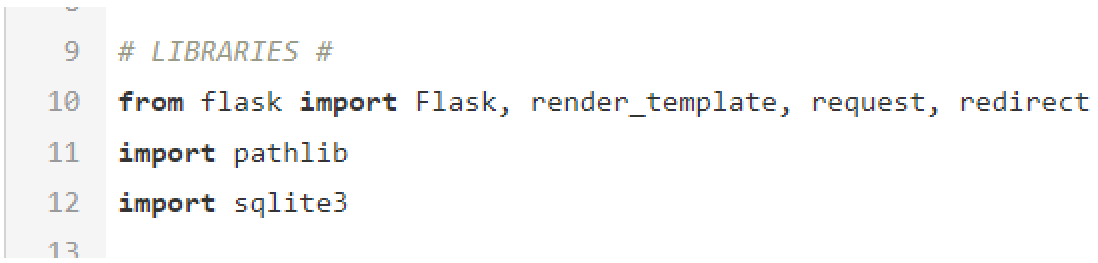
Techniques used for development:

- Additional libraries
    - Pathlib, flask, sqlite3
- Creation of database
    - Creates members database
- Add data to database
    - Add the new members to the members database
- Search for specific data in database
    - Fuzzys searches the values
- Data extraction
    - Extract data from the database to show on view members
- Delete data from database
    - Delete data
- 2 dimensional arrays
    - When fetching data from the database
- Parsing a text file
    - Determining the correct login information
- Recursion
    - The login page when an incorrect username or password is entered
- For loops
    - Used many times throughout the program

<u>Additional Libraries</u>

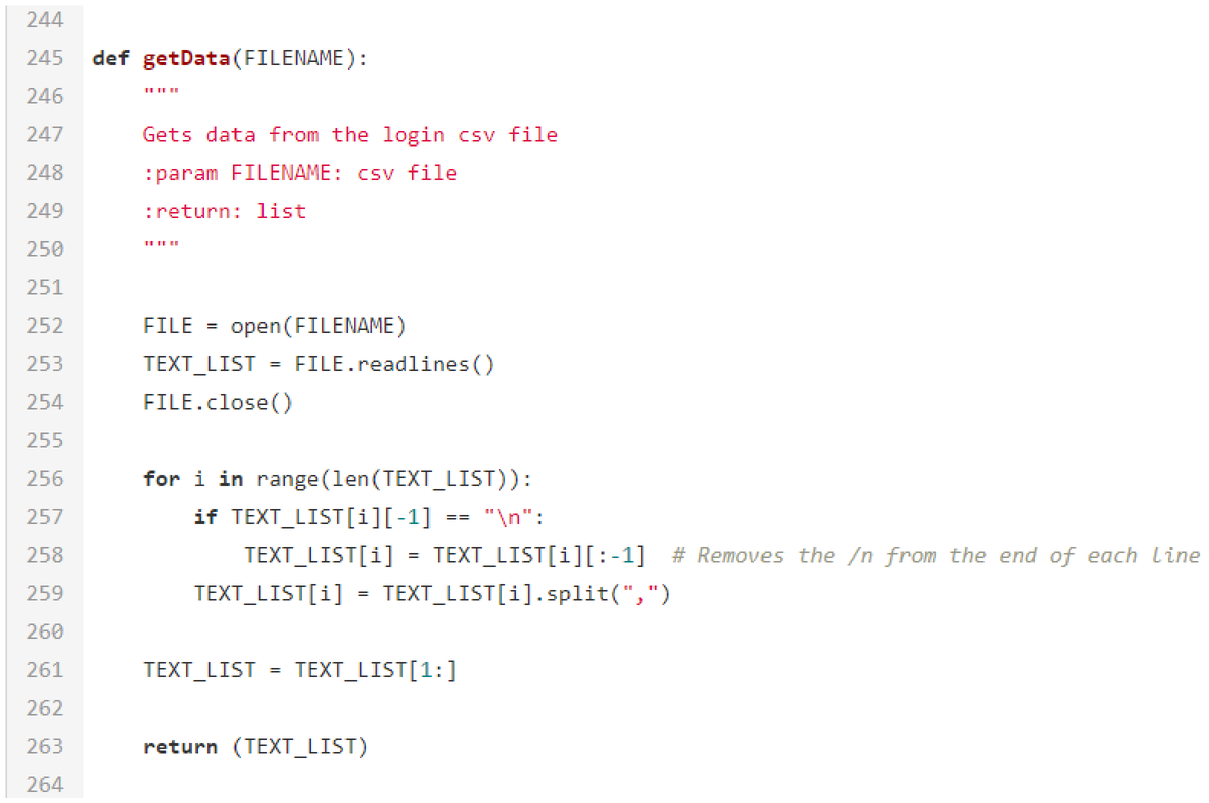


```python
# LIBRARIES #
from flask import Flask, render_template, request, redirect
import pathlib
import sqlite3
```

The three libraries I used for this project were flask, pathlib, and sqlite3.

- flask is a microframework for the frontend of the program
- sqlite3 is a library that allows me to execute SQL commands from Python
- pathlib is used to detect if a datafile already exists. If it does, pathlib will ensure that another database file doesn't get created

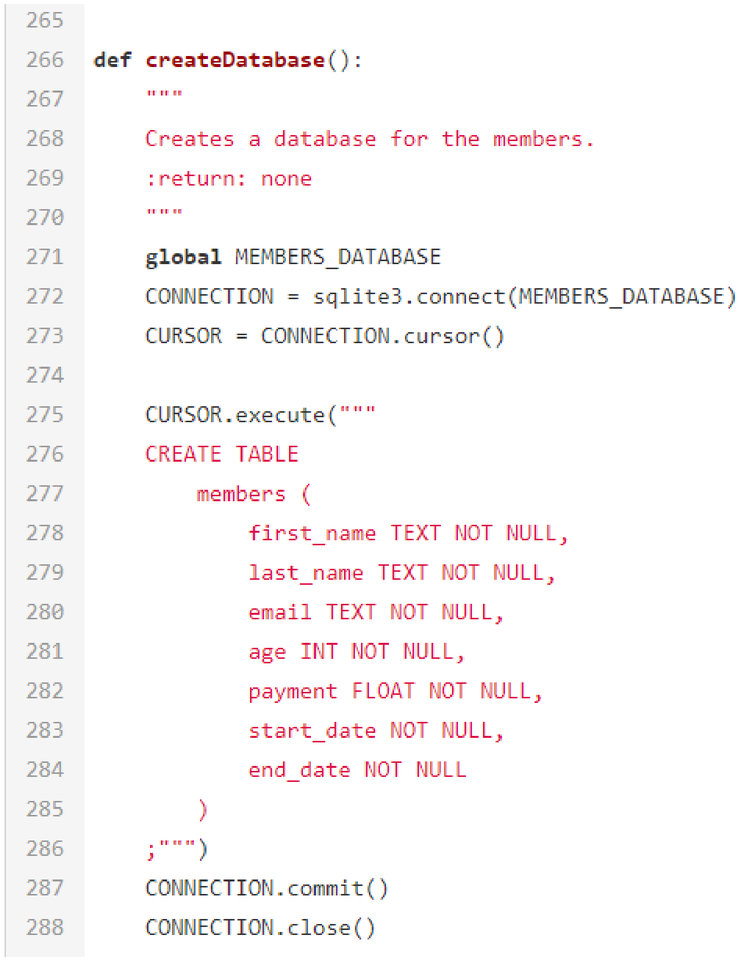
<u>Reading a text file</u>



```python

def getData(FILENAME):
    """
    Gets data from the login csv file
    :param FILENAME: csv file
    :return: list
    """

    FILE = open(FILENAME)
    TEXT_LIST = FILE.readlines()
    FILE.close()

    for i in range(len(TEXT_LIST)):
        if TEXT_LIST[i][-1] == "\n":
            TEXT_LIST[i] = TEXT_LIST[i][:-1]  # Removes the /n from the end of each line
        TEXT_LIST[i] = TEXT_LIST[i].split(",")

    TEXT_LIST = TEXT_LIST[1:]

    return (TEXT_LIST)

```

In this function, the login CSV file provided by my client is being read, and turned into a 2D array for later use in the login page where the program can check if the correct login information is entered.

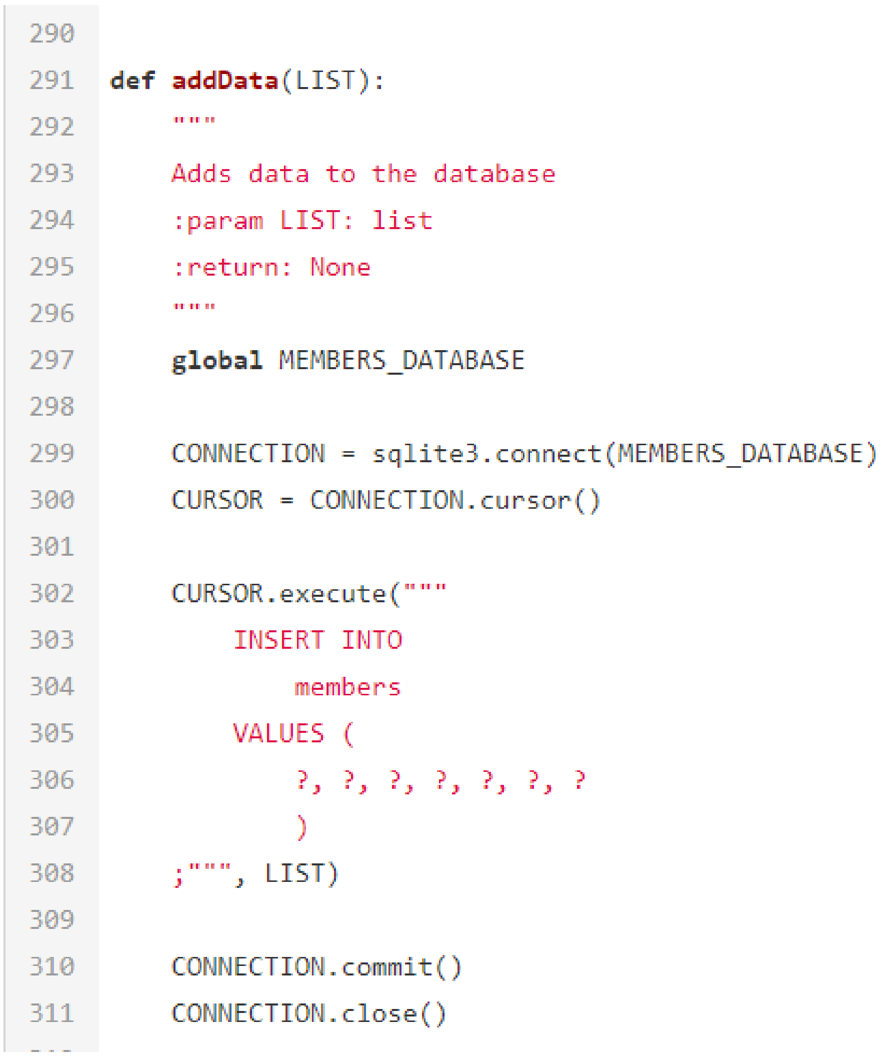
<u>Creation of database</u>



```python

def createDatabase():
    """
    Creates a database for the members.
    :return: none
    """
    global MEMBERS_DATABASE
    CONNECTION = sqlite3.connect(MEMBERS_DATABASE)
    CURSOR = CONNECTION.cursor()

    CURSOR.execute("""
    CREATE TABLE
        members (
            first_name TEXT NOT NULL,
            last_name TEXT NOT NULL,
            email TEXT NOT NULL,
            age INT NOT NULL,
            payment FLOAT NOT NULL,
            start_date NOT NULL,
            end_date NOT NULL
        )
    ;""")
    CONNECTION.commit()
    CONNECTION.close()
```

Here, a database called “members” is being created using the sqlite3 library. This is the database where all the information of the members of the organization will be stored and will be displayed in the view members tab in the website.

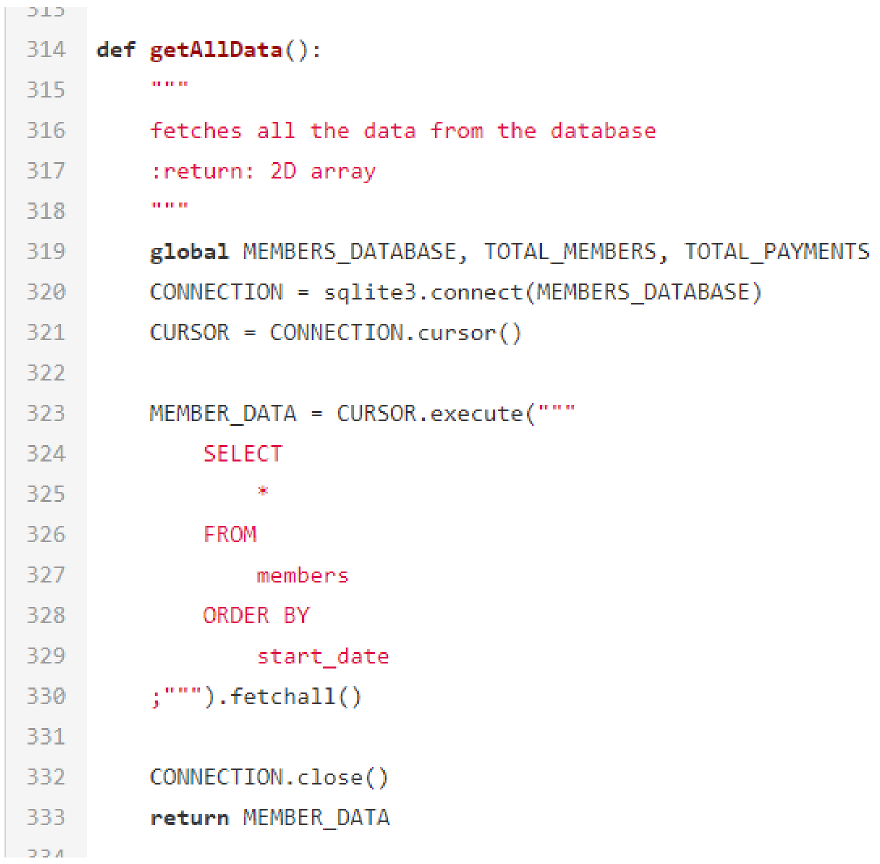
Add data to database



```python

def addData(LIST):
    """
    Adds data to the database
    :param LIST: list
    :return: None
    """
    global MEMBERS_DATABASE

    CONNECTION = sqlite3.connect(MEMBERS_DATABASE)
    CURSOR = CONNECTION.cursor()

    CURSOR.execute("""
        INSERT INTO
            members
        VALUES (
            ?, ?, ?, ?, ?, ?, ?
            )
    ;""", LIST)

    CONNECTION.commit()
    CONNECTION.close()
```

Here, data from the LIST parameter is being added into the database. The ‘?’ is used to add the information because it is a much more secure method of adding the data since it prevents SQL injection.

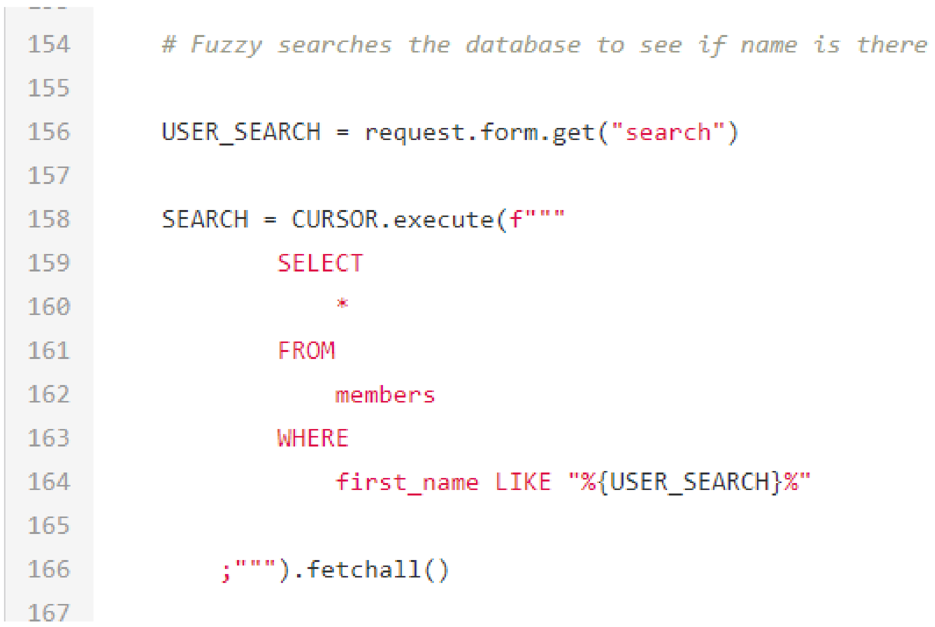
<u>Data Extraction</u>



```python
def getAllData():
    """
    fetches all the data from the database
    :return: 2D array
    """
    global MEMBERS_DATABASE, TOTAL_MEMBERS, TOTAL_PAYMENTS
    CONNECTION = sqlite3.connect(MEMBERS_DATABASE)
    CURSOR = CONNECTION.cursor()

    MEMBER_DATA = CURSOR.execute("""
        SELECT
            *
        FROM
            members
        ORDER BY
            start_date
    ;""").fetchall()

    CONNECTION.close()
    return MEMBER_DATA
```

This function fetches all the data from the database and orders it by the date the membership is bought. This comes useful in the view members tab where it is all displayed.

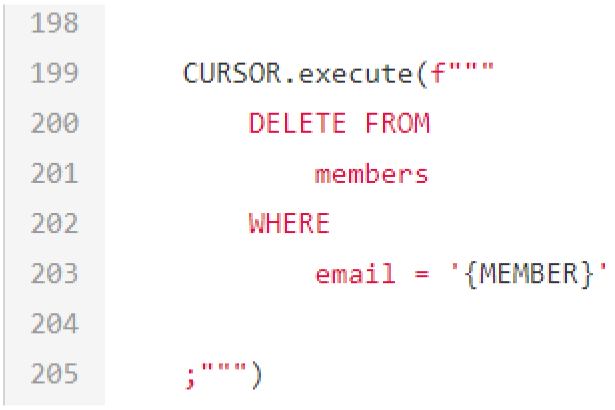
Search for data in database:



```python
    # Fuzzy searches the database to see if name is there

    USER_SEARCH = request.form.get("search")

    SEARCH = CURSOR.execute(f"""
            SELECT
                *
            FROM
                members
            WHERE
                first_name LIKE "%{USER_SEARCH}%"

        ;""").fetchall()

```

Fuzzy searches the database on what the user types and then fetches all the members that match the fuzzy search.

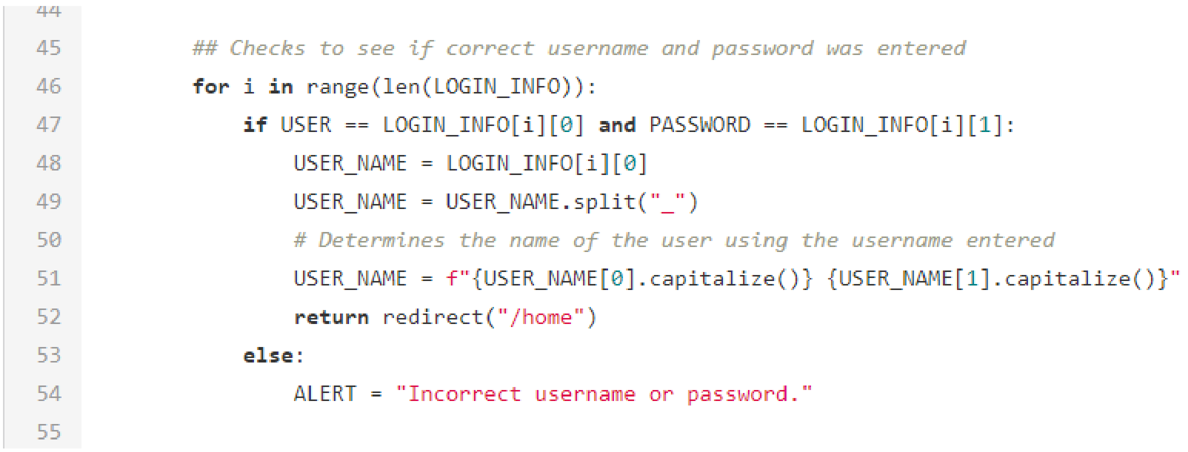
<u>Delete from database:</u>



```
CURSOR.execute(f"""
    DELETE FROM
        members
    WHERE
        email = '{MEMBER}'

;""")
```

Here, the member is being deleted from the “members” database on the parameter that the email is the email of the member being deleted.

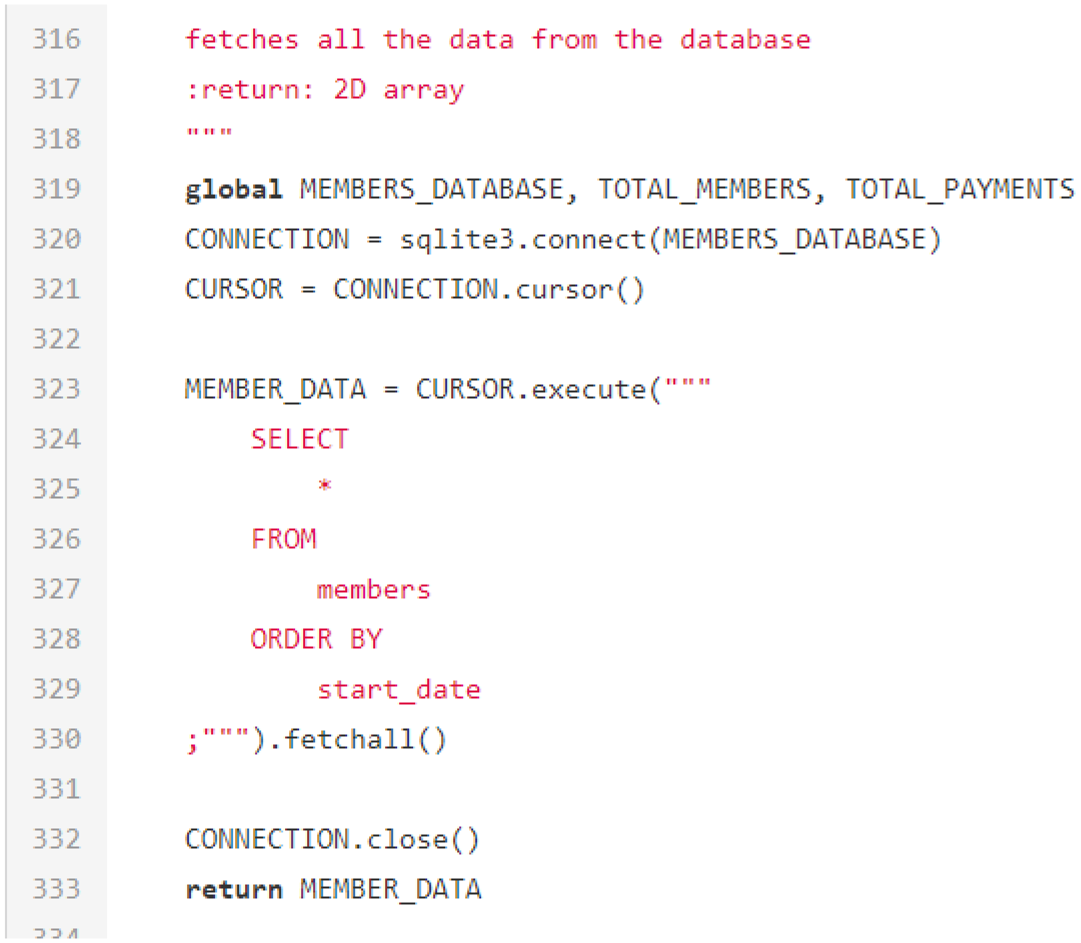
<u>For Loops</u>



```
## Checks to see if correct username and password was entered
for i in range(len(LOGIN_INFO)):
    if USER == LOGIN_INFO[i][0] and PASSWORD == LOGIN_INFO[i][1]:
        USER_NAME = LOGIN_INFO[i][0]
        USER_NAME = USER_NAME.split("_")
        # Determines the name of the user using the username entered
        USER_NAME = f"{USER_NAME[0].capitalize()} {USER_NAME[1].capitalize()}"
        return redirect("/home")
    else:
        ALERT = "Incorrect username or password."

```

One of the instances that a for loop is used in the program. Here the for loop is being used to determine if the correct login information is being entered in the login page of the website.

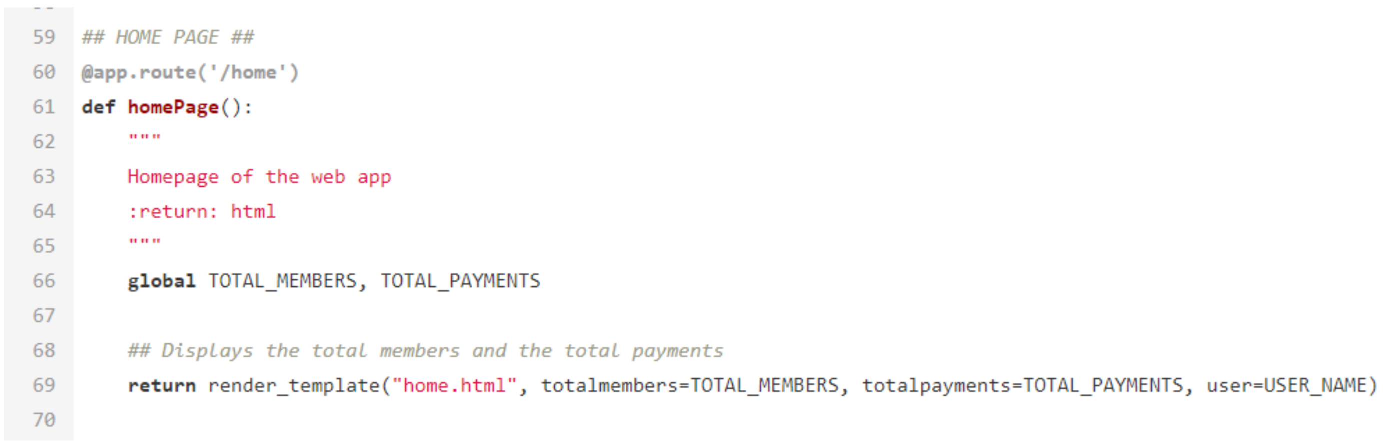
<u>2 dimensional arrays</u>



```
    fetches all the data from the database
    :return: 2D array
    """
    global MEMBERS_DATABASE, TOTAL_MEMBERS, TOTAL_PAYMENTS
    CONNECTION = sqlite3.connect(MEMBERS_DATABASE)
    CURSOR = CONNECTION.cursor()

    MEMBER_DATA = CURSOR.execute("""
        SELECT
            *
        FROM
            members
        ORDER BY
            start_date
    ;""").fetchall()

    CONNECTION.close()
    return MEMBER_DATA
```

This function fetches all the data from the database, and returns that in a 2 dimensional array for that data to be used.

Recursion



```
## HOME PAGE ##
@app.route('/home')
def homePage():
    """
    Homepage of the web app
    :return: html
    """
    global TOTAL_MEMBERS, TOTAL_PAYMENTS

    ## Displays the total members and the total payments
    return render_template("home.html", totalmembers=TOTAL_MEMBERS, totalpayments=TOTAL_PAYMENTS, user=USER_NAME)

```

Here the home page function is returning itself, which is defined as being recursion. The home page function is known as a recursive function.