if ns_2 not in deleted_note_list:
                        active_section_note_2 = [active_sec, ns_2]
                        break
                    else:
                        continue
            else:
                print("active_section_note_2 not responsive/working.")
            activelog_section_note_list.append(active_section_note_2)

        else:
            print("not working purpose_2 loop")
    else:
```

```
        print("not working entire delete function ")
    restart("maindisplay", list_widgets_2)


# conditions for proceeding forward
if purpose_delete_section_note == "section":
    name_label = "Section Deletion"
    instructions = "Please select the section from below:"
    list_sections = list()
    if len(deleted_section_list) != 0:
        for s in entrylog_section:
            if s not in deleted_section_list:
                list_sections.append(s)
            else:
                pass
        value_combobox = list_sections
    else:
        value_combobox = entrylog_section
elif purpose_delete_section_note == "note":
    name_label = "Note Deletion"
    instructions = "Please select the note from below:"
    present_section = present('find_section')
    list_notes_unverified = dict_section_note[present_section]
    list_notes_verified = list()
    if len(deleted_notes_section_dict[present_section]) != 0:
        for n in list_notes_unverified:
            if n not in deleted_notes_section_dict[present_section]:
                list_notes_verified.append(n)
            else:
                pass
```

```
            else:
                list_notes_verified = list_notes_unverified
            value_combobox = list_notes_verified
        else:
            name_label = "XYZ"
            instructions = "XYZ"
            value_combobox = []

        if len(value_combobox) > 1:
            # separator widget
            separator_ = ttk.Separator(feature_panel, orient='horizontal')
            separator_.pack(side="top")
            feature_panel.add(separator_)

                        label_name = Label(feature_panel, text=name_label, font="calbri 16",
bg='#999999')
            label_name.pack(side="top")
            feature_panel.add(label_name)

            if purpose_delete_section_note == "note":
                # extra label for section verification
                present_section = present('find_section')
                text_selected_section = "Section selected: " + str(present_section)
                                select_section = Label(feature_panel, text=text_selected_section,
wraplength=100, bg='#999999')
                # placement of widget
                select_section.pack(side="top")
                feature_panel.add(select_section)
            else:
                pass
```

```
                label_instruction = Label(feature_panel, text=instructions, wraplength=100,
bg='#999999')
            _Next_ = Button(
               feature_panel, text="Next", pady=4, command=lambda:
               [
                  text_frame_configure('data_textframe_into_notefile'),
                  feature_panel_activity(False),
                  delete(purpose_delete_section_note, list_widgets_next),
                  text_frame_configure('data_notefile_into_textframe')
               ]
            )
            _Back_ = Button(
               feature_panel, text="Back", pady=4, command=lambda:
               [
                  text_frame_configure('data_textframe_into_notefile'),
                  feature_panel_activity(False),
                  restart("maindisplay", list_widgets_next)
               ]
            )
            combobox_ = ttk.Combobox(feature_panel, text="Options")
            combobox_["values"] = value_combobox

            # widget placement
            label_instruction.pack(side="top")
            combobox_.pack(side="top")
            _Next_.pack(side="right")
            _Back_.pack(side="left")
```

```
            feature_panel.add(label_instruction)
            feature_panel.add(combobox_)
            feature_panel.add(_Next_)
            feature_panel.add(_Back_)

            label_pack_ = Label(feature_panel, text="", bg="#999999")
            label_pack_.pack(fill=X)
            feature_panel.add(label_pack_)

        elif len(value_combobox) == 1 and purpose_delete_section_note == "sections":
                    text_messagebox = "The application suggest that at least one " + purpose_delete_section_note + \
                    " be kept undeleted for use."
            messagebox.showerror("AIY Note", text_messagebox)
        else:
            print("len(value_combobox) : ", len(value_combobox))
                    text_messagebox = "The application suggest that at least one " + purpose_delete_section_note + \
                    " be kept undeleted for use."
            messagebox.showerror("AIY Note", text_messagebox)

    if direct_not == "direct_section":
        delete_section_note("section")
    elif direct_not == "direct_note":
        delete_section_note("note")
    elif direct_not is None:
        text_remove_label = "Select what you want to delete:"
                remove_labal_instruction = Label(feature_panel, text=text_remove_label, wraplength=100, pady=4,
                                 bg="#999999")
```

```
        remove_labal_instruction.pack(side='top')
        feature_panel.add(remove_labal_instruction)

        section_minus = Button(feature_panel, text="- Section", pady=4,
                               command=lambda: delete_section_note("section"))
        note_minus = Button(feature_panel, text="- Note", pady=4,
                            command=lambda: delete_section_note("note"))

        section_minus.pack(side="top")
        feature_panel.add(section_minus)
        note_minus.pack(side="top")
        feature_panel.add(note_minus)
    else:
        pass

elif purpose_ == "Recovery":
    def _recovery_function(section_note):
        # program that helps in recovering deleted or removed notes and section
        proceed_recovery = None
        if section_note == "section":
            if len(deleted_section_list) == 0:
                text_messagebox = "There is currently no section to be recovered."
                messagebox.showerror("AIY Note", text_messagebox)
                proceed_recovery = False
            else:
                proceed_recovery = True
        elif section_note == "note":
            present_section_rf = present('find_section')
            list_notes_ = deleted_notes_section_dict[present_section_rf]
```

```
            if len(list_notes_) == 0:
                text_messagebox = "There is currently no note to be recovered."
                messagebox.showerror("AIY Note", text_messagebox)
                proceed_recovery = False
            else:
                proceed_recovery = True


        def recover(purpose_2, list_widgets_2):
            # program that helps in recovering deleted or removed notes and section
            if combobox_.get() != "":
                if purpose_2 == "section":
                    deleted_section_list.remove(combobox_.get())
                    re_active_section_note = [combobox_.get(), 'note_1']
                    activelog_section_note_list.append(re_active_section_note)
                elif purpose_2 == "note":
                    present_section_r = present('find_section')
                    list_note = deleted_notes_section_dict[present_section_r]
                    if combobox_.get() in list_note:
                        list_note.remove(combobox_.get())
                        deleted_notes_section_dict[present_section_r] = list_note
                                        activelog_section_note_list.append([present_section_r,
combobox_.get()])
                    else:
                        print("not working purpose_2 loop")
            else:
                print("not working entire delete function ")
            restart("maindisplay", list_widgets_2)


        if proceed_recovery is True:
```

```
            # separator widget
            separator_ = ttk.Separator(feature_panel, orient='horizontal')
            separator_.pack(side="top")
            feature_panel.add(separator_)

            if section_note == "section":
                    section_recovery_label = Label(feature_panel, text="Section Recovery",
wraplength=100,
                                                   bg="#999999")
                section_text_label = Label(feature_panel, text="Select the section to recover:",
wraplength=100,
                                           bg="#999999")

                section_recovery_label.pack(side="top")
                section_text_label.pack(side="top")
                feature_panel.add(section_recovery_label)
                feature_panel.add(section_text_label)
                value_combobox = deleted_section_list

            elif section_note == "note":
                    note_recovery_label = Label(feature_panel, text="Section Recovery",
wraplength=100,
                                                bg="#999999")
                print("748", activelog_section_note_list)
                section_current_2 = activelog_section_note_list[-1][0]
                    current_section_text_label = Label(feature_panel, text="Current section:",
bg="#999999")
                    current_section_label = Label(feature_panel, text=section_current_2,
bg="#999999")
                note_text_label = Label(feature_panel, text="Select the note to recover:",
wraplength=100,
```

```
                    bg="#999999")

    note_recovery_label.pack(side="top")
    current_section_text_label.pack(side="top")
    current_section_label.pack(side="top")
    note_text_label.pack(side="top")

    feature_panel.add(note_recovery_label)
    feature_panel.add(current_section_text_label)
    feature_panel.add(current_section_label)
    feature_panel.add(note_text_label)
    value_combobox = deleted_notes_section_dict[section_current_2]

_Next_ = Button(
    feature_panel, text="Next", pady=4, command=lambda:
    [
        text_frame_configure('data_textframe_into_notefile'),
        feature_panel_activity(False),
        recover(section_note, list_widgets_next),
        text_frame_configure('data_notefile_into_textframe')
    ]
)

_Back_ = Button(
    feature_panel, text="Back", pady=4, command=lambda:
    [
        text_frame_configure('data_textframe_into_notefile'),
        feature_panel_activity(False),
        restart("maindisplay", list_widgets_next),
```

```
                text_frame_configure('data_notefile_into_textframe')
            ]
        )
        combobox_ = ttk.Combobox(feature_panel, text="Options")
        combobox_["values"] = value_combobox

        label_recover_text = Label(feature_panel, text="", bg='#999999')

        combobox_.pack(side="top")
        _Back_.pack(side="top")
        _Next_.pack(side="top")
        label_recover_text.pack(fill=X)

        feature_panel.add(combobox_)
        feature_panel.add(_Back_)
        feature_panel.add(_Next_)
        feature_panel.add(label_recover_text)

    else:
        pass

text_recovery = "Select what you want to recover:"
        recovery_label = Label(feature_panel, text=text_recovery, wraplength=100, bg="#999999", font="calbri 16")
section_recover = Button(
                    feature_panel, text="Section", pady=4, command=lambda: _recovery_function("section")
)
        note_recover = Button(feature_panel, text='Note', pady=4, command=lambda: _recovery_function("note"))
```

```
        recovery_label.pack(side="top")
        section_recover.pack(side="top")
        note_recover.pack(side="top")

        feature_panel.add(recovery_label)
        feature_panel.add(section_recover)
        feature_panel.add(note_recover)

    elif purpose_ == 'About':
        text_about = (
                "This is a computer project undertaken by a few students of The Foundation School, Gunjur,Bangalore."
            "Our aim is to provide a note taking application using python, SQL database and text files."
        )
        about_label = Label(feature_panel, text=text_about, wraplength=100, bg='#999999')
        about_label.pack(side="top")
        feature_panel.add(about_label)
    else:
        pass

    _Quit_ = Button(
                                    feature_panel, text="Quit", pady=4, command=lambda: [feature_panel_activity(False), feature_panel.destroy()]
    )
    _Quit_.pack(side="top")
    feature_panel.add(_Quit_)

    label_pack = Label(feature_panel, text="", bg="#999999")
```

```
        label_pack.pack(fill=X)
        feature_panel.add(label_pack)
    elif purpose_ == "destroy_feature_panel":
        feature_panel.destroy()
    else:
        pass


def feature_panel_activity(boolian, purpose_=None, list_widgets=None, direct_not=None):
    # program to keep track of the use going to take place as well as the present use
    global active_feature_panel, active_button_feature_panel
    active_feature_panel = boolian

    if active_feature_panel is False:
        active_button_feature_panel = []
    else:
        pass

    if active_feature_panel is True:
        if len(active_button_feature_panel) == 0:
            if purpose_ is not None:
                active_button_feature_panel.append(purpose_)
                if direct_not is not None:
                    functions_panel_features(purpose_, list_widgets, direct_not)
                else:
                    functions_panel_features(purpose_, list_widgets)
            else:
                pass
```

```
        elif len(active_button_feature_panel) > 0:
            functions_panel_features("destroy_feature_panel")
            active_button_feature_panel.pop()
            active_button_feature_panel.append(purpose_)
            if direct_not is not None:
                functions_panel_features(purpose_, list_widgets, direct_not)
            else:
                functions_panel_features(purpose_, list_widgets)
        else:
            pass
    else:
        pass



def present(purpose, add_what_section_note=None, section_note_name=None):
    # program that helps in knowing which note of which section is being displayed on the screen
    global activelog_section_note_list
    if purpose == "find_section":
        return activelog_section_note_list[-1][0]
    elif purpose == "find_note":
        return activelog_section_note_list[-1][1]


    elif purpose == "add":
        if add_what_section_note is not None:
            if add_what_section_note == "section":
                activelog_section_note_list.append([section_note_name, 'note_1'])
            elif add_what_section_note == "note":
                activelog_section_note_list.append([present('find_section'), section_note_name])
```

```
        else:
            pass
    else:
        pass

    # Maintainence of activelog_log
    if len(activelog_section_note_list) > 3:
        activelog_section_note_list.pop(0)
    else:
        pass


def text_frame_configure(purpose_tfc, present_section=None, present_note_section=None):
    # program for the working of the text frame and its features
    print("text_frame_configure")
    print("906", activelog_section_note_list)
    print("present_section, present_note_section : ", present_section, present_note_section)

    def text_file_existance(path):
        # program to check if path of text file exist
        try:
            f = open(path, "r")
            text_file_data_ = f.read()
            print(text_file_data_)
            file_accessible = True
            f.close()
        except IOError:
            print("File not accessible")
            file_accessible = False
```

```
    return file_accessible


def textframe_data_notefile_interchange(purpose_tdni, note_directory=None):
    # program that helps in either appending data from text frame to text file or vice versa
                        list_file_existance_option = ["textframe_data_into_notefile",
'notefile_data_into_textframe']
    print("textframe_data_notefile_interchange purpose: ", purpose_tdni)
    print("note_directory: ", note_directory)
    if purpose_tdni in list_file_existance_option:
        file_accessible = text_file_existance(note_directory)
        print('file_accessible: ', file_accessible)
        if file_accessible is not None:
            if file_accessible is False:
                with open(note_directory, 'a') as f:
                    f.write("")
                f.close()
            else:
                pass
        else:
            pass


        if purpose_tdni == "notefile_data_into_textframe":
            print("starting to extract data from notefile")
            with open(note_directory, "r") as file_note:
                data_infile = file_note.read()
                print("data_infile : ", data_infile)
                print("data extracted")
                my_text.insert(END, data_infile)
                print("data inserted into text frame")
```

```
            file_note.close()


        elif purpose_tdni == "textframe_data_into_notefile":
            print("starting to extract data from textframe")
            textframe_data = my_text.get('1.0', END).splitlines()
            print("textframe_data: ", textframe_data)
            print("data extracted")
            textframe_new_data = list()
            with open(note_directory, "w") as file_note:
                for textd in textframe_data:
                    if len(textd) != 0:
                        new_data = textd + "\n"
                    else:
                        new_data = textd
                    textframe_new_data.append(new_data)
                print("textframe_new_data: ", textframe_new_data)
                file_note.writelines(textframe_new_data)
                print("data inserted into notefile")
            file_note.close()


    elif purpose_tdni == "delete_textframe":
        my_text.delete('1.0', 'end')
    else:
        print("not working")


if purpose_tfc == 'data_notefile_into_textframe':
    print("969", activelog_section_note_list)
    if present_section is None:
        present_section = activelog_section_note_list[-1][0]
```

```
        else:
            pass

        if present_note_section is None:
            present_note_section = activelog_section_note_list[-1][1]
        else:
            pass

        dict_note = directory_section_note_dict[present_section]
        note_directory_present = \
             dict_note[present_note_section] + "/" + present_section + " " + present_note_section
+ ".txt"
                                textframe_data_notefile_interchange('notefile_data_into_textframe',
note_directory_present)

    elif purpose_tfc == "data_textframe_into_notefile":
        print("985", activelog_section_note_list)
        if present_section is None:
            present_section = activelog_section_note_list[-1][0]
        else:
            pass

        if present_note_section is None:
            present_note_section = activelog_section_note_list[-1][1]
        else:
            pass

        dict_note = directory_section_note_dict[present_section]
        note_directory_present = \
```

```
            dict_note[present_note_section] + "/" + present_section + " " + present_note_section
+ ".txt"
                                textframe_data_notefile_interchange('textframe_data_into_notefile',
note_directory_present)


    elif purpose_tfc == "clear_textframe":
        textframe_data_notefile_interchange("delete_textframe")



def modify_panel(section_note, list_widgets):
    # program to track and modify the section and note panel
    global no_section, no_note
    count = 0
    if section_note == "section":
        for es in entrylog_section:
            if es not in deleted_section_list:
                count += 1
            else:
                pass
    elif section_note == "note":
        _section_name_ = present("find_section")
        _list_notes_ = dict_section_note[_section_name_]
        for ln in _list_notes_:
            if ln not in deleted_notes_section_dict[_section_name_]:
                count += 1
    else:
        pass


    if count == 20:
        section_note_proceed = False
```

```
    text_message = "Application has a limit of only 20 " + section_note + \
                   "s to be kept in use. If you want to add a new " \
                   + section_note + "s you must delete an existing section."
    messagebox.showerror("AIY Note", text_message)
else:
    section_note_proceed = True
    pass

if section_note_proceed is True:
    if section_note == 'section':
        no_section += 1
        section_name_2 = "section_" + str(no_section)
        dict_section_note[section_name_2] = ['note_1', 'note_2']
        activelog_section_note_list.append([section_name_2, 'note_1'])
        restart("maindisplay", list_widgets)
    elif section_note == 'note':
        section_name_2 = present("find_section")
        note_list = dict_section_note[section_name_2]
        no_notes = len(note_list)
        new_note = "note_" + str(no_notes + 1)
        note_list.append(new_note)
        dict_section_note[section_name_2] = note_list
        activelog_section_note_list.append([section_name_2, new_note])
        restart("maindisplay", list_widgets)
    else:
        pass
else:
    pass
```

```
def directory_section_note(purpose):
    # program that helps in creation of folders in which text files are stored and well as their directories
    # section_note path creation
    # require - account_no_user of the user
    def directory_creator_finder(path):
        if not os.path.exists(path):
            os.makedirs(path)
        else:
            pass

    if purpose == "creation of folder directory":
        if account_no_user is not None:
            for dsn in dict_section_note:
                note_dict = dict()
                for dn in dict_section_note[dsn]:
                    new_path = "AIY/" + str(account_no_user) + "/" + str(dsn) + "/" + str(dn)
                    if new_path not in directory_path_entry_log:
                        directory_path_entry_log.append(new_path)
                    else:
                        pass
                    note_dict[dn] = new_path
                    directory_section_note_dict[dsn] = note_dict
                    directory_creator_finder(new_path)
        else:
            pass
```

```
def maindisplay():
    # program that helps in the working and tracking of the maindisplay window
    print("maindisplay activate")
    print("username_", username_)
    print("account_no_user", account_no_user)

    # sql_part for section and note list
    print("1091", activelog_section_note_list)
    sql_part("maindisplay_entry", None)
    print("1093", activelog_section_note_list)

    # main display
    displaywindow.title("XYZ's Handbook")
    displaywindow.geometry('1000x1000')
    displaywindow["bg"] = '#000000'
    displaywindow.resizable(True, True)

    # function panel bg ="#4d4d4d"
        function_panel = PanedWindow(displaywindow, orient="horizontal", height=60, width=150, bd=4, bg="#000000")
    function_panel.pack(side=TOP, expand=False, fill=X)

    # section panel bg="#999999"
     section_panel = PanedWindow(displaywindow, orient='vertical', height=800, width=150, bd=2, bg="#999999")
    section_panel.pack(side=LEFT, expand="False")
    section_panel.configure(sashrelief=FLAT)

    # note panel bg="#bfbfbf"
```

```
    note_panel = PanedWindow(displaywindow, orient='vertical', height=800, width=150, bd=2, bg="#bfbfbf")
    note_panel.pack(side=LEFT, expand="False")
    note_panel.configure(sashrelief=FLAT)

    # text frame
    text_frame = Frame(displaywindow)
    text_frame.pack(side=RIGHT, expand=True, fill=BOTH)

    # Text scrollbar
    text_scroll_y = Scrollbar(text_frame, orient='vertical')
    text_scroll_y.pack(side=RIGHT, fill=Y)

    text_scroll_x = Scrollbar(text_frame, orient='horizontal')
    text_scroll_x.pack(side="bottom", fill=X)

    # Text box
    global my_text
            my_text = Text(text_frame, width=50, height=100, font="calbri 16", selectbackground="#b3ccff",
             selectforeground="black", undo=True, yscrollcommand=text_scroll_y.set,
             xscrollcommand=text_scroll_x.set)
    my_text.pack(expand=True, fill=BOTH)

    # configuration of the scrollbar
    text_scroll_x.configure(command=my_text.xview)
    text_scroll_y.configure(command=my_text.yview)

    global list_maindisplay_widgets
    list_maindisplay_widgets = [section_panel, note_panel, text_frame, function_panel]
```

```
    # image_2
    logo_label = Label(displaywindow, image=img_3, bg='#000000')
    logo_label.place(relx=0.001, rely=0.001)

    separator_1 = ttk.Separator(function_panel, orient='vertical')
    separator_1.place(relx=0.105, rely=0, relheight=1)

        account_name_label = Label(function_panel, text="Username : ", bg="#000000", foreground="white")
         _account_name_display = Label(function_panel, text=username_, bg="#000000", foreground="white")
         account_no_label = Label(function_panel, text="Account No: ", bg="#000000", foreground="white")
       _account_no_display = Label(function_panel, text=account_no_user, bg="#000000", foreground="white")

    account_name_label.place(relx=0.11, rely=0.1)
    _account_name_display.place(relx=0.195, rely=0.1)
    account_no_label.place(relx=0.11, rely=0.5)
    _account_no_display.place(relx=0.195, rely=0.5)

    separator_2 = ttk.Separator(function_panel, orient='vertical')
    separator_2.place(relx=0.265, rely=0, relheight=1)

    print("1157", activelog_section_note_list)
    current_displayed_section = activelog_section_note_list[-1][0]
    current_displayed_note = activelog_section_note_list[-1][1]
        current_section = Label(function_panel, text="Current Section: ", bg="#000000", foreground="white")
```

```
        current_note = Label(function_panel, text="Current Note: ", bg="#000000", foreground="white")
    _section_current_ = Label(function_panel, text=current_displayed_section, bg="#000000", foreground="white")
        _note_current_ = Label(function_panel, text=current_displayed_note, bg="#000000", foreground="white")


    current_section.place(relx=0.27, rely=0.1)
    _section_current_.place(relx=0.38, rely=0.1)
    current_note.place(relx=0.27, rely=0.5)
    _note_current_.place(relx=0.38, rely=0.5)


    separator_3 = ttk.Separator(function_panel, orient='vertical')
    separator_3.place(relx=0.46, rely=0, relheight=1)


    save_ = Button(function_panel, text="Save", pady=4,
                command=lambda: [text_frame_configure("data_textframe_into_notefile"),
                                 text_frame_configure('clear_textframe'),
                                 text_frame_configure('data_notefile_into_textframe'),
                                 sql_part("exit_program", None)])
    quit_ = Button(function_panel, text="Quit", pady=4,
                command=lambda: [text_frame_configure("data_textframe_into_notefile"),
                                 text_frame_configure('clear_textframe'),
                                 text_frame_configure('data_notefile_into_textframe'),
                                 sql_part("exit_program", None),
                                 displaywindow.destroy()])
    add_ = Button(function_panel, text=" + Add", pady=4,
                                       command=lambda: [feature_panel_activity(True, "+ Add", list_maindisplay_widgets)])
    remove_ = Button(function_panel, text="- Remove", pady=4,
```

```
                                    command=lambda: [feature_panel_activity(True, "- Remove",
list_maindisplay_widgets)])
    recover_ = Button(function_panel, text="Recover", pady=4,
                                     command=lambda: [feature_panel_activity(True, "Recovery",
list_maindisplay_widgets)])
    about_ = Button(function_panel, text="About", pady=4,
                                        command=lambda: [feature_panel_activity(True, "About",
list_maindisplay_widgets)])


    save_.place(rely=0.1, relx=0.475)
    quit_.place(rely=0.1, relx=0.535)
    add_.place(rely=0.1, relx=0.59)
    remove_.place(rely=0.1, relx=0.66)
    recover_.place(rely=0.1, relx=0.75)
    about_.place(rely=0.1, relx=0.83)


    def section_note_formation(section_or_note, text_name, present_name):
        # program that helps in formation of notes and sections
        print("1204", activelog_section_note_list)
        global entrylog_section, entrylog_note
        if section_or_note == "section":
            section_i = Button(
                section_panel, text=text_name, pady=4, command=lambda:
                [text_frame_configure('data_textframe_into_notefile'),
                 text_frame_configure('clear_textframe'),
                 present("add", "section", present_name),
                 restart("maindisplay", list_maindisplay_widgets),
                 text_frame_configure('clear_textframe'),
                 text_frame_configure('data_notefile_into_textframe')
                 ]
```

```
    )
    section_i.pack(side=TOP)
    section_panel.add(section_i)

    if present_name not in entrylog_section:
        entrylog_section.append(present_name)
        deleted_notes_section_dict[present_name] = []
    else:
        pass
elif section_or_note == "note":
    note_i = Button(
        note_panel, text=text_name, pady=4, command=lambda:
        [text_frame_configure('data_textframe_into_notefile'),
         text_frame_configure('clear_textframe'),
         present("add", "note", present_name),
         restart("maindisplay", list_maindisplay_widgets),
         text_frame_configure('clear_textframe'),
         text_frame_configure('data_notefile_into_textframe')
         ]
    )
    note_i.pack(side=TOP)
    note_panel.add(note_i)

    if present_name not in entrylog_note:
        entrylog_note.append(present_name)
    else:
        pass
else:
    pass
```

```
# section panel
# buttons and labels
l_section = Label(section_panel, text="Section", font="calbri 16", bg="#999999")
l_section.pack(side="top")
section_panel.add(l_section)

# section formation
for no_s in range(0, no_section):
    name_section_text = "Section " + str(no_s + 1)
    name_section_present = "section_" + str(no_s + 1)
    if len(deleted_section_list) != 0:
        if name_section_present not in deleted_section_list:
            section_note_formation("section", name_section_text, name_section_present)
        else:
            pass
    else:
        section_note_formation("section", name_section_text, name_section_present)

# To add new sections
section_add = Button(section_panel, text="+ Section", pady=4,
                     command=lambda: [text_frame_configure('data_textframe_into_notefile'),
                                      text_frame_configure('clear_textframe'),
                                      modify_panel('section', list_maindisplay_widgets),
                                      text_frame_configure('clear_textframe'),
                                      text_frame_configure('data_notefile_into_textframe')])

section_add.pack(side='bottom')
section_panel.add(section_add)
```

```
section_subtract = Button(
    section_panel, text="- Section", pady=4, command=lambda:
    [feature_panel_activity(True, "- Remove", list_maindisplay_widgets, "direct_section")]
)
section_subtract.pack(side='bottom')
section_panel.add(section_subtract)

label_section = Label(section_panel, text="", bg="#999999")
label_section.pack(fill=X)
section_panel.add(label_section)

# Note's panel
# buttons and labels
l_note = Label(note_panel, text="Notes", font="calbri 16", bg="#bfbfbf")
l_note.pack(side="top")
note_panel.add(l_note)

# note formation
section_name_1 = present("find_section")
note_list = dict_section_note[section_name_1]
print("section active: ", section_name_1)
print("no_note_section: ", note_list)

for lnote in note_list:
    list_number = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "0"]
    new_string = ""
    for jn in lnote:
        if jn in list_number:
```

```
            new_string += str(jn)
        else:
            pass
    name_note_text = "Note " + str(new_string)
    name_note_present = lnote
    print(section_name_1)
    print(deleted_notes_section_dict)

    if len(deleted_notes_section_dict[section_name_1]) != 0:
        list_notes_section_del = deleted_notes_section_dict[section_name_1]
        if name_note_present not in list_notes_section_del:
            section_note_formation("note", name_note_text, name_note_present)
        else:
            pass
    else:
        section_note_formation("note", name_note_text, name_note_present)

# To add new notes
note_add = Button(note_panel, text="+ Note", pady=4,
                  command=lambda: [text_frame_configure('data_textframe_into_notefile'),
                                   text_frame_configure('clear_textframe'),
                                   modify_panel('note', list_maindisplay_widgets),
                                   text_frame_configure('data_notefile_into_textframe')])
note_add.pack(side='bottom')
note_panel.add(note_add)

note_subtract = Button(note_panel, text="- Note", pady=4,
                                    command=lambda: [feature_panel_activity(True, "- Remove",
list_maindisplay_widgets,
```

```
                                                "direct_note")])
    note_subtract.pack(side='bottom')
    note_panel.add(note_subtract)

    l_note_empty = Label(note_panel, text="", bg="#bfbfbf")
    l_note_empty.pack(fill=X)
    note_panel.add(l_note_empty)

    directory_section_note('creation of folder directory')
    print("1337", activelog_section_note_list)

    def start_note_text_frame(count):
        # program that helps in section note text frame working and tracking
        if count == 0:
            global activelog_section_note_list
            print("1343", activelog_section_note_list)
             text_frame_configure("data_notefile_into_textframe", activelog_section_note_list[0]
[0],
                                 activelog_section_note_list[0][1])
        else:
            pass

    # Keeping track of  maindisplay_function_activity
    global count_maindisplay_function_activated
    start_note_text_frame(count_maindisplay_function_activated)
    count_maindisplay_function_activated += 1



def activation_procedure_1(window, list_entry):
```

```
    # program that checks if each entry of an particular window is filled or not
    print("1374", activelog_section_note_list)
    global proceed_activation
    if len(list_entry) > 0:
        if window == "displaywindow":
            for ile in list_entry:
                if ile.get() == "":
                    proceed_activation = False
                    messagebox.showerror("AIY Note", "Please fill the form completely.")
                    break
                else:
                    proceed_activation = True
        else:
            for ile_ in list_entry:
                if ile_ == "":
                    proceed_activation = False
                    messagebox.showerror("AIY Note", "Please fill the form completely.")
                    break
                else:
                    proceed_activation = True
    else:
        proceed_activation = False



def activation_procedure_2(window, list_entry_2):
        # program that helps in instructing which function to call upon while proceeding from certain windows
    print("activation_procedure_2 activate")
    # present window closes if proceed = True and the respective next window opens up
```

```
if proceed_activation is True:
    # for trial run of whole system and real login window
    if window == "login_displaywindow":
        print("list_entry activation procedure", list_entry_2)
        if list_entry_2[0] == "AIY_login":
            if list_entry_2[1] == "AIY_password":
                global username_, account_no_user
                username_ = "XAY"
                account_no_user = "AIY0001"
                maindisplay()
        else:
            print("seaching through SQL database")
            loginwindow_proceed = sql_part('loginwindow', list_entry_2)
            if loginwindow_proceed is True:
                if account_no_user is not None:
                    maindisplay()
                else:
                    print("sql part not working")
            else:
                messagebox.showerror("AIY Note", "Please try again.")

    elif window == "account_creation_window":
        sql_part("account_creation", list_entry_2)

    elif window == "forget account":
        sql_part('forget_account', list_entry_2)
else:
    print("Not proceeding further (activation procedure)")
```

```
# email typed or not verification for otp system and beyond
def verification_email_entry(window, username, email_id, purpose, button_destroy=None):
    # program that checks if email entry and username_ entry is filled or not
    if window == "forget_account" or window == "account_creation_window":
        if email_id != "" and username != "":
            otp_system(username, email_id, purpose)
            if button_destroy is not None:
                button_destroy.destroy()
            else:
                pass
                # it means that no button is getting destroyed
        elif email_id != "" or username != "":
            messagebox.showerror("AIY Note", "Please fill in the email id part and username and
retry")
        else:
            messagebox.showerror("AIY Note", "Please fill in the email id part and username and
retry")
    else:
        messagebox.showerror("AIY Note", "Error")



def otp_system(username_receiver, email_id, otp_purpose):
    # program that helps in sending otps through email and keeping track of pincode
    username = username_receiver
    email_reciever = email_id
    sender_email = "
    password = ""


    # pin_code segment
```

```
    global pincode
    pincode = 0
    pincode = str(random.randint(1000, 9999))

    print("starting to send email")
    with smtplib.SMTP('smtp.gmail.com', 587) as smtp:
        smtp.ehlo()
        smtp.starttls()

        smtp.login(sender_email, password)
        subject = 'AIY Note'

        # message and custom subject segment
        if otp_purpose == 'account_activation_otp':
            subject = 'AIY Note Account Activation'
            body = 'Dear ' + username + ',' \
                                    '\n The account activation process has begun. For creation of the
account,' \
                                     ' we require you to enter the below displayed pin code into the
program,' \
                                "\n " + pincode + \
                   '\n Thank you for your time and consideration.' \
                   '\n Thank you,' \
                   '\n AIY Note'
        elif otp_purpose == 'forget_account':
            subject = 'AIY Note Forget Account'
            body = 'Dear ' + username + ',' \
                                      '\n The account recovery has been begun. For authenticity we
require you to ' \
                                'enter the below display pin code into the program,' \
```

```
                    "\n " + pincode + \
            '\n Thank you for your time and consideration.' \
            '\n Thank you,' \
            '\n AIY Note'
    elif otp_purpose == "retry_otp":
        subject = 'AIY Note New Pin Code'
        body = 'Dear ' + username + ',' \
                    '\n The new pin code given below, please try to re-entry the new code.' \
                    "\n " + pincode + \
            '\n Thank you for your time and consideration.' \
            '\n Thank you,' \
            '\n AIY Note'

    msg = f'subject:{subject}\n\n{body}'
    smtp.sendmail(sender_email, email_reciever, msg)
    smtp.quit()
  print("email sent")


def loginwindow(back_from_window=None, list_window_widget=None):
  # program that helps in coming back to login window from other windows
  # for future back program buttons on other windows
  if back_from_window is not None:
            back_from_window_list = ["forget_account", "account_creation_window", "forget_message_window"]
    if back_from_window in back_from_window_list:
      if list_window_widget is not None:
        for ilww in list_window_widget:
          ilww.destroy()
```

```
        else:
            pass
    else:
        pass
else:
    pass

displaywindow["bg"] = '#bfbfbf'
# Welcome label
wel = Label(displaywindow, text="Welcome To AIY Note", font="zapfino 16 bold")
signin = Label(displaywindow, text="Sign in", font="calbri 16")
wel.place(relx=0.2)
signin.place(relx=0.3, rely=0.2)

# login and entry
loginid = Label(displaywindow, text="Username:")
loginid.place(relx=0.1, rely=0.3)
loginpassword = Label(displaywindow, text="Password:")
loginpassword.place(relx=0.1, rely=0.4)
# entry widgets
id_loginwindow = Entry(displaywindow)
password_loginwindow = Entry(displaywindow, show="*")

id_loginwindow.place(relx=0.3, rely=0.3)
password_loginwindow.place(relx=0.3, rely=0.4)

# Buttons
next_l = Button(
    displaywindow, text="Next", command=lambda:
```

```
        [
            widgets_destroy(loginwindow_list_widgets),
                                activation_procedure_1('login_displaywindow',  [id_loginwindow.get(),
password_loginwindow.get()]),
                                activation_procedure_2('login_displaywindow',  [id_loginwindow.get(),
password_loginwindow.get()])
        ]
    )


    forgotpassword_l = Button(
        displaywindow, text="Forgot password", command=lambda:
        [widgets_destroy(loginwindow_list_widgets), forget_password()]
    )
    createaccount_l = Button(
        displaywindow, text="Create account", command=lambda:
        [widgets_destroy(loginwindow_list_widgets), account_creation()]
    )


    next_l.place(relx=0.7, rely=0.6)
    forgotpassword_l.place(relx=0.1, rely=0.6)
    createaccount_l.place(relx=0.4, rely=0.6)
    global loginwindow_list_widgets
    loginwindow_list_widgets = (
        wel, signin, loginid, loginpassword, id_loginwindow, password_loginwindow, next_l,
forgotpassword_l,
        createaccount_l)



def account_creation():
    # program that controls the working of create account display window
```

```
    displaywindow["bg"] = '#bfbfbf'

    # destroy login window's widgets
    widgets_destroy(loginwindow_list_widgets)

    # label for creation of new accounts
      account_creation_l = Label(displaywindow, text="Account Creation", font="zapfino 16 bold")
    user_name = Label(displaywindow, text="Username:")
    dob = Label(displaywindow, text="Date of Birth:")
    password = Label(displaywindow, text="Password:")
    confirm_password = Label(displaywindow, text="Confirm password:")
    email_id = Label(displaywindow, text="Email id:")
    ph_no = Label(displaywindow, text="Phone.no:")

    # Entry widgets for creation of new accounts
    global confirm_password_, otp_create_
    user_name_entry = Entry(displaywindow)
    dob_ = Entry(displaywindow)
    password_ = Entry(displaywindow)
    confirm_password_ = Entry(displaywindow)
    email_id_ = Entry(displaywindow)
    ph_no_ = Entry(displaywindow)

    # placement of label and entry widget
    # Label placement
    account_creation_l.place(relx=0.3)
    user_name.place(relx=0.1, rely=0.2)
    dob.place(relx=0.1, rely=0.3)
```

```
password.place(relx=0.1, rely=0.4)
confirm_password.place(relx=0.1, rely=0.5)
email_id.place(relx=0.1, rely=0.6)
ph_no.place(relx=0.1, rely=0.7)

# Entry widget placement
user_name_entry.place(relx=0.4, rely=0.2)
dob_.place(relx=0.4, rely=0.3)
password_.place(relx=0.4, rely=0.4)
confirm_password_.place(relx=0.4, rely=0.5)
email_id_.place(relx=0.4, rely=0.6)
ph_no_.place(relx=0.4, rely=0.7)

# OTP system activation
otp_create = Label(displaywindow, text="OTP Pin:", font="Calbri 14")
otp_create_ = Entry(displaywindow)

# OTP placement
otp_create.place(relx=0.1, rely=0.8)
otp_create_.place(relx=0.4, rely=0.8)

def verification_password_otp(list_verification_password_otp):
    # program that checks if otp and pincode ae same or not
    print("verification_password_otp activate")
    if proceed_activation is True:
        proceed_sql_account_creation_1 = False
        proceed_sql_account_creation_2 = False
        # password verification
        if len(list_verification_password_otp) > 0:
```

```
        if list_verification_password_otp[0] != "":
            if list_verification_password_otp[1] != "":
                print("password, confirm_password :", list_verification_password_otp[0],
                      list_verification_password_otp[1])
                                                    if  str(list_verification_password_otp[0])  !=
str(list_verification_password_otp[1]):
                    confirm_password_.delete(0, END)
                        messagebox.showerror("AIY Note", "Confirm passwword is not equal to
password,"
                                             " please try again.")
                else:
                    proceed_sql_account_creation_1 = True
            else:
                pass
        else:
            pass

        # otp verification
        if list_verification_password_otp[2] != "":
            if list_verification_password_otp[3] != "":
                print("otp_create_, pincode :", list_verification_password_otp[2],
                      list_verification_password_otp[3])
                if list_verification_password_otp[2] != list_verification_password_otp[3]:
                    otp_create_.delete(0, END)
                        messagebox.showerror("AIY Note", "OTP is not equal to assigned OTP,
please try again.")
                else:
                    proceed_sql_account_creation_2 = True
            else:
                pass
```

```
            else:
                pass


             if proceed_sql_account_creation_1 is True and proceed_sql_account_creation_2 is
True:
                print("1646 activation_procedure_2")
                activation_procedure_2(
                    'account_creation_window',
                    [
                        list_verification_password_otp[4][0],
                        list_verification_password_otp[4][1],
                        list_verification_password_otp[4][2],
                        list_verification_password_otp[4][3],
                        list_verification_password_otp[4][4]
                    ]
                )
            else:
                print("verification_password_otp is not working")
        else:
            print("verification_password_otp is not working")
    else:
        print("Not proceeding further (verification_password_otp)")


# buttons for account_creation_window
send_otp_create = Button(
    displaywindow, text="Send otp", command=lambda:
    [
        otp_create_.delete(0, END),
        verification_email_entry(
```

```
            "account_creation_window",
            user_name_entry.get(),
            email_id_.get(),
            'account_activation_otp',
            send_otp_create
        )
    ]
)

_Next_ = Button(
    displaywindow, text="Next", command=lambda:
    [
        activation_procedure_1(
            'account_creation_window',
            [
                user_name_entry.get(),
                dob_.get(),
                password_.get(),
                ph_no_.get(),
                email_id_.get()
            ]

        ),
        verification_password_otp(
            [
                password_.get(),
                confirm_password_.get(),
                otp_create_.get(),
                pincode,
```

```
                [
                    user_name_entry.get(),
                    dob_.get(),
                    password_.get(),
                    ph_no_.get(),
                    email_id_.get()
                ]
            ]
        )
    ]
)

_Resend_pincode_create = Button(
    displaywindow, text="Resend otp", command=lambda:
    [
        otp_create_.delete(0, END),
        verification_email_entry(
            "account_creation_window",
            user_name_entry.get(),
            email_id_.get(),
            'retry_otp'
        )
    ]
)
list_widgets_account_creation = (
        account_creation_l, user_name, dob, password, confirm_password, email_id, ph_no, user_name_entry, dob_,
                password_, confirm_password_, email_id_, ph_no_, otp_create, otp_create_, send_otp_create, _Next_,
    _Resend_pincode_create)
```

```
    back_login = Button(displaywindow, text="Back",
                                                    command=lambda:  [back_login.destroy(),
widgets_destroy(list_widgets_account_creation),
                                    loginwindow("account_creation_window")])

    # buttons placements
    send_otp_create.place(relx=0.25, rely=0.9)
    _Next_.place(relx=0.7, rely=0.9)
    _Resend_pincode_create.place(relx=0.45, rely=0.9)
    back_login.place(relx=0.1, rely=0.9)



def forget_password():
    # program that controls the working of forget account display window
    displaywindow["bg"] = '#bfbfbf'

    # destroy login window's widgets
    widgets_destroy(loginwindow_list_widgets)

    # Label and entry widgets for the forget_account
    # Label widgets
    _Forget_label = Label(displaywindow, text="Forget_account", font="zapfino 16 bold")
    _Username_forget = Label(displaywindow, text="Username:")
    _Last_password_forget = Label(displaywindow, text="Password used: ")
    _DoB_forget = Label(displaywindow, text="Date of birth:")
    ph_no_forget = Label(displaywindow, text="Phone.no:")
    email_forget = Label(displaywindow, text="Email id:")
```

```
# Entry widgets
username_forget_ = Entry(displaywindow)
last_password_forget_ = Entry(displaywindow)
_DoB_forget_ = Entry(displaywindow)
_Ph_no_forget_ = Entry(displaywindow)
email_forget_ = Entry(displaywindow)

# OTP system recover
otp_forget = Label(displaywindow, text="OTP Pin:")
otp_forget_ = Entry(displaywindow)

# Placement of label and entry widgets
# Label segment of forget_account
_Forget_label.place(relx=0.3)
_Username_forget.place(relx=0.1, rely=0.2)
_Last_password_forget.place(relx=0.1, rely=0.3)
_DoB_forget.place(relx=0.1, rely=0.4)
ph_no_forget.place(relx=0.1, rely=0.5)
email_forget.place(relx=0.1, rely=0.6)

# Entry segment of forget_account
username_forget_.place(relx=0.4, rely=0.2)
last_password_forget_.place(relx=0.4, rely=0.3)
_DoB_forget_.place(relx=0.4, rely=0.4)
_Ph_no_forget_.place(relx=0.4, rely=0.5)
email_forget_.place(relx=0.4, rely=0.6)

# OTP segment of forget_account
otp_forget.place(relx=0.1, rely=0.7)
```

```
    otp_forget_.place(relx=0.4, rely=0.7)


    def otp_verify(list_otp_forget=None):
        # program that checks if otp and pincode ae same or not
        if proceed_activation is True:
            global otp_forget_
            if list_otp_forget[0] != "":
                if list_otp_forget[1] != "":
                    if list_otp_forget[0] != list_otp_forget[1]:
                        otp_forget_.delete(0, END)
                            messagebox.showerror("AIY Note", "OTP is not equal to assigned OTP,
please try again.")
                    elif list_otp_forget[0] == list_otp_forget[1]:
                        forget_info = list_otp_forget[2]
                        correct_account_forget_user = sql_part('forget_account', forget_info)
                        if correct_account_forget_user == 'not found':
                            text_messagebox_cafu_f = "The application could not find your account," \
                                                     " either try again or"  \
                                                     "create a new account. "
                            messagebox.showerror("AIY Note", text_messagebox_cafu_f)


                        elif correct_account_forget_user is not None:
                            forget_message_window(correct_account_forget_user)
                            # use message window to display the info of forget account if present
                else:
                    pass
            else:
                messagebox.showerror("AIY Note", "Please fill the form completely.")
        else:
```

```
        print("Not proceeding further (otp_verify)")


global _Retry_pincode
_Retry_pincode = Button(
    displaywindow, text="Resend otp", command=lambda:
    [
        otp_forget_.delete(0, END),
        verification_email_entry(
            "forget_account",
            username_forget_.get(),
            email_forget_.get(),
            'retry_otp'
        )
    ]
)
_Retry_pincode.place(relx=0.45, rely=0.8)


#   widgets for forget account
send_otp_forget = Button(
    displaywindow, text="Send otp", command=lambda:
    [
        otp_forget_.delete(0, END),
        verification_email_entry(
            "forget_account",
            username_forget_.get(),
            email_forget_.get(),
            'forget_account', send_otp_forget
        )
    ]
```

```
    )


    account_no_forget = None
    list_widgets_forget_account_1 = (
        _Forget_label, _Username_forget, _Last_password_forget, _DoB_forget, ph_no_forget,
email_forget,
                    username_forget_,  last_password_forget_,  _DoB_forget_,  _Ph_no_forget_,
email_forget_, otp_forget,
        otp_forget_, _Retry_pincode, send_otp_forget)
    _Next_ = Button(
        displaywindow, text="Next", command=lambda:
        [
            activation_procedure_1(
                "forget_window",
                [
                    username_forget_.get(),
                    _DoB_forget_.get(),
                    last_password_forget_.get(),
                    _Ph_no_forget_.get(),
                    email_forget_.get()
                ]
            ),
            otp_verify(
                [
                    otp_forget_.get(),
                    pincode,
                    [
                        username_forget_.get(),
                        _DoB_forget_.get(),
```

```
                last_password_forget_.get(),
                account_no_forget,
                _Ph_no_forget_.get(),
                email_forget_.get()
            ]
        ]
    ),
    widgets_destroy(list_widgets_forget_account_1),
    _Next_.destroy(),
    back_login.destroy()
  ]
)

  global list_widget_forget_account
  list_widgets_forget_account = (
    _Forget_label, _Username_forget, _Last_password_forget, _DoB_forget, ph_no_forget, email_forget,
                username_forget_, last_password_forget_, _DoB_forget_, _Ph_no_forget_, email_forget_, otp_forget,
    otp_forget_, _Next_, _Retry_pincode, send_otp_forget)

  back_login = Button(displaywindow, text="Back",
                                                command=lambda: [back_login.destroy(), widgets_destroy(list_widgets_forget_account),
                            loginwindow("forget_account")])

  # Buttons placements of forget account
  _Next_.place(relx=0.7, rely=0.8)
  send_otp_forget.place(relx=0.25, rely=0.8)
```

```
    back_login.place(relx=0.1, rely=0.8)



def forget_message_window(list_forget_message):
    # program that helps in displaying information about the forget account
    displaywindow["bg"] = '#bfbfbf'

    # label for creation of new accounts
     account_creation_fmw = Label(displaywindow, text="Account Recovery", font="zapfino
16 bold")
    user_name_fmw = Label(displaywindow, text="Username:")
    dob_fmw = Label(displaywindow, text="Date of Birth:")
    password_fmw = Label(displaywindow, text="Password:")
    account_no_fmw = Label(displaywindow, text="Account no")
    email_id_fmw = Label(displaywindow, text="Phone no:")
    ph_no_fmw = Label(displaywindow, text="Email id:")

    # Entry widgets for displaying forget data
    user_name_display = Label(displaywindow, text=str(list_forget_message[0]))
    dob_display = Label(displaywindow, text=str(list_forget_message[1]))
    password_display = Label(displaywindow, text=str(list_forget_message[2]))
    account_no_display = Label(displaywindow, text=str(list_forget_message[3]))
    email_id_display = Label(displaywindow, text=str(list_forget_message[4]))
    ph_no_display = Label(displaywindow, text=str(list_forget_message[5]))

    # placement of label and entry widget
    # Label placement
    account_creation_fmw.place(relx=0.3)
    user_name_fmw.place(relx=0.1, rely=0.2)
```

```
    dob_fmw.place(relx=0.1, rely=0.3)
    password_fmw.place(relx=0.1, rely=0.4)
    account_no_fmw.place(relx=0.1, rely=0.5)
    email_id_fmw.place(relx=0.1, rely=0.6)
    ph_no_fmw.place(relx=0.1, rely=0.7)

    # Entry widget placement
    user_name_display.place(relx=0.4, rely=0.2)
    dob_display.place(relx=0.4, rely=0.3)
    password_display.place(relx=0.4, rely=0.4)
    account_no_display.place(relx=0.4, rely=0.5)
    email_id_display.place(relx=0.4, rely=0.6)
    ph_no_display.place(relx=0.4, rely=0.7)

    list_widget_forget_message_window = (
            account_creation_fmw, user_name_fmw, dob_fmw, password_fmw, email_id_fmw,
ph_no_fmw, account_no_fmw,
        user_name_display, dob_display, password_display, email_id_display, ph_no_display,
account_no_display
    )

    back_login_fmw = Button(displaywindow, text="Back",
                        command=lambda: [back_login_fmw.destroy(),
                                         widgets_destroy(list_widget_forget_message_window),
                                         loginwindow("forget_message_window")])

    # buttons placements
    back_login_fmw.place(relx=0.1, rely=0.9)
```

```
def _loginwindow_1():
    # program that controls the working of login account display window
    displaywindow["bg"] = '#bfbfbf'
    global proceed_activation
    proceed_activation = False
    # Welcome label
    wel = Label(displaywindow, text="Welcome To AIY Note", font="zapfino 16 bold")
    signin = Label(displaywindow, text="Sign in", font="calbri 16")
    wel.place(relx=0.2)
    signin.place(relx=0.3, rely=0.2)

    # login and entry
    loginid = Label(displaywindow, text="Username:")
    loginid.place(relx=0.1, rely=0.3)
    loginpassword = Label(displaywindow, text="Password:")
    loginpassword.place(relx=0.1, rely=0.4)
    # entry widgets
    id_ = Entry(displaywindow)
    password_ = Entry(displaywindow, show="*")

    id_.place(relx=0.3, rely=0.3)
    password_.place(relx=0.3, rely=0.4)

    # Buttons
    next_l1 = Button(
        displaywindow, text="Next", command=lambda:
        [
            activation_procedure_1('login_displaywindow', [id_.get(), password_.get()]),
            activation_procedure_2('login_displaywindow', [id_.get(), password_.get()]),
```

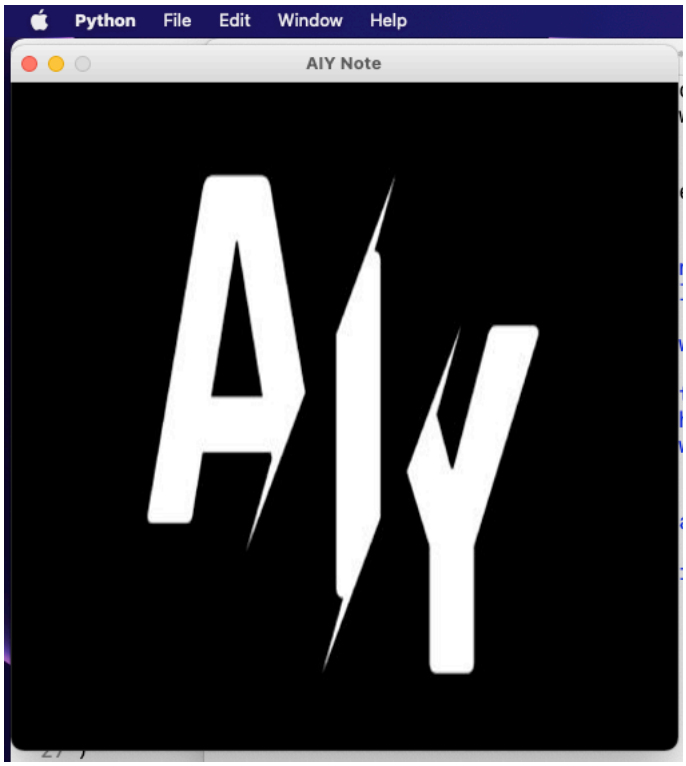
```
        ]


    )

    forgotpassword_l1 = Button(
        displaywindow, text="Forgot password", command=lambda:
        [widgets_destroy(loginwindow_list_widgets), forget_password()]
    )

    createaccount_l1 = Button(
        displaywindow, text="Create account", command=lambda:
        [widgets_destroy(loginwindow_list_widgets), account_creation()]
    )

    next_l1.place(relx=0.7, rely=0.6)
    forgotpassword_l1.place(relx=0.1, rely=0.6)
    createaccount_l1.place(relx=0.4, rely=0.6)
    global loginwindow_list_widgets
    loginwindow_list_widgets = (
            wel, signin, loginid, loginpassword, id_, password_, next_l1, forgotpassword_l1,
createaccount_l1)


# image
# Read the Image
image = Image.open(location_AIY_Logo)
# Resize the image using resize() method
resize_image = image.resize((600, 600))
img = ImageTk.PhotoImage(resize_image)
image.close()
```

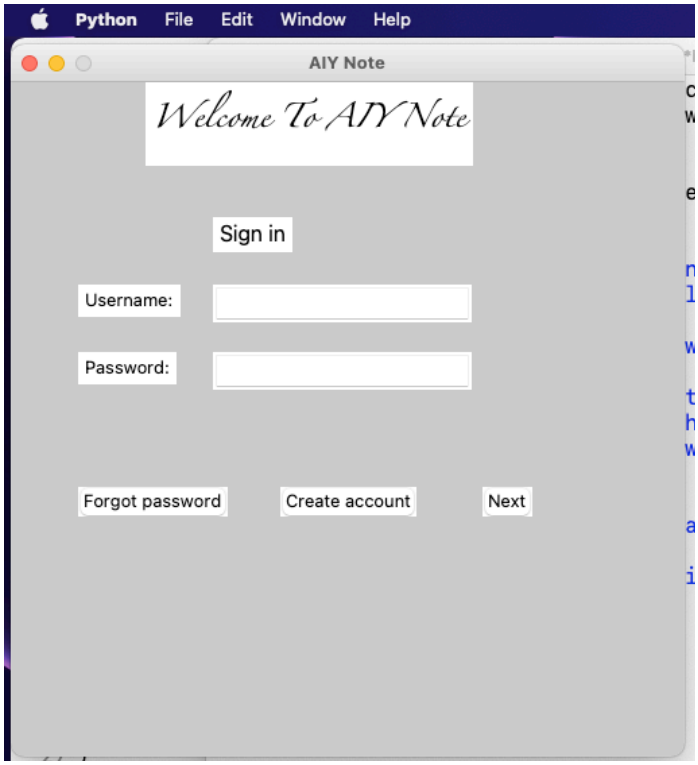
```
image_3 = Image.open(location_AIY_Logo)

resize_image_3 = image_3.resize((100, 50))

img_3 = ImageTk.PhotoImage(resize_image_3)


logo_button = Button(displaywindow, image=img, command=lambda: [logo_button.destroy(), _loginwindow_1()])

logo_button.pack()


displaywindow.mainloop()

mydb.commit()

my_c.close()
```

Program Interface

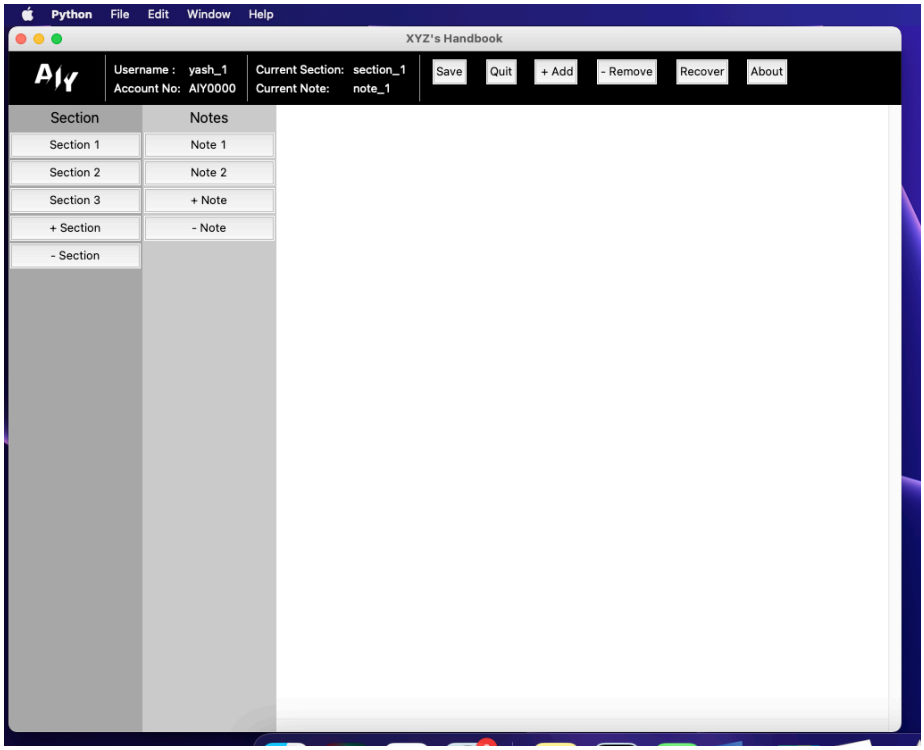
Program interface 1:

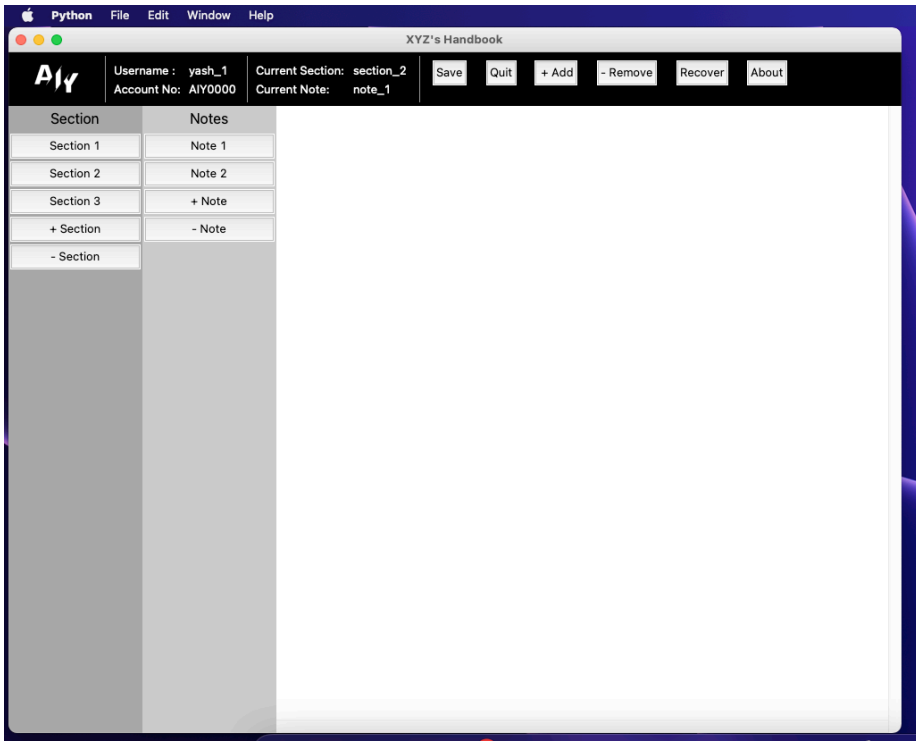
Login Window and Note Window




Python File Edit Window Help
AIY Note
Welcome To AIY Note
Sign in
Username:
Password:
Forgot password
Create account
Next

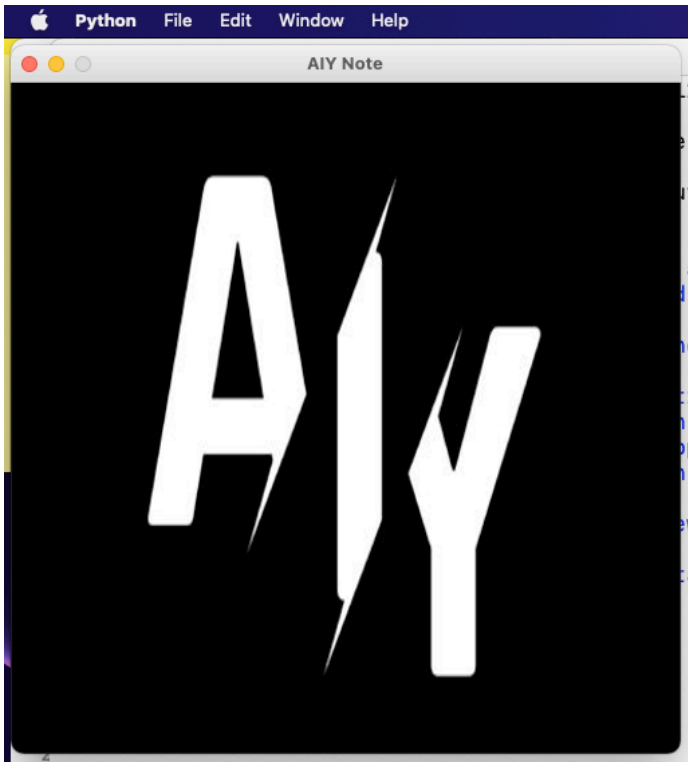

Python File Edit Window Help
AIY Note
Welcome To AIY Note
Sign in
Username:
yash_1
Password:
***
Forgot password
Create account
Next

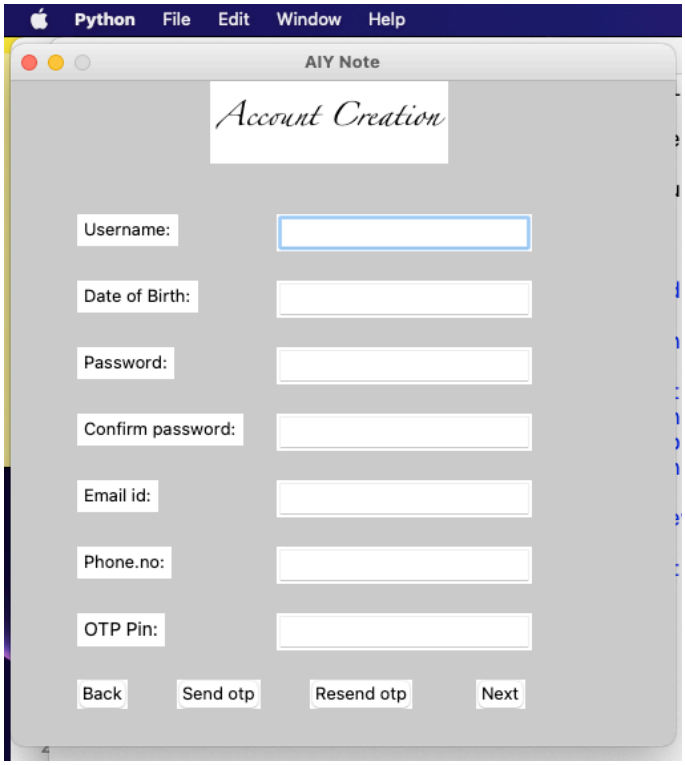

Python File Edit Window Help
XYZ's Handbook
Username : yash_1
Account No: AIY0000
Current Section: section_1
Current Note: note_1
Save
Quit
+ Add
- Remove
Recover
About
Section
Notes
Section 1
Section 2
Section 3
+ Section
- Section
Note 1
Note 2
+ Note
- Note

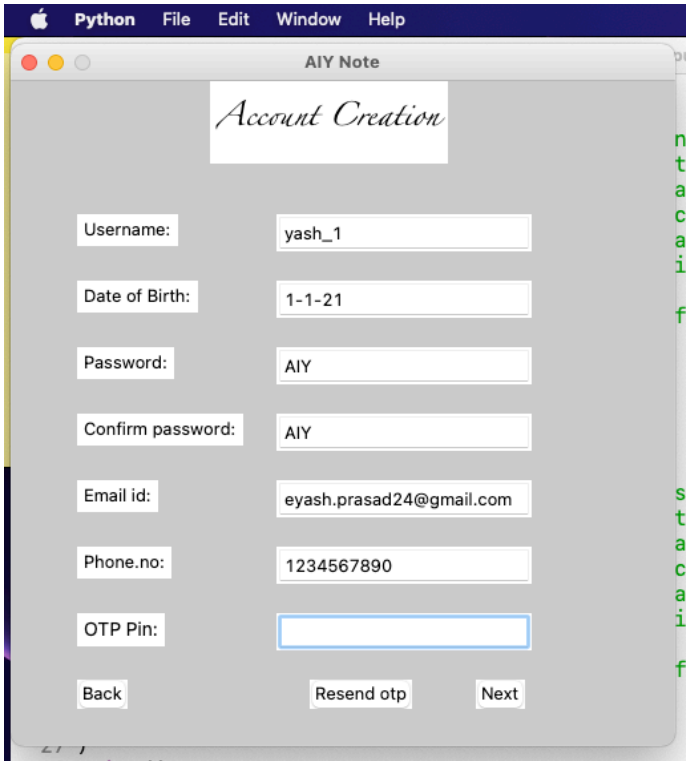

Python File Edit Window Help
XYZ's Handbook
Username : yash_1
Account No: AIY0000
Current Section: section_2
Current Note: note_1
Save
Quit
+ Add
- Remove
Recover
About
Section
Notes
Section 1
Section 2
Section 3
+ Section
- Section
Note 1
Note 2
+ Note
- Note

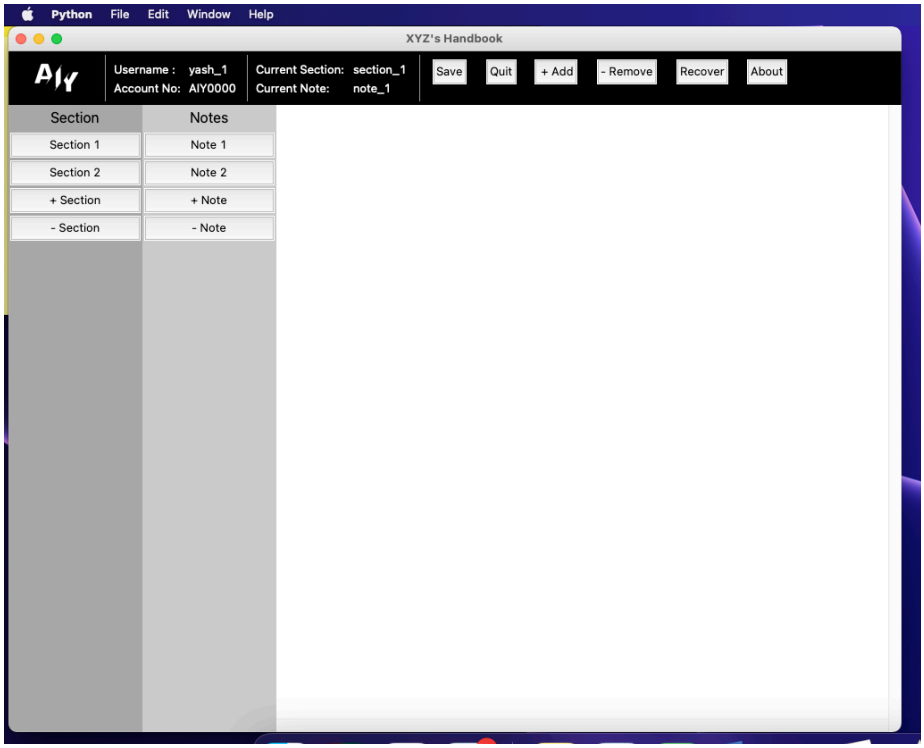
Program interface 2:

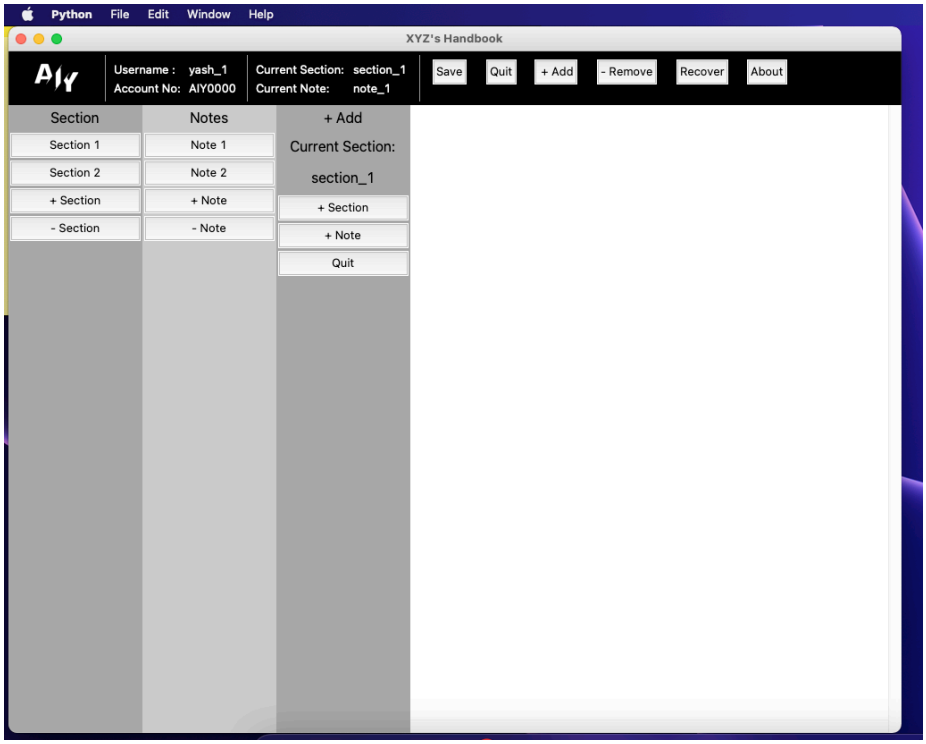
Create Account Window and Note Window

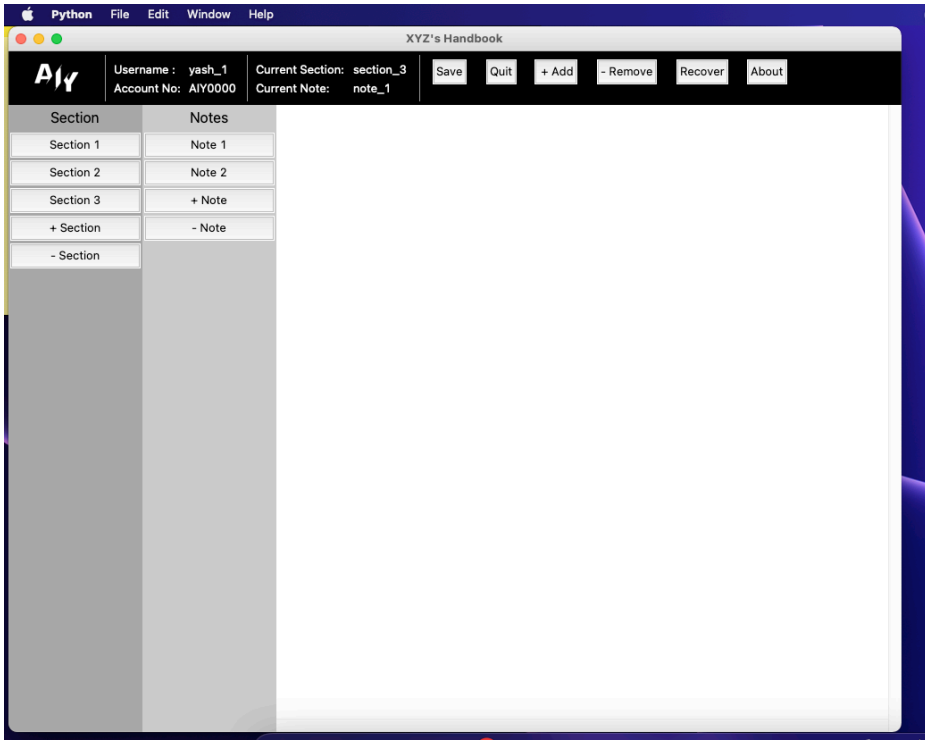


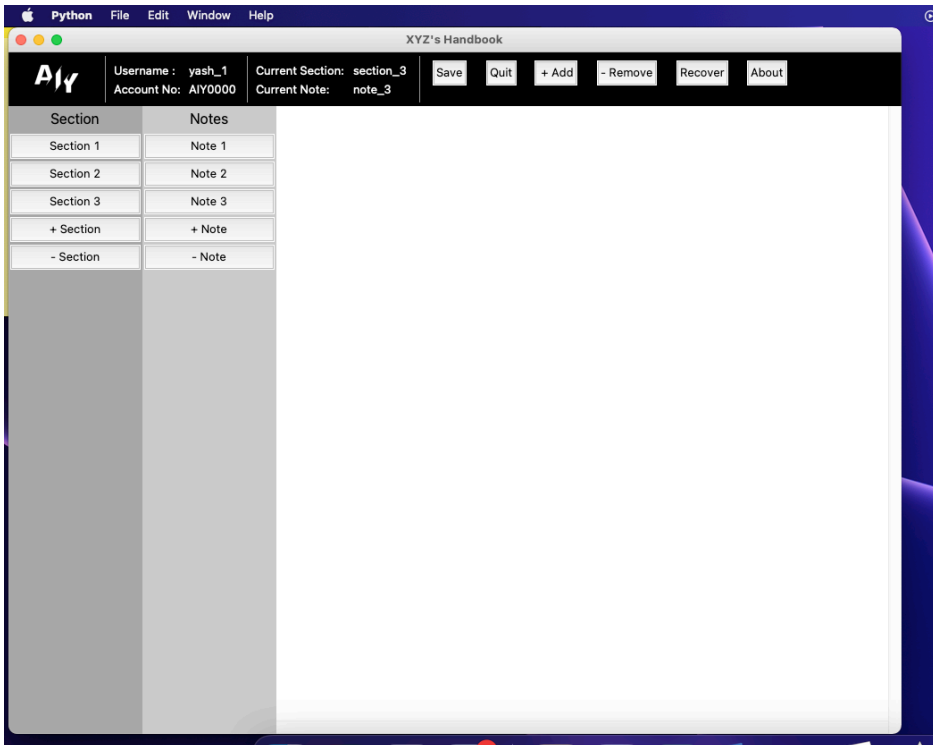

Python File Edit Window Help
AIY Note
Account Creation
Username:
Date of Birth:
Password:
Confirm password:
Email id:
Phone.no:
OTP Pin:
Back
Send otp
Resend otp
Next

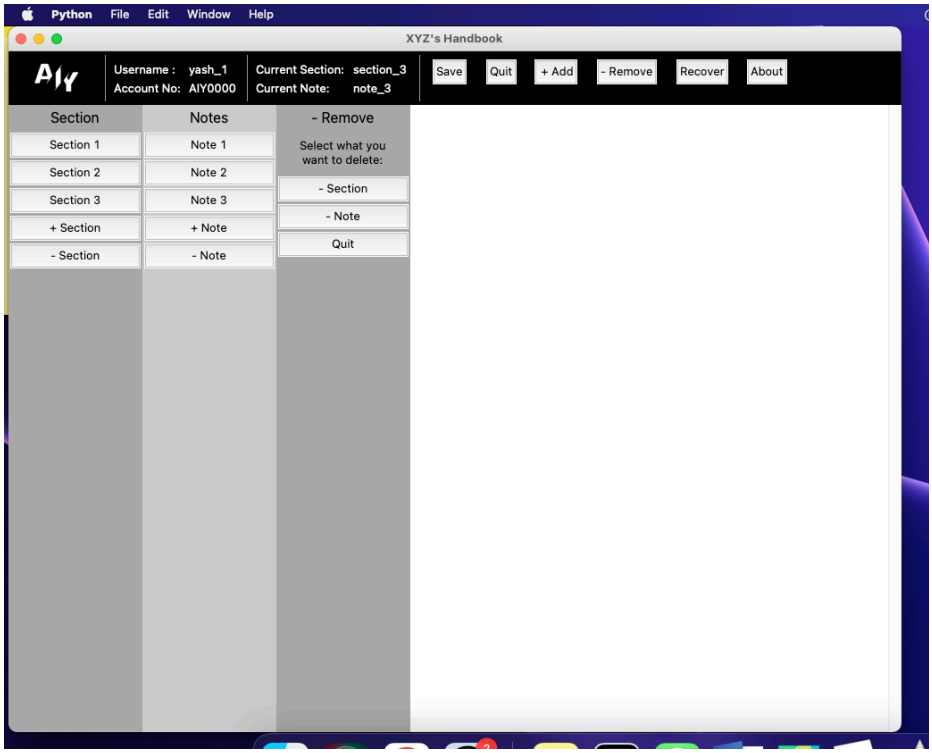

Python File Edit Window Help
AIY Note
Account Creation
Username:
yash_1
Date of Birth:
1-1-21
Password:
AIY
Confirm password:
AIY
Email id:
eyash.prasad24@gmail.com
Phone.no:
1234567890
OTP Pin:
Back
Resend otp
Next

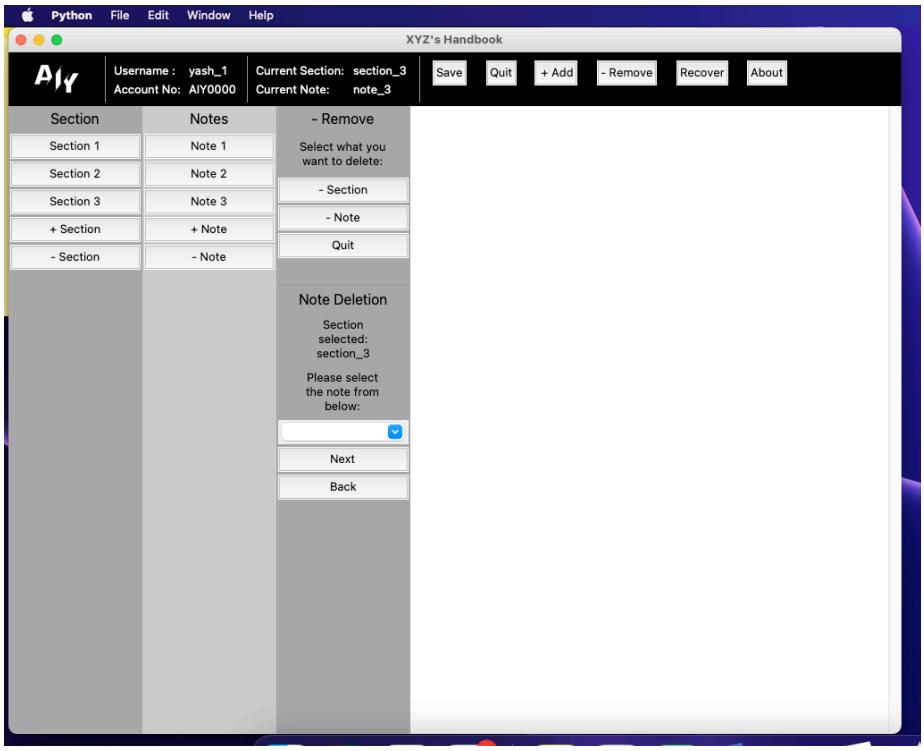

Python File Edit Window Help
XYZ's Handbook
Username : yash_1
Account No: AIY0000
Current Section: section_1
Current Note: note_1
Save
Quit
+ Add
- Remove
Recover
About
Section
Notes
Section 1
Note 1
Section 2
Note 2
+ Section
+ Note
- Section
- Note

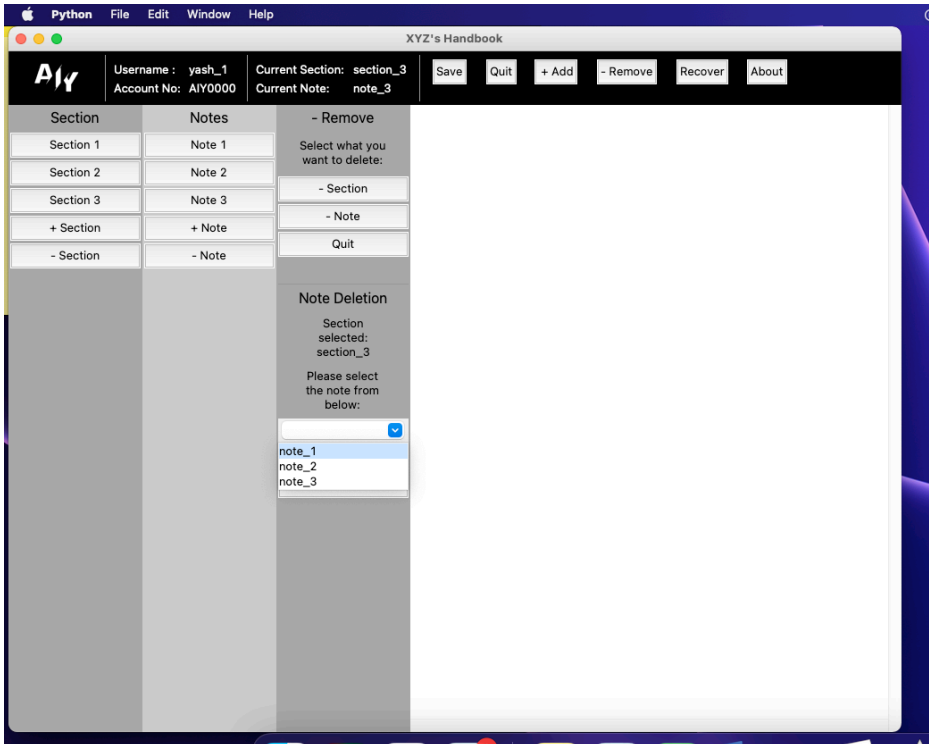

Python File Edit Window Help
XYZ's Handbook
Username : yash_1
Account No: AIY0000
Current Section: section_1
Current Note: note_1
Save
Quit
+ Add
- Remove
Recover
About
Section
Notes
+ Add
Current Section:
section_1
Section 1
Note 1
Section 2
Note 2
+ Section
+ Note
- Section
- Note
+ Section
+ Note
Quit

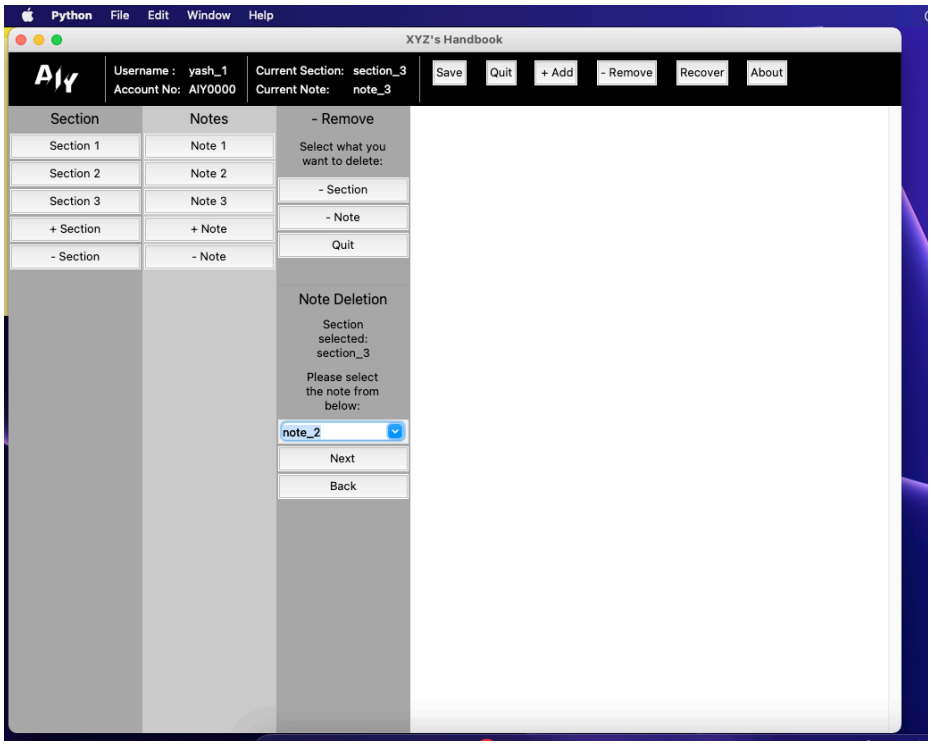

Python File Edit Window Help
XYZ's Handbook
AIY
Username : yash_1
Account No: AIY0000
Current Section: section_3
Current Note: note_1
Save
Quit
+ Add
- Remove
Recover
About
Section
Notes
Section 1
Section 2
Section 3
+ Section
- Section
Note 1
Note 2
+ Note
- Note

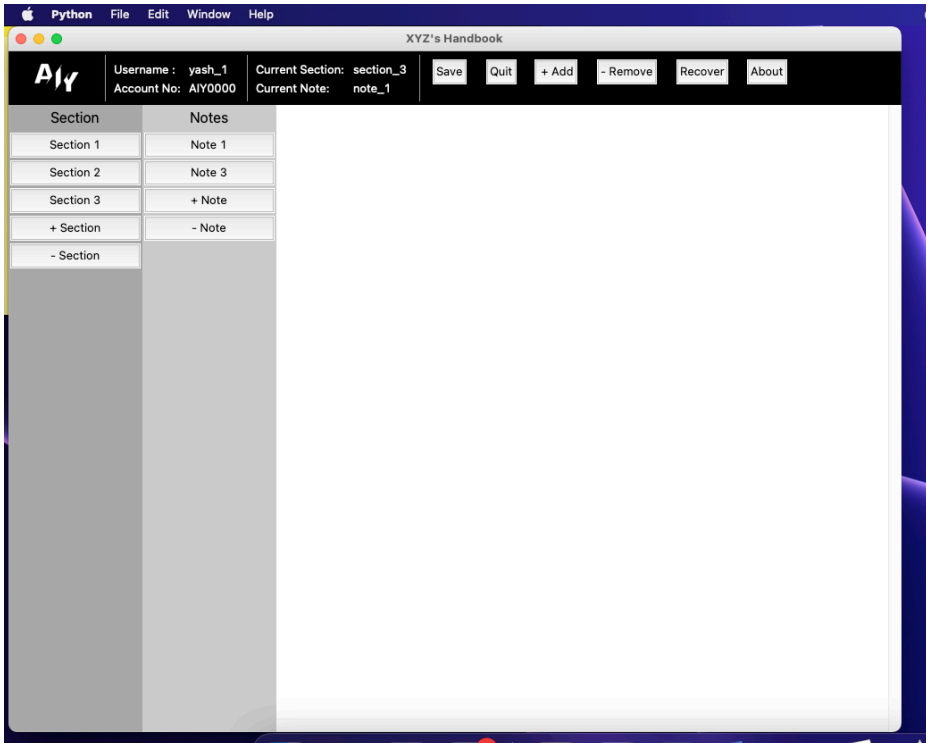

Python File Edit Window Help
XYZ's Handbook
AIY
Username : yash_1
Account No: AIY0000
Current Section: section_3
Current Note: note_3
Save
Quit
+ Add
- Remove
Recover
About
Section
Notes
Section 1
Section 2
Section 3
+ Section
- Section
Note 1
Note 2
Note 3
+ Note
- Note

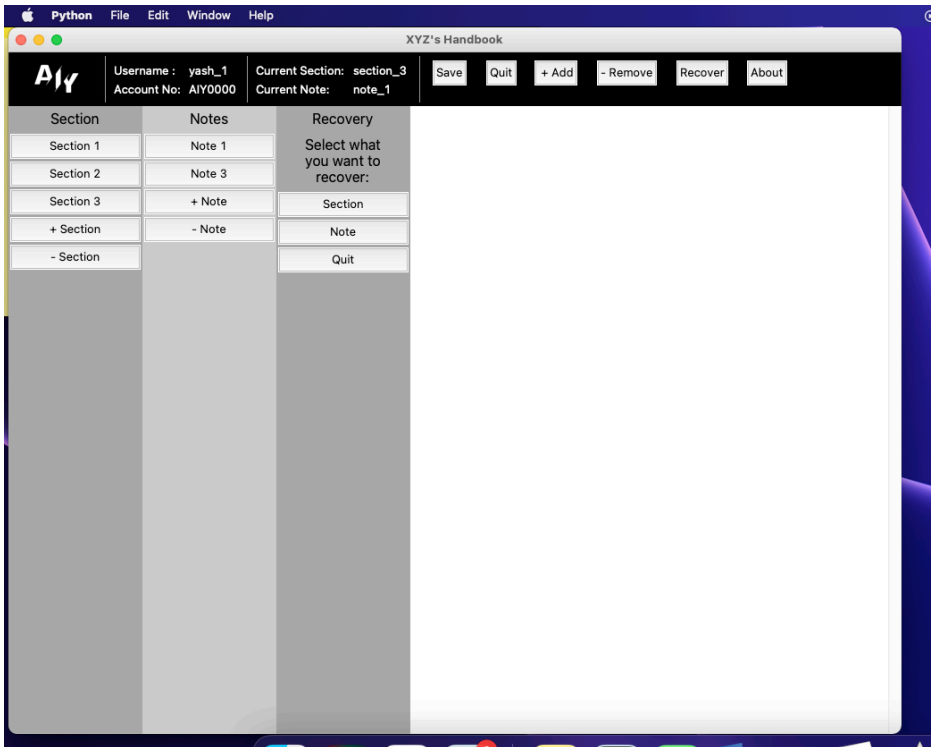

Python File Edit Window Help
XYZ's Handbook
AIY
Username : yash_1
Account No: AIY0000
Current Section: section_3
Current Note: note_3
Save
Quit
+ Add
- Remove
Recover
About
Section
Notes
- Remove
Section 1
Note 1
Select what you want to delete:
Section 2
Note 2
- Section
Section 3
Note 3
- Note
+ Section
+ Note
Quit
- Section
- Note

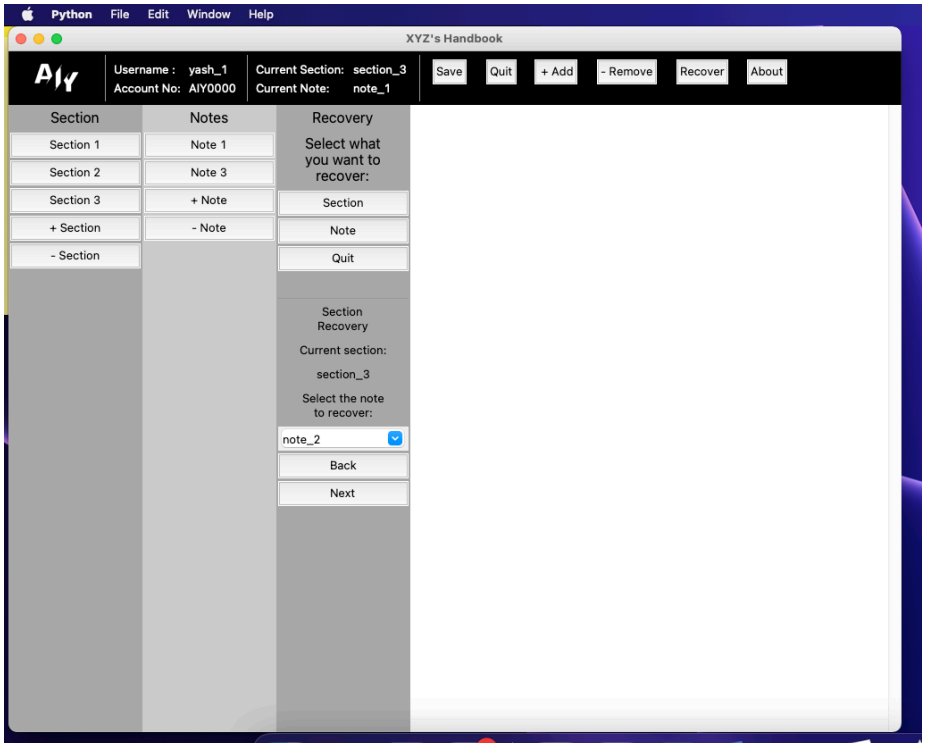

Python File Edit Window Help
XYZ's Handbook
AIY
Username : yash_1
Account No: AIY0000
Current Section: section_3
Current Note: note_3
Save
Quit
+ Add
- Remove
Recover
About
Section
Notes
- Remove
Section 1
Note 1
Select what you want to delete:
Section 2
Note 2
- Section
Section 3
Note 3
- Note
+ Section
+ Note
Quit
- Section
- Note
Note Deletion
Section selected: section_3
Please select the note from below:
Next
Back

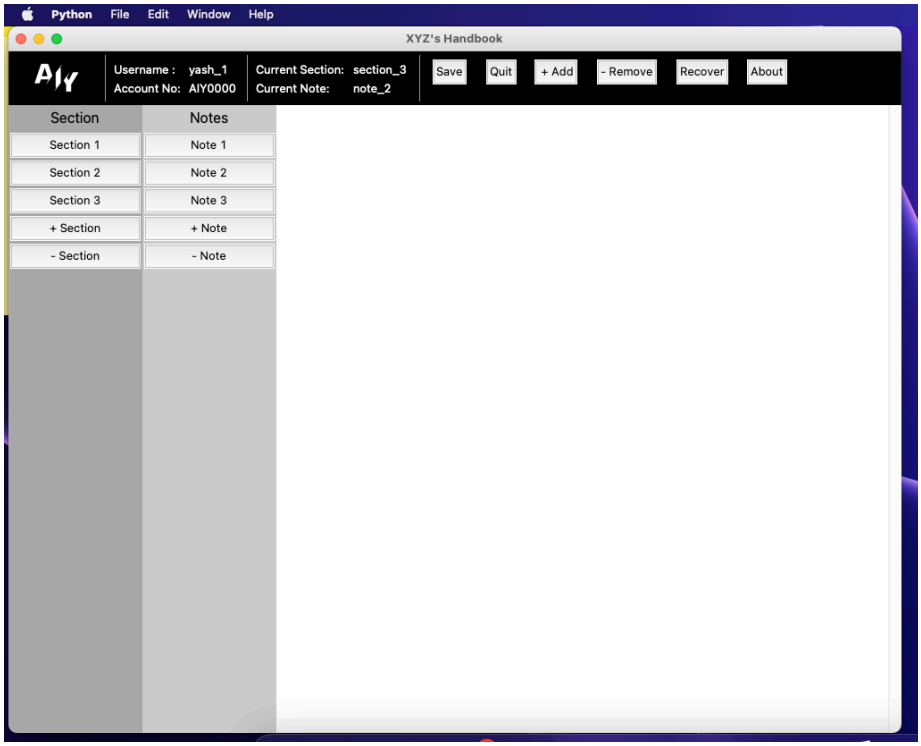

Python File Edit Window Help
XYZ's Handbook
AIY
Username : yash_1
Account No: AIY0000
Current Section: section_3
Current Note: note_3
Save
Quit
+ Add
- Remove
Recover
About
Section
Section 1
Section 2
Section 3
+ Section
- Section
Notes
Note 1
Note 2
Note 3
+ Note
- Note
- Remove
Select what you want to delete:
- Section
- Note
Quit
Note Deletion
Section selected: section_3
Please select the note from below:
note_1
note_2
note_3

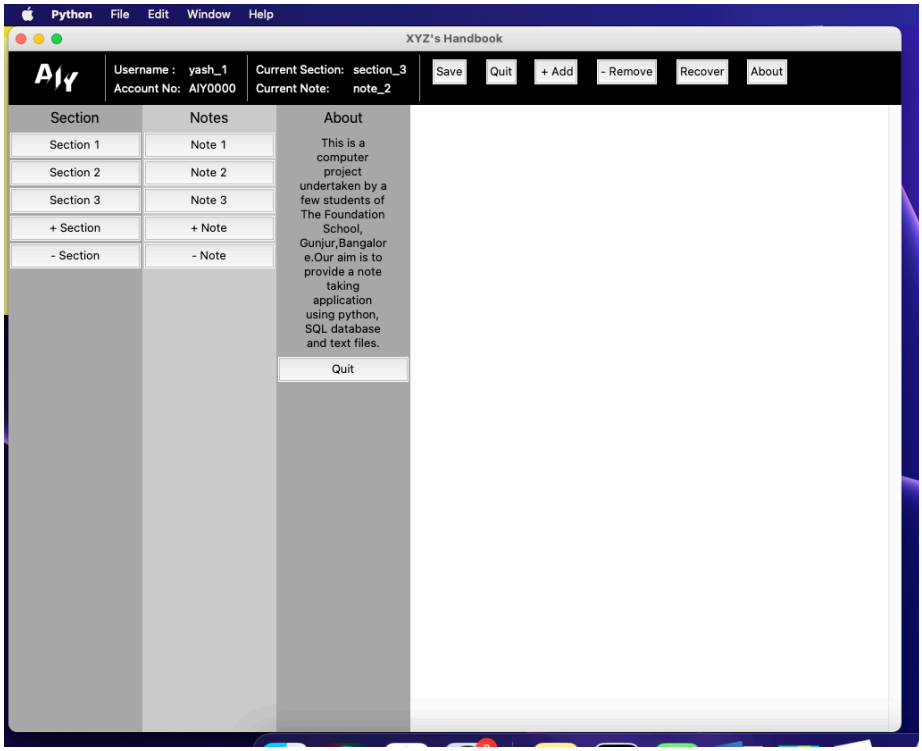

Python File Edit Window Help
XYZ's Handbook
AIY
Username : yash_1
Account No: AIY0000
Current Section: section_3
Current Note: note_3
Save
Quit
+ Add
- Remove
Recover
About
Section
Section 1
Section 2
Section 3
+ Section
- Section
Notes
Note 1
Note 2
Note 3
+ Note
- Note
- Remove
Select what you want to delete:
- Section
- Note
Quit
Note Deletion
Section selected: section_3
Please select the note from below:
note_2
Next
Back

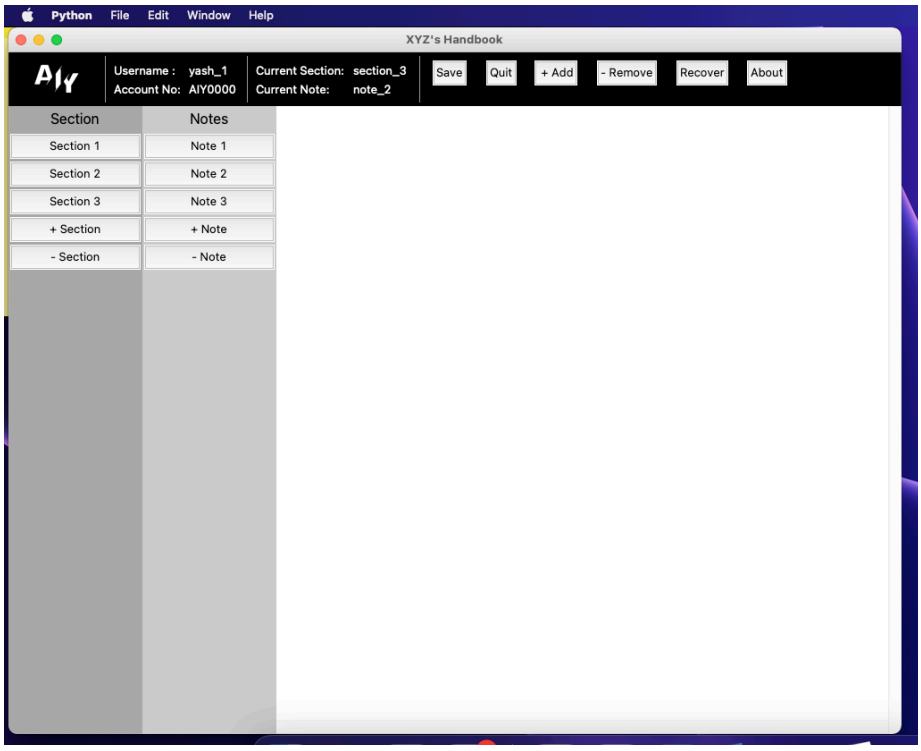

Python File Edit Window Help
XYZ's Handbook
AIY
Username : yash_1
Account No: AIY0000
Current Section: section_3
Current Note: note_1
Save
Quit
+ Add
- Remove
Recover
About
Section
Notes
Section 1
Section 2
Section 3
+ Section
- Section
Note 1
Note 3
+ Note
- Note


Python File Edit Window Help
XYZ's Handbook
AIY
Username : yash_1
Account No: AIY0000
Current Section: section_3
Current Note: note_1
Save
Quit
+ Add
- Remove
Recover
About
Section
Notes
Recovery
Section 1
Section 2
Section 3
+ Section
- Section
Note 1
Note 3
+ Note
- Note
Select what you want to recover:
Section
Note
Quit

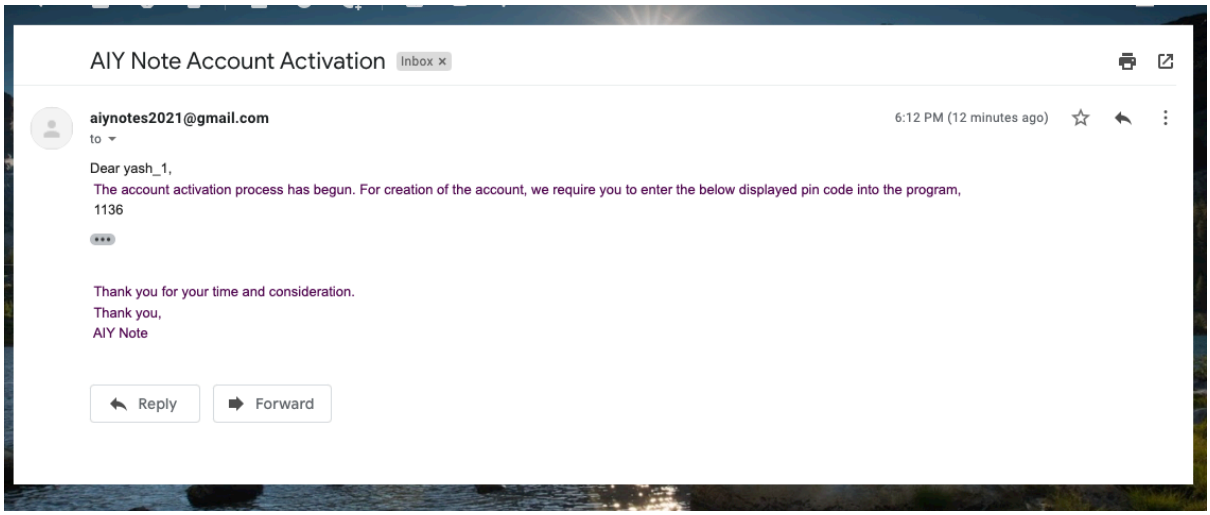

Python File Edit Window Help
XYZ's Handbook
AIY
Username : yash_1
Account No: AIY0000
Current Section: section_3
Current Note: note_1
Save
Quit
+ Add
- Remove
Recover
About
Section
Section 1
Section 2
Section 3
+ Section
- Section
Notes
Note 1
Note 3
+ Note
- Note
Recovery
Select what you want to recover:
Section
Note
Quit
Section Recovery
Current section:
section_3
Select the note to recover:
note_2
Back
Next


Python File Edit Window Help
XYZ's Handbook
AIY
Username : yash_1
Account No: AIY0000
Current Section: section_3
Current Note: note_2
Save
Quit
+ Add
- Remove
Recover
About
Section
Section 1
Section 2
Section 3
+ Section
- Section
Notes
Note 1
Note 2
Note 3
+ Note
- Note


Python File Edit Window Help
XYZ's Handbook
AIY
Username : yash_1
Account No: AIY0000
Current Section: section_3
Current Note: note_2
Save
Quit
+ Add
- Remove
Recover
About
Section
Notes
About
Section 1
Note 1
Section 2
Note 2
Section 3
Note 3
+ Section
+ Note
- Section
- Note
This is a computer project undertaken by a few students of The Foundation School, Gunjur,Bangalore.Our aim is to provide a note taking application using python, SQL database and text files.
Quit

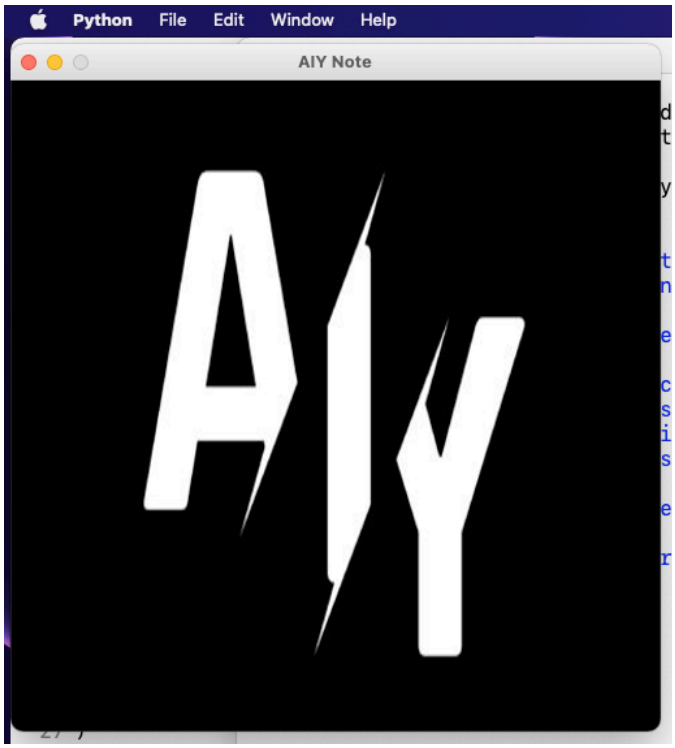

Python File Edit Window Help
XYZ's Handbook
AIY
Username : yash_1
Account No: AIY0000
Current Section: section_3
Current Note: note_2
Save
Quit
+ Add
- Remove
Recover
About
Section
Notes
Section 1
Note 1
Section 2
Note 2
Section 3
Note 3
+ Section
+ Note
- Section
- Note

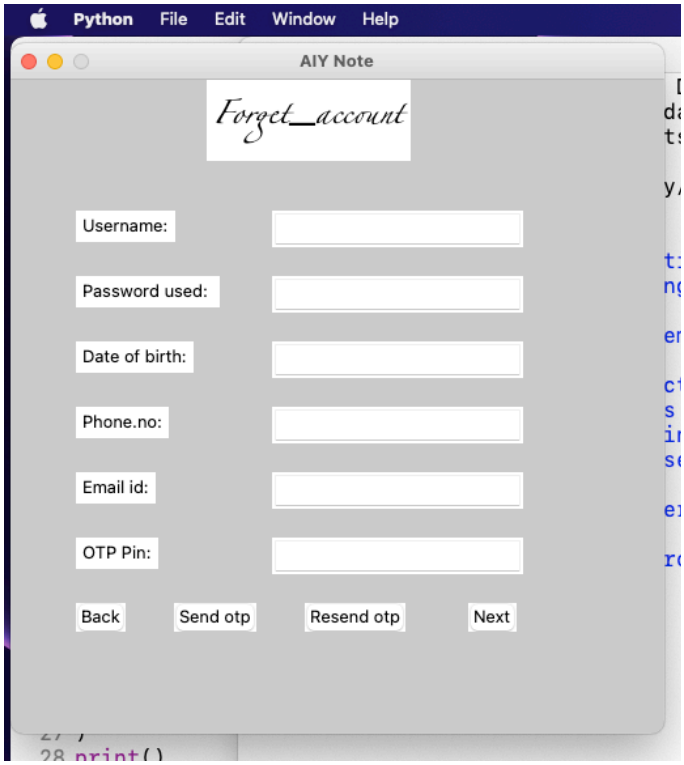
OTP confirmation email:



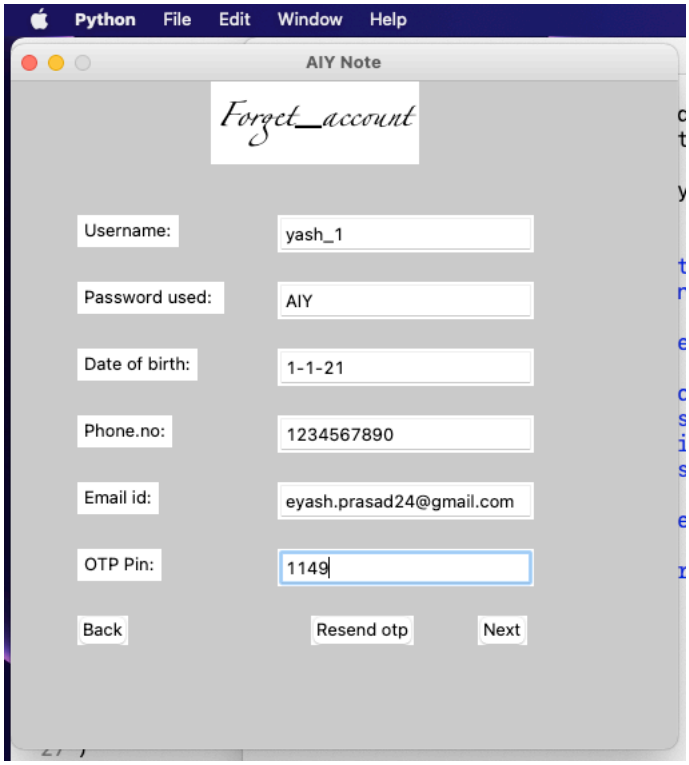
Program interface 3:

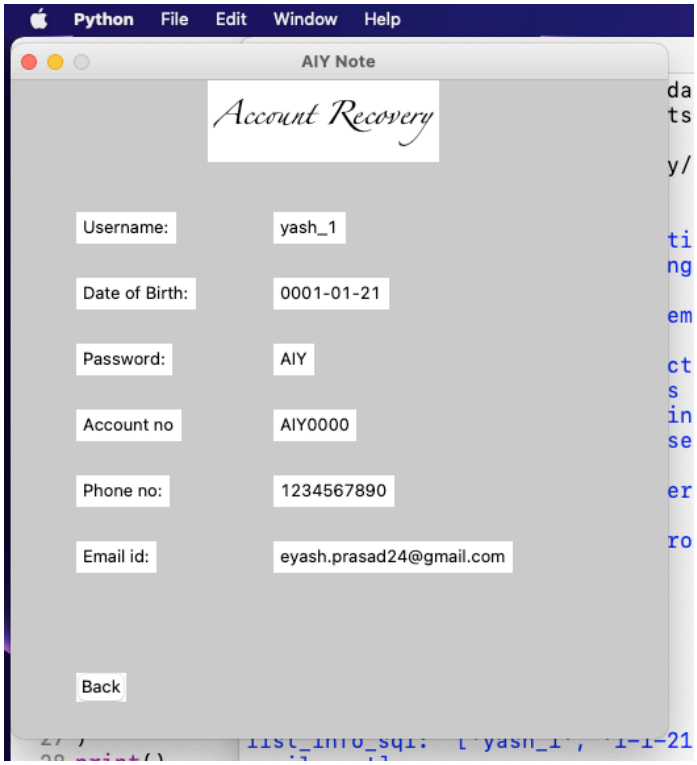
Forget Account Window



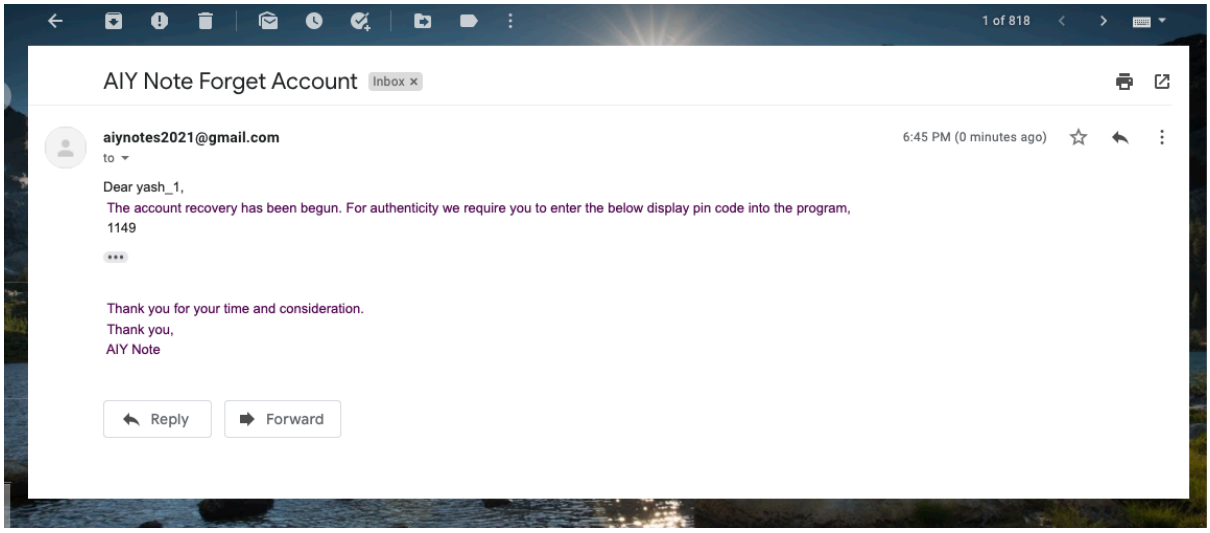

Python File Edit Window Help
AIY Note
Forget_account
Username:
Password used:
Date of birth:
Phone.no:
Email id:
OTP Pin:
Back
Send otp
Resend otp
Next


Python File Edit Window Help
AIY Note
Forget_account
Username:
yash_1
Password used:
AIY
Date of birth:
1-1-21
Phone.no:
1234567890
Email id:
eyash.prasad24@gmail.com
OTP Pin:
Back
Resend otp
Next


Python File Edit Window Help
AIY Note
Forget_account
Username:
yash_1
Password used:
AIY
Date of birth:
1-1-21
Phone.no:
1234567890
Email id:
eyash.prasad24@gmail.com
OTP Pin:
1149
Back
Resend otp
Next


Python File Edit Window Help
AIY Note
Account Recovery
Username:
yash_1
Date of Birth:
0001-01-21
Password:
AIY
Account no
AIY0000
Phone no:
1234567890
Email id:
eyash.prasad24@gmail.com
Back

OTP confirmation email:

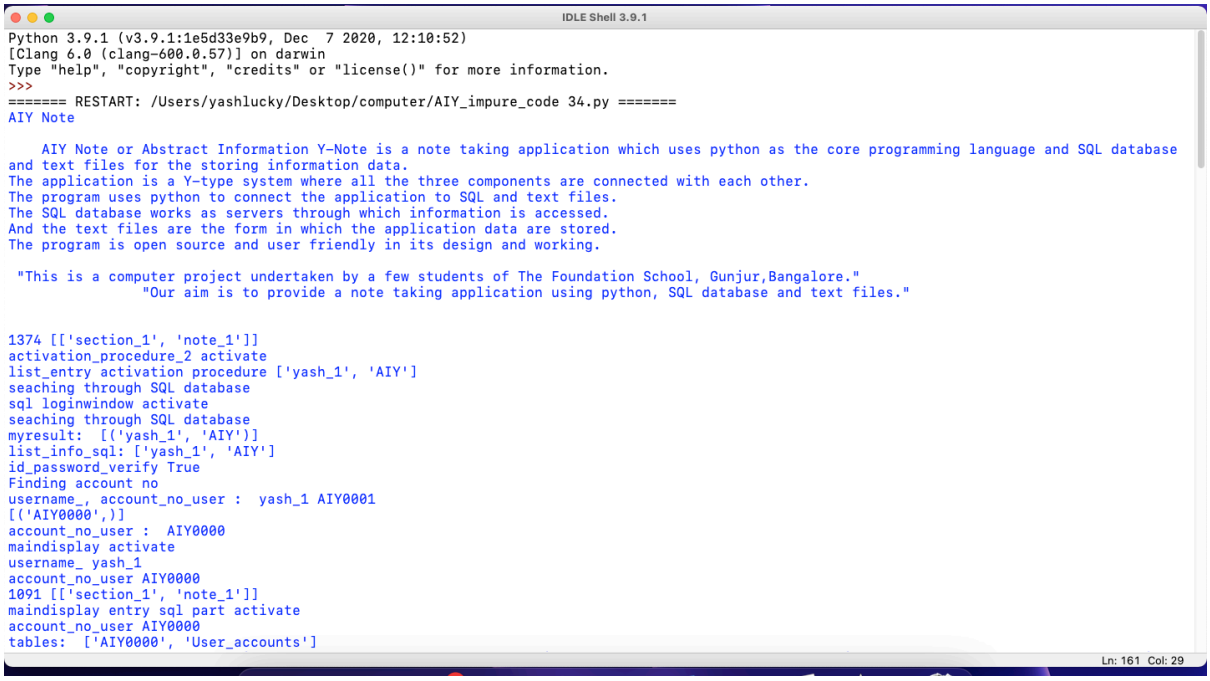


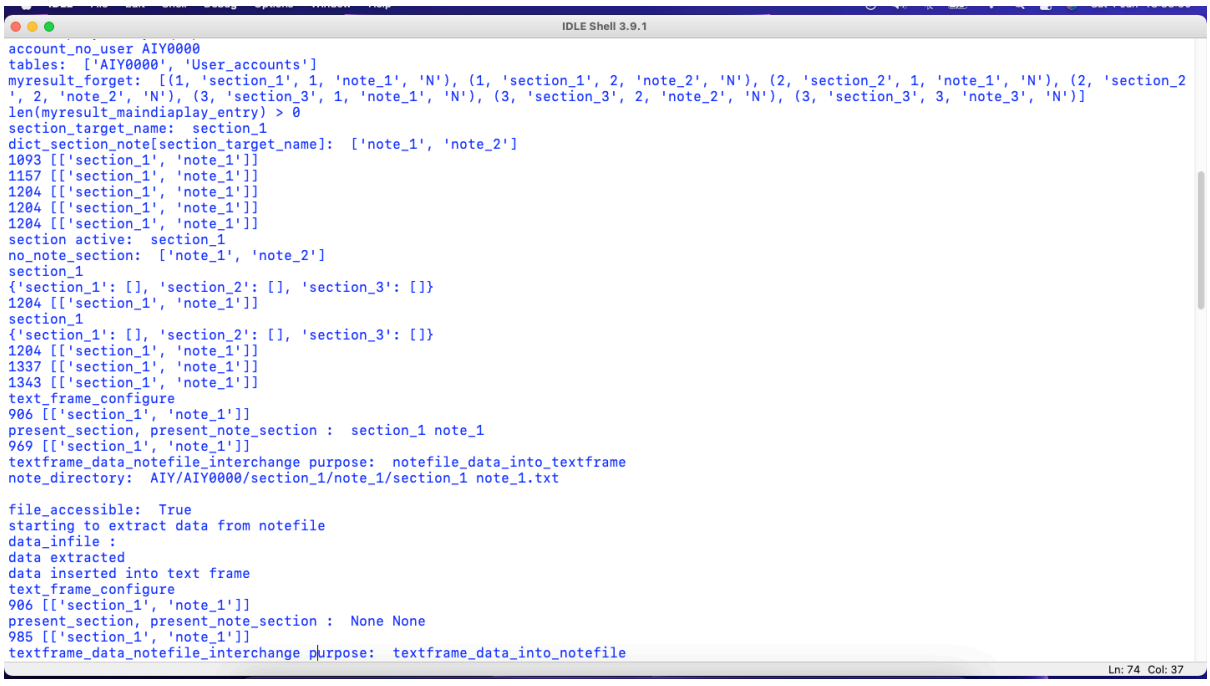
Program Output

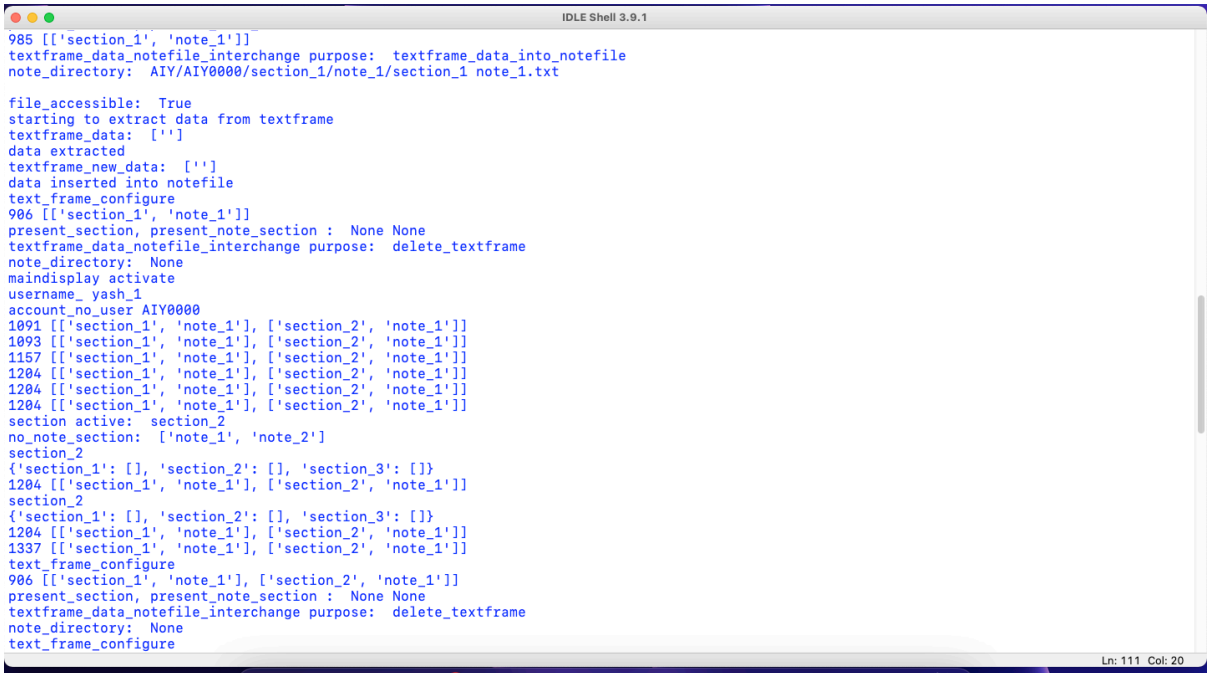
Program Output 1:

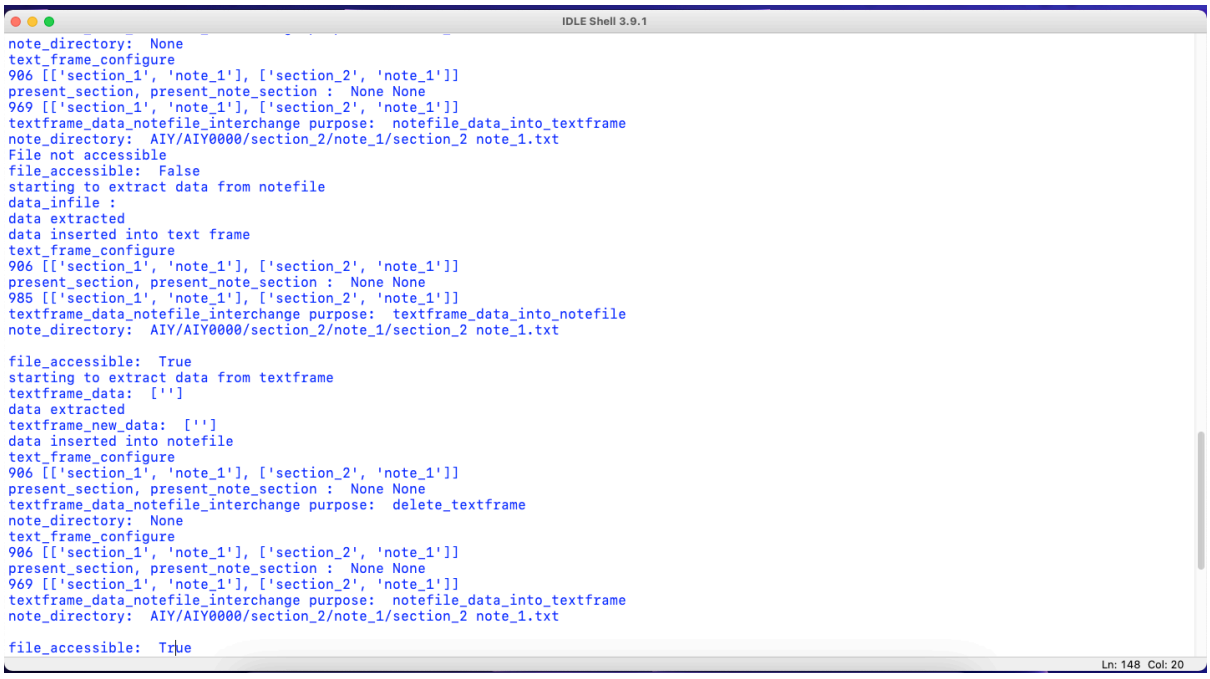
Login Window and Note Window

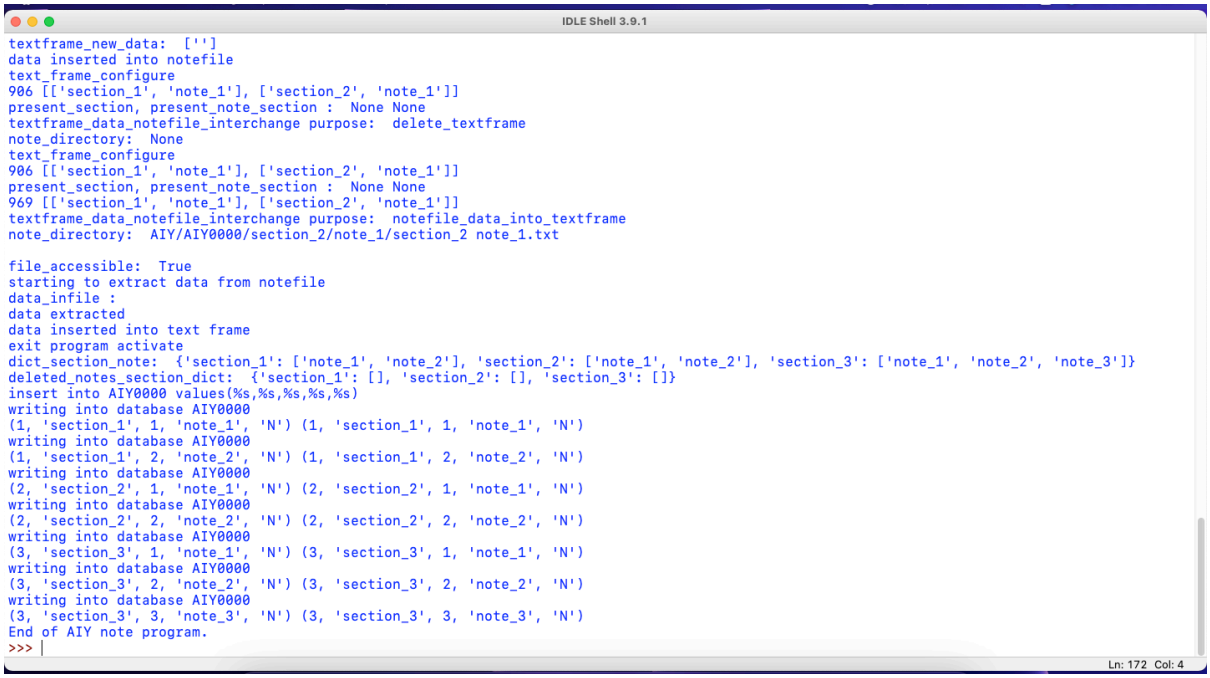
Python Output screen:






IDLE Shell 3.9.1
985 [['section_1', 'note_1']]
textframe_data_notefile_interchange purpose: textframe_data_into_notefile
note_directory: AIY/AIY0000/section_1/note_1/section_1 note_1.txt
file_accessible: True
starting to extract data from textframe
textframe_data: ['']
data extracted
textframe_new_data: ['']
data inserted into notefile
text_frame_configure
906 [['section_1', 'note_1']]
present_section, present_note_section : None None
textframe_data_notefile_interchange purpose: delete_textframe
note_directory: None
maindisplay activate
username_ yash_1
account_no_user AIY0000
1091 [['section_1', 'note_1'], ['section_2', 'note_1']]
1093 [['section_1', 'note_1'], ['section_2', 'note_1']]
1157 [['section_1', 'note_1'], ['section_2', 'note_1']]
1204 [['section_1', 'note_1'], ['section_2', 'note_1']]
1204 [['section_1', 'note_1'], ['section_2', 'note_1']]
1204 [['section_1', 'note_1'], ['section_2', 'note_1']]
section active: section_2
no_note_section: ['note_1', 'note_2']
section_2
{'section_1': [], 'section_2': [], 'section_3': []}
1204 [['section_1', 'note_1'], ['section_2', 'note_1']]
section_2
{'section_1': [], 'section_2': [], 'section_3': []}
1204 [['section_1', 'note_1'], ['section_2', 'note_1']]
1337 [['section_1', 'note_1'], ['section_2', 'note_1']]
text_frame_configure
906 [['section_1', 'note_1'], ['section_2', 'note_1']]
present_section, present_note_section : None None
textframe_data_notefile_interchange purpose: delete_textframe
note_directory: None
text_frame_configure
Ln: 111 Col: 20

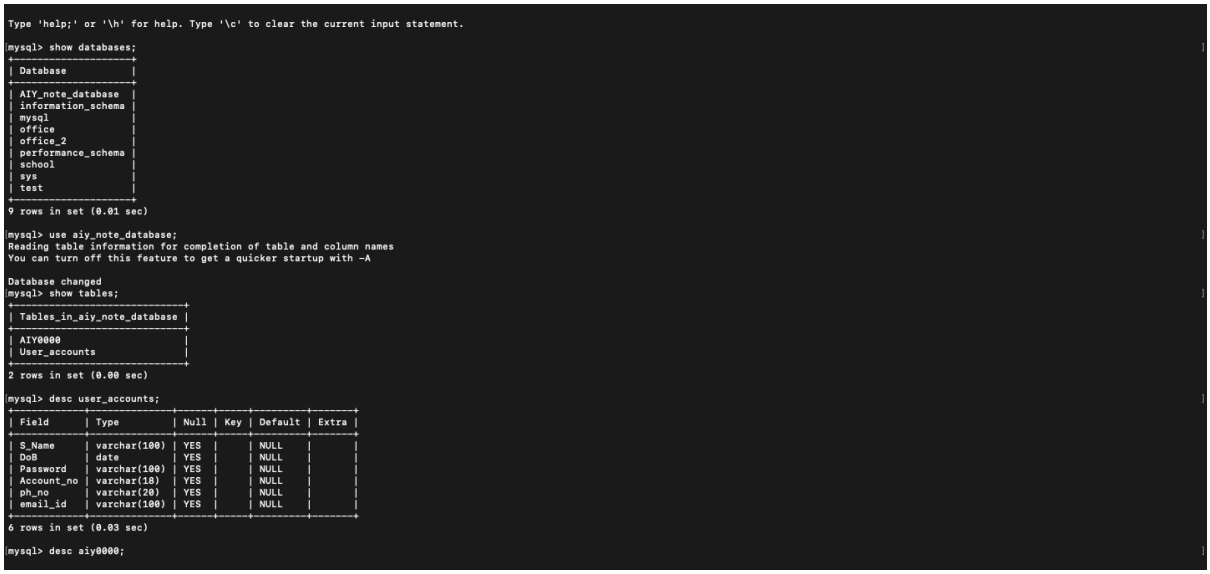

IDLE Shell 3.9.1
note_directory: None
text_frame_configure
906 [['section_1', 'note_1'], ['section_2', 'note_1']]
present_section, present_note_section : None None
969 [['section_1', 'note_1'], ['section_2', 'note_1']]
textframe_data_notefile_interchange purpose: notefile_data_into_textframe
note_directory: AIY/AIY0000/section_2/note_1/section_2 note_1.txt
File not accessible
file_accessible: False
starting to extract data from notefile
data_infile :
data extracted
data inserted into text frame
text_frame_configure
906 [['section_1', 'note_1'], ['section_2', 'note_1']]
present_section, present_note_section : None None
985 [['section_1', 'note_1'], ['section_2', 'note_1']]
textframe_data_notefile_interchange purpose: textframe_data_into_notefile
note_directory: AIY/AIY0000/section_2/note_1/section_2 note_1.txt
file_accessible: True
starting to extract data from textframe
textframe_data: ['']
data extracted
textframe_new_data: ['']
data inserted into notefile
text_frame_configure
906 [['section_1', 'note_1'], ['section_2', 'note_1']]
present_section, present_note_section : None None
textframe_data_notefile_interchange purpose: delete_textframe
note_directory: None
text_frame_configure
906 [['section_1', 'note_1'], ['section_2', 'note_1']]
present_section, present_note_section : None None
969 [['section_1', 'note_1'], ['section_2', 'note_1']]
textframe_data_notefile_interchange purpose: notefile_data_into_textframe
note_directory: AIY/AIY0000/section_2/note_1/section_2 note_1.txt
file_accessible: True
Ln: 148 Col: 20


IDLE Shell 3.9.1
textframe_new_data: ['']
data inserted into notefile
text_frame_configure
906 [['section_1', 'note_1'], ['section_2', 'note_1']]
present_section, present_note_section : None None
textframe_data_notefile_interchange purpose: delete_textframe
note_directory: None
text_frame_configure
906 [['section_1', 'note_1'], ['section_2', 'note_1']]
present_section, present_note_section : None None
969 [['section_1', 'note_1'], ['section_2', 'note_1']]
textframe_data_notefile_interchange purpose: notefile_data_into_textframe
note_directory: AIY/AIY0000/section_2/note_1/section_2 note_1.txt
file_accessible: True
starting to extract data from notefile
data_infile :
data extracted
data inserted into text frame
exit program activate
dict_section_note: {'section_1': ['note_1', 'note_2'], 'section_2': ['note_1', 'note_2'], 'section_3': ['note_1', 'note_2', 'note_3']}
deleted_notes_section_dict: {'section_1': [], 'section_2': [], 'section_3': []}
insert into AIY0000 values(%s,%s,%s,%s,%s)
writing into database AIY0000
(1, 'section_1', 1, 'note_1', 'N') (1, 'section_1', 1, 'note_1', 'N')
writing into database AIY0000
(1, 'section_1', 2, 'note_2', 'N') (1, 'section_1', 2, 'note_2', 'N')
writing into database AIY0000
(2, 'section_2', 1, 'note_1', 'N') (2, 'section_2', 1, 'note_1', 'N')
writing into database AIY0000
(2, 'section_2', 2, 'note_2', 'N') (2, 'section_2', 2, 'note_2', 'N')
writing into database AIY0000
(3, 'section_3', 1, 'note_1', 'N') (3, 'section_3', 1, 'note_1', 'N')
writing into database AIY0000
(3, 'section_3', 2, 'note_2', 'N') (3, 'section_3', 2, 'note_2', 'N')
writing into database AIY0000
(3, 'section_3', 3, 'note_3', 'N') (3, 'section_3', 3, 'note_3', 'N')
End of AIY note program.
>>>
Ln: 172 Col: 4

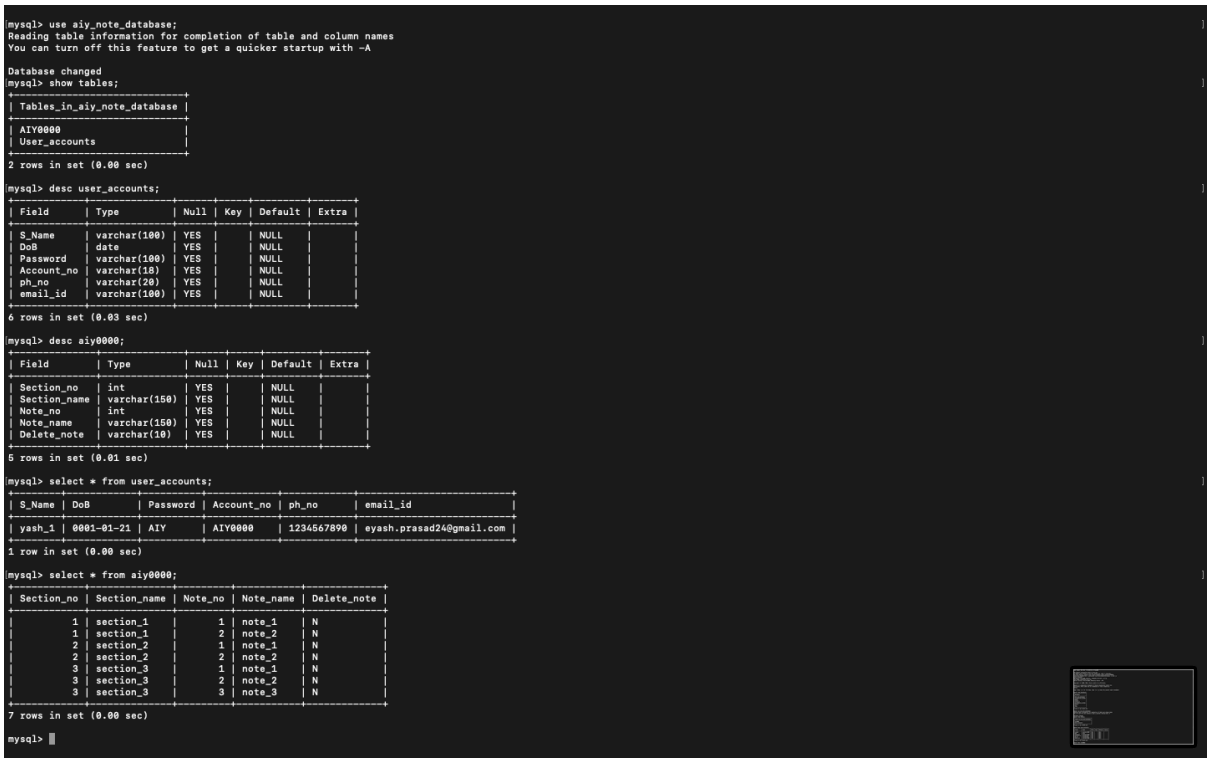
Program Output 2:

Create Account Window and Note Window

SQL screen:

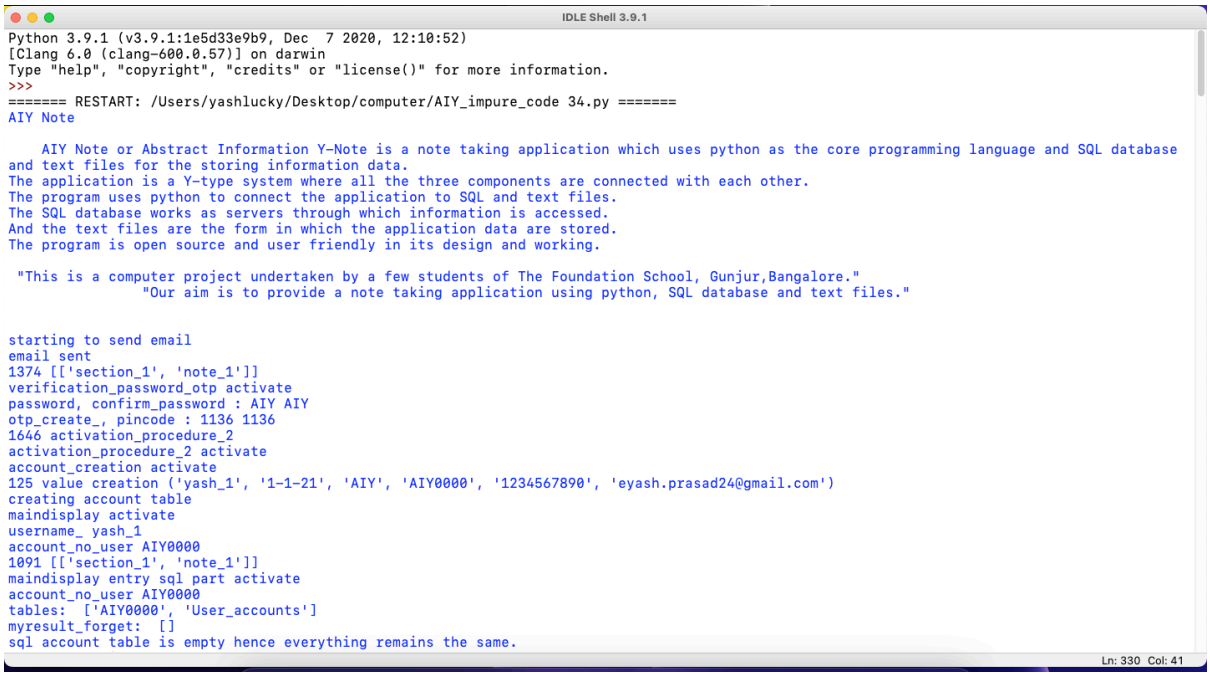


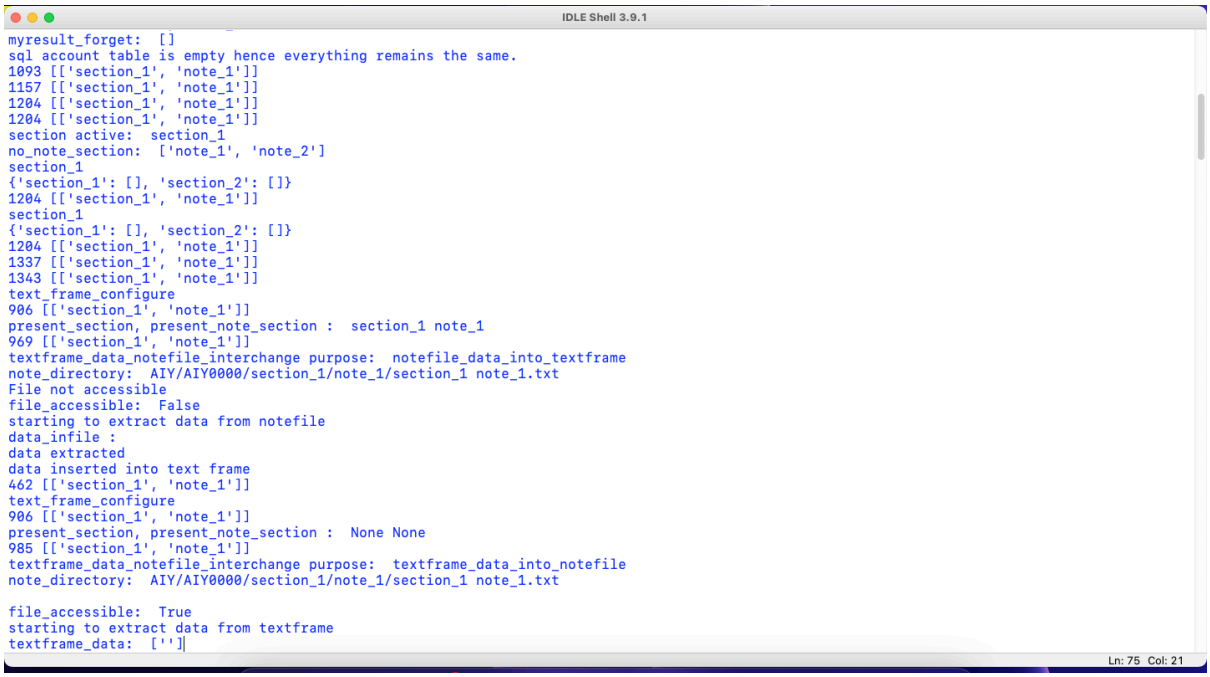
```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| AIY_note_database  |
| information_schema |
| mysql              |
| office             |
| office_2           |
| performance_schema |
| school             |
| sys                |
| test               |
+--------------------+
9 rows in set (0.01 sec)

mysql> use aiy_note_database;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------------------+
| Tables_in_aiy_note_database |
+-----------------------------+
| AIY0000                     |
| User_accounts               |
+-----------------------------+
2 rows in set (0.00 sec)

mysql> desc user_accounts;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| S_Name     | varchar(100) | YES  |     | NULL    |       |
| DoB        | date         | YES  |     | NULL    |       |
| Password   | varchar(100) | YES  |     | NULL    |       |
| Account_no | varchar(18)  | YES  |     | NULL    |       |
| ph_no      | varchar(20)  | YES  |     | NULL    |       |
| email_id   | varchar(100) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
6 rows in set (0.03 sec)

mysql> desc aiy0000;
```

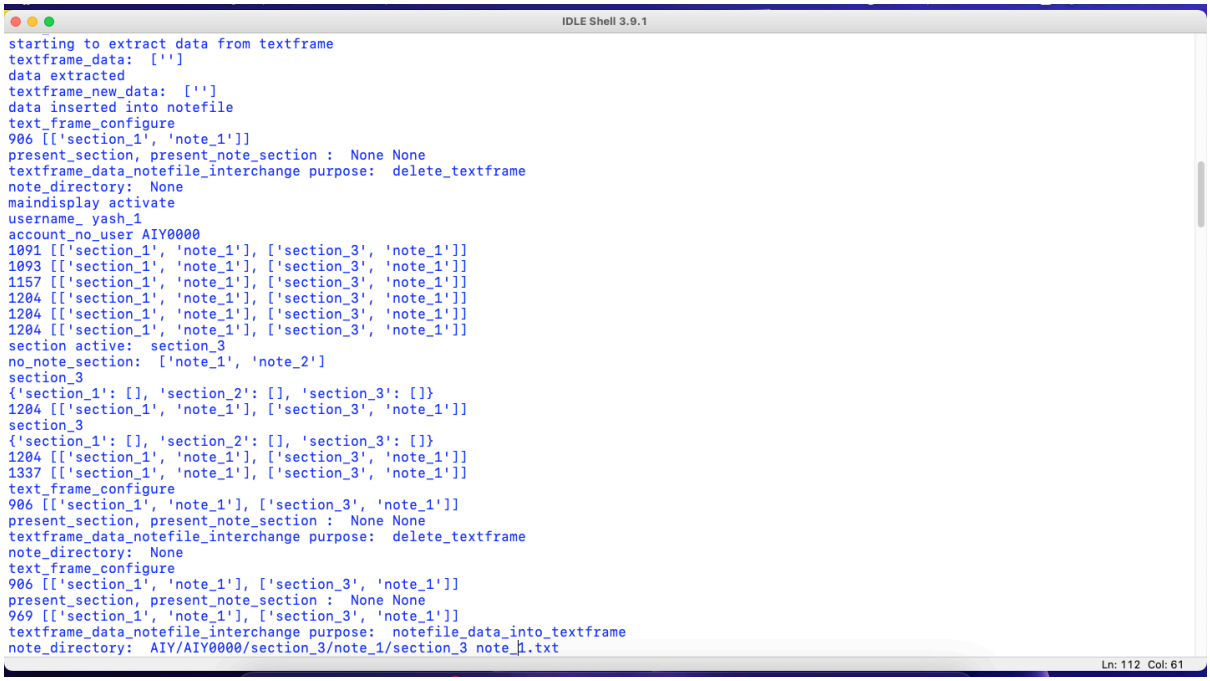


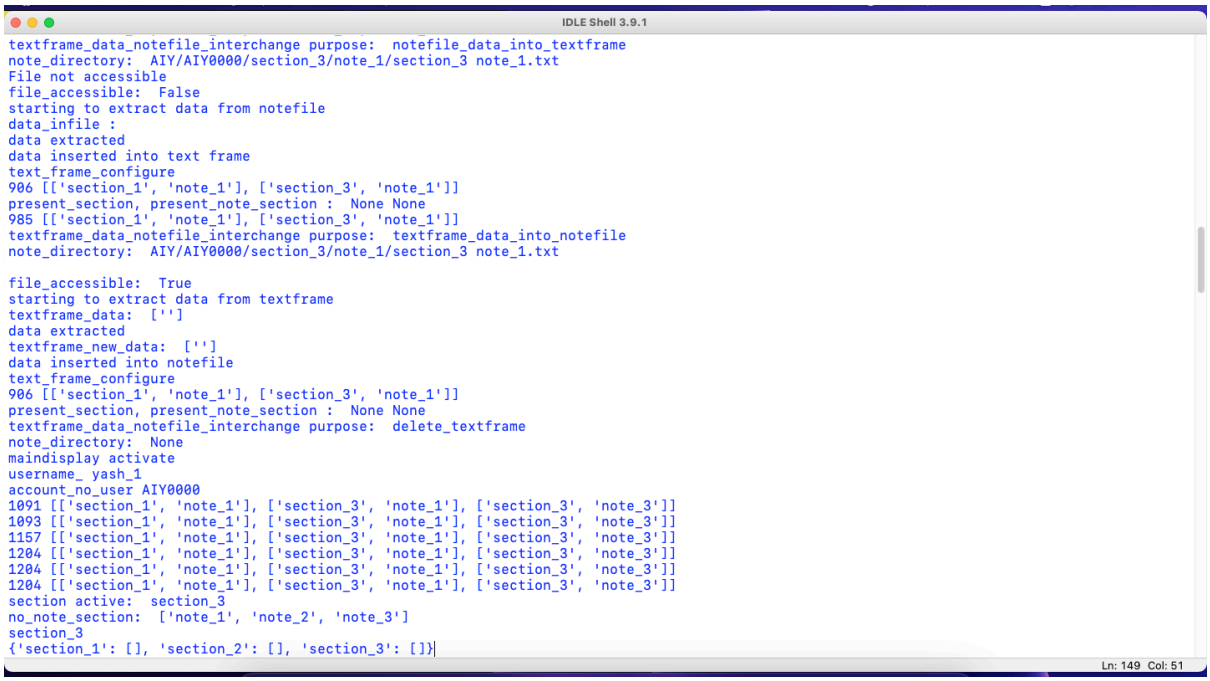
```
mysql> use aiy_note_database;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------------------+
| Tables_in_aiy_note_database |
+-----------------------------+
| AIY0000                     |
| User_accounts               |
+-----------------------------+
2 rows in set (0.00 sec)

mysql> desc user_accounts;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| S_Name     | varchar(100) | YES  |     | NULL    |       |
| DoB        | date         | YES  |     | NULL    |       |
| Password   | varchar(100) | YES  |     | NULL    |       |
| Account_no | varchar(18)  | YES  |     | NULL    |       |
| ph_no      | varchar(20)  | YES  |     | NULL    |       |
| email_id   | varchar(100) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
6 rows in set (0.03 sec)

mysql> desc aiy0000;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| Section_no   | int          | YES  |     | NULL    |       |
| Section_name | varchar(150) | YES  |     | NULL    |       |
| Note_no      | int          | YES  |     | NULL    |       |
| Note_name    | varchar(150) | YES  |     | NULL    |       |
| Delete_note  | varchar(10)  | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
5 rows in set (0.01 sec)

mysql> select * from user_accounts;
+--------+------------+----------+------------+------------+--------------------------+
| S_Name | DoB        | Password | Account_no | ph_no      | email_id                 |
+--------+------------+----------+------------+------------+--------------------------+
| yash_1 | 0001-01-21 | AIY      | AIY0000    | 1234567890 | eyash.prasad24@gmail.com |
+--------+------------+----------+------------+------------+--------------------------+
1 row in set (0.00 sec)

mysql> select * from aiy0000;
+------------+--------------+---------+-----------+-------------+
| Section_no | Section_name | Note_no | Note_name | Delete_note |
+------------+--------------+---------+-----------+-------------+
|          1 | section_1    |       1 | note_1    | N           |
|          1 | section_1    |       2 | note_2    | N           |
|          2 | section_2    |       1 | note_1    | N           |
|          2 | section_2    |       2 | note_2    | N           |
|          3 | section_3    |       1 | note_1    | N           |
|          3 | section_3    |       2 | note_2    | N           |
|          3 | section_3    |       3 | note_3    | N           |
+------------+--------------+---------+-----------+-------------+
7 rows in set (0.00 sec)

mysql>
```

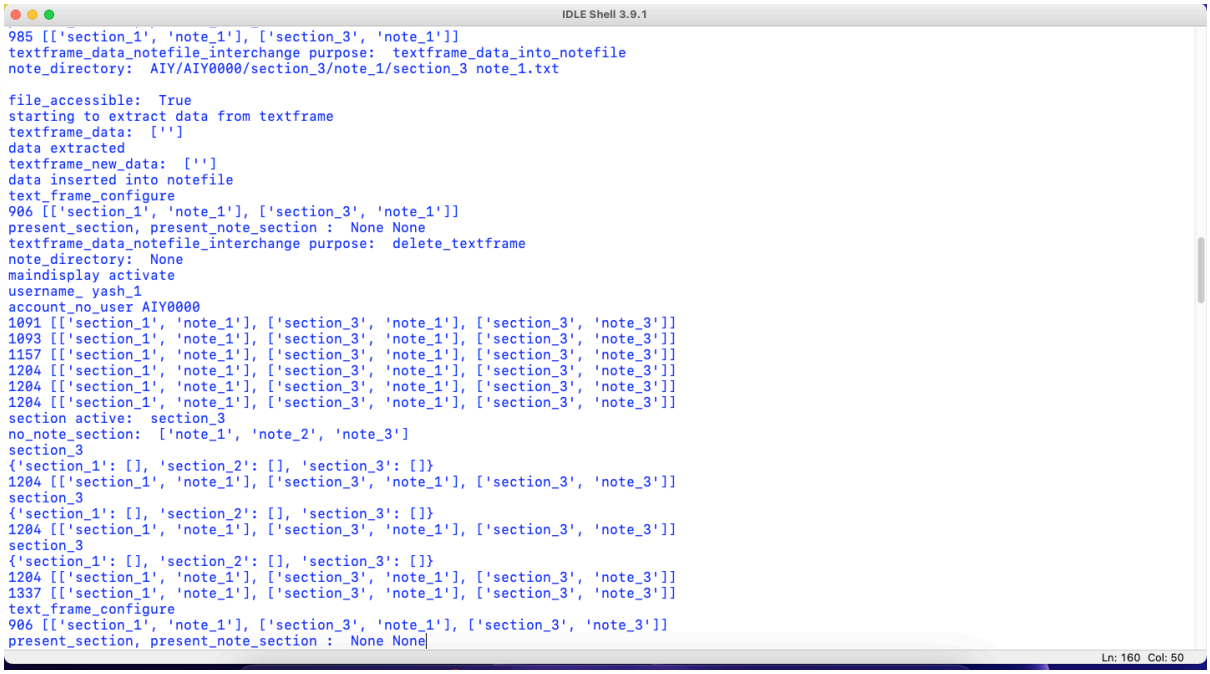
Python output screen:

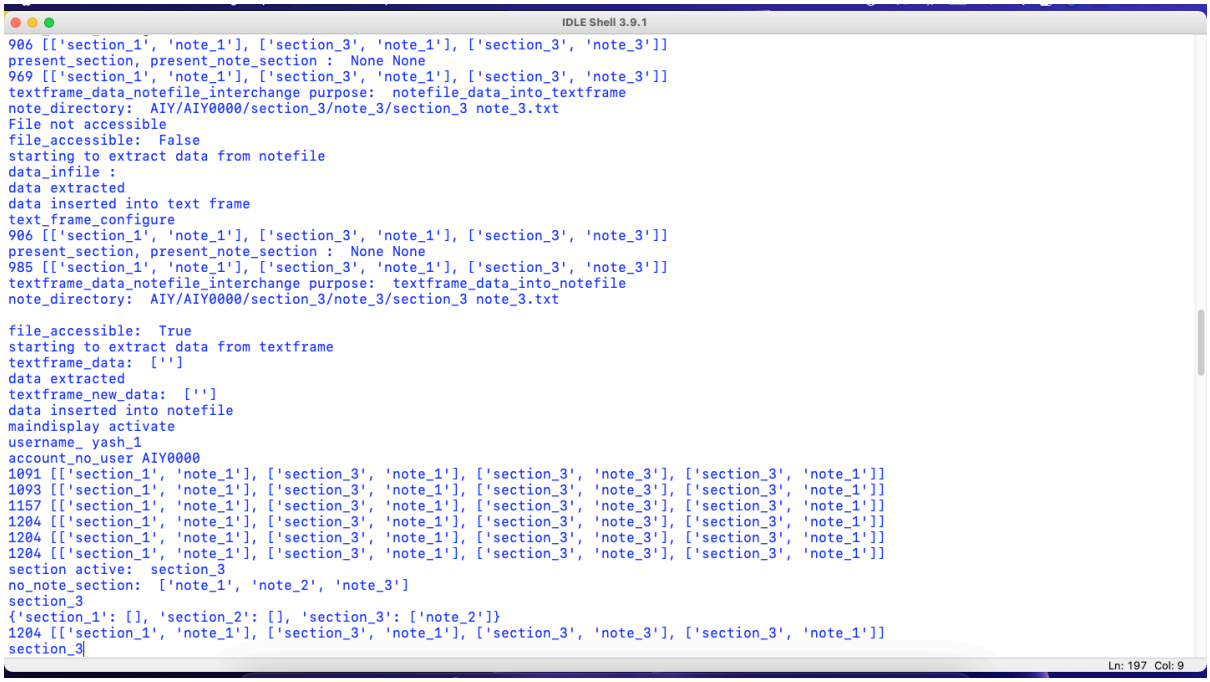


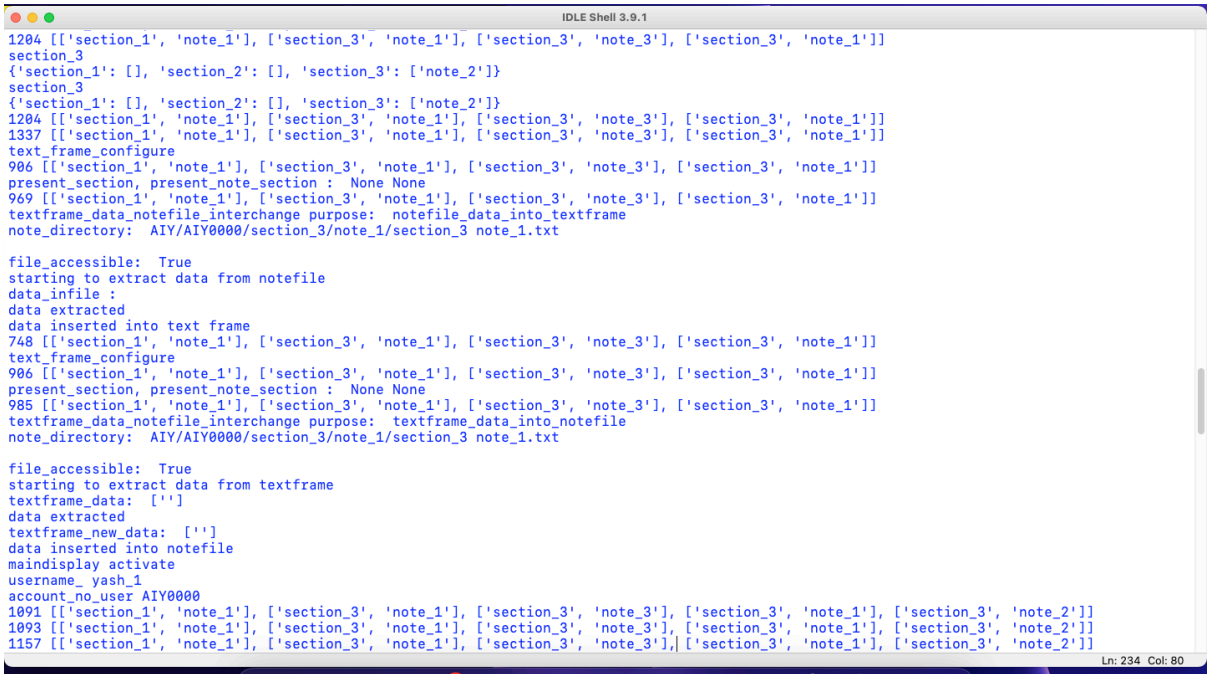



IDLE Shell 3.9.1
starting to extract data from textframe
textframe_data: ['']
data extracted
textframe_new_data: ['']
data inserted into notefile
text_frame_configure
906 [['section_1', 'note_1']]
present_section, present_note_section : None None
textframe_data_notefile_interchange purpose: delete_textframe
note_directory: None
maindisplay activate
username_ yash_1
account_no_user AIY0000
1091 [['section_1', 'note_1'], ['section_3', 'note_1']]
1093 [['section_1', 'note_1'], ['section_3', 'note_1']]
1157 [['section_1', 'note_1'], ['section_3', 'note_1']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1']]
section active: section_3
no_note_section: ['note_1', 'note_2']
section_3
{'section_1': [], 'section_2': [], 'section_3': []}
1204 [['section_1', 'note_1'], ['section_3', 'note_1']]
section_3
{'section_1': [], 'section_2': [], 'section_3': []}
1204 [['section_1', 'note_1'], ['section_3', 'note_1']]
1337 [['section_1', 'note_1'], ['section_3', 'note_1']]
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1']]
present_section, present_note_section : None None
textframe_data_notefile_interchange purpose: delete_textframe
note_directory: None
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1']]
present_section, present_note_section : None None
969 [['section_1', 'note_1'], ['section_3', 'note_1']]
textframe_data_notefile_interchange purpose: notefile_data_into_textframe
note_directory: AIY/AIY0000/section_3/note_1/section_3 note_1.txt
Ln: 112 Col: 61

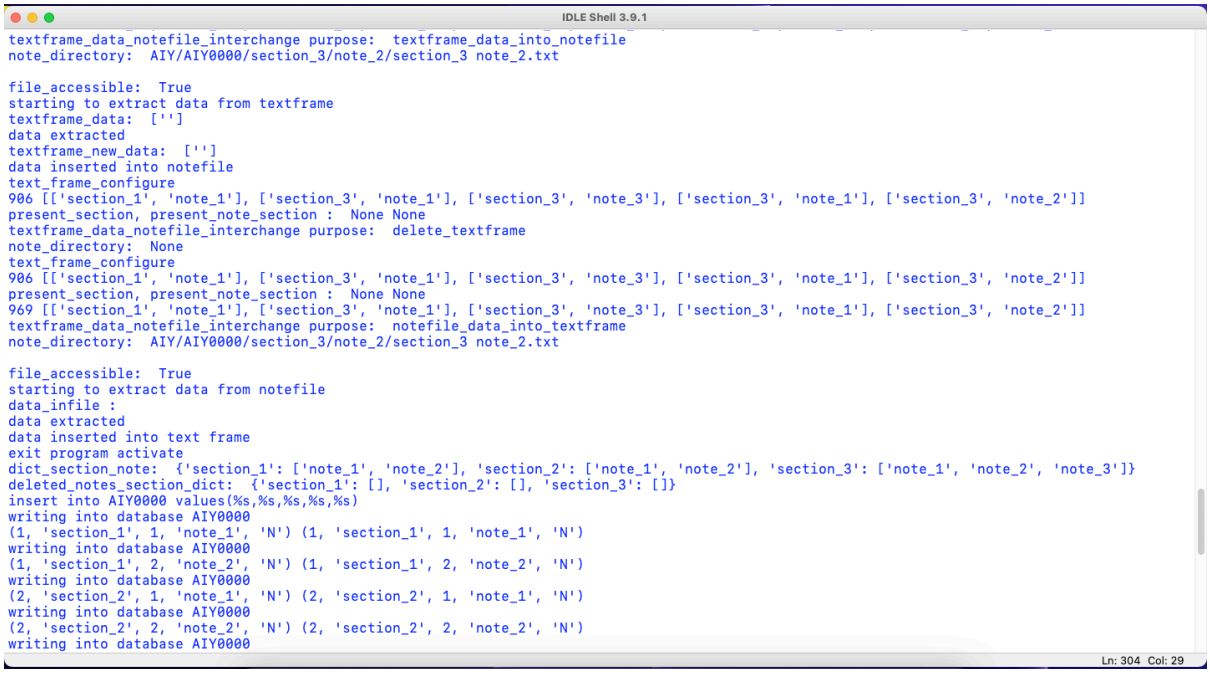

IDLE Shell 3.9.1
textframe_data_notefile_interchange purpose: notefile_data_into_textframe
note_directory: AIY/AIY0000/section_3/note_1/section_3 note_1.txt
File not accessible
file_accessible: False
starting to extract data from notefile
data_infile :
data extracted
data inserted into text frame
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1']]
present_section, present_note_section : None None
985 [['section_1', 'note_1'], ['section_3', 'note_1']]
textframe_data_notefile_interchange purpose: textframe_data_into_notefile
note_directory: AIY/AIY0000/section_3/note_1/section_3 note_1.txt
file_accessible: True
starting to extract data from textframe
textframe_data: ['']
data extracted
textframe_new_data: ['']
data inserted into notefile
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1']]
present_section, present_note_section : None None
textframe_data_notefile_interchange purpose: delete_textframe
note_directory: None
maindisplay activate
username_ yash_1
account_no_user AIY0000
1091 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
1093 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
1157 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
section active: section_3
no_note_section: ['note_1', 'note_2', 'note_3']
section_3
{'section_1': [], 'section_2': [], 'section_3': []}
Ln: 149 Col: 51

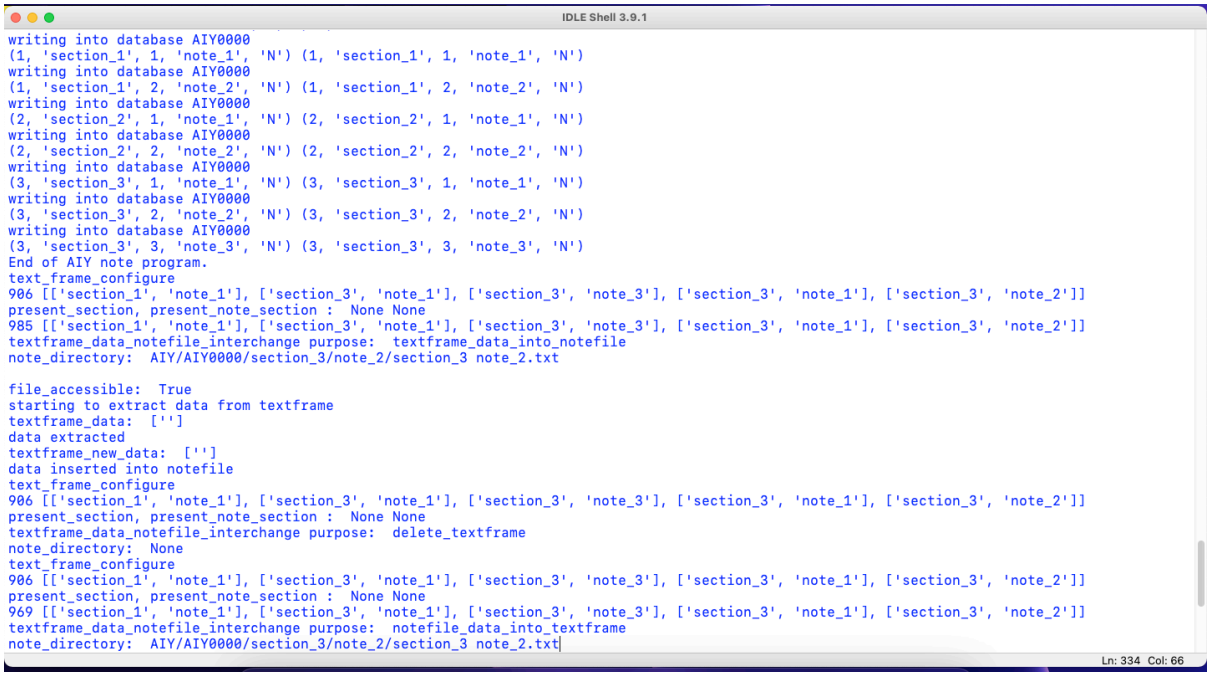

IDLE Shell 3.9.1
985 [['section_1', 'note_1'], ['section_3', 'note_1']]
textframe_data_notefile_interchange purpose: textframe_data_into_notefile
note_directory: AIY/AIY0000/section_3/note_1/section_3 note_1.txt
file_accessible: True
starting to extract data from textframe
textframe_data: ['']
data extracted
textframe_new_data: ['']
data inserted into notefile
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1']]
present_section, present_note_section : None None
textframe_data_notefile_interchange purpose: delete_textframe
note_directory: None
maindisplay activate
username_ yash_1
account_no_user AIY0000
1091 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
1093 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
1157 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
section active: section_3
no_note_section: ['note_1', 'note_2', 'note_3']
section_3
{'section_1': [], 'section_2': [], 'section_3': []}
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
section_3
{'section_1': [], 'section_2': [], 'section_3': []}
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
section_3
{'section_1': [], 'section_2': [], 'section_3': []}
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
1337 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
present_section, present_note_section : None None
Ln: 160 Col: 50

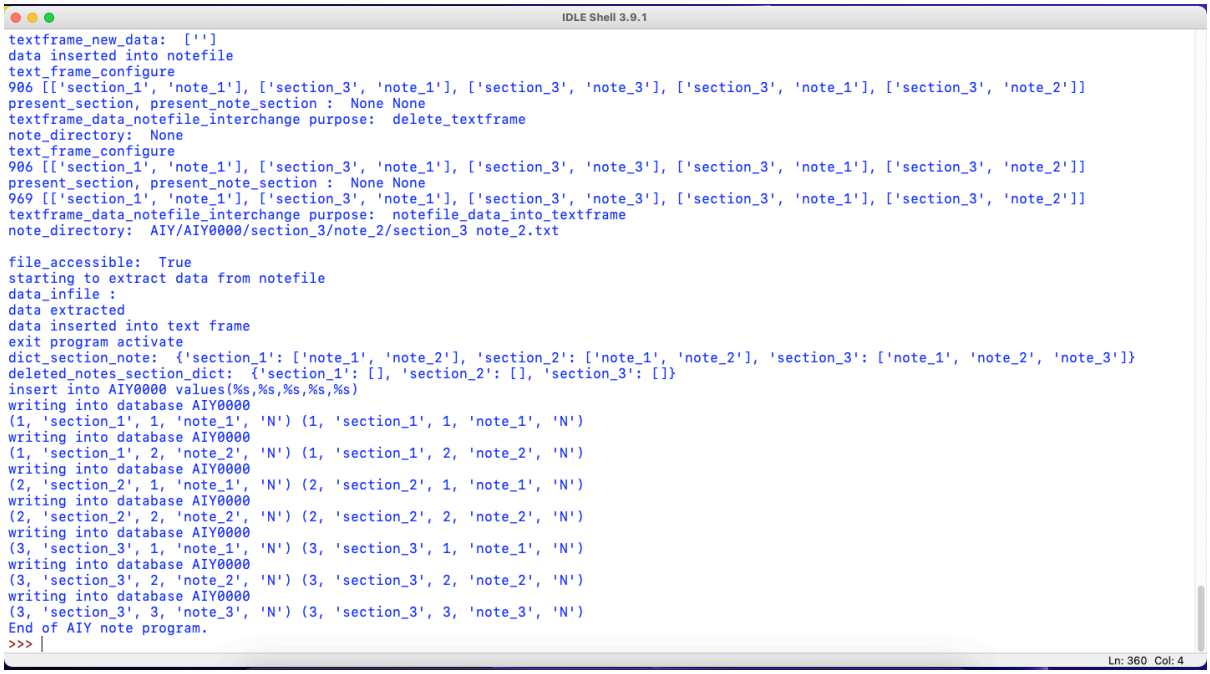

IDLE Shell 3.9.1
906 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
present_section, present_note_section : None None
969 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
textframe_data_notefile_interchange purpose: notefile_data_into_textframe
note_directory: AIY/AIY0000/section_3/note_3/section_3 note_3.txt
File not accessible
file_accessible: False
starting to extract data from notefile
data_infile :
data extracted
data inserted into text frame
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
present_section, present_note_section : None None
985 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3']]
textframe_data_notefile_interchange purpose: textframe_data_into_notefile
note_directory: AIY/AIY0000/section_3/note_3/section_3 note_3.txt
file_accessible: True
starting to extract data from textframe
textframe_data: ['']
data extracted
textframe_new_data: ['']
data inserted into notefile
maindisplay activate
username_ yash_1
account_no_user AIY0000
1091 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
1093 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
1157 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
section active: section_3
no_note_section: ['note_1', 'note_2', 'note_3']
section_3
{'section_1': [], 'section_2': [], 'section_3': ['note_2']}
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
section_3
Ln: 197 Col: 9


IDLE Shell 3.9.1
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
section_3
{'section_1': [], 'section_2': [], 'section_3': ['note_2']}
section_3
{'section_1': [], 'section_2': [], 'section_3': ['note_2']}
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
1337 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
present_section, present_note_section : None None
969 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
textframe_data_notefile_interchange purpose: notefile_data_into_textframe
note_directory: AIY/AIY0000/section_3/note_1/section_3 note_1.txt
file_accessible: True
starting to extract data from notefile
data_infile :
data extracted
data inserted into text frame
748 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
present_section, present_note_section : None None
985 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1']]
textframe_data_notefile_interchange purpose: textframe_data_into_notefile
note_directory: AIY/AIY0000/section_3/note_1/section_3 note_1.txt
file_accessible: True
starting to extract data from textframe
textframe_data: ['']
data extracted
textframe_new_data: ['']
data inserted into notefile
maindisplay activate
username_ yash_1
account_no_user AIY0000
1091 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
1093 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
1157 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
Ln: 234 Col: 80


IDLE Shell 3.9.1
username_ yash_1
account_no_user AIY0000
1091 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
1093 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
1157 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
section active: section_3
no_note_section: ['note_1', 'note_2', 'note_3']
section_3
{'section_1': [], 'section_2': [], 'section_3': []}
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
section_3
{'section_1': [], 'section_2': [], 'section_3': []}
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
section_3
{'section_1': [], 'section_2': [], 'section_3': []}
1204 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
1337 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
present_section, present_note_section : None None
969 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
textframe_data_notefile_interchange purpose: notefile_data_into_textframe
note_directory: AIY/AIY0000/section_3/note_2/section_3 note_2.txt
File not accessible
file_accessible: False
starting to extract data from notefile
data_infile :
data extracted
data inserted into text frame
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
present_section, present_note_section : None None
985 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
textframe_data_notefile_interchange purpose: textframe_data_into_notefile
note_directory: AIY/AIY0000/section_3/note_2/section_3 note_2.txt
Ln: 268 Col: 0


IDLE Shell 3.9.1
textframe_data_notefile_interchange purpose: textframe_data_into_notefile
note_directory: AIY/AIY0000/section_3/note_2/section_3 note_2.txt
file_accessible: True
starting to extract data from textframe
textframe_data: ['']
data extracted
textframe_new_data: ['']
data inserted into notefile
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
present_section, present_note_section : None None
textframe_data_notefile_interchange purpose: delete_textframe
note_directory: None
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
present_section, present_note_section : None None
969 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
textframe_data_notefile_interchange purpose: notefile_data_into_textframe
note_directory: AIY/AIY0000/section_3/note_2/section_3 note_2.txt
file_accessible: True
starting to extract data from notefile
data_infile :
data extracted
data inserted into text frame
exit program activate
dict_section_note: {'section_1': ['note_1', 'note_2'], 'section_2': ['note_1', 'note_2'], 'section_3': ['note_1', 'note_2', 'note_3']}
deleted_notes_section_dict: {'section_1': [], 'section_2': [], 'section_3': []}
insert into AIY0000 values(%s,%s,%s,%s,%s)
writing into database AIY0000
(1, 'section_1', 1, 'note_1', 'N') (1, 'section_1', 1, 'note_1', 'N')
writing into database AIY0000
(1, 'section_1', 2, 'note_2', 'N') (1, 'section_1', 2, 'note_2', 'N')
writing into database AIY0000
(2, 'section_2', 1, 'note_1', 'N') (2, 'section_2', 1, 'note_1', 'N')
writing into database AIY0000
(2, 'section_2', 2, 'note_2', 'N') (2, 'section_2', 2, 'note_2', 'N')
writing into database AIY0000
Ln: 304 Col: 29

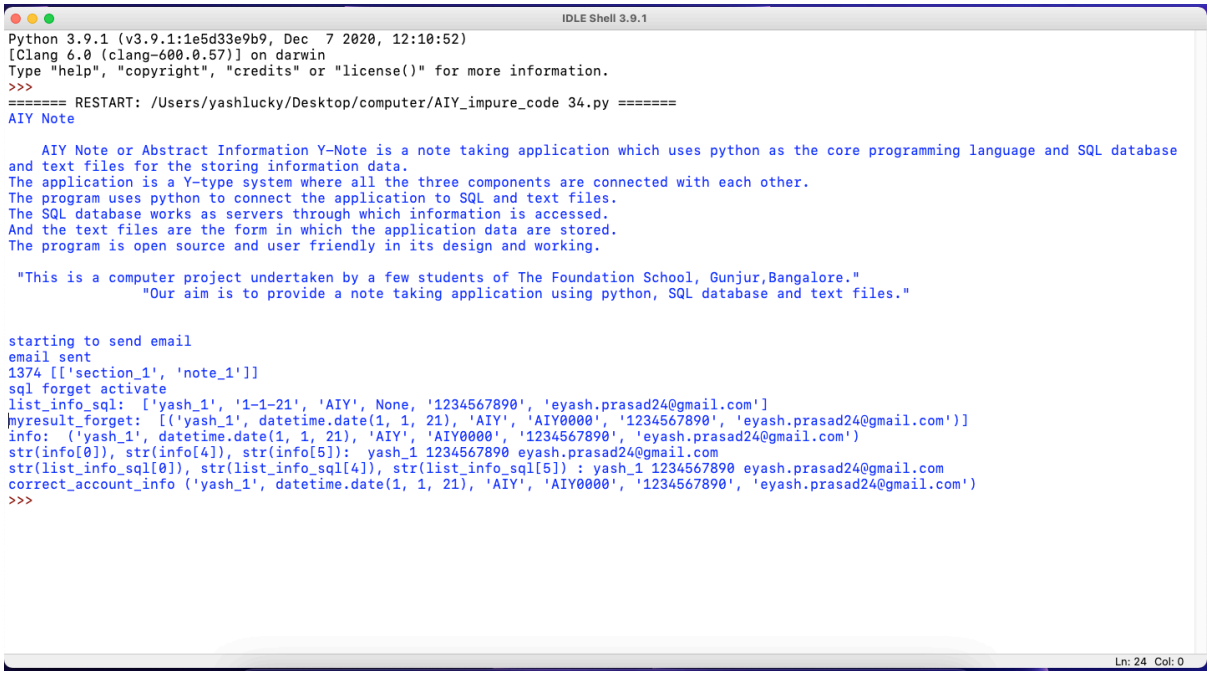

IDLE Shell 3.9.1
writing into database AIY0000
(1, 'section_1', 1, 'note_1', 'N') (1, 'section_1', 1, 'note_1', 'N')
writing into database AIY0000
(1, 'section_1', 2, 'note_2', 'N') (1, 'section_1', 2, 'note_2', 'N')
writing into database AIY0000
(2, 'section_2', 1, 'note_1', 'N') (2, 'section_2', 1, 'note_1', 'N')
writing into database AIY0000
(2, 'section_2', 2, 'note_2', 'N') (2, 'section_2', 2, 'note_2', 'N')
writing into database AIY0000
(3, 'section_3', 1, 'note_1', 'N') (3, 'section_3', 1, 'note_1', 'N')
writing into database AIY0000
(3, 'section_3', 2, 'note_2', 'N') (3, 'section_3', 2, 'note_2', 'N')
writing into database AIY0000
(3, 'section_3', 3, 'note_3', 'N') (3, 'section_3', 3, 'note_3', 'N')
End of AIY note program.
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
present_section, present_note_section : None None
985 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
textframe_data_notefile_interchange purpose: textframe_data_into_notefile
note_directory: AIY/AIY0000/section_3/note_2/section_3 note_2.txt
file_accessible: True
starting to extract data from textframe
textframe_data: ['']
data extracted
textframe_new_data: ['']
data inserted into notefile
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
present_section, present_note_section : None None
textframe_data_notefile_interchange purpose: delete_textframe
note_directory: None
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
present_section, present_note_section : None None
969 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
textframe_data_notefile_interchange purpose: notefile_data_into_textframe
note_directory: AIY/AIY0000/section_3/note_2/section_3 note_2.txt
Ln: 334 Col: 66


IDLE Shell 3.9.1
textframe_new_data: ['']
data inserted into notefile
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
present_section, present_note_section : None None
textframe_data_notefile_interchange purpose: delete_textframe
note_directory: None
text_frame_configure
906 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
present_section, present_note_section : None None
969 [['section_1', 'note_1'], ['section_3', 'note_1'], ['section_3', 'note_3'], ['section_3', 'note_1'], ['section_3', 'note_2']]
textframe_data_notefile_interchange purpose: notefile_data_into_textframe
note_directory: AIY/AIY0000/section_3/note_2/section_3 note_2.txt
file_accessible: True
starting to extract data from notefile
data_infile :
data extracted
data inserted into text frame
exit program activate
dict_section_note: {'section_1': ['note_1', 'note_2'], 'section_2': ['note_1', 'note_2'], 'section_3': ['note_1', 'note_2', 'note_3']}
deleted_notes_section_dict: {'section_1': [], 'section_2': [], 'section_3': []}
insert into AIY0000 values(%s,%s,%s,%s,%s)
writing into database AIY0000
(1, 'section_1', 1, 'note_1', 'N') (1, 'section_1', 1, 'note_1', 'N')
writing into database AIY0000
(1, 'section_1', 2, 'note_2', 'N') (1, 'section_1', 2, 'note_2', 'N')
writing into database AIY0000
(2, 'section_2', 1, 'note_1', 'N') (2, 'section_2', 1, 'note_1', 'N')
writing into database AIY0000
(2, 'section_2', 2, 'note_2', 'N') (2, 'section_2', 2, 'note_2', 'N')
writing into database AIY0000
(3, 'section_3', 1, 'note_1', 'N') (3, 'section_3', 1, 'note_1', 'N')
writing into database AIY0000
(3, 'section_3', 2, 'note_2', 'N') (3, 'section_3', 2, 'note_2', 'N')
writing into database AIY0000
(3, 'section_3', 3, 'note_3', 'N') (3, 'section_3', 3, 'note_3', 'N')
End of AIY note program.
>>>
Ln: 360 Col: 4

Program Output 3:

Forget Account Window

Python output screen:



Bibliography

Logo window:

- https://www.geeksforgeeks.org/how-to-resize-image-in-python-tkinter/
- https://stackoverflow.com/questions/31751464/how-do-i-close-an-image-opened-in-pillow
- https://www.activestate.com/resources/quick-reads/how-to-add-images-in-tkinter/

SQL and SQL connectivity:

- https://www.w3schools.com/sql/sql_syntax.asp
- https://www.tutorialspoint.com/python_data_access/python_mysql_create_table.htm
- https://www.w3schools.com/python/python_mysql_select.asp
- https://pynative.com/python-mysql-select-query-to-fetch-data/

GUI Website:

- https://www.tutorialspoint.com/python/python_gui_programming.htm
- https://www.geeksforgeeks.org/how-to-set-font-for-text-in-tkinter/
- https://stackoverflow.com/questions/55558649/python-how-to-change-text-of-already-created-tab-in-tkinter-notebook
- https://stackoverflow.com/questions/51758308/tkinter-making-a-button-panel-contained-in-a-class
- https://www.delftstack.com/howto/python-tkinter/how-to-bind-multiple-commands-to-tkinter-button/#:~:text=The%20Tkinter%20button%20has%20only,is%20bound%20to%20this%20command%20.&text=This%20lambda%20function%20will%20execute,and%20funcC%20one%20by%20one.
- https://www.delftstack.com/howto/python-tkinter/how-to-bind-multiple-commands-to-tkinter-button/#:~:text=The%20Tkinter%20button%20has%20only,is%20bound%20to%20this%20command%20.
- https://www.delftstack.com/howto/python-tkinter/how-to-close-a-tkinter-window-with-a-button/
- https://www.kite.com/python/answers/how-to-create-functions-with-optional-arguments-in-python#use-args-parameter
- https://stackoverflow.com/questions/9539921/how-do-i-create-a-python-function-with-optional-arguments
- https://www.geeksforgeeks.org/python-setting-and-retrieving-values-of-tkinter-variable/
- https://xiith.com/python-tkinter-program-to-get-value-from-combobox/

- https://stackoverflow.com/questions/41522427/get-combobox-value-in-python
- https://www.pythontutorial.net/tkinter/tkinter-combobox/
- https://www.geeksforgeeks.org/tkinter-separator-widget/
- https://www.youtube.com/watch?v=3E_fK5hCUnI
- https://www.youtube.com/watch?v=OPUSBBD2OJw
- https://stackoverflow.com/questions/23901641/wxpython-how-to-check-if-a-frame-is-already-visible
- https://stackoverflow.com/questions/42212920/how-to-add-tag-to-a-new-line-in-tkinter-text
- https://stackoverflow.com/questions/17746817/how-to-read-the-inputline-by-line-from-a-multiline-tkinter-textbox-in-python
- https://www.youtube.com/watch?v=VmlgrrXAqb4
- https://www.delftstack.com/howto/python-tkinter/how-to-clear-tkinter-text-box-widget/
- https://www.youtube.com/watch?v=UlQRXJWUNBA
- https://www.youtube.com/watch?v=ZS2_v_zsPTg
- https://stackoverflow.com/questions/8449053/how-to-make-menubar-cut-copy-paste-with-python-tkinter
- https://www.youtube.com/watch?v=4bMU1xAolSg
- https://www.geeksforgeeks.org/python-panedwindow-widget-in-tkinter/#:~:text=The%20PanedWindow%20widget%20is%20a,or%20more%20child%20widgets%20panes.
- https://www.youtube.com/watch?v=9Hyltpk2tSM
- https://www.youtube.com/watch?v=GW0w97-urkg
- https://www.youtube.com/watch?v=yqo_PcecZXQ

Python programming and Others:

- https://www.w3schools.com/colors/colors_picker.asp
- https://www.w3schools.com/howto/howto_css_searchbar.asp
- https://www.geeksforgeeks.org/python-opencv-cv2-imwrite-method/
- https://stackoverflow.com/questions/855493/why-do-i-get-a-referenced-before-assignment-error-when-assigning-to-a-global-v
- https://www.codegrepper.com/code-examples/python/python+create+directory+if+it+doesn%27t+exist
- https://www.tutorialspoint.com/python/python_date_time.htm
- https://stackoverflow.com/questions/82831/how-do-i-check-whether-a-file-exists-without-exceptions
- https://linuxize.com/post/python-check-if-file-exists/

Email connectivity:

- https://www.tutorialspoint.com/python/python_sending_email.htm
- https://www.codeproject.com/Questions/885446/How-to-resolve-this-issue-failure-sending-mail-csh
- https://stackoverflow.com/questions/13115724/new-to-python-gmail-smtp-error

- https://www.youtube.com/watch?v=JRCJ6RtE3xU
- https://www.youtube.com/results?search_query=sending+emails+through+python+
- https://github.com/CoreyMSchafer/code_snippets/blob/master/Python/Emails/mail-demo.py
- https://www.techwithtim.net/tutorials/sending-emails-with-python/
- https://stackoverflow.com/questions/51925384/unable-to-get-local-issuer-certificate-when-using-requests-in-python
- https://www.youtube.com/watch?v=_z2GGBILZmQ
- https://www.youtube.com/watch?v=vLoHf6oKKPI
- https://www.youtube.com/watch?v=YUCrtEvxEVY
- https://www.youtube.com/watch?v=YJagw7GfZcE