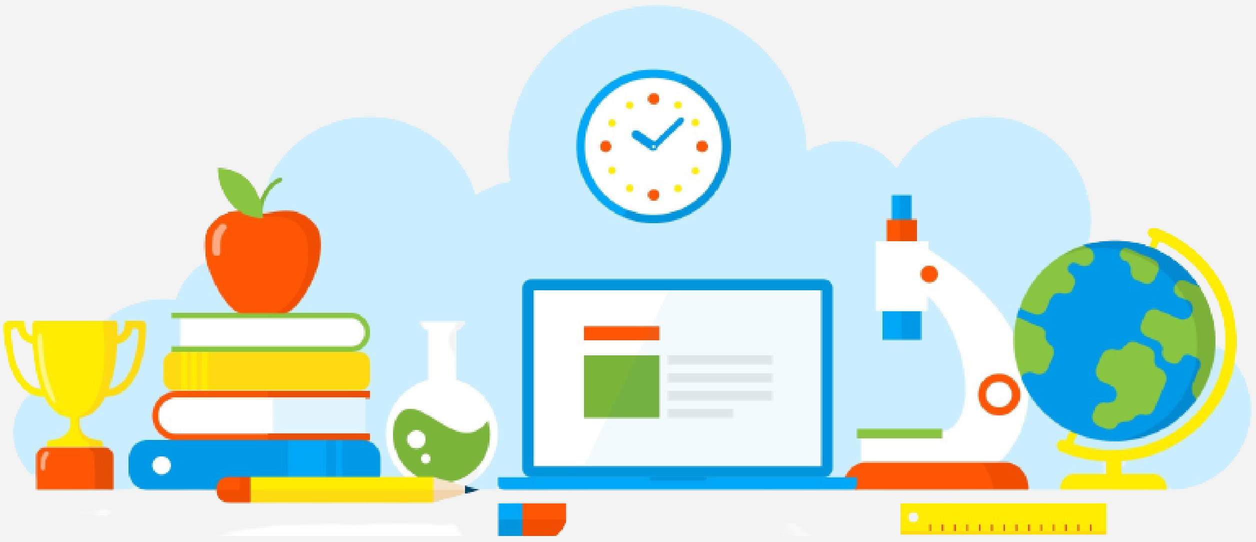


COMPUTER SCIENCE PROJECT WORK

LIBRARY MANAGEMENT SYSTEM

Made By:

Manas Madan	: 20
Sanyam Singh	: 35
Uday Kalra	: 41

CERTIFICATE

This is to certify that

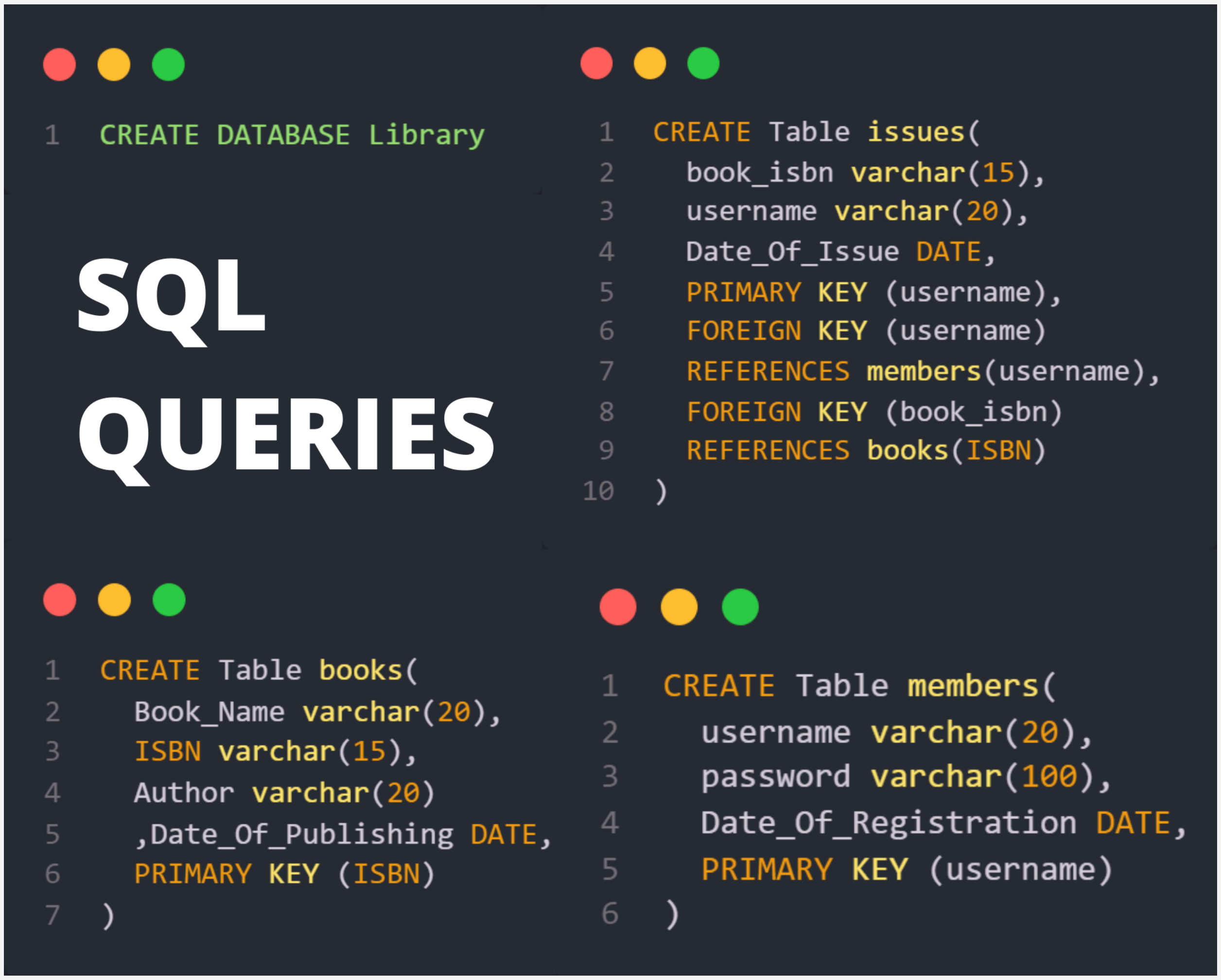
Manas Madan, Sanyam Singh & **Uday Kalra** of class 12 - C have successfully completed the **Library Management Project** under the Guidance of **Mrs. Nutan Parashar**.

Acknowledgement

In the accomplishment of this project successfully we would like to express our special thanks to our Computer Science teacher
Mrs. Nutan Parashar for their able guidance and support in completing of our project

Date. 28 12 2021

SQL QUERIES USED:



```sql
CREATE DATABASE Library
```

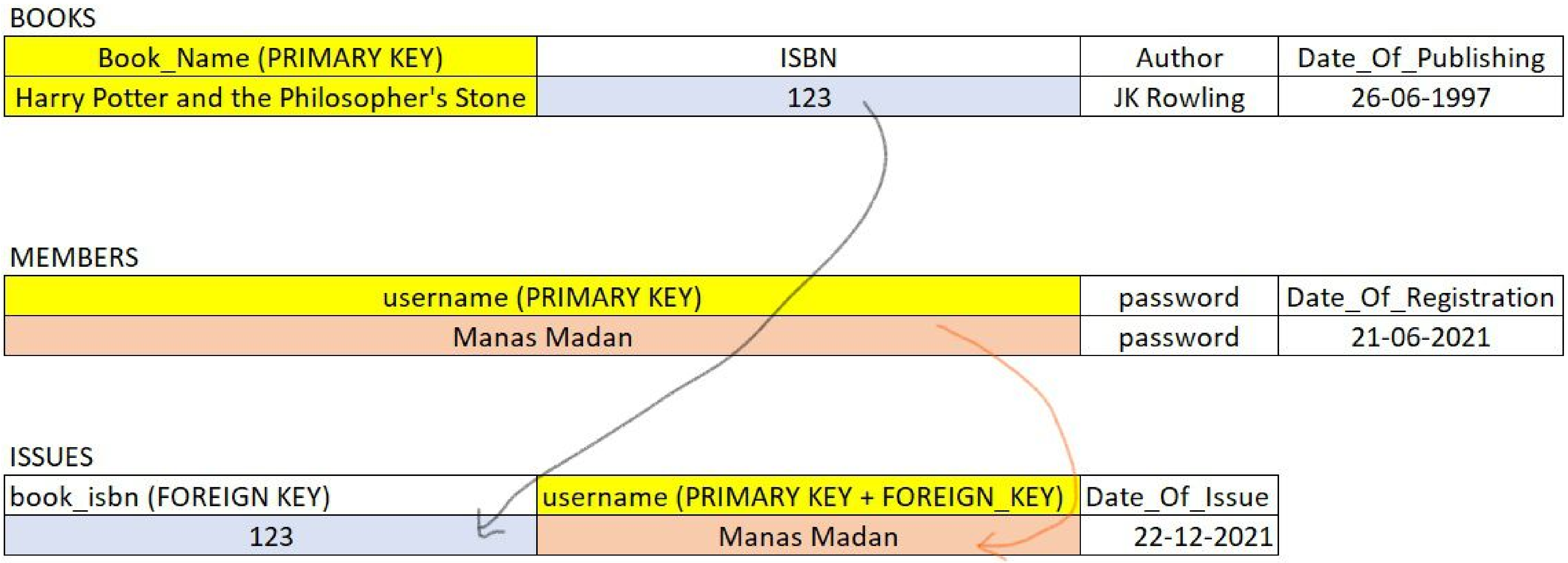
```sql
CREATE Table issues(
  book_isbn varchar(15),
  username varchar(20),
  Date_Of_Issue DATE,
  PRIMARY KEY (username),
  FOREIGN KEY (username)
  REFERENCES members(username),
  FOREIGN KEY (book_isbn)
  REFERENCES books(ISBN)
)
```

```sql
CREATE Table books(
  Book_Name varchar(20),
  ISBN varchar(15),
  Author varchar(20)
  ,Date_Of_Publishing DATE,
  PRIMARY KEY (ISBN)
)
```

```sql
CREATE Table members(
  username varchar(20),
  password varchar(100),
  Date_Of_Registration DATE,
  PRIMARY KEY (username)
)
```

A Separate File setup.py has been created that does all this for you.

1. It Creates The Database Library
2. Then It Creates Three Tables Under Database Library:
    a. Books
    b. Members
    c. Issues



BOOKS

Book_Name (PRIMARY KEY)	ISBN	Author	Date_Of_Publishing
Harry Potter and the Philosopher's Stone	123	JK Rowling	26-06-1997

MEMBERS

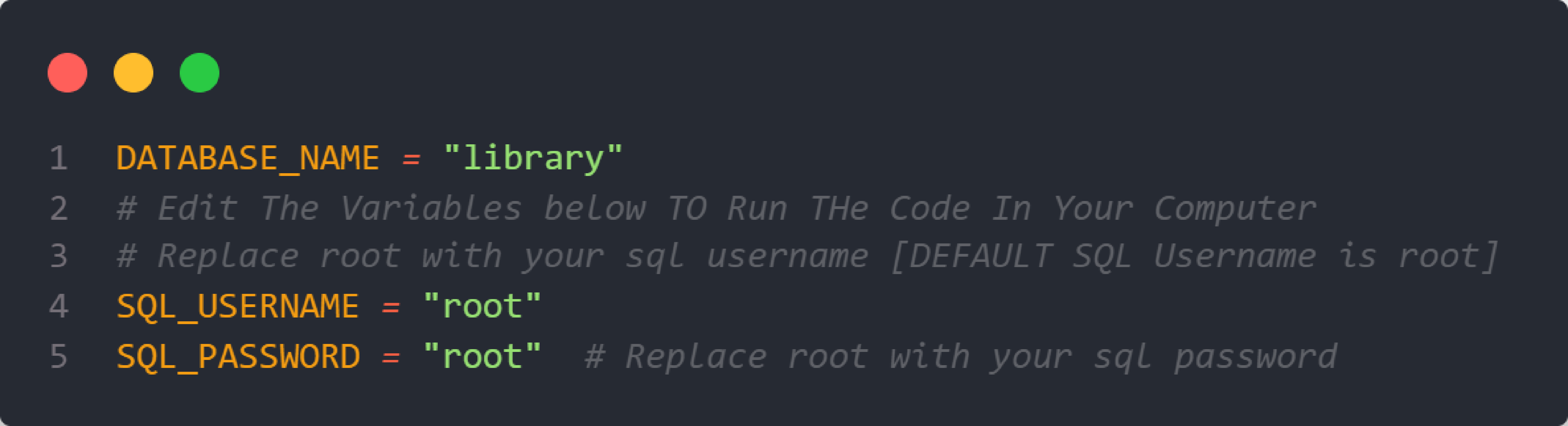
username (PRIMARY KEY)	password	Date_Of_Registration
Manas Madan	password	21-06-2021

ISSUES

book_isbn (FOREIGN KEY)	username (PRIMARY KEY + FOREIGN_KEY)	Date_Of_Issue
123	Manas Madan	22-12-2021

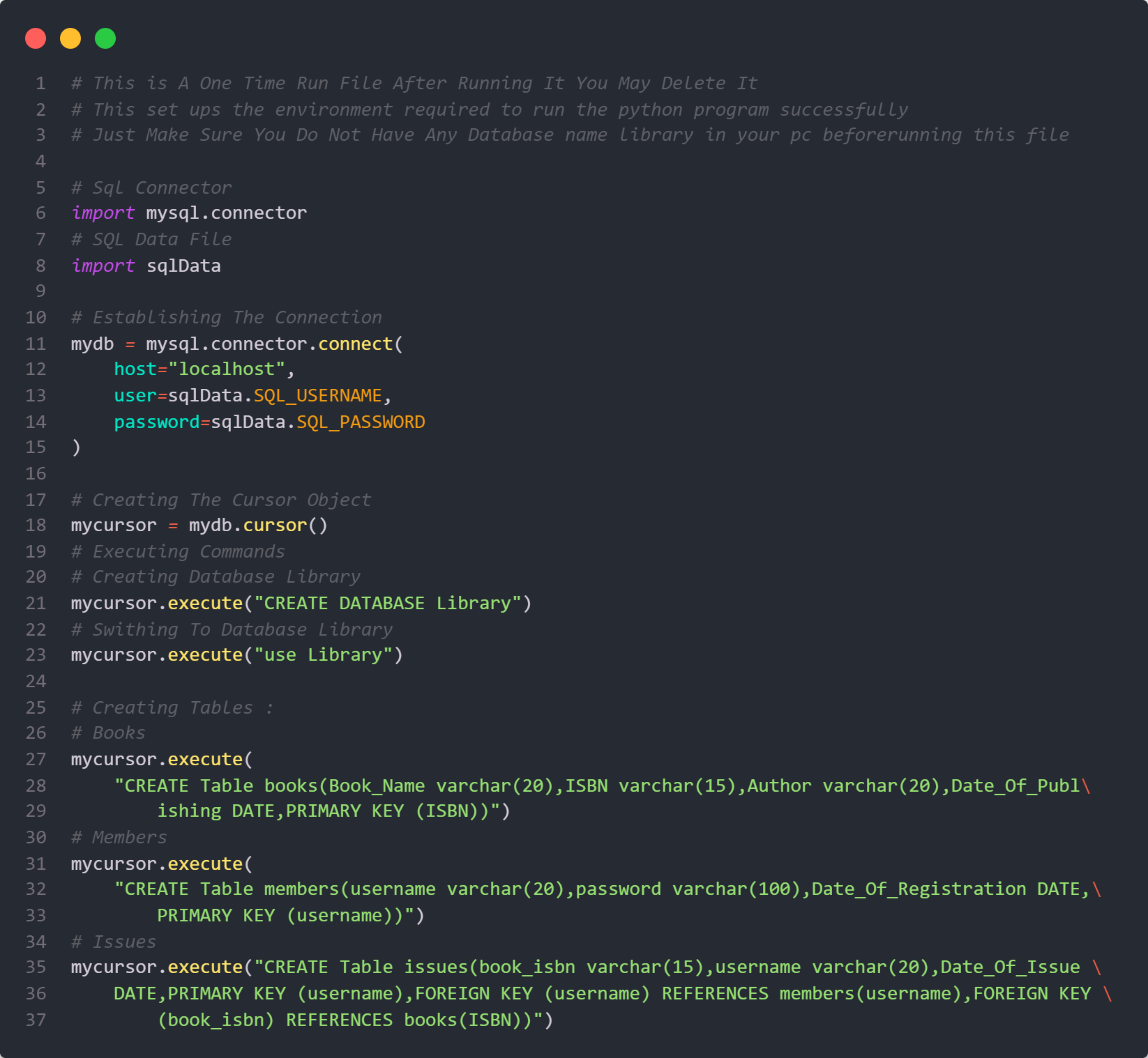
Python Code

sqlData.py



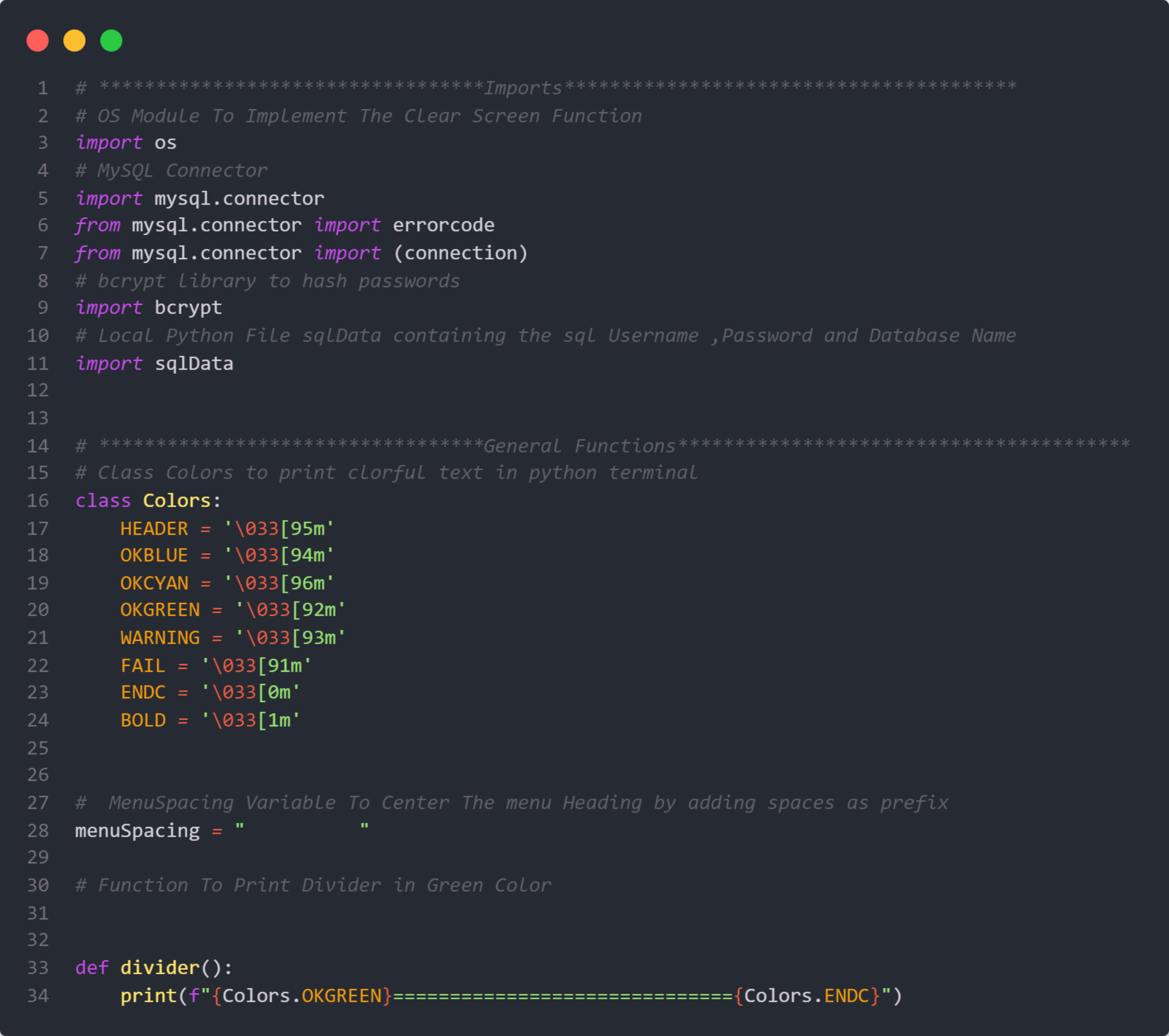
```python
DATABASE_NAME = "library"
# Edit The Variables below TO Run THe Code In Your Computer
# Replace root with your sql username [DEFAULT SQL Username is root]
SQL_USERNAME = "root"
SQL_PASSWORD = "root"  # Replace root with your sql password
```

setup.py

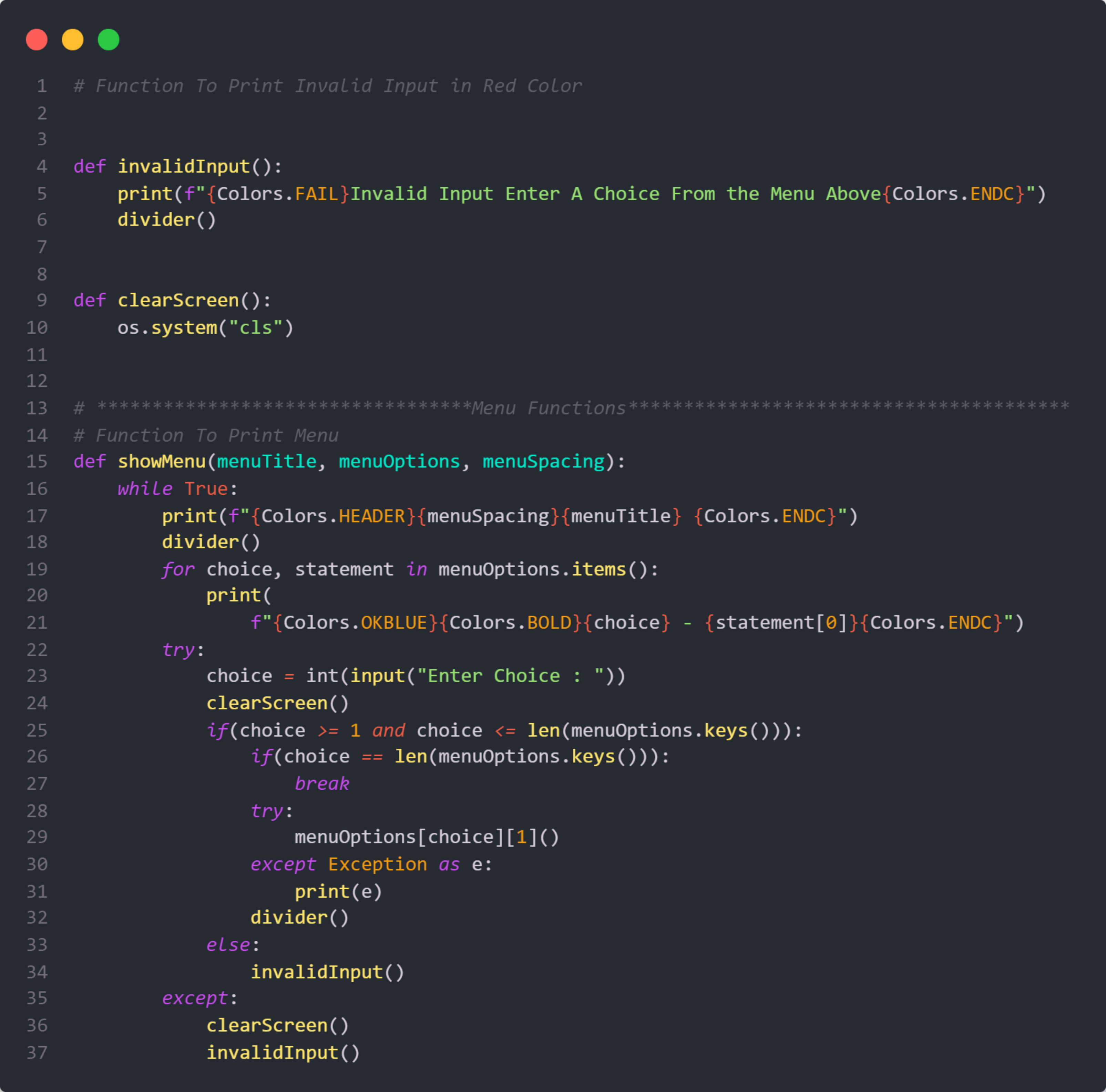


```python
# This is A One Time Run File After Running It You May Delete It
# This set ups the environment required to run the python program successfully
# Just Make Sure You Do Not Have Any Database name library in your pc beforerunning this file

# Sql Connector
import mysql.connector
# SQL Data File
import sqlData

# Establishing The Connection
mydb = mysql.connector.connect(
    host="localhost",
    user=sqlData.SQL_USERNAME,
    password=sqlData.SQL_PASSWORD
)

# Creating The Cursor Object
mycursor = mydb.cursor()
# Executing Commands
# Creating Database Library
mycursor.execute("CREATE DATABASE Library")
# Swithing To Database Library
mycursor.execute("use Library")

# Creating Tables :
# Books
mycursor.execute(
    "CREATE Table books(Book_Name varchar(20),ISBN varchar(15),Author varchar(20),Date_Of_Publ\
        ishing DATE,PRIMARY KEY (ISBN))")
# Members
mycursor.execute(
    "CREATE Table members(username varchar(20),password varchar(100),Date_Of_Registration DATE,\
        PRIMARY KEY (username))")
# Issues
mycursor.execute("CREATE Table issues(book_isbn varchar(15),username varchar(20),Date_Of_Issue \
    DATE,PRIMARY KEY (username),FOREIGN KEY (username) REFERENCES members(username),FOREIGN KEY \
        (book_isbn) REFERENCES books(ISBN))")
```

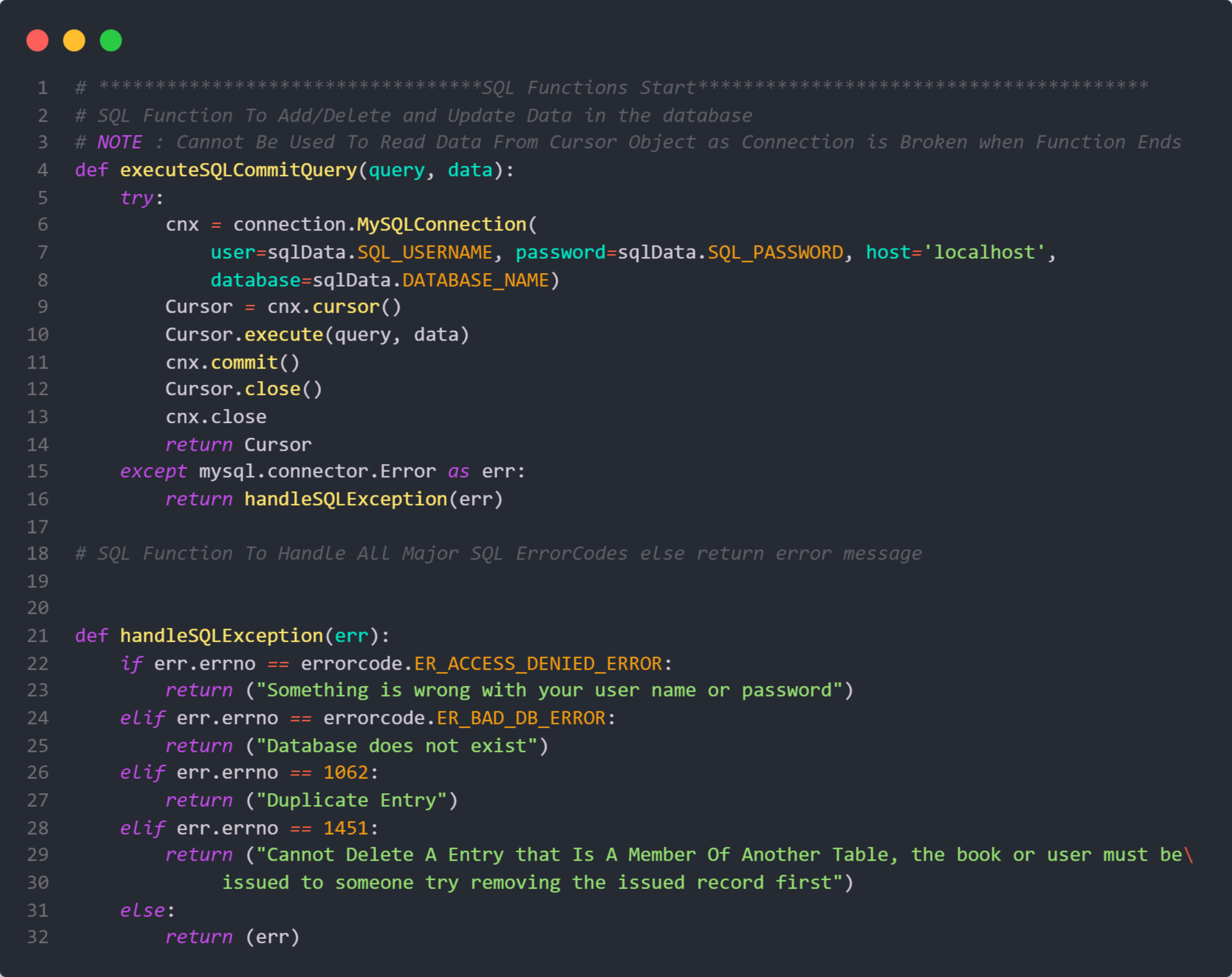
utils.py



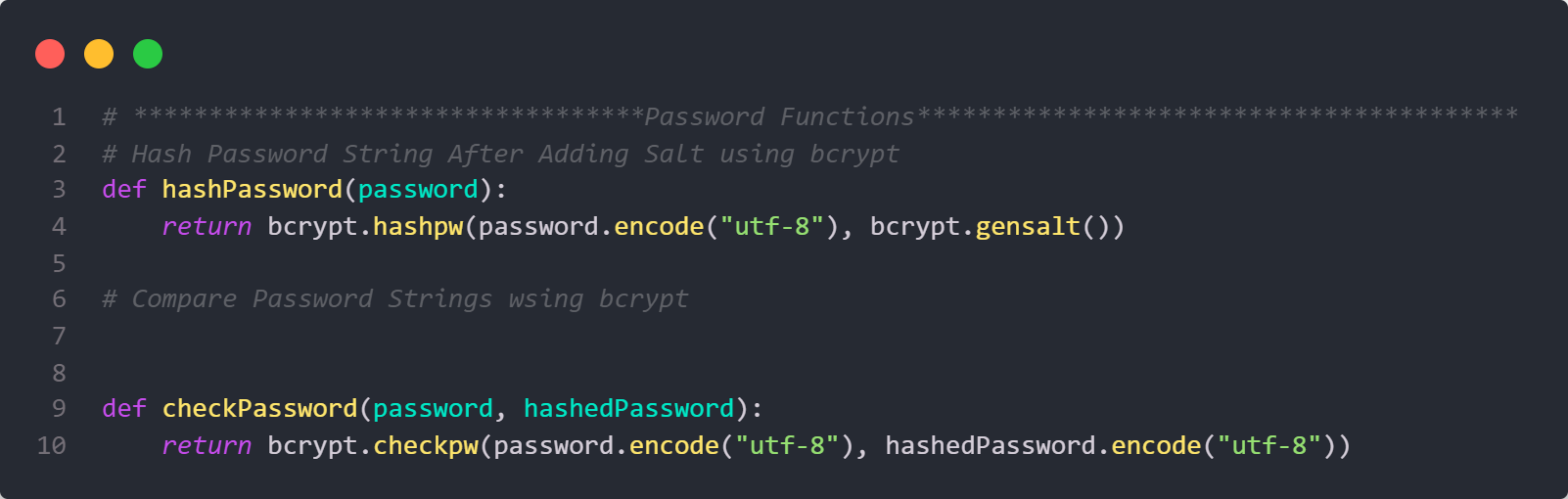
```python
# **********************************Imports****************************************
# OS Module To Implement The Clear Screen Function
import os
# MySQL Connector
import mysql.connector
from mysql.connector import errorcode
from mysql.connector import (connection)
# bcrypt library to hash passwords
import bcrypt
# Local Python File sqlData containing the sql Username ,Password and Database Name
import sqlData


# **********************************General Functions****************************************
# Class Colors to print clorful text in python terminal
class Colors:
    HEADER = '\033[95m'
    OKBLUE = '\033[94m'
    OKCYAN = '\033[96m'
    OKGREEN = '\033[92m'
    WARNING = '\033[93m'
    FAIL = '\033[91m'
    ENDC = '\033[0m'
    BOLD = '\033[1m'


#  MenuSpacing Variable To Center The menu Heading by adding spaces as prefix
menuSpacing = "          "

# Function To Print Divider in Green Color



def divider():
    print(f"{Colors.OKGREEN}=============================={Colors.ENDC}")
```



```python
# Function To Print Invalid Input in Red Color


def invalidInput():
    print(f"{Colors.FAIL}Invalid Input Enter A Choice From the Menu Above{Colors.ENDC}")
    divider()


def clearScreen():
    os.system("cls")


# ********************************Menu Functions*************************************
# Function To Print Menu
def showMenu(menuTitle, menuOptions, menuSpacing):
    while True:
        print(f"{Colors.HEADER}{menuSpacing}{menuTitle} {Colors.ENDC}")
        divider()
        for choice, statement in menuOptions.items():
            print(
                f"{Colors.OKBLUE}{Colors.BOLD}{choice} - {statement[0]}{Colors.ENDC}")
        try:
            choice = int(input("Enter Choice : "))
            clearScreen()
            if(choice >= 1 and choice <= len(menuOptions.keys())):
                if(choice == len(menuOptions.keys())):
                    break
                try:
                    menuOptions[choice][1]()
                except Exception as e:
                    print(e)
                divider()
            else:
                invalidInput()
        except:
            clearScreen()
            invalidInput()
```

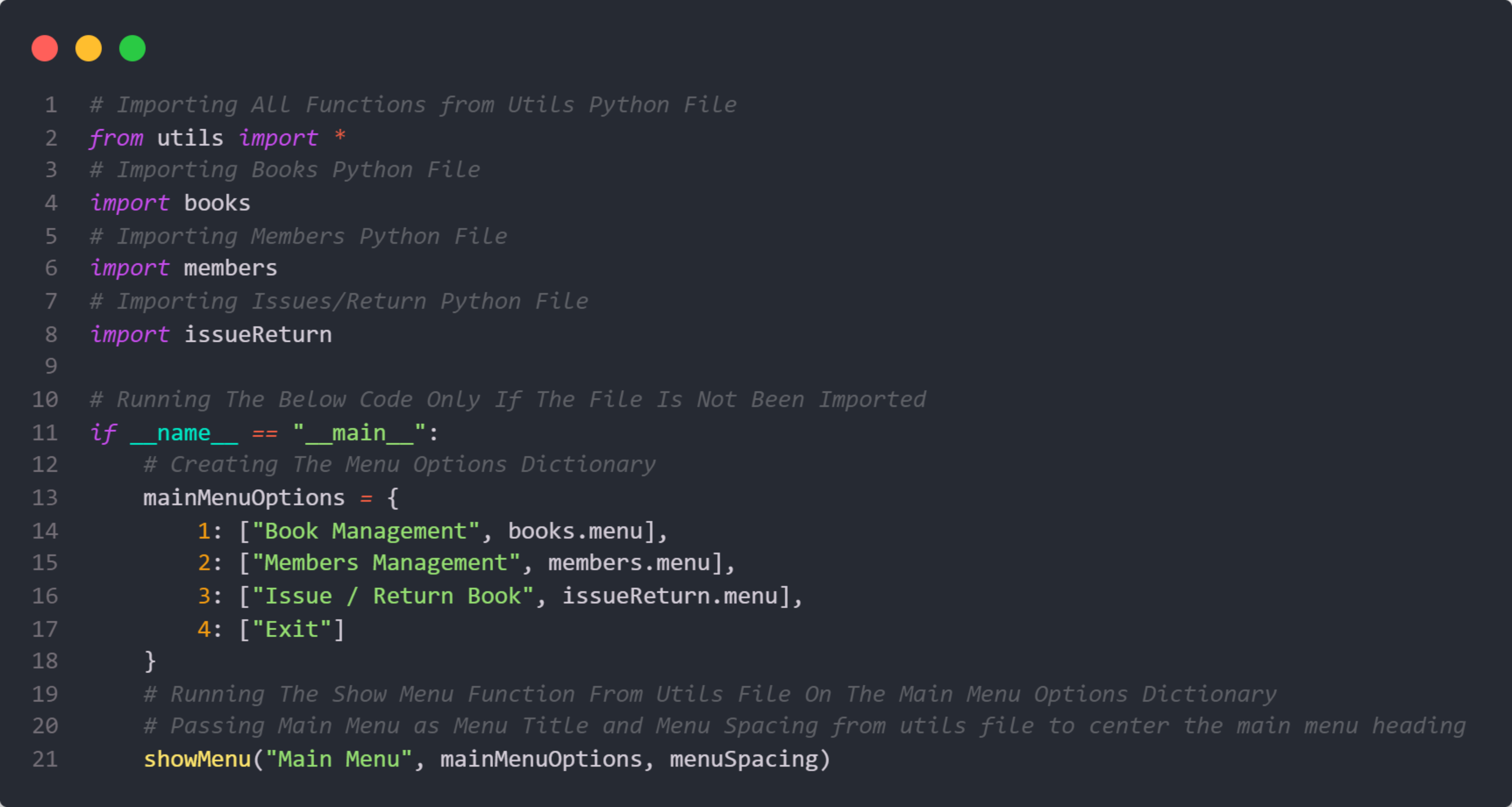


```python
# ********************************SQL Functions Start************************************
# SQL Function To Add/Delete and Update Data in the database
# NOTE : Cannot Be Used To Read Data From Cursor Object as Connection is Broken when Function Ends
def executeSQLCommitQuery(query, data):
    try:
        cnx = connection.MySQLConnection(
            user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD, host='localhost',
            database=sqlData.DATABASE_NAME)
        Cursor = cnx.cursor()
        Cursor.execute(query, data)
        cnx.commit()
        Cursor.close()
        cnx.close
        return Cursor
    except mysql.connector.Error as err:
        return handleSQLException(err)

# SQL Function To Handle All Major SQL ErrorCodes else return error message


def handleSQLException(err):
    if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
        return ("Something is wrong with your user name or password")
    elif err.errno == errorcode.ER_BAD_DB_ERROR:
        return ("Database does not exist")
    elif err.errno == 1062:
        return ("Duplicate Entry")
    elif err.errno == 1451:
        return ("Cannot Delete A Entry that Is A Member Of Another Table, the book or user must be\
             issued to someone try removing the issued record first")
    else:
        return (err)
```

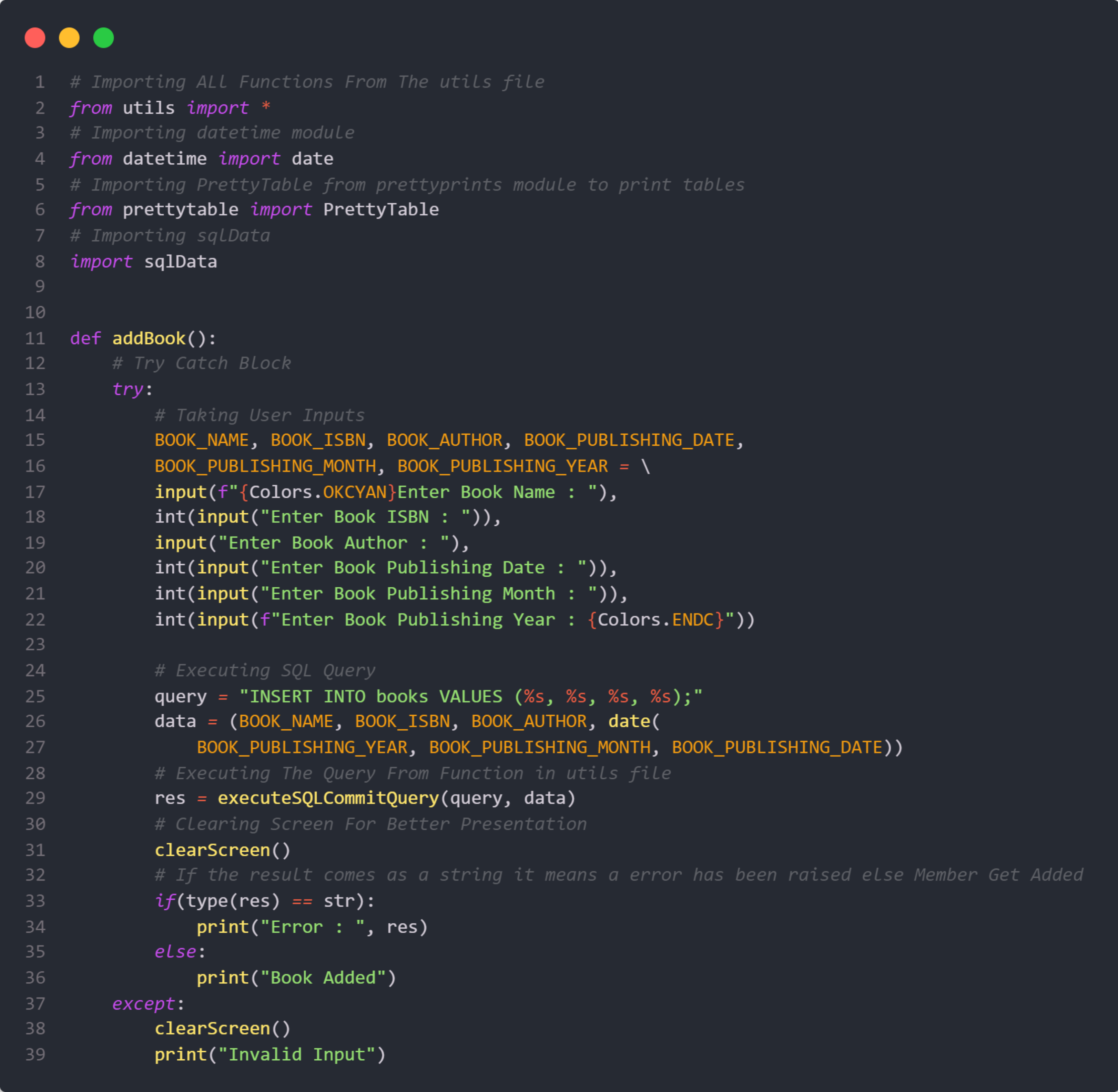


```python
# *********************************Password Functions****************************************
# Hash Password String After Adding Salt using bcrypt
def hashPassword(password):
    return bcrypt.hashpw(password.encode("utf-8"), bcrypt.gensalt())

# Compare Password Strings wsing bcrypt


def checkPassword(password, hashedPassword):
    return bcrypt.checkpw(password.encode("utf-8"), hashedPassword.encode("utf-8"))
```

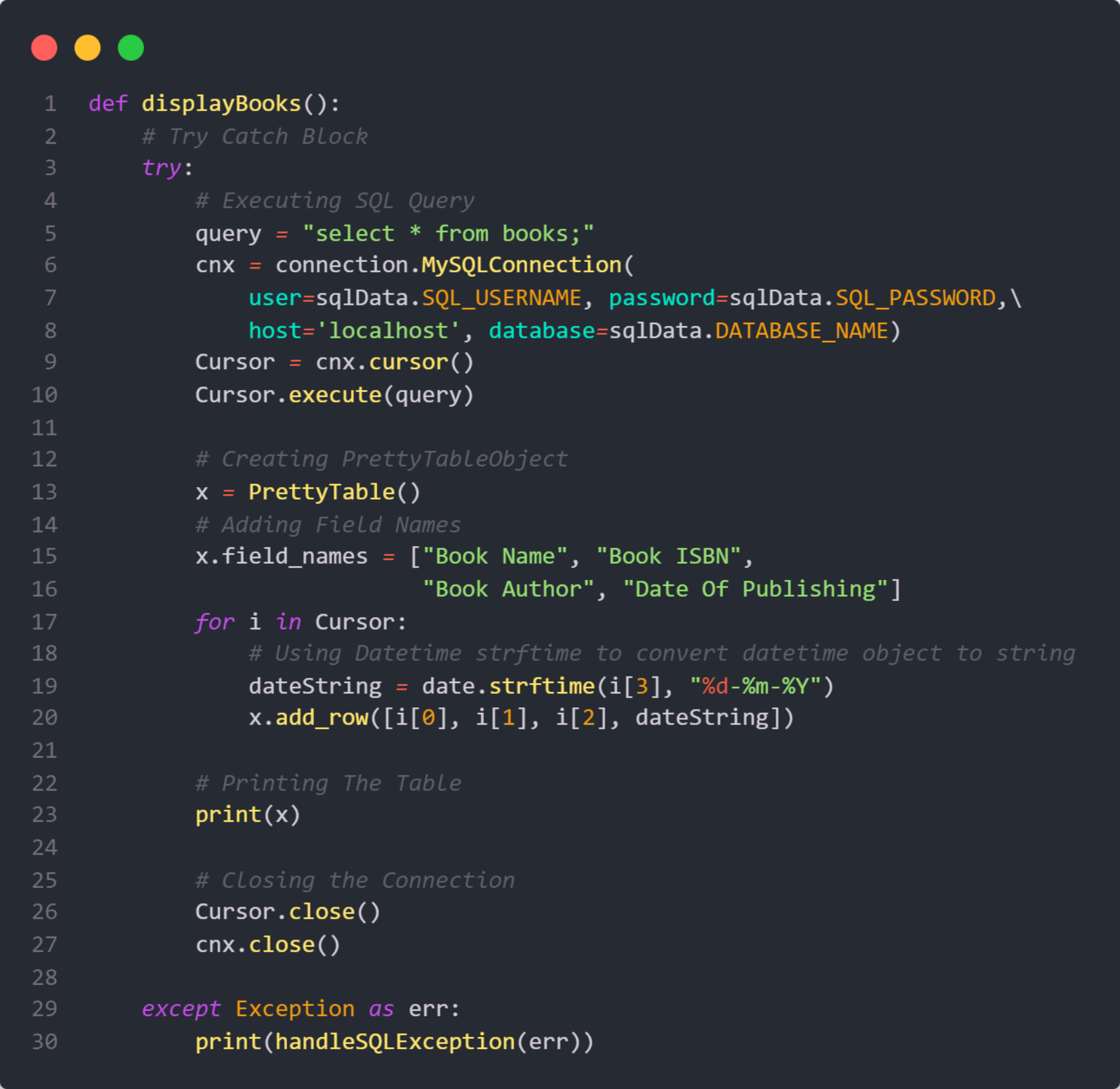
main.py


```python
# Importing All Functions from Utils Python File
from utils import *
# Importing Books Python File
import books
# Importing Members Python File
import members
# Importing Issues/Return Python File
import issueReturn

# Running The Below Code Only If The File Is Not Been Imported
if __name__ == "__main__":
    # Creating The Menu Options Dictionary
    mainMenuOptions = {
        1: ["Book Management", books.menu],
        2: ["Members Management", members.menu],
        3: ["Issue / Return Book", issueReturn.menu],
        4: ["Exit"]
    }
    # Running The Show Menu Function From Utils File On The Main Menu Options Dictionary
    # Passing Main Menu as Menu Title and Menu Spacing from utils file to center the main menu heading
    showMenu("Main Menu", mainMenuOptions, menuSpacing)
```

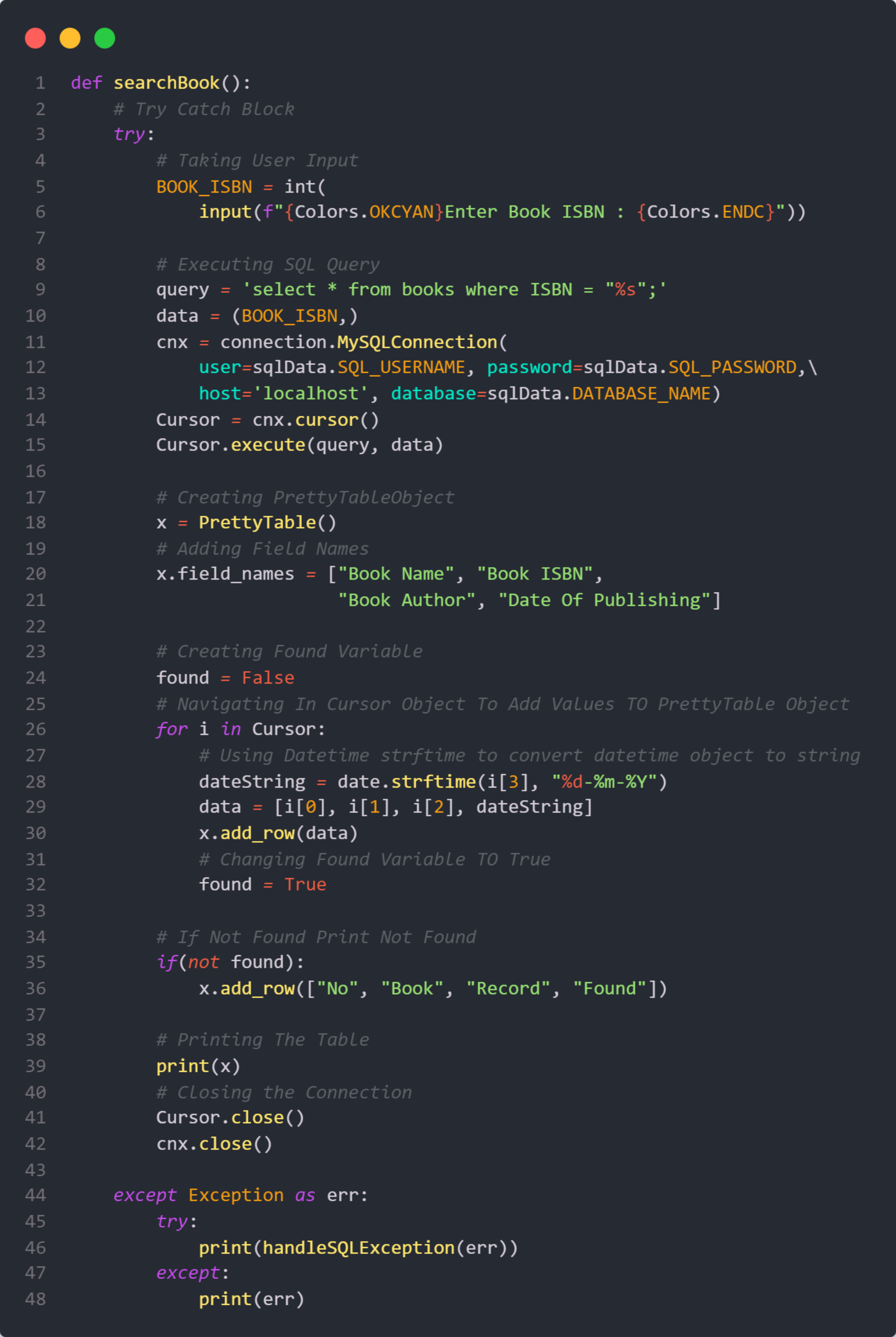
books.py



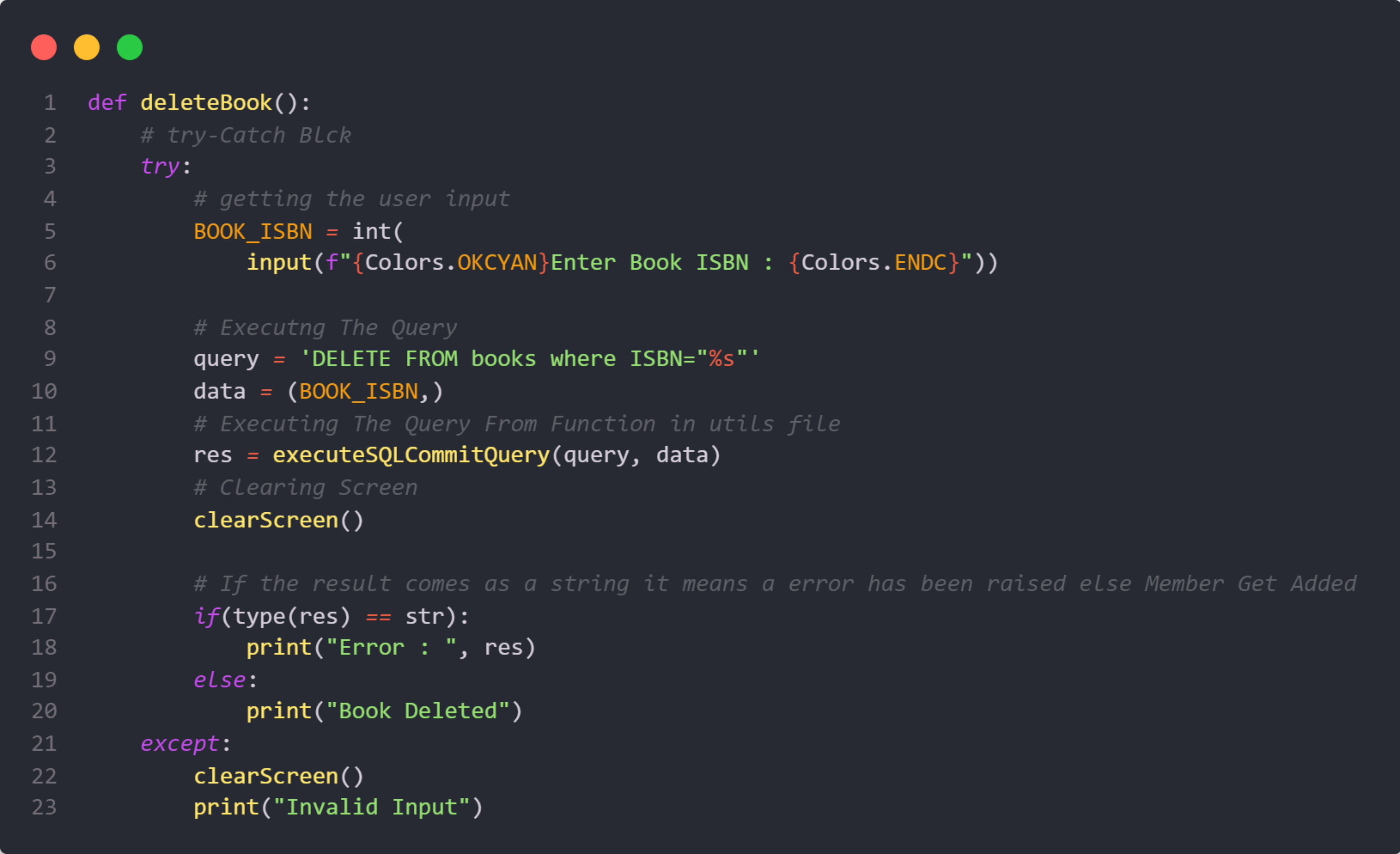
```python
# Importing ALL Functions From The utils file
from utils import *
# Importing datetime module
from datetime import date
# Importing PrettyTable from prettyprints module to print tables
from prettytable import PrettyTable
# Importing sqlData
import sqlData


def addBook():
    # Try Catch Block
    try:
        # Taking User Inputs
        BOOK_NAME, BOOK_ISBN, BOOK_AUTHOR, BOOK_PUBLISHING_DATE,
        BOOK_PUBLISHING_MONTH, BOOK_PUBLISHING_YEAR = \
        input(f"{Colors.OKCYAN}Enter Book Name : "),
        int(input("Enter Book ISBN : ")),
        input("Enter Book Author : "),
        int(input("Enter Book Publishing Date : ")),
        int(input("Enter Book Publishing Month : ")),
        int(input(f"Enter Book Publishing Year : {Colors.ENDC}"))

        # Executing SQL Query
        query = "INSERT INTO books VALUES (%s, %s, %s, %s);"
        data = (BOOK_NAME, BOOK_ISBN, BOOK_AUTHOR, date(
            BOOK_PUBLISHING_YEAR, BOOK_PUBLISHING_MONTH, BOOK_PUBLISHING_DATE))
        # Executing The Query From Function in utils file
        res = executeSQLCommitQuery(query, data)
        # Clearing Screen For Better Presentation
        clearScreen()
        # If the result comes as a string it means a error has been raised else Member Get Added
        if(type(res) == str):
            print("Error : ", res)
        else:
            print("Book Added")
    except:
        clearScreen()
        print("Invalid Input")
```



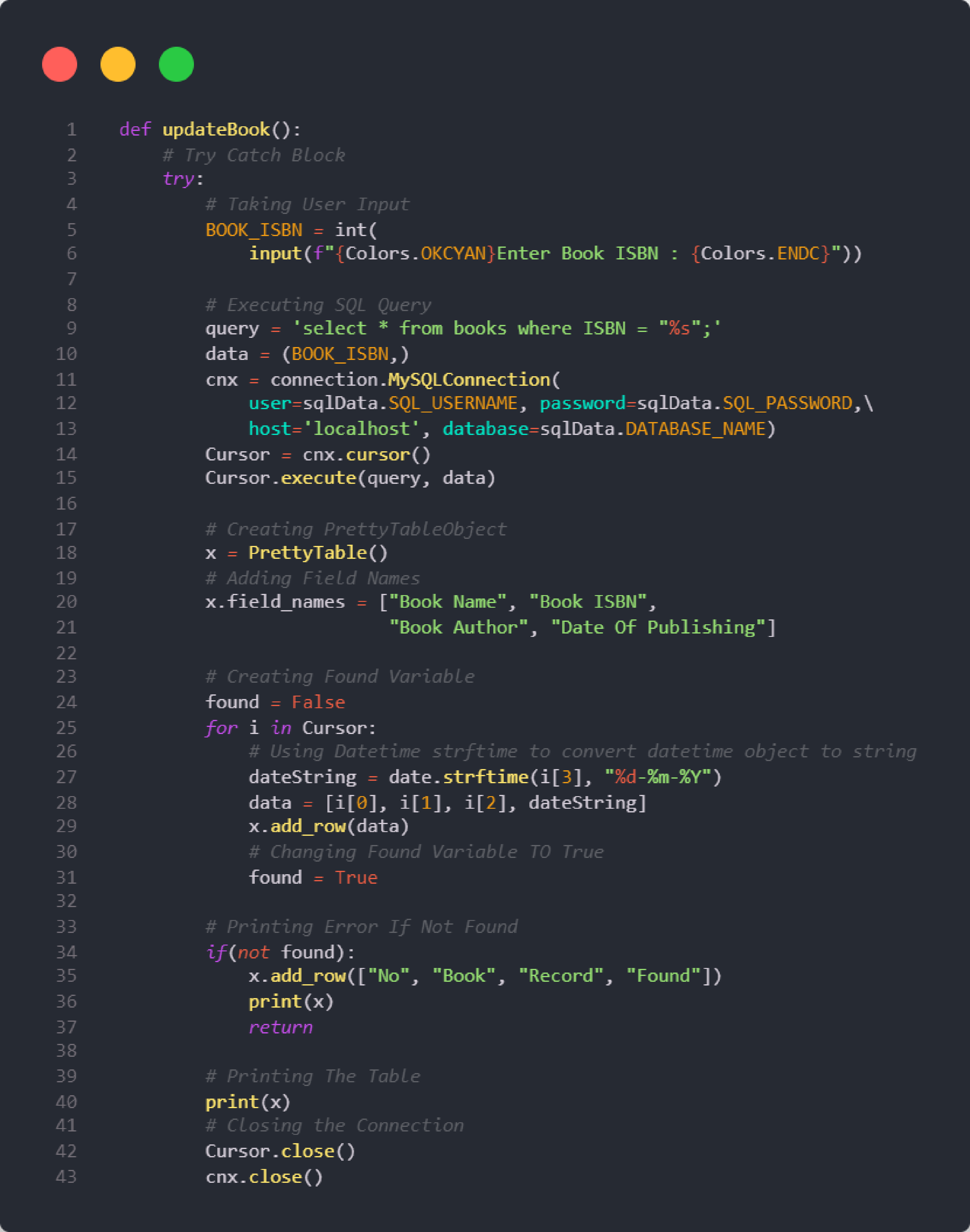
```python
def displayBooks():
    # Try Catch Block
    try:
        # Executing SQL Query
        query = "select * from books;"
        cnx = connection.MySQLConnection(
            user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
            host='localhost', database=sqlData.DATABASE_NAME)
        Cursor = cnx.cursor()
        Cursor.execute(query)

        # Creating PrettyTableObject
        x = PrettyTable()
        # Adding Field Names
        x.field_names = ["Book Name", "Book ISBN",
                         "Book Author", "Date Of Publishing"]
        for i in Cursor:
            # Using Datetime strftime to convert datetime object to string
            dateString = date.strftime(i[3], "%d-%m-%Y")
            x.add_row([i[0], i[1], i[2], dateString])

        # Printing The Table
        print(x)

        # Closing the Connection
        Cursor.close()
        cnx.close()

    except Exception as err:
        print(handleSQLException(err))
```



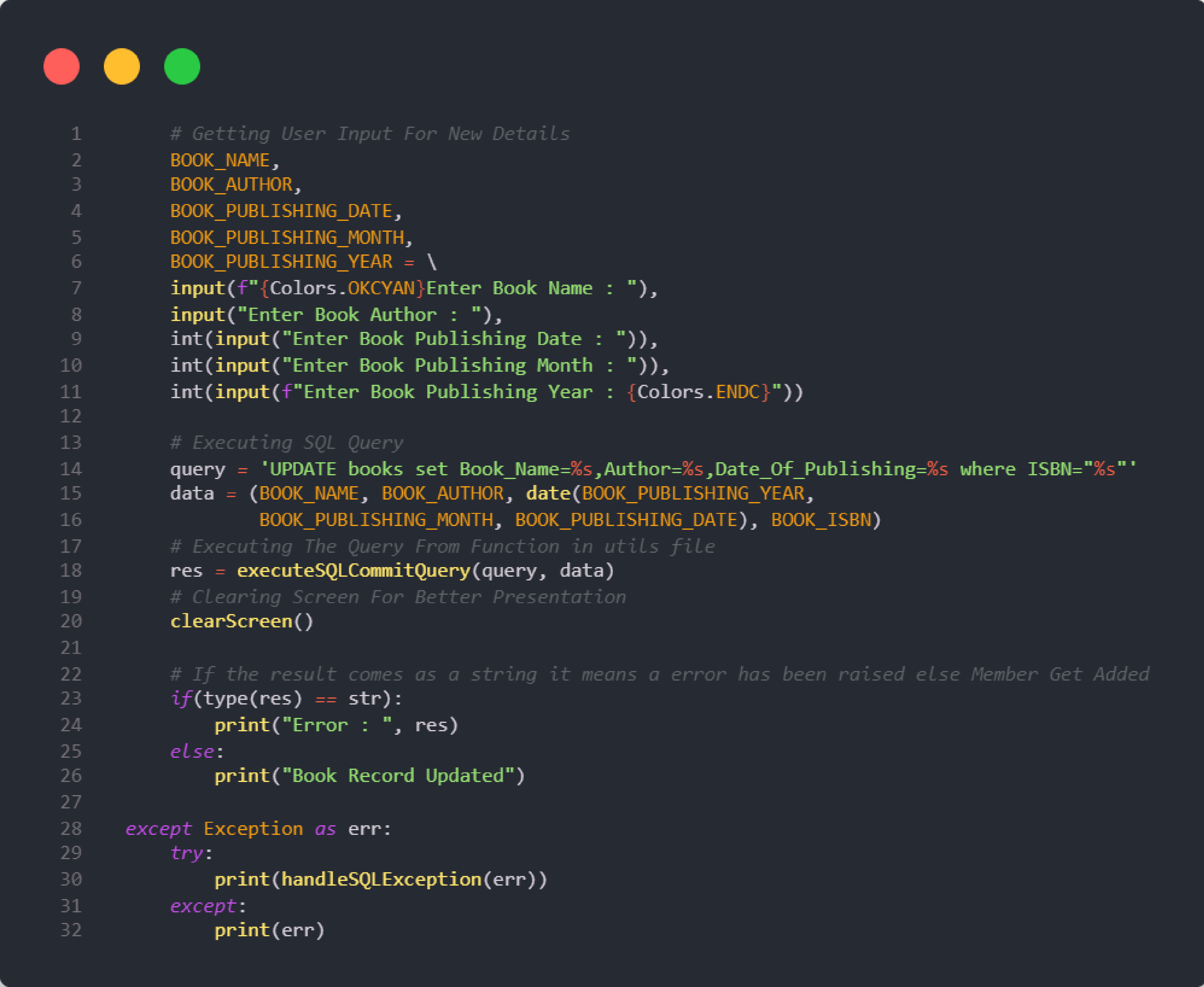
```python
def searchBook():
    # Try Catch Block
    try:
        # Taking User Input
        BOOK_ISBN = int(
            input(f"{Colors.OKCYAN}Enter Book ISBN : {Colors.ENDC}"))

        # Executing SQL Query
        query = 'select * from books where ISBN = "%s";'
        data = (BOOK_ISBN,)
        cnx = connection.MySQLConnection(
            user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
            host='localhost', database=sqlData.DATABASE_NAME)
        Cursor = cnx.cursor()
        Cursor.execute(query, data)

        # Creating PrettyTableObject
        x = PrettyTable()
        # Adding Field Names
        x.field_names = ["Book Name", "Book ISBN",
                         "Book Author", "Date Of Publishing"]

        # Creating Found Variable
        found = False
        # Navigating In Cursor Object To Add Values TO PrettyTable Object
        for i in Cursor:
            # Using Datetime strftime to convert datetime object to string
            dateString = date.strftime(i[3], "%d-%m-%Y")
            data = [i[0], i[1], i[2], dateString]
            x.add_row(data)
            # Changing Found Variable TO True
            found = True

        # If Not Found Print Not Found
        if(not found):
            x.add_row(["No", "Book", "Record", "Found"])

        # Printing The Table
        print(x)
        # Closing the Connection
        Cursor.close()
        cnx.close()

    except Exception as err:
        try:
            print(handleSQLException(err))
        except:
            print(err)
```



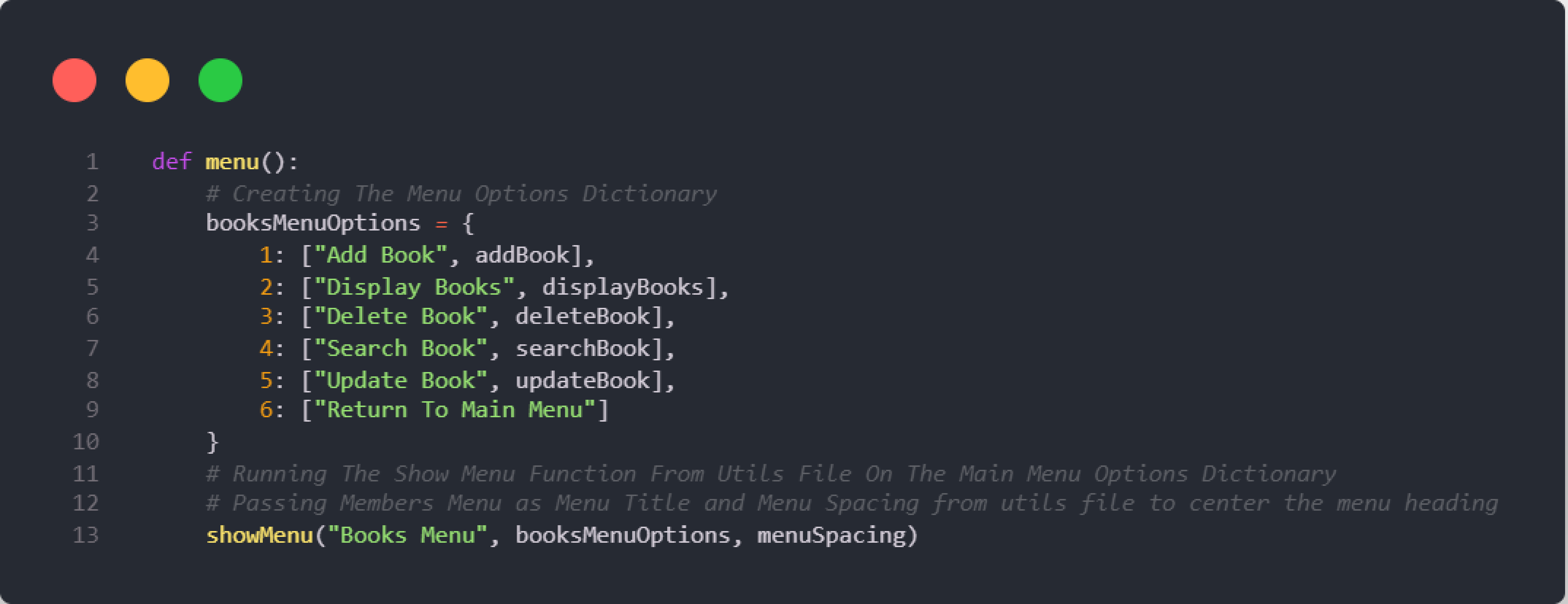
```python
def deleteBook():
    # try-Catch Blck
    try:
        # getting the user input
        BOOK_ISBN = int(
            input(f"{Colors.OKCYAN}Enter Book ISBN : {Colors.ENDC}"))

        # Executng The Query
        query = 'DELETE FROM books where ISBN="%s"'
        data = (BOOK_ISBN,)
        # Executing The Query From Function in utils file
        res = executeSQLCommitQuery(query, data)
        # Clearing Screen
        clearScreen()

        # If the result comes as a string it means a error has been raised else Member Get Added
        if(type(res) == str):
            print("Error : ", res)
        else:
            print("Book Deleted")
    except:
        clearScreen()
        print("Invalid Input")
```



```python
def updateBook():
    # Try Catch Block
    try:
        # Taking User Input
        BOOK_ISBN = int(
            input(f"{Colors.OKCYAN}Enter Book ISBN : {Colors.ENDC}"))

        # Executing SQL Query
        query = 'select * from books where ISBN = "%s";'
        data = (BOOK_ISBN,)
        cnx = connection.MySQLConnection(
            user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
            host='localhost', database=sqlData.DATABASE_NAME)
        Cursor = cnx.cursor()
        Cursor.execute(query, data)

        # Creating PrettyTableObject
        x = PrettyTable()
        # Adding Field Names
        x.field_names = ["Book Name", "Book ISBN",
                         "Book Author", "Date Of Publishing"]

        # Creating Found Variable
        found = False
        for i in Cursor:
            # Using Datetime strftime to convert datetime object to string
            dateString = date.strftime(i[3], "%d-%m-%Y")
            data = [i[0], i[1], i[2], dateString]
            x.add_row(data)
            # Changing Found Variable TO True
            found = True

        # Printing Error If Not Found
        if(not found):
            x.add_row(["No", "Book", "Record", "Found"])
            print(x)
            return

        # Printing The Table
        print(x)
        # Closing the Connection
        Cursor.close()
        cnx.close()
```

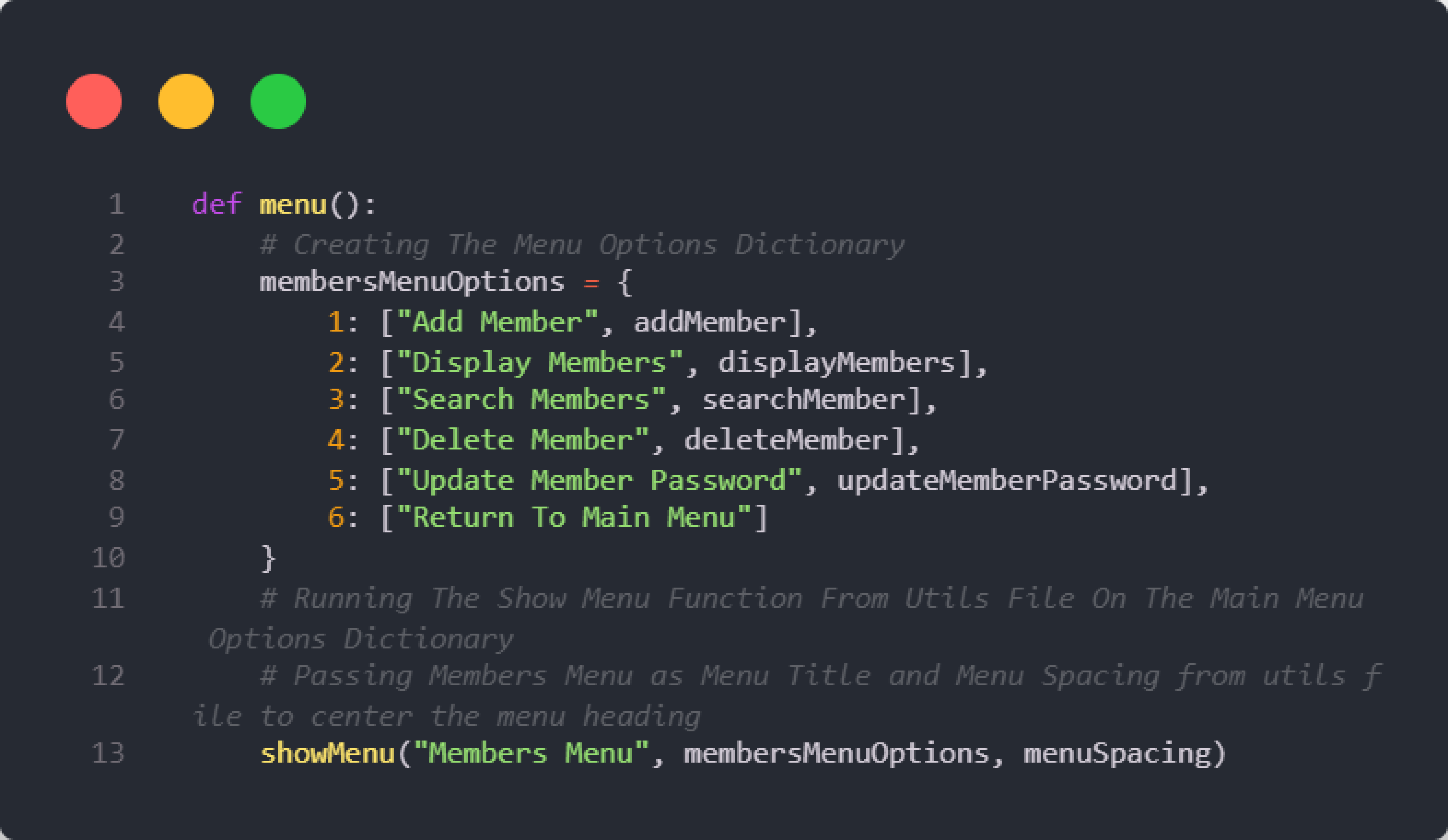


```python
        # Getting User Input For New Details
        BOOK_NAME,
        BOOK_AUTHOR,
        BOOK_PUBLISHING_DATE,
        BOOK_PUBLISHING_MONTH,
        BOOK_PUBLISHING_YEAR = \
        input(f"{Colors.OKCYAN}Enter Book Name : "),
        input("Enter Book Author : "),
        int(input("Enter Book Publishing Date : ")),
        int(input("Enter Book Publishing Month : ")),
        int(input(f"Enter Book Publishing Year : {Colors.ENDC}"))

        # Executing SQL Query
        query = 'UPDATE books set Book_Name=%s,Author=%s,Date_Of_Publishing=%s where ISBN="%s"'
        data = (BOOK_NAME, BOOK_AUTHOR, date(BOOK_PUBLISHING_YEAR,
                BOOK_PUBLISHING_MONTH, BOOK_PUBLISHING_DATE), BOOK_ISBN)
        # Executing The Query From Function in utils file
        res = executeSQLCommitQuery(query, data)
        # Clearing Screen For Better Presentation
        clearScreen()

        # If the result comes as a string it means a error has been raised else Member Get Added
        if(type(res) == str):
            print("Error : ", res)
        else:
            print("Book Record Updated")

    except Exception as err:
        try:
            print(handleSQLException(err))
        except:
            print(err)
```

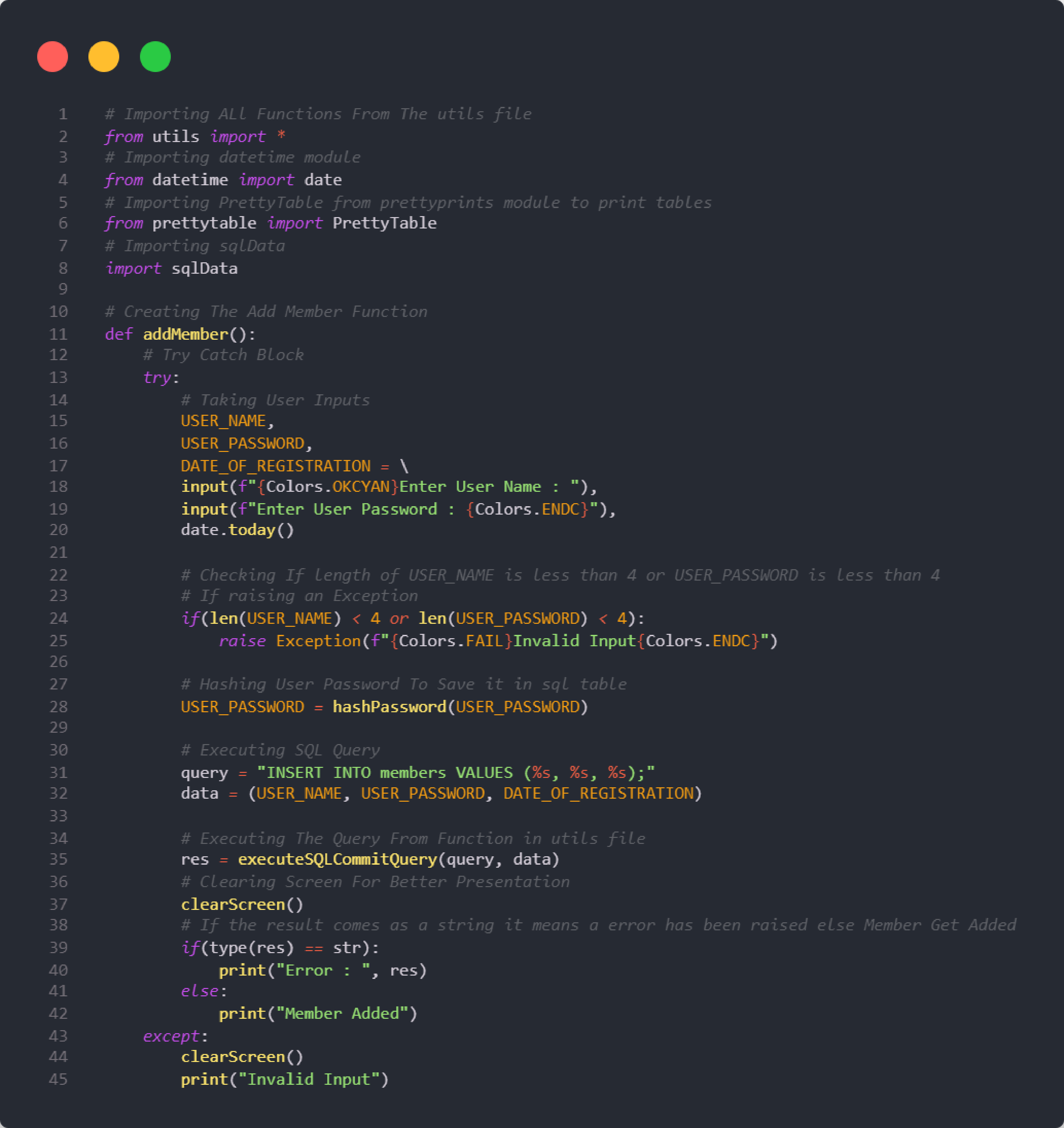


```python
def menu():
    # Creating The Menu Options Dictionary
    booksMenuOptions = {
        1: ["Add Book", addBook],
        2: ["Display Books", displayBooks],
        3: ["Delete Book", deleteBook],
        4: ["Search Book", searchBook],
        5: ["Update Book", updateBook],
        6: ["Return To Main Menu"]
    }
    # Running The Show Menu Function From Utils File On The Main Menu Options Dictionary
    # Passing Members Menu as Menu Title and Menu Spacing from utils file to center the menu heading
    showMenu("Books Menu", booksMenuOptions, menuSpacing)
```

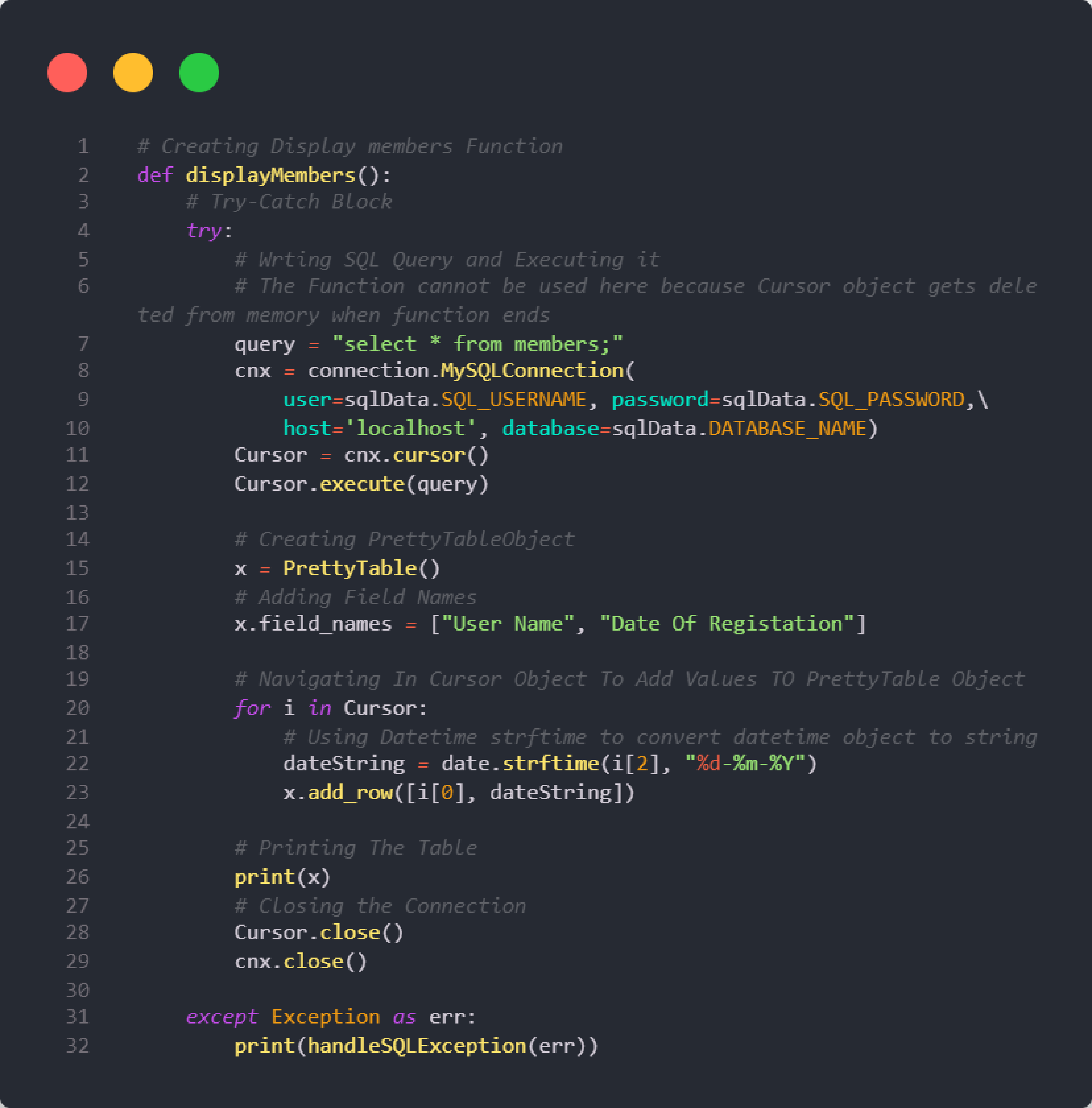
members.py



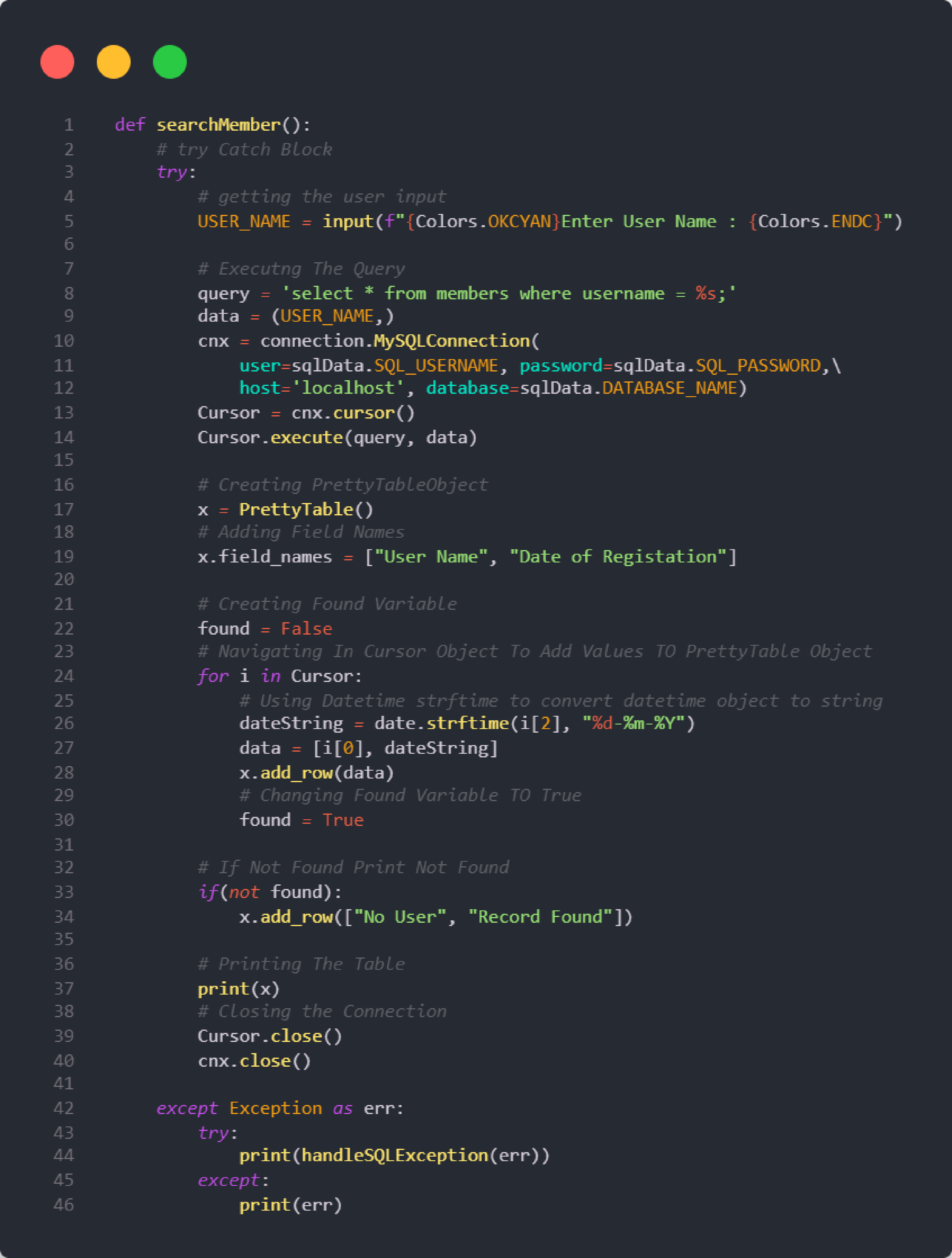
```python
def menu():
    # Creating The Menu Options Dictionary
    membersMenuOptions = {
        1: ["Add Member", addMember],
        2: ["Display Members", displayMembers],
        3: ["Search Members", searchMember],
        4: ["Delete Member", deleteMember],
        5: ["Update Member Password", updateMemberPassword],
        6: ["Return To Main Menu"]
    }
    # Running The Show Menu Function From Utils File On The Main Menu Options Dictionary
    # Passing Members Menu as Menu Title and Menu Spacing from utils file to center the menu heading
    showMenu("Members Menu", membersMenuOptions, menuSpacing)
```



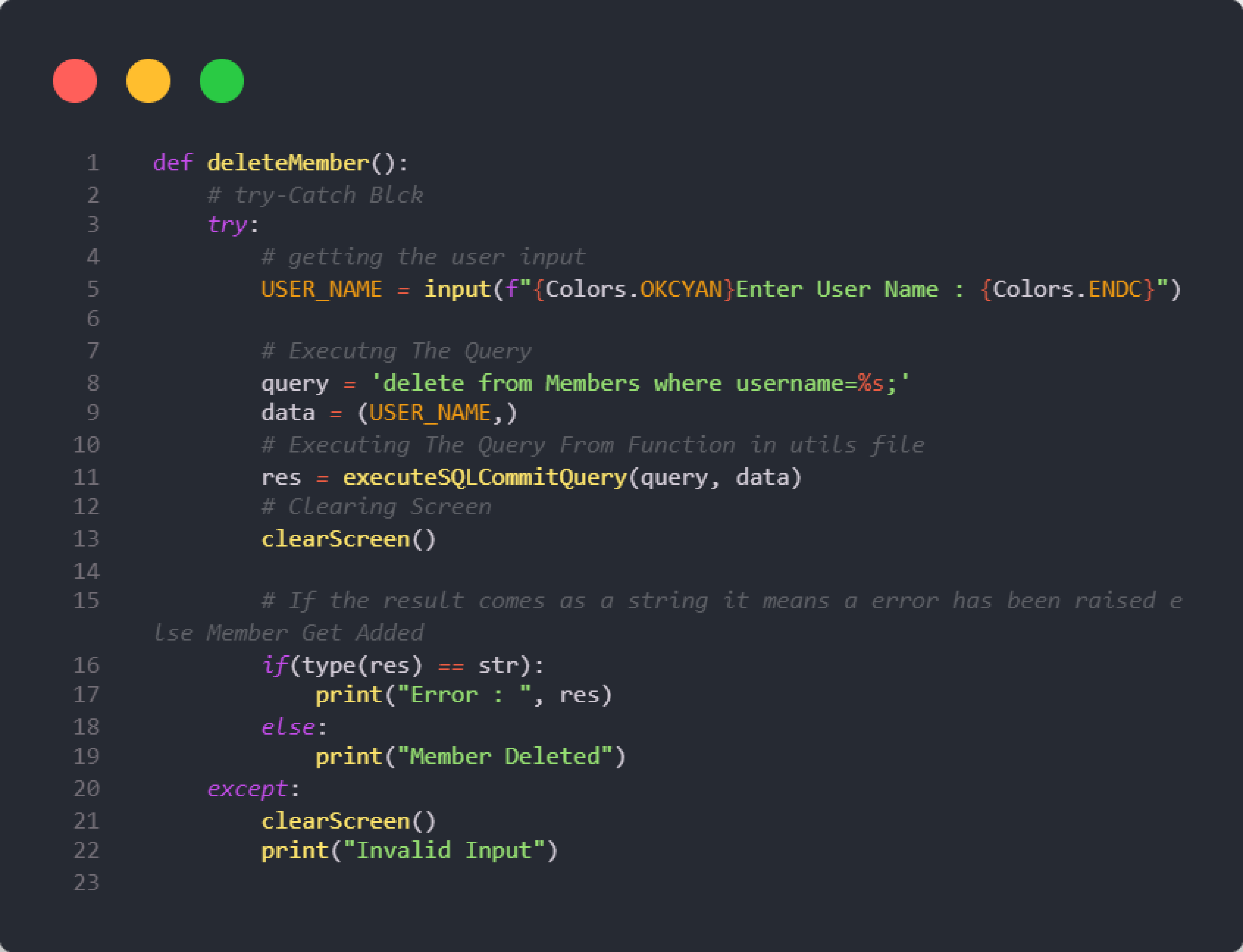
```python
# Importing ALL Functions From The utils file
from utils import *
# Importing datetime module
from datetime import date
# Importing PrettyTable from prettyprints module to print tables
from prettytable import PrettyTable
# Importing sqlData
import sqlData

# Creating The Add Member Function
def addMember():
    # Try Catch Block
    try:
        # Taking User Inputs
        USER_NAME,
        USER_PASSWORD,
        DATE_OF_REGISTRATION = \
        input(f"{Colors.OKCYAN}Enter User Name : "),
        input(f"Enter User Password : {Colors.ENDC}"),
        date.today()

        # Checking If length of USER_NAME is less than 4 or USER_PASSWORD is less than 4
        # If raising an Exception
        if(len(USER_NAME) < 4 or len(USER_PASSWORD) < 4):
            raise Exception(f"{Colors.FAIL}Invalid Input{Colors.ENDC}")

        # Hashing User Password To Save it in sql table
        USER_PASSWORD = hashPassword(USER_PASSWORD)

        # Executing SQL Query
        query = "INSERT INTO members VALUES (%s, %s, %s);"
        data = (USER_NAME, USER_PASSWORD, DATE_OF_REGISTRATION)

        # Executing The Query From Function in utils file
        res = executeSQLCommitQuery(query, data)
        # Clearing Screen For Better Presentation
        clearScreen()
        # If the result comes as a string it means a error has been raised else Member Get Added
        if(type(res) == str):
            print("Error : ", res)
        else:
            print("Member Added")
    except:
        clearScreen()
        print("Invalid Input")
```



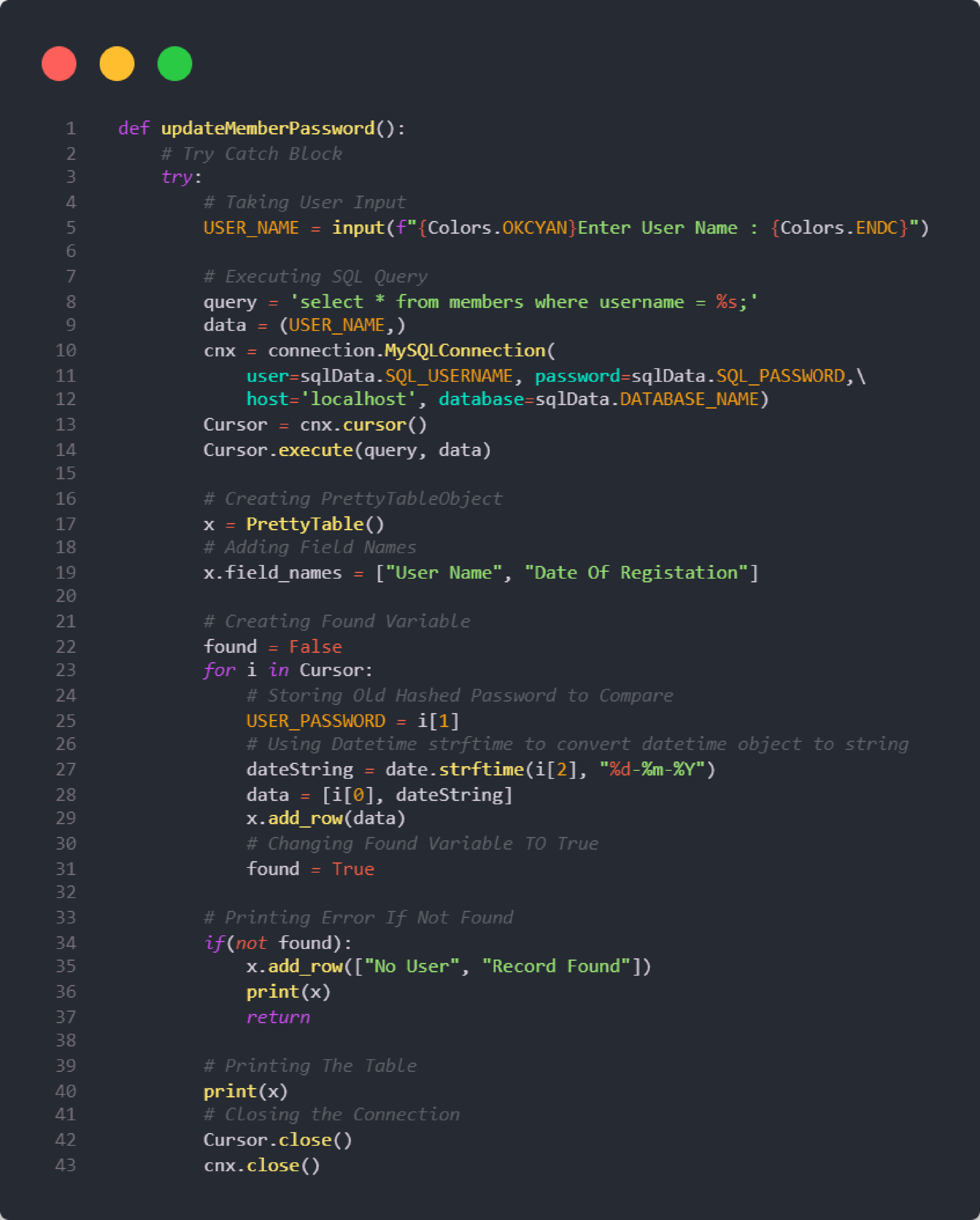
```python
# Creating Display members Function
def displayMembers():
    # Try-Catch Block
    try:
        # Wrting SQL Query and Executing it
        # The Function cannot be used here because Cursor object gets deleted from memory when function ends
        query = "select * from members;"
        cnx = connection.MySQLConnection(
            user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
            host='localhost', database=sqlData.DATABASE_NAME)
        Cursor = cnx.cursor()
        Cursor.execute(query)

        # Creating PrettyTableObject
        x = PrettyTable()
        # Adding Field Names
        x.field_names = ["User Name", "Date Of Registation"]

        # Navigating In Cursor Object To Add Values TO PrettyTable Object
        for i in Cursor:
            # Using Datetime strftime to convert datetime object to string
            dateString = date.strftime(i[2], "%d-%m-%Y")
            x.add_row([i[0], dateString])

        # Printing The Table
        print(x)
        # Closing the Connection
        Cursor.close()
        cnx.close()

    except Exception as err:
        print(handleSQLException(err))
```



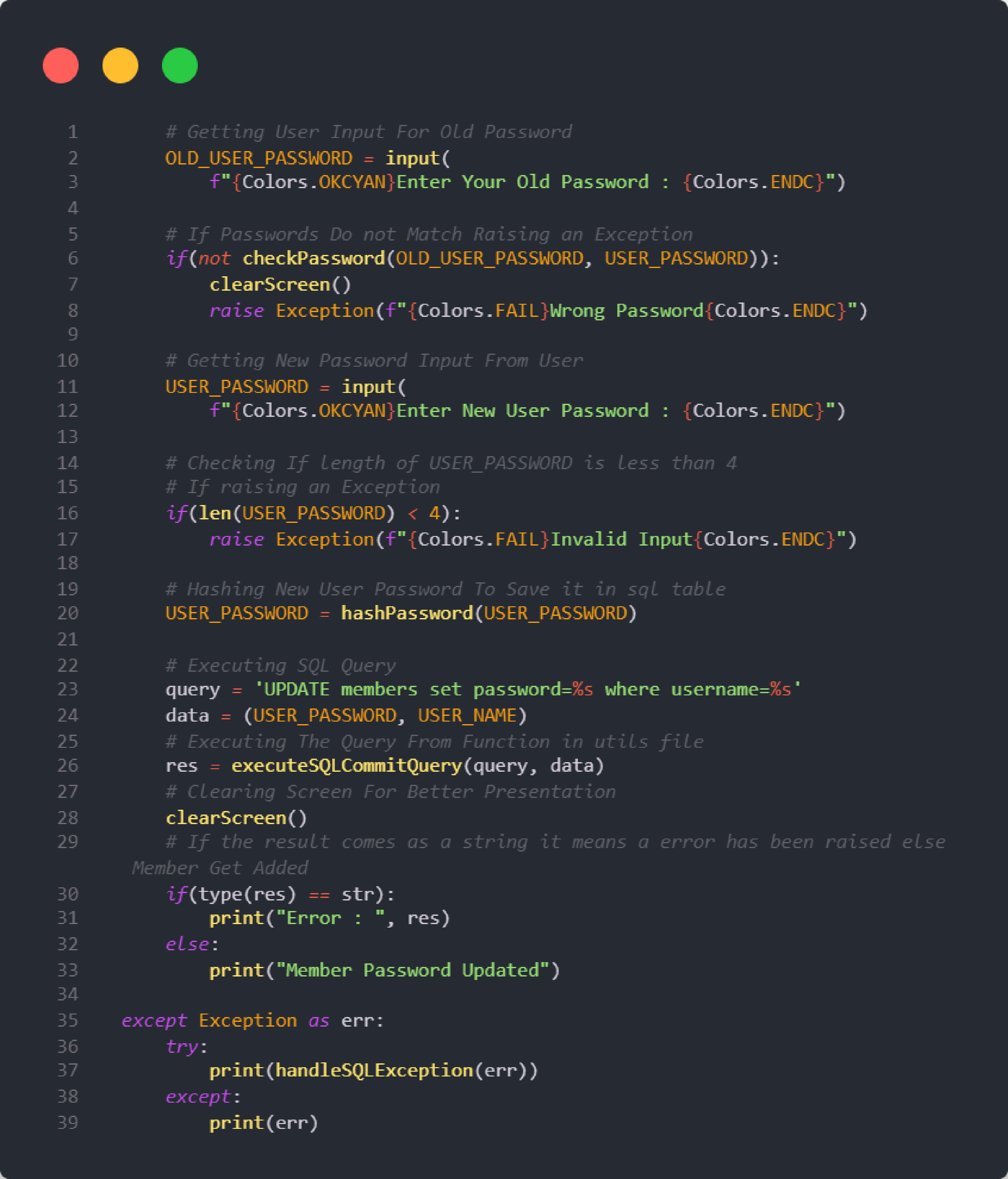
```python
def searchMember():
    # try Catch Block
    try:
        # getting the user input
        USER_NAME = input(f"{Colors.OKCYAN}Enter User Name : {Colors.ENDC}")

        # Executng The Query
        query = 'select * from members where username = %s;'
        data = (USER_NAME,)
        cnx = connection.MySQLConnection(
            user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
            host='localhost', database=sqlData.DATABASE_NAME)
        Cursor = cnx.cursor()
        Cursor.execute(query, data)

        # Creating PrettyTableObject
        x = PrettyTable()
        # Adding Field Names
        x.field_names = ["User Name", "Date of Registation"]

        # Creating Found Variable
        found = False
        # Navigating In Cursor Object To Add Values TO PrettyTable Object
        for i in Cursor:
            # Using Datetime strftime to convert datetime object to string
            dateString = date.strftime(i[2], "%d-%m-%Y")
            data = [i[0], dateString]
            x.add_row(data)
            # Changing Found Variable TO True
            found = True

        # If Not Found Print Not Found
        if(not found):
            x.add_row(["No User", "Record Found"])

        # Printing The Table
        print(x)
        # Closing the Connection
        Cursor.close()
        cnx.close()

    except Exception as err:
        try:
            print(handleSQLException(err))
        except:
            print(err)
```



```python
def deleteMember():
    # try-Catch Blck
    try:
        # getting the user input
        USER_NAME = input(f"{Colors.OKCYAN}Enter User Name : {Colors.ENDC}")

        # Executng The Query
        query = 'delete from Members where username=%s;'
        data = (USER_NAME,)
        # Executing The Query From Function in utils file
        res = executeSQLCommitQuery(query, data)
        # Clearing Screen
        clearScreen()

        # If the result comes as a string it means a error has been raised else Member Get Added
        if(type(res) == str):
            print("Error : ", res)
        else:
            print("Member Deleted")
    except:
        clearScreen()
        print("Invalid Input")

```

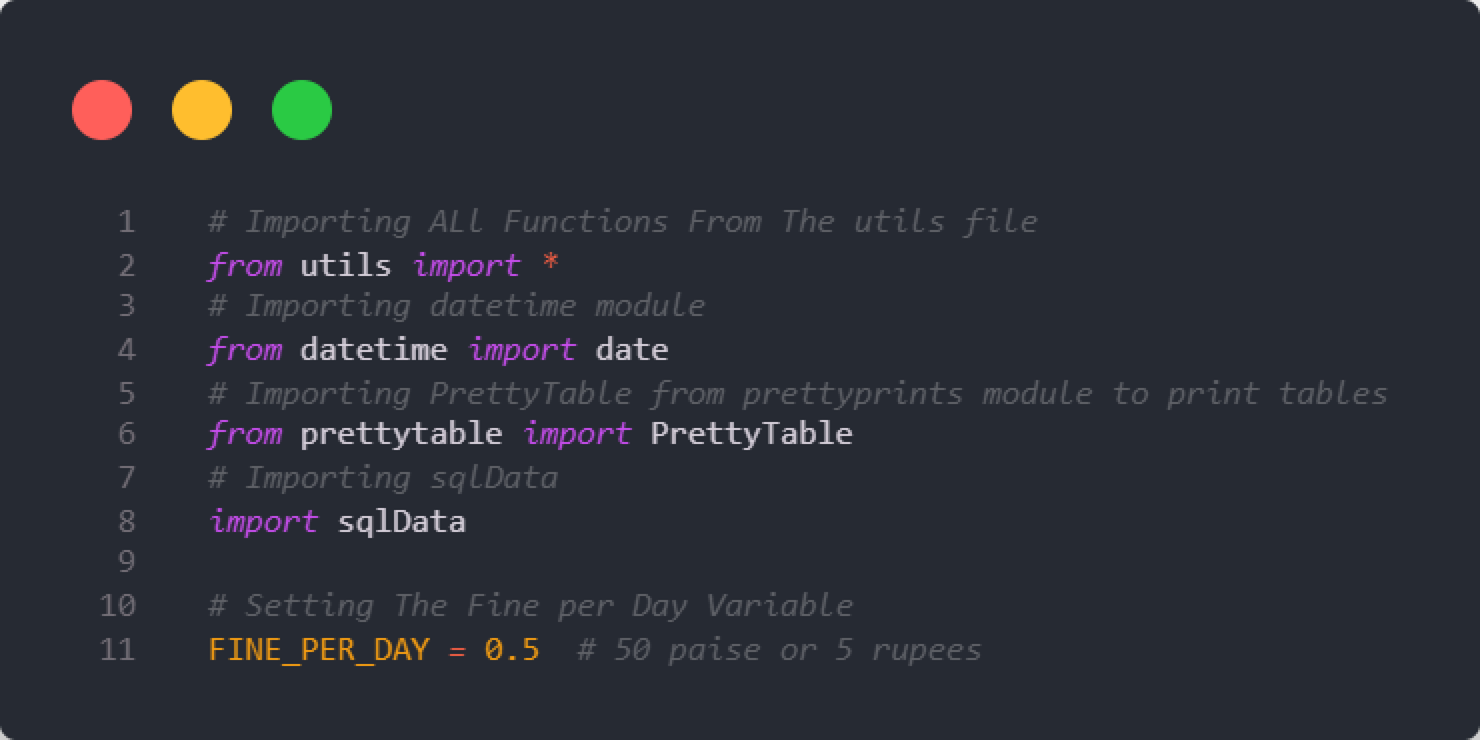


```python
def updateMemberPassword():
    # Try Catch Block
    try:
        # Taking User Input
        USER_NAME = input(f"{Colors.OKCYAN}Enter User Name : {Colors.ENDC}")

        # Executing SQL Query
        query = 'select * from members where username = %s;'
        data = (USER_NAME,)
        cnx = connection.MySQLConnection(
            user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
            host='localhost', database=sqlData.DATABASE_NAME)
        Cursor = cnx.cursor()
        Cursor.execute(query, data)

        # Creating PrettyTableObject
        x = PrettyTable()
        # Adding Field Names
        x.field_names = ["User Name", "Date Of Registation"]

        # Creating Found Variable
        found = False
        for i in Cursor:
            # Storing Old Hashed Password to Compare
            USER_PASSWORD = i[1]
            # Using Datetime strftime to convert datetime object to string
            dateString = date.strftime(i[2], "%d-%m-%Y")
            data = [i[0], dateString]
            x.add_row(data)
            # Changing Found Variable TO True
            found = True

        # Printing Error If Not Found
        if(not found):
            x.add_row(["No User", "Record Found"])
            print(x)
            return

        # Printing The Table
        print(x)
        # Closing the Connection
        Cursor.close()
        cnx.close()
```

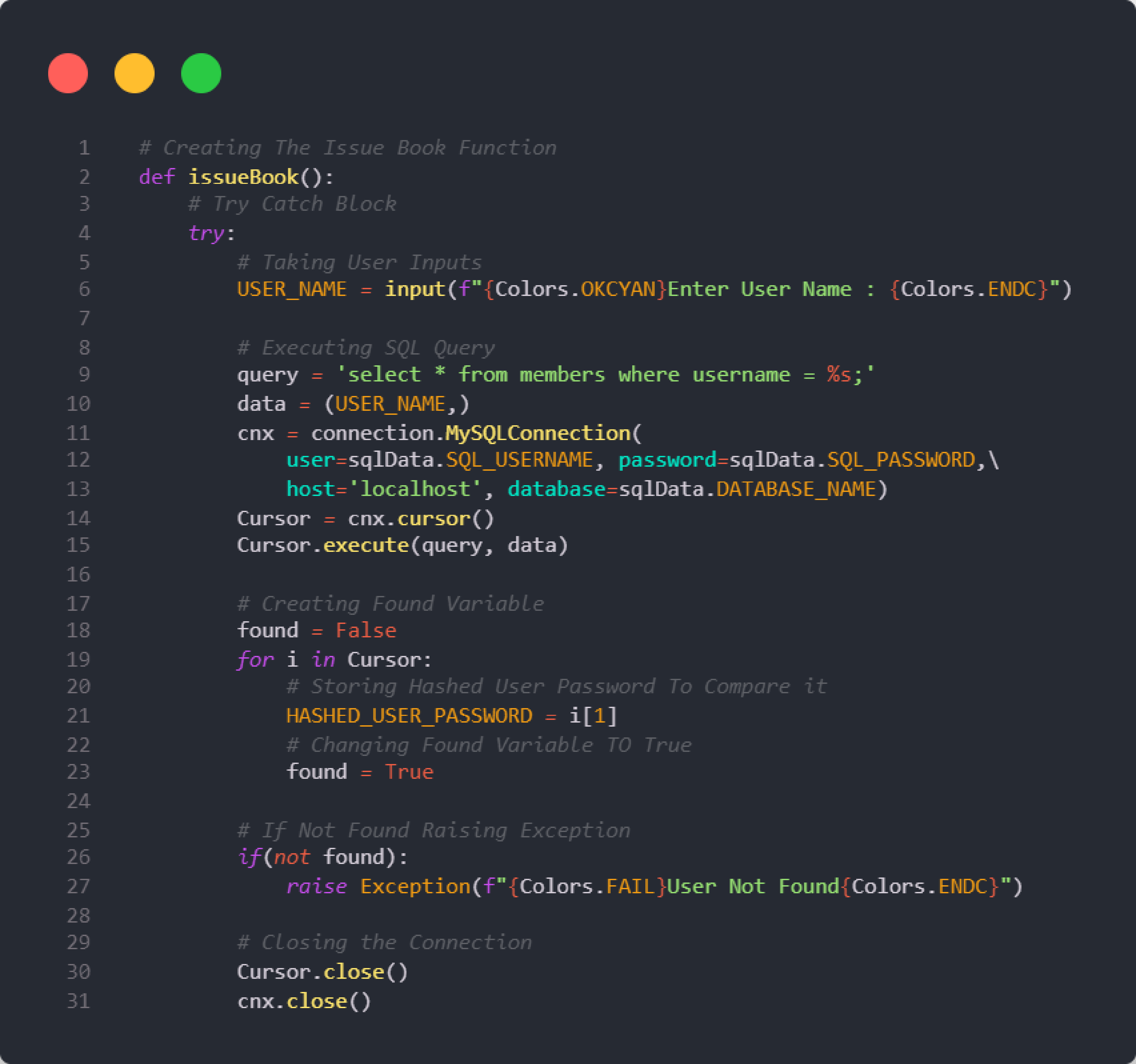


```python
        # Getting User Input For Old Password
        OLD_USER_PASSWORD = input(
            f"{Colors.OKCYAN}Enter Your Old Password : {Colors.ENDC}")

        # If Passwords Do not Match Raising an Exception
        if(not checkPassword(OLD_USER_PASSWORD, USER_PASSWORD)):
            clearScreen()
            raise Exception(f"{Colors.FAIL}Wrong Password{Colors.ENDC}")

        # Getting New Password Input From User
        USER_PASSWORD = input(
            f"{Colors.OKCYAN}Enter New User Password : {Colors.ENDC}")

        # Checking If length of USER_PASSWORD is less than 4
        # If raising an Exception
        if(len(USER_PASSWORD) < 4):
            raise Exception(f"{Colors.FAIL}Invalid Input{Colors.ENDC}")

        # Hashing New User Password To Save it in sql table
        USER_PASSWORD = hashPassword(USER_PASSWORD)

        # Executing SQL Query
        query = 'UPDATE members set password=%s where username=%s'
        data = (USER_PASSWORD, USER_NAME)
        # Executing The Query From Function in utils file
        res = executeSQLCommitQuery(query, data)
        # Clearing Screen For Better Presentation
        clearScreen()
        # If the result comes as a string it means a error has been raised else Member Get Added
        if(type(res) == str):
            print("Error : ", res)
        else:
            print("Member Password Updated")

    except Exception as err:
        try:
            print(handleSQLException(err))
        except:
            print(err)
```

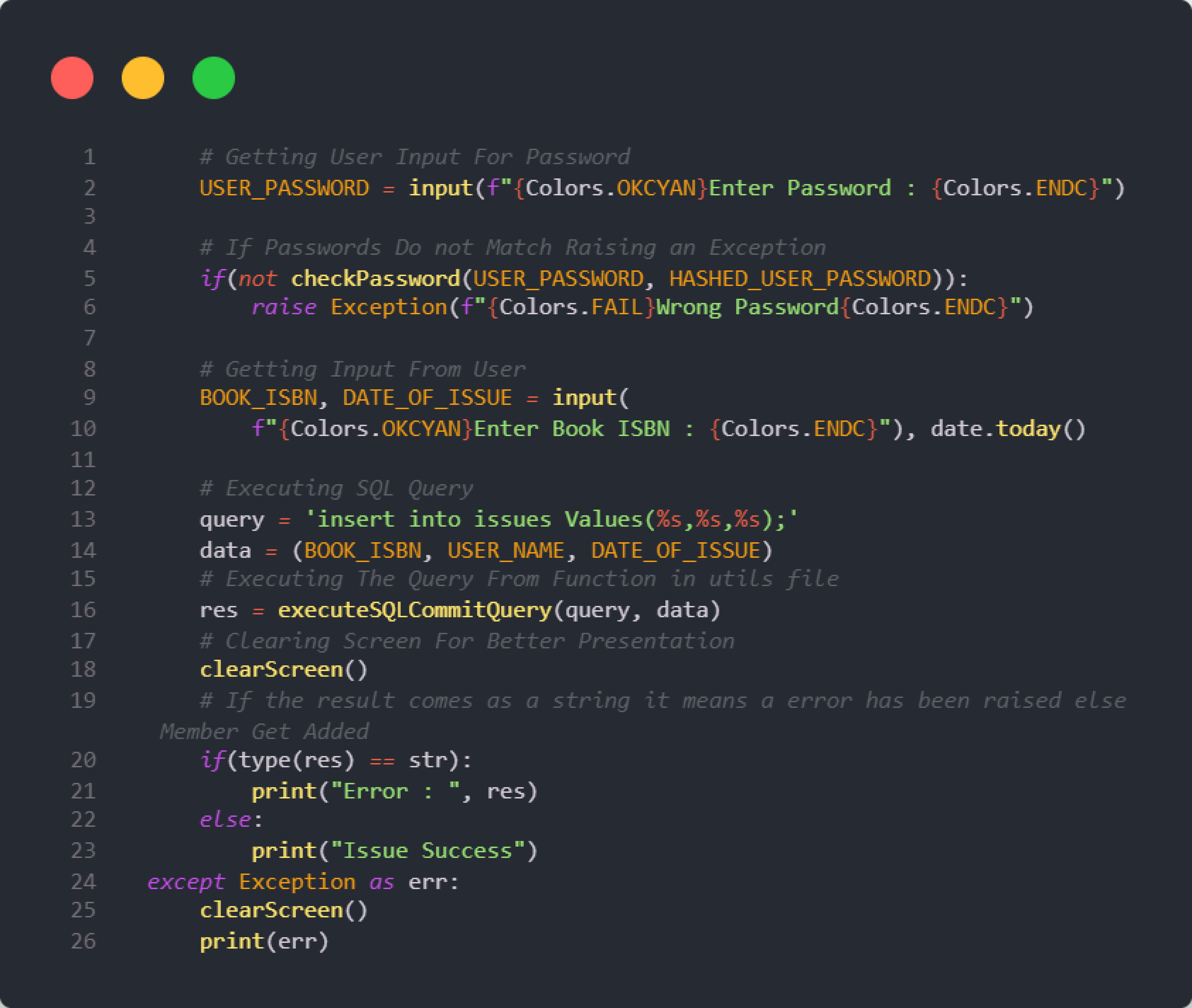
issueReturn.py



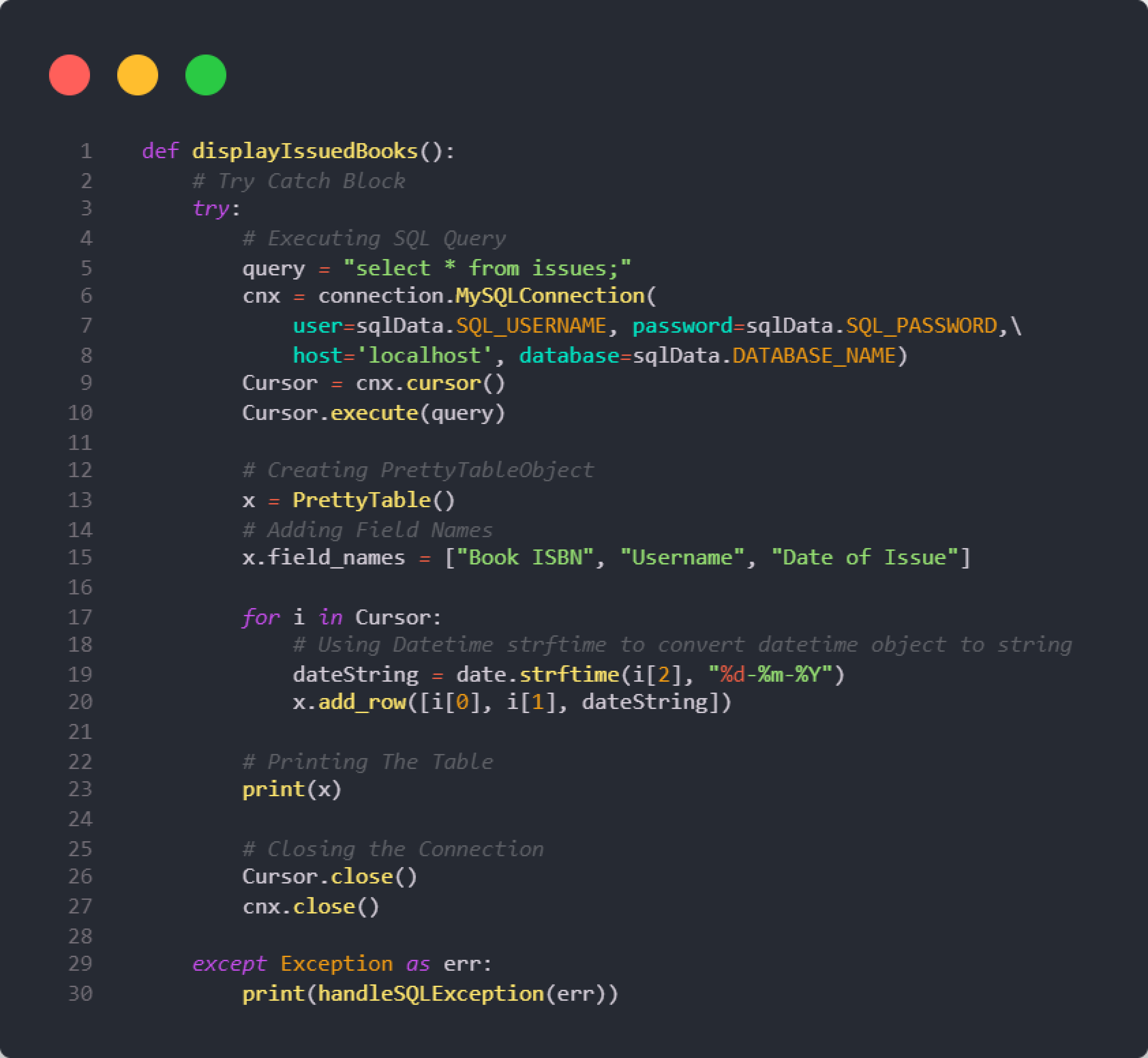
```python
# Importing ALL Functions From The utils file
from utils import *
# Importing datetime module
from datetime import date
# Importing PrettyTable from prettyprints module to print tables
from prettytable import PrettyTable
# Importing sqlData
import sqlData

# Setting The Fine per Day Variable
FINE_PER_DAY = 0.5  # 50 paise or 5 rupees
```



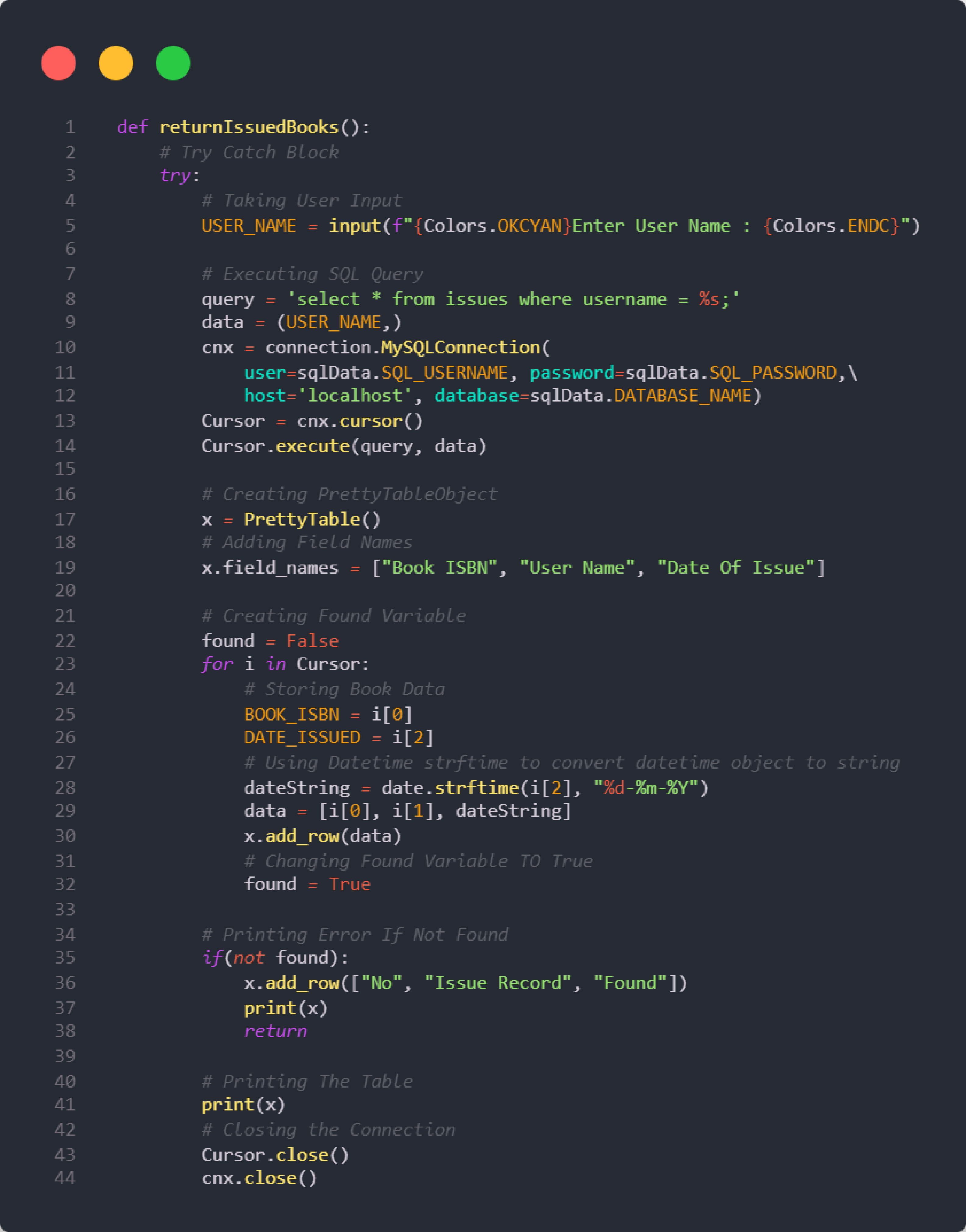
```python
# Creating The Issue Book Function
def issueBook():
    # Try Catch Block
    try:
        # Taking User Inputs
        USER_NAME = input(f"{Colors.OKCYAN}Enter User Name : {Colors.ENDC}")

        # Executing SQL Query
        query = 'select * from members where username = %s;'
        data = (USER_NAME,)
        cnx = connection.MySQLConnection(
            user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
            host='localhost', database=sqlData.DATABASE_NAME)
        Cursor = cnx.cursor()
        Cursor.execute(query, data)

        # Creating Found Variable
        found = False
        for i in Cursor:
            # Storing Hashed User Password To Compare it
            HASHED_USER_PASSWORD = i[1]
            # Changing Found Variable TO True
            found = True

        # If Not Found Raising Exception
        if(not found):
            raise Exception(f"{Colors.FAIL}User Not Found{Colors.ENDC}")

        # Closing the Connection
        Cursor.close()
        cnx.close()
```



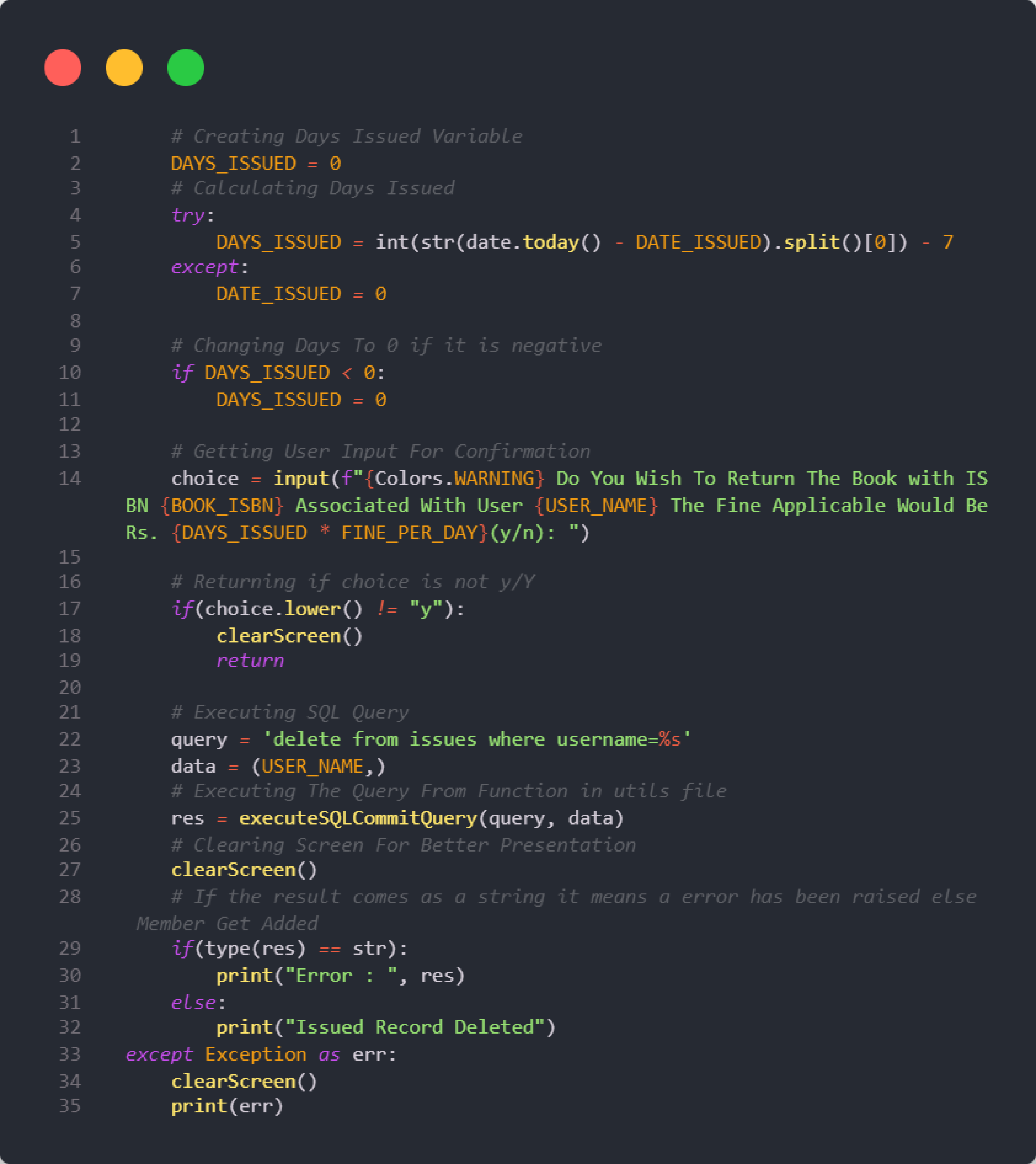
```python
        # Getting User Input For Password
        USER_PASSWORD = input(f"{Colors.OKCYAN}Enter Password : {Colors.ENDC}")

        # If Passwords Do not Match Raising an Exception
        if(not checkPassword(USER_PASSWORD, HASHED_USER_PASSWORD)):
            raise Exception(f"{Colors.FAIL}Wrong Password{Colors.ENDC}")

        # Getting Input From User
        BOOK_ISBN, DATE_OF_ISSUE = input(
            f"{Colors.OKCYAN}Enter Book ISBN : {Colors.ENDC}"), date.today()

        # Executing SQL Query
        query = 'insert into issues Values(%s,%s,%s);'
        data = (BOOK_ISBN, USER_NAME, DATE_OF_ISSUE)
        # Executing The Query From Function in utils file
        res = executeSQLCommitQuery(query, data)
        # Clearing Screen For Better Presentation
        clearScreen()
        # If the result comes as a string it means a error has been raised else Member Get Added
        if(type(res) == str):
            print("Error : ", res)
        else:
            print("Issue Success")
    except Exception as err:
        clearScreen()
        print(err)
```



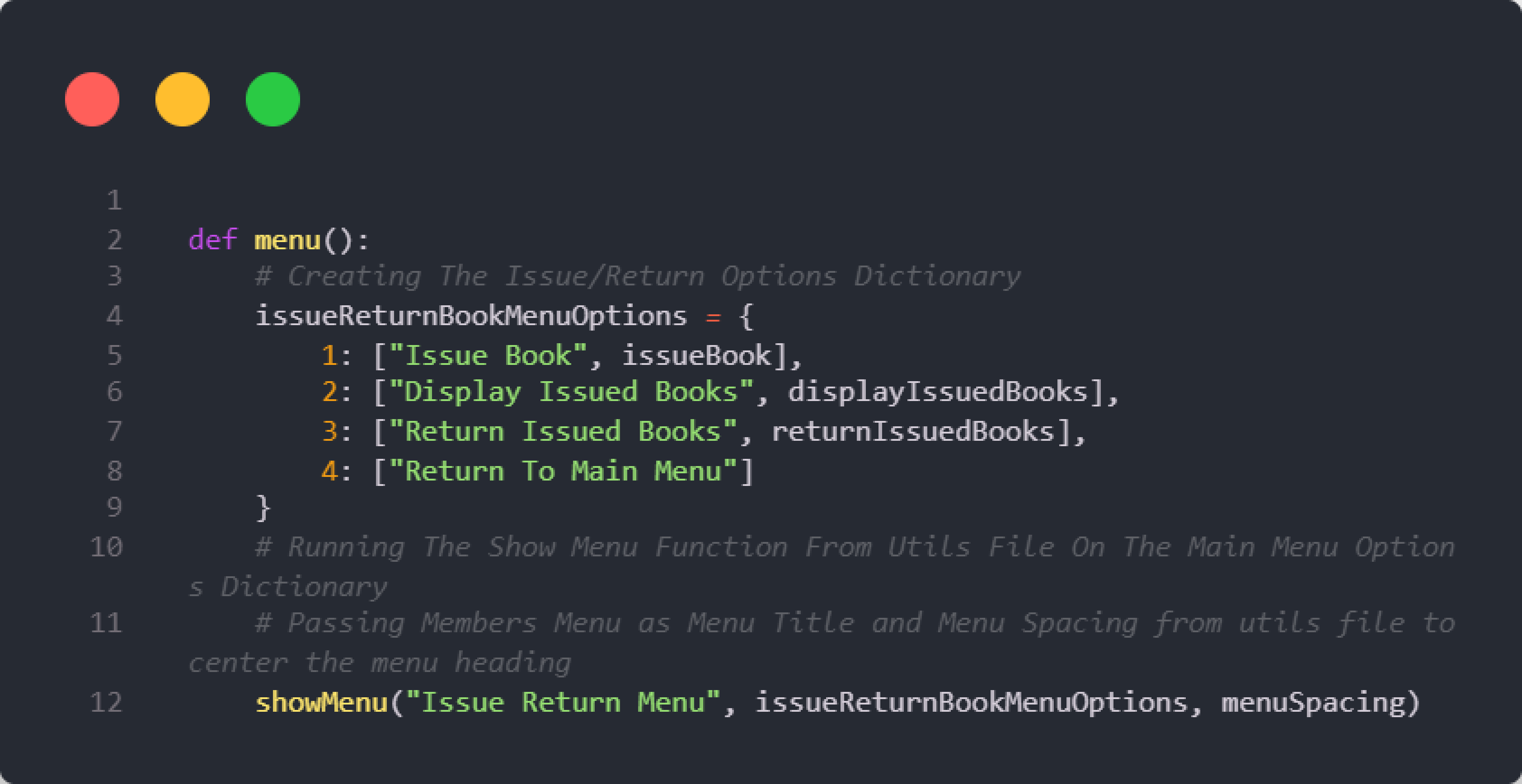
```python
def displayIssuedBooks():
    # Try Catch Block
    try:
        # Executing SQL Query
        query = "select * from issues;"
        cnx = connection.MySQLConnection(
            user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
            host='localhost', database=sqlData.DATABASE_NAME)
        Cursor = cnx.cursor()
        Cursor.execute(query)

        # Creating PrettyTableObject
        x = PrettyTable()
        # Adding Field Names
        x.field_names = ["Book ISBN", "Username", "Date of Issue"]

        for i in Cursor:
            # Using Datetime strftime to convert datetime object to string
            dateString = date.strftime(i[2], "%d-%m-%Y")
            x.add_row([i[0], i[1], dateString])

        # Printing The Table
        print(x)

        # Closing the Connection
        Cursor.close()
        cnx.close()

    except Exception as err:
        print(handleSQLException(err))
```



```python
def returnIssuedBooks():
    # Try Catch Block
    try:
        # Taking User Input
        USER_NAME = input(f"{Colors.OKCYAN}Enter User Name : {Colors.ENDC}")

        # Executing SQL Query
        query = 'select * from issues where username = %s;'
        data = (USER_NAME,)
        cnx = connection.MySQLConnection(
            user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
            host='localhost', database=sqlData.DATABASE_NAME)
        Cursor = cnx.cursor()
        Cursor.execute(query, data)

        # Creating PrettyTableObject
        x = PrettyTable()
        # Adding Field Names
        x.field_names = ["Book ISBN", "User Name", "Date Of Issue"]

        # Creating Found Variable
        found = False
        for i in Cursor:
            # Storing Book Data
            BOOK_ISBN = i[0]
            DATE_ISSUED = i[2]
            # Using Datetime strftime to convert datetime object to string
            dateString = date.strftime(i[2], "%d-%m-%Y")
            data = [i[0], i[1], dateString]
            x.add_row(data)
            # Changing Found Variable TO True
            found = True

        # Printing Error If Not Found
        if(not found):
            x.add_row(["No", "Issue Record", "Found"])
            print(x)
            return

        # Printing The Table
        print(x)
        # Closing the Connection
        Cursor.close()
        cnx.close()
```



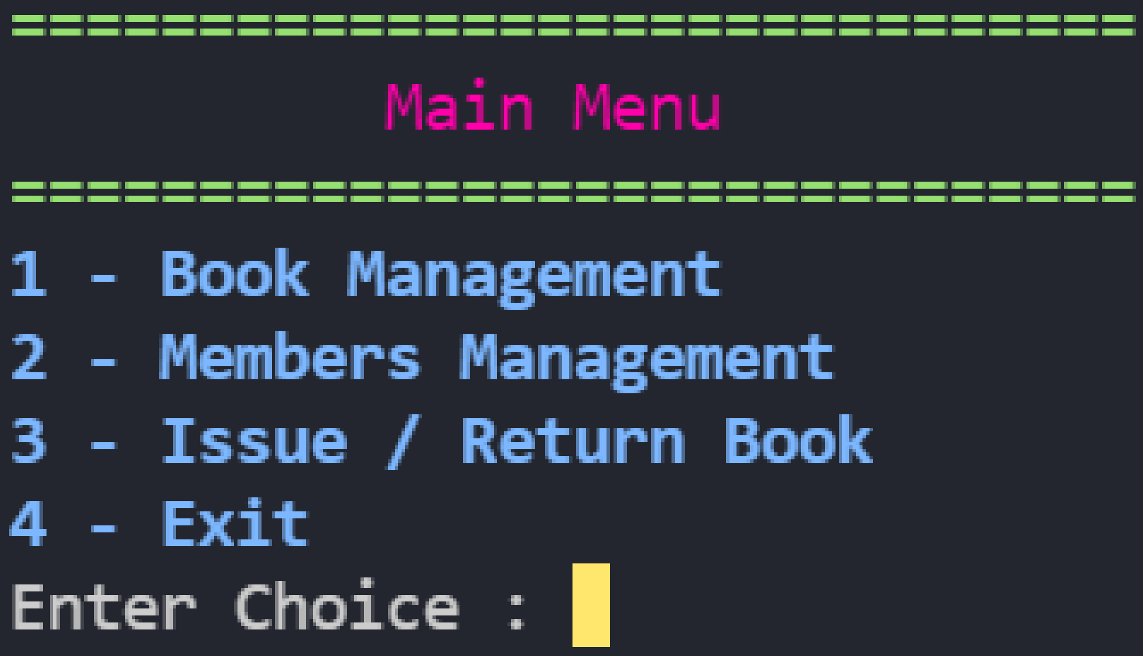
```python
        # Creating Days Issued Variable
        DAYS_ISSUED = 0
        # Calculating Days Issued
        try:
            DAYS_ISSUED = int(str(date.today() - DATE_ISSUED).split()[0]) - 7
        except:
            DATE_ISSUED = 0

        # Changing Days To 0 if it is negative
        if DAYS_ISSUED < 0:
            DAYS_ISSUED = 0

        # Getting User Input For Confirmation
        choice = input(f"{Colors.WARNING} Do You Wish To Return The Book with ISBN {BOOK_ISBN} Associated With User {USER_NAME} The Fine Applicable Would Be Rs. {DAYS_ISSUED * FINE_PER_DAY}(y/n): ")

        # Returning if choice is not y/Y
        if(choice.lower() != "y"):
            clearScreen()
            return

        # Executing SQL Query
        query = 'delete from issues where username=%s'
        data = (USER_NAME,)
        # Executing The Query From Function in utils file
        res = executeSQLCommitQuery(query, data)
        # Clearing Screen For Better Presentation
        clearScreen()
        # If the result comes as a string it means a error has been raised else Member Get Added
        if(type(res) == str):
            print("Error : ", res)
        else:
            print("Issued Record Deleted")
    except Exception as err:
        clearScreen()
        print(err)
```



```python

def menu():
    # Creating The Issue/Return Options Dictionary
    issueReturnBookMenuOptions = {
        1: ["Issue Book", issueBook],
        2: ["Display Issued Books", displayIssuedBooks],
        3: ["Return Issued Books", returnIssuedBooks],
        4: ["Return To Main Menu"]
    }
    # Running The Show Menu Function From Utils File On The Main Menu Options Dictionary
    # Passing Members Menu as Menu Title and Menu Spacing from utils file to center the menu heading
    showMenu("Issue Return Menu", issueReturnBookMenuOptions, menuSpacing)
```

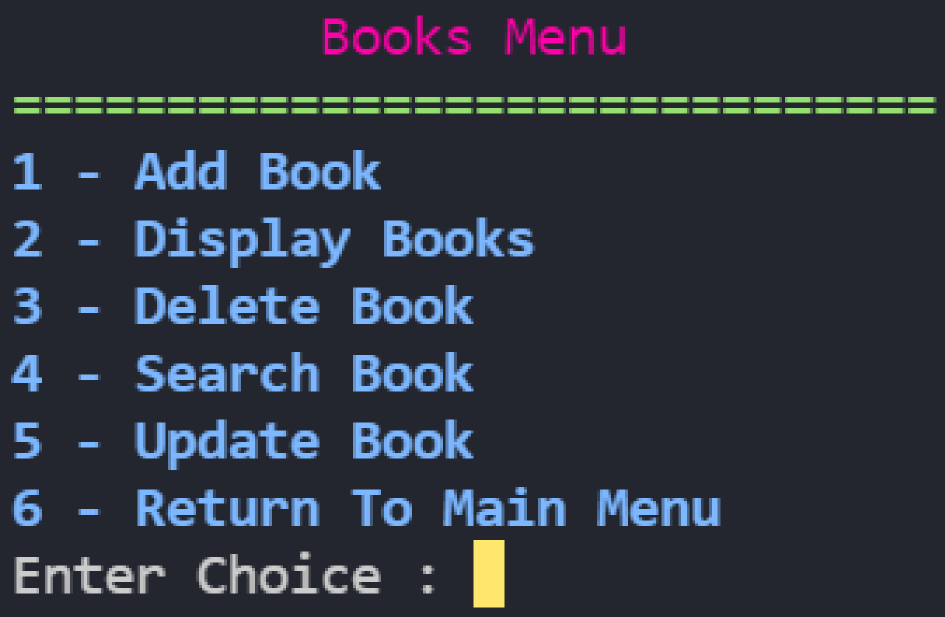
Output

MAIN MENU



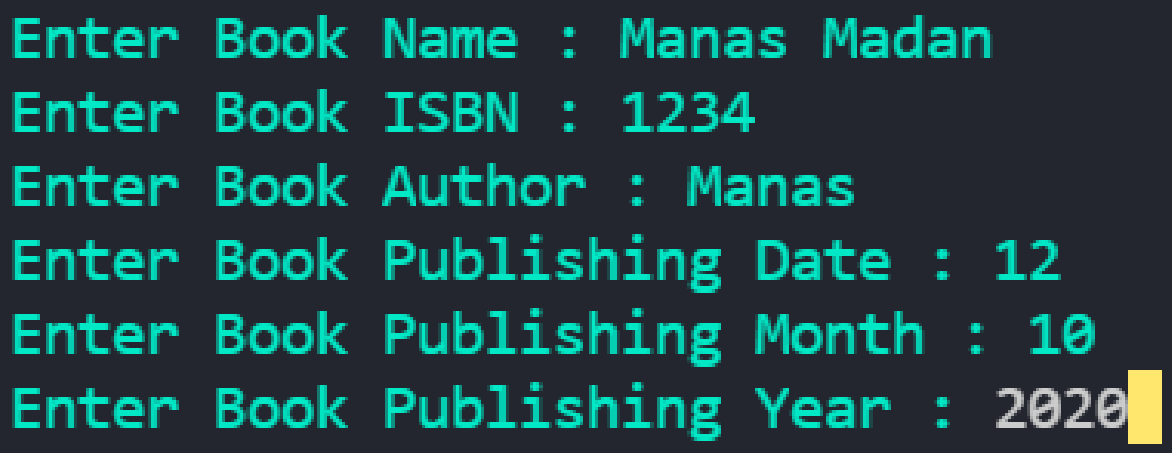
```
==============================
          Main Menu
==============================
1 - Book Management
2 - Members Management
3 - Issue / Return Book
4 - Exit
Enter Choice : 
```

BOOKS MENU



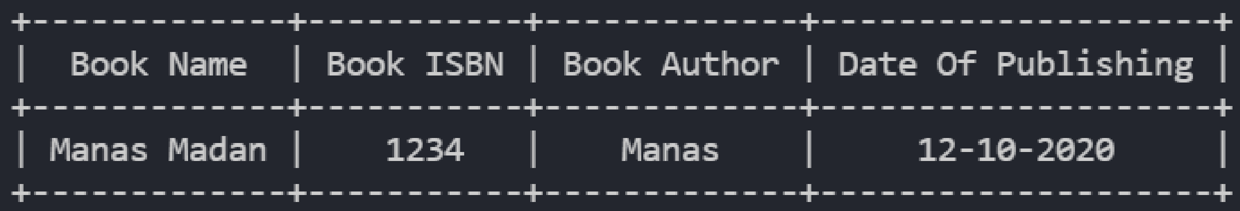
```
          Books Menu
==============================
1 - Add Book
2 - Display Books
3 - Delete Book
4 - Search Book
5 - Update Book
6 - Return To Main Menu
Enter Choice : 
```

ADD BOOK



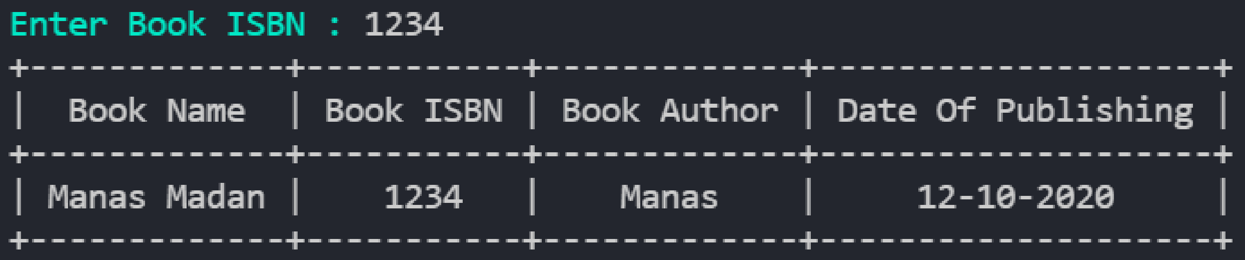
```
Enter Book Name : Manas Madan
Enter Book ISBN : 1234
Enter Book Author : Manas
Enter Book Publishing Date : 12
Enter Book Publishing Month : 10
Enter Book Publishing Year : 2020
```

DISPLAY BOOKS

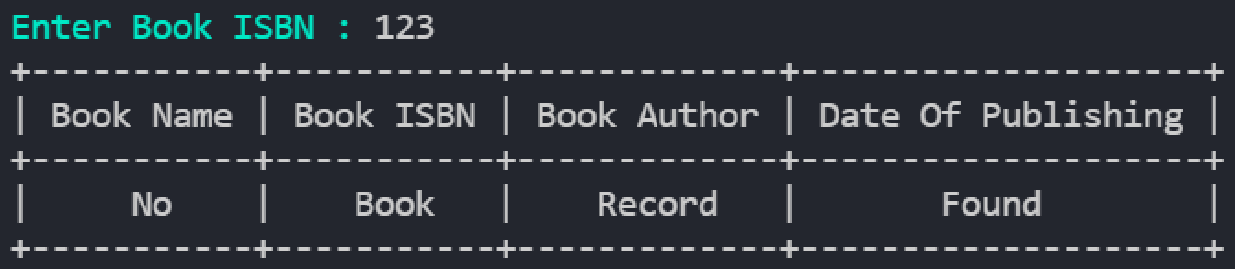


```
+-------------+-----------+-------------+-------------------+
|  Book Name  | Book ISBN | Book Author | Date Of Publishing |
+-------------+-----------+-------------+-------------------+
| Manas Madan |    1234   |    Manas    |     12-10-2020     |
+-------------+-----------+-------------+-------------------+
```

SEARCH BOOKS

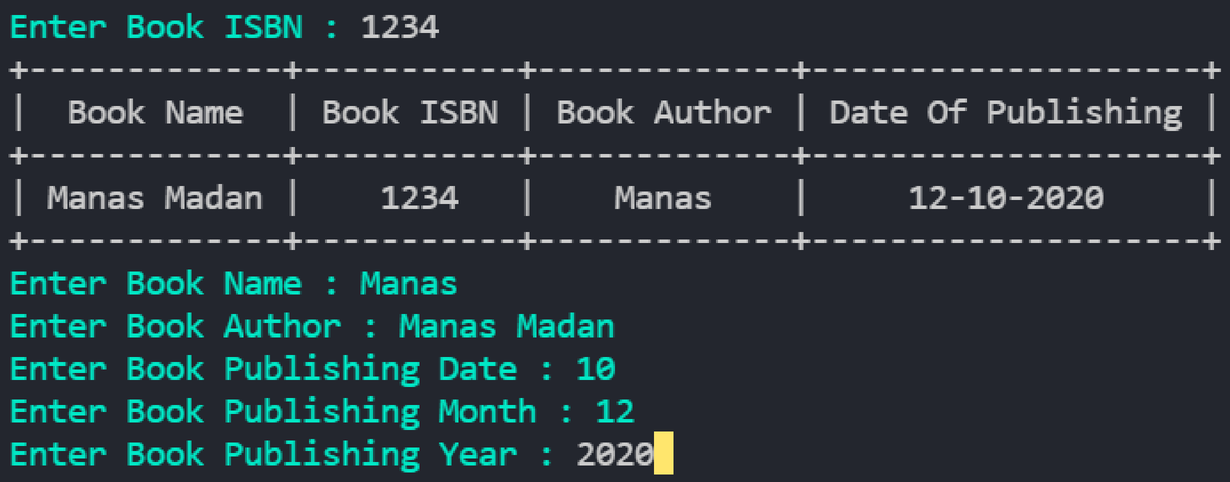


```
Enter Book ISBN : 1234
+-------------+-----------+-------------+-------------------+
|  Book Name  | Book ISBN | Book Author | Date Of Publishing |
+-------------+-----------+-------------+-------------------+
| Manas Madan |    1234   |    Manas    |     12-10-2020     |
+-------------+-----------+-------------+-------------------+
```

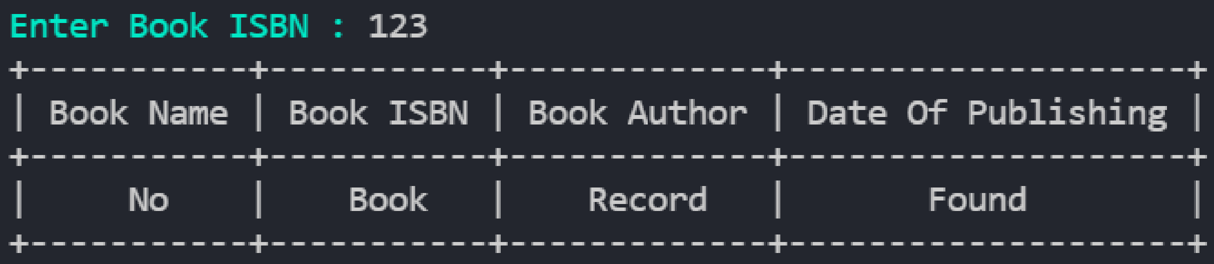


```
Enter Book ISBN : 123
+-----------+-----------+-------------+--------------------+
| Book Name | Book ISBN | Book Author | Date Of Publishing |
+-----------+-----------+-------------+--------------------+
|     No    |    Book   |    Record   |        Found       |
+-----------+-----------+-------------+--------------------+
```

UPDATE BOOKS

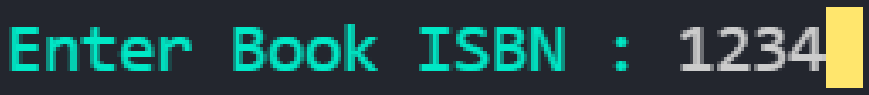


```
Enter Book ISBN : 1234
+------------+----------+------------+--------------------+
|  Book Name  | Book ISBN | Book Author | Date Of Publishing |
+------------+----------+------------+--------------------+
| Manas Madan |    1234   |    Manas    |     12-10-2020     |
+------------+----------+------------+--------------------+
Enter Book Name : Manas
Enter Book Author : Manas Madan
Enter Book Publishing Date : 10
Enter Book Publishing Month : 12
Enter Book Publishing Year : 2020
```



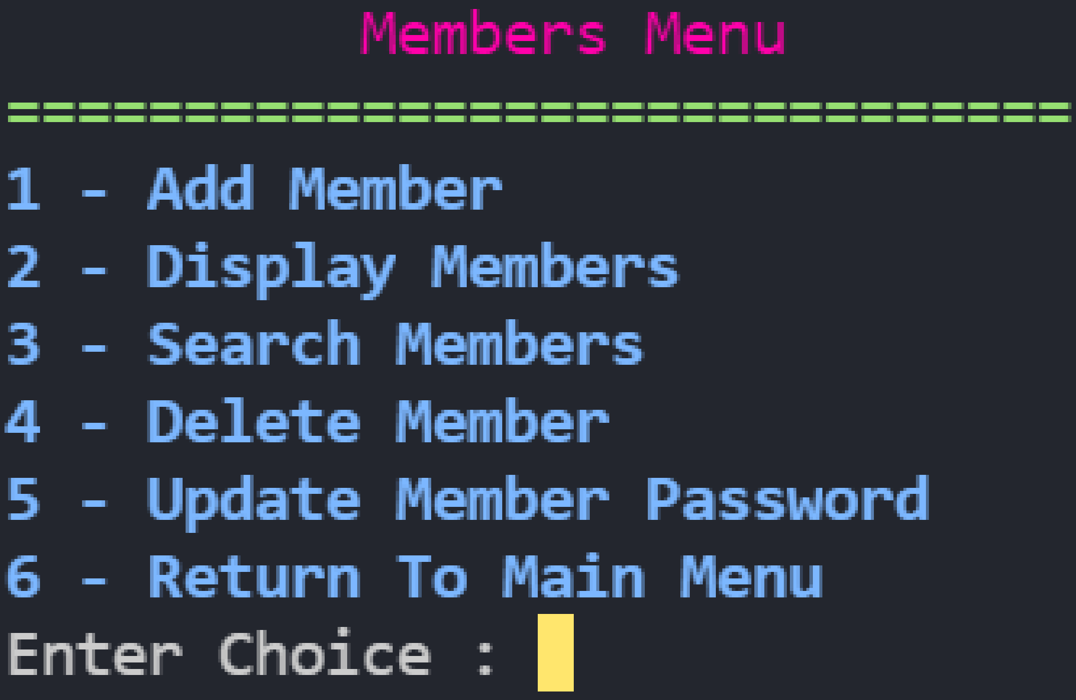
```
Enter Book ISBN : 123
+-----------+----------+------------+--------------------+
| Book Name | Book ISBN | Book Author | Date Of Publishing |
+-----------+----------+------------+--------------------+
|     No    |    Book   |    Record   |        Found       |
+-----------+----------+------------+--------------------+
```

DELETE BOOK



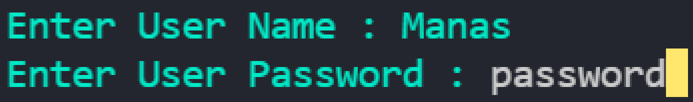
```
Enter Book ISBN : 1234
```

MEMBERS MENU

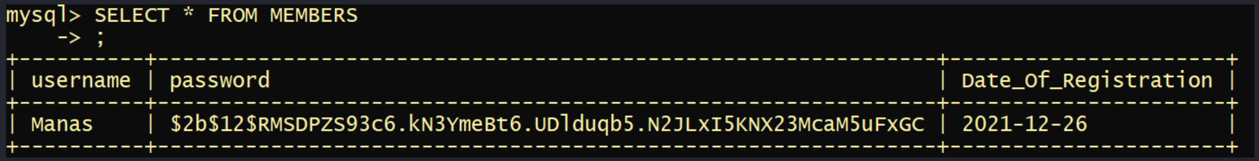


```
          Members Menu
==============================
1 - Add Member
2 - Display Members
3 - Search Members
4 - Delete Member
5 - Update Member Password
6 - Return To Main Menu
Enter Choice :
```

ADD MEMBER

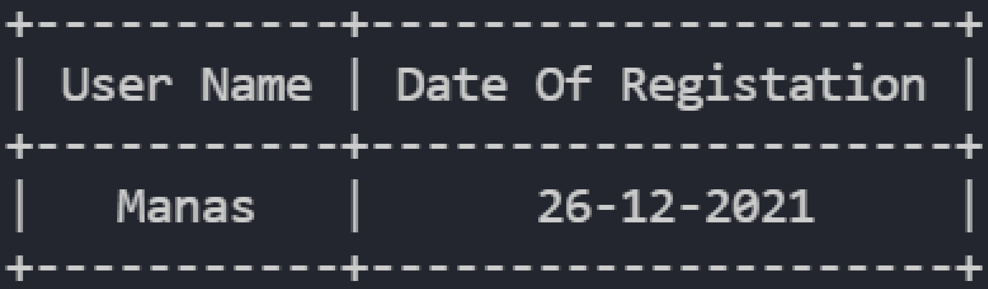


```
Enter User Name : Manas
Enter User Password : password
```



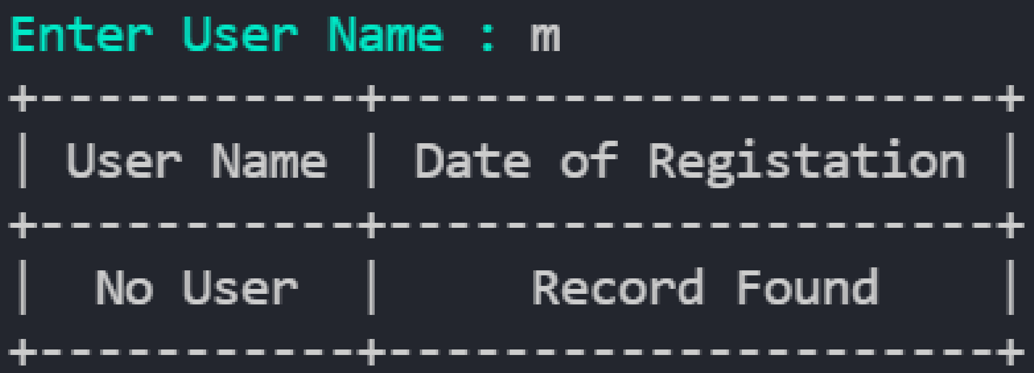
```
mysql> SELECT * FROM MEMBERS
    -> ;
+----------+--------------------------------------------------------------+----------------------+
| username | password                                                     | Date_Of_Registration |
+----------+--------------------------------------------------------------+----------------------+
| Manas    | $2b$12$RMSDPZS93c6.kN3YmeBt6.UDlduqb5.N2JLxI5KNX23McaM5uFxGC | 2021-12-26           |
+----------+--------------------------------------------------------------+----------------------+
```

DISPLAY MEMBERS

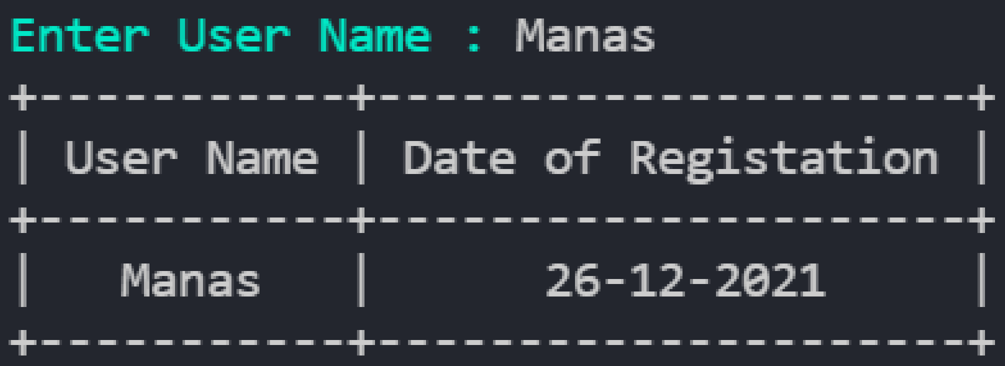


```
+-----------+---------------------+
| User Name | Date Of Registation |
+-----------+---------------------+
|   Manas   |     26-12-2021      |
+-----------+---------------------+
```

SEARCH MEMBERS

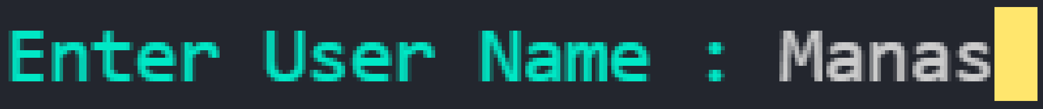


```
Enter User Name : m
+-----------+----------------------+
| User Name | Date of Registation |
+-----------+----------------------+
|  No User  |     Record Found     |
+-----------+----------------------+
```



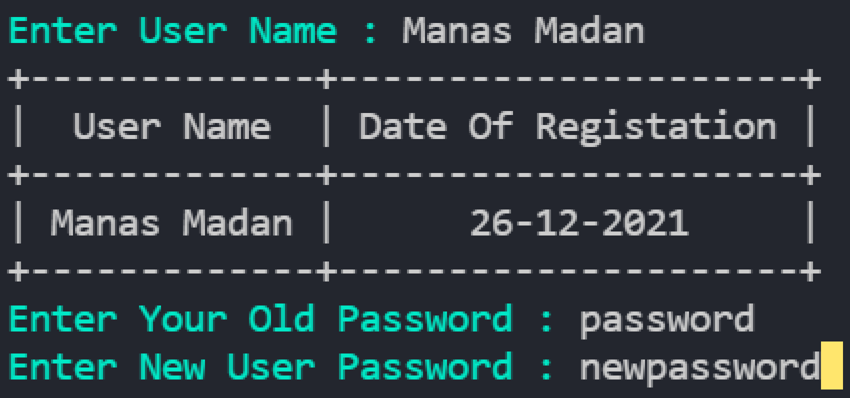
```
Enter User Name : Manas
+-----------+----------------------+
| User Name | Date of Registation |
+-----------+----------------------+
|   Manas   |      26-12-2021      |
+-----------+----------------------+
```

DELETE MEMBER



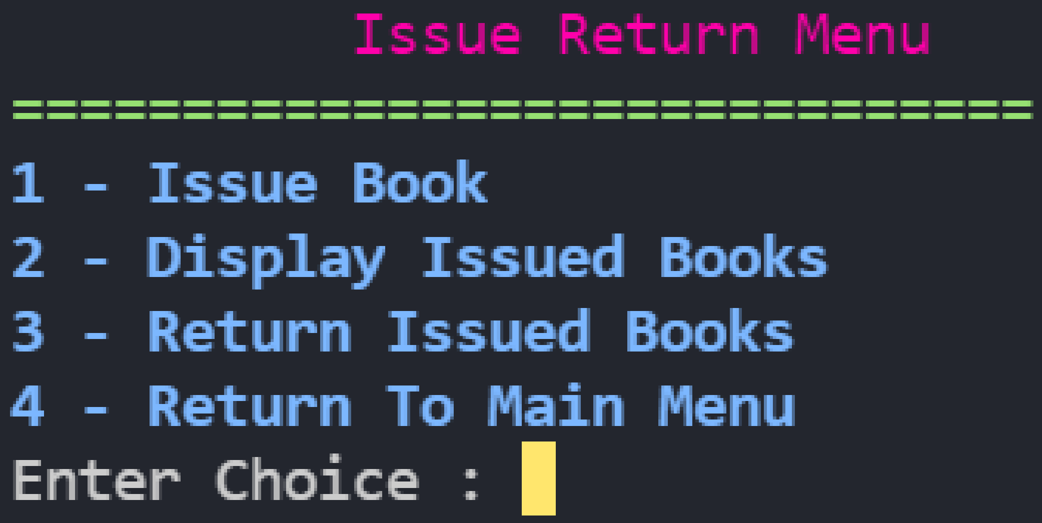
```
Enter User Name : Manas
```

UPDATE MEMBER PASSWORD



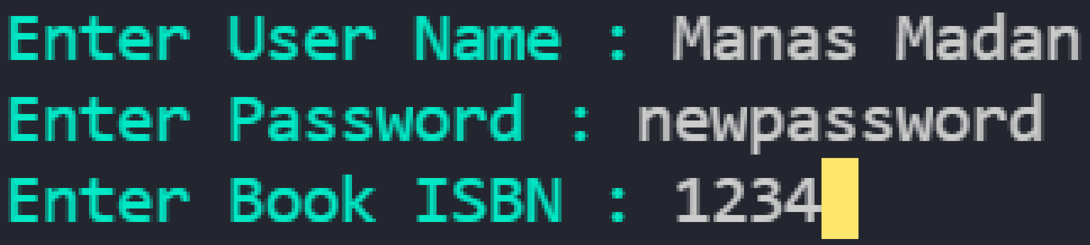
```
Enter User Name : Manas Madan
+-------------+---------------------+
|  User Name  | Date Of Registation |
+-------------+---------------------+
| Manas Madan |     26-12-2021      |
+-------------+---------------------+
Enter Your Old Password : password
Enter New User Password : newpassword
```

ISSUE MENU



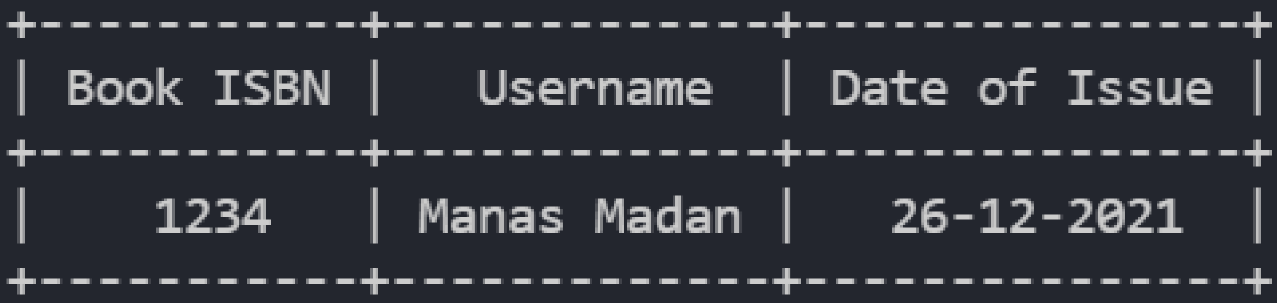
```
           Issue Return Menu
==============================
1 - Issue Book
2 - Display Issued Books
3 - Return Issued Books
4 - Return To Main Menu
Enter Choice :
```

ISSUE BOOK



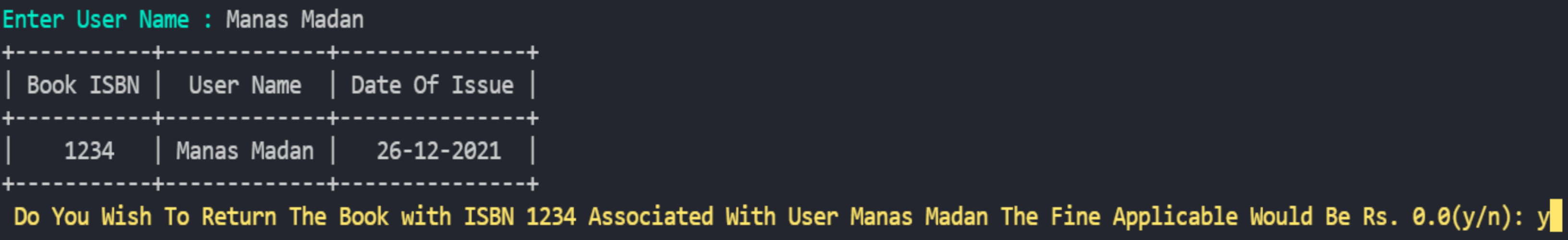
```
Enter User Name : Manas Madan
Enter Password : newpassword
Enter Book ISBN : 1234
```

DISPLAY ISSUED BOOKS



```
+-----------+-------------+---------------+
| Book ISBN |  Username   | Date of Issue |
+-----------+-------------+---------------+
|   1234    | Manas Madan |  26-12-2021   |
+-----------+-------------+---------------+
```

RETURN ISSUED BOOK



```
Enter User Name : Manas Madan
+-----------+------------+--------------+
| Book ISBN |  User Name | Date Of Issue |
+-----------+------------+--------------+
|    1234   | Manas Madan |   26-12-2021  |
+-----------+------------+--------------+
 Do You Wish To Return The Book with ISBN 1234 Associated With User Manas Madan The Fine Applicable Would Be Rs. 0.0(y/n): y
```

Modules Used

- bcrypt - 3.2.0
    - To Hash Passwords
- mysql-connector-python - 8.0.27
    - To Establish Connection between python code and mysql database
- prettytable - 2.5.0
    - To print sql tables beautifully
- datetime - inbuilt
    - To add in DATE column of sql

BIBLIOGRAPHY & REFERENCES

- SQL Reference
- Colored Python Terminal
- Python and MySQL Connector Reference
- MySQL Python Connector Docs