Menu:

```
#student ID:	f225694

#menu

from bookCheckout import *

from bookReturn import *

from bookSearch import *

from bookSelect import *

import time

print("Welcome to the library, how can we help")

options = ['SEARCH','CHECKOUT','RETURN','SELECT']


def menu():

   '''asks the user what they want to do and efficiently decides whhat function to take them to

      throughout the file.'''

   options1 = input("What would you like to do? Enter \n\t'Search' to search for a book: \n\t'Checkout'"

            "to checkout a book: \n\t'Return' to return a book: \n\t'Select' to select a book for purchase: \n:")

   options2 = options1.upper() #converts the input to uppercase, making the input not case sensitive

   if options2 == options[0]:

      print("SEARCH")

      book_search()

      time.sleep(0.5) #this adds a time delay before calling the next function so it bahaves more naturally

      check()

      #go to search function

   elif options2 == options[1]:

      print("CHECKOUT")

      memberID()

      time.sleep(0.5)

      check()

      #go to checkout function
```

```
    elif options2 == options[2]:
        print("RETURN")
        Book_Return()
        time.sleep(0.5)
        check()
        #go to return function
    elif options2 == options[3]:
        print("SELECT")
        MemberID()
        time.sleep(0.5)
        check()
        #go to select function
    else:
        print("Error, please try again")
        menu()



def check():
    '''this function is used to see if the user would wish to perform further actions
       within the system or if they are finsished and would like to quit'''
    check = input("Enter 'Q' if finished: \nEnter menu to return to menu: \n\t: ")
    check1 = check.upper()#This also converts the input to uppercase, making the input not case
sensitive
    if check1 == 'Q':
        exit()
    elif check1 == 'MENU':
        print("Going back to menu")
        time.sleep(0.5)
        menu()
```

```
if __name__ == "__main__":
   print(menu())
   print(check())
```

Database:

```
#student ID:     f225694
#database.py
import time
log = []
shelf = []
def sortdata():
   '''the sortdata function opens the files and takes the data from each file
      and splits it up so each line(book) is its own sublist in the list Data
      or log.'''
   f = open("books1.txt","r")
   for line in f:
      a = line.strip() #removes the white space on each line
      b = a.split(";") #splits each line at the semi colons for when its written into data
      shelf.append(b)
   f.close()
   L = open("logfile.txt","r")
   for line in L:
      a = line.strip()
      b = a.split(";")
      log.append(b)
   f.close()
   #must close the file everytime you open it, otherwise the contents inside may be lost.
```

```
search_data_list = ['ERROR']
def search_data(search):
    '''in this function its primary purpose is to search through the data put into log
       and shelf so it can determine whther the book is available or not deoending on
       the state of the log file.'''
    sortdata() #data must be organised first in order to be searched through
    for book in shelf:
        if search in book:
            book.remove(book[-1])#these manipulate the sublist so its in the correct format to then be
written into the list
            book.remove(book[-1])
            book.remove(book[-1])
            book.append('[AVAILABLE]')#initially all books are available until it checks the log file for
transaction history
            search_data_list.remove(search_data_list[0])
            search_data_list.append(book)

    for book in log:
        if search in book and book[-2] == '-':
            book.remove(book[-1])
            book.remove(book[-1])
            book.remove(book[-1])
            book.append('[UNAVAILABLE]')
            search_data_list.append(book)
    print(search_data_list)



def checkout():
    '''the checkout function within database, asks for an input from the user and
       initially checks the log file to see if it has already been taken out and
```

```
      is yet to be returned, if so it will give the user the option to reserve the book.
      otherise the book is checked out and data from shelf is manipulated to fit into logfile'''
    sortdata()
    search = input("Search the ID of the book you would like to take out: ")
    for book in shelf:
        if book[0] == search:
            print("")
            print(book)
            print("")
            check = input("Would you like to take this book out: ")
            if check == 'no':
                checkout()
            else:
                for book in log:
                    if book[0] == search and book[2] != "-" and book[-2] == "-":
                        rsrv = input("\nBook has already been checked out, Would you like to reserve the book: 
") #gives the user a choice and an option to reserve a book
                        if rsrv == 'yes' and book[-1] != '[RESERVED]':
                            OP = open("logfile.txt","a+")
                            book.remove(book[-1])
                            book.append("[RESERVED]")
                            join = ';'.join(book)
                            OP.seek(0) #moves curser back to the start of file.
                            OPR = OP.read(100)
                            if len(OPR) > 0:
                                OP.write("\n")
                            OP.write(join)
                            OP.close()
                            print(book,"\nits being reserved for you")
                            return None
                        elif rsrv == 'no':
```

```
            checkout()
        elif book[-1] == '[RESERVED]':
            print("book is already being reserved")
            time.sleep(0.5)
            checkout2 = input("Would you like to checkout a different book? ")
            if checkout2 == 'yes':
                checkout()
            else:
                break
    #break
for book in shelf:
    if book[0] == search:
        book.remove(book[-1])
        book.remove(book[-1])
        book.remove(book[-1])
        book.append(day)
        book.append('-')
        book.append('-')
        OP = open("logfile.txt","a+")
        join = ';'.join(book)
        OP.seek(0)
        OPR = OP.read(100)
        if len(OPR) > 0:
            OP.write("\n")
        OP.write(join)
        OP.close()
        print("book has been successfully checked out\n")
        time.sleep(0.5)
        return None
```

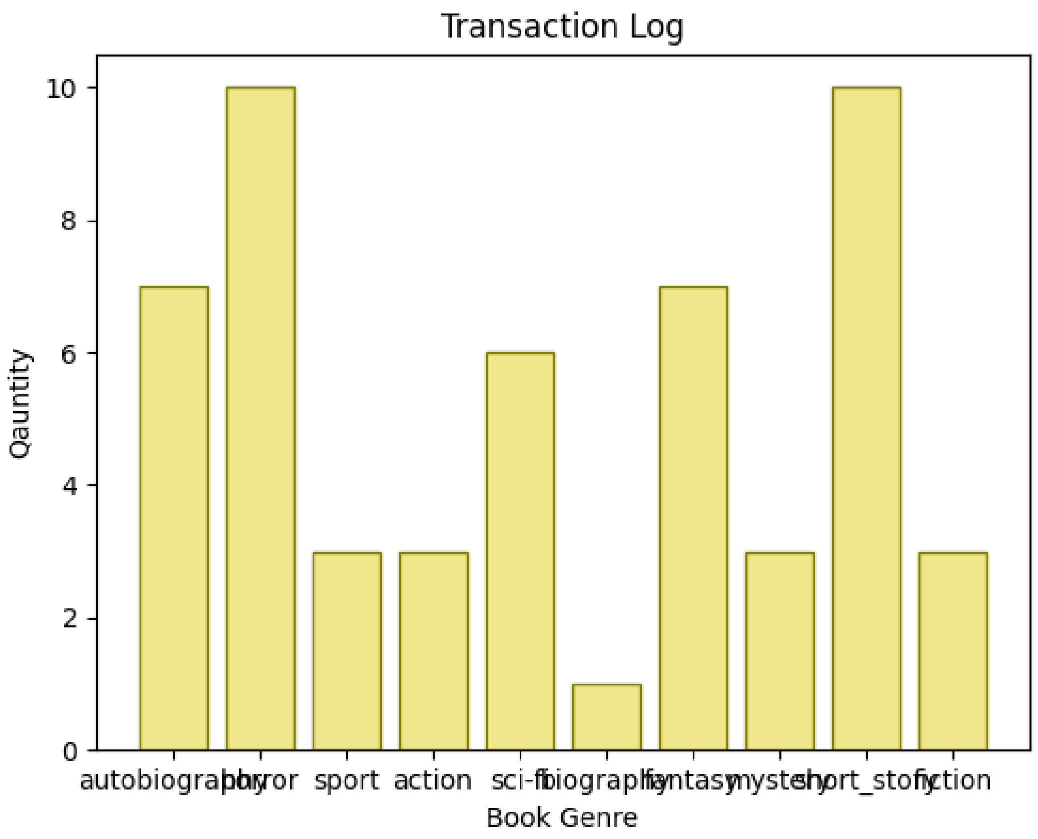
```
#this is a function that allows the user to use the current date when checking out
#and returnig books in and out of the library
import datetime
datetime_1 = datetime.date.today()
day = str(datetime_1)


def returns(BookID):
   '''here the parameter is used of the book id from an input statement and its compared
      with criteria which tells the programme whether the book has already been returned
      or that it needs returning and has been returned successfully/'''
   sortdata()
   for book in log:
      if book[0] == BookID and book[-2] == '-':
         book.remove(book[-1])
         book.remove(book[-1])
         book.append(day)
         OP = open("logfile.txt","a+")
         join = ';'.join(book)
         OP.seek(0)
         OPR = OP.read(100)
         if len(OPR) > 0:
            OP.write("\n")
         OP.write(join)
         OP.close()
         break

      elif book[0] == BookID and book[-2] != '-': #book has already been returned
         print("invalid request")
         time.sleep(0.5)
         break
```

```
if __name__ == "__main__":
    print(sortdata()) #should print none because there arent any necessary print statements
    print(returns("001")) #should print none as the print statement is in another file.
    print(checkout()) #should ask the user to enter the book id thye wish to take out.
    print(search_data("action")) #should print all the action books and whether they are available or
not.
```

Book Select:

```
#student ID:     f225694
#bookSelect.py
from database import *
#- recommends the popular book genres that are being checked out the most
#- the books in that genre are listed and an average price is given and the
 #value of all the current books in that genre are given
#- based on the budget and the average price the programme should output how
 #mnay books it recommends buying.
def MemberID():
    '''this part of bookselect gets the user to enter there 4 digit ID in order
       to login, once they've done that then it takes them to the transaction log'''
    MID = input("please enter your member ID in order to proceed: ")
    MID1 = MID.isdecimal()
    MID1 = str(MID1)
    MID2 = len(MID)
    if MID1 == "True" and MID2 == 4:
        print("")
        print("correct")
        print("")
        print("Here is a visual reprisentation of the transaction log")
        print("")
        time.sleep(0.5)
        transaction_log()
    else:
        print("Error, incorrect ID. \tPlease try again")
        time.sleep(0.5)
        MemberID()



import operator
def transaction_log():
    '''this function recommends the popular book genres that are being checked out the most
       and its also represented visually through a bar chart which is initially shown to user.
       The books in those genre's are listed and an average price is given and the value of all
       the current books in that genre are given. Based on the budget and the average price the
       programme should output how many books it recommends buying.'''
    sortdata()
    autobio_count = 0
    horror_count = 0
    sport_count = 0
    action_count = 0
    sci_fi_count = 0
    biography_count = 0
    fantasy_count = 0
    mystery_count = 0
    short_story_count = 0
    fiction_count = 0
    for book in log:
        if book[1] == "autobiography" and book[-2] == "-":
            autobio_count = autobio_count + 1
        elif book[1] == "horror" and book[-2] == "-":
            horror_count = horror_count + 1
        elif book[1] == "sport" and book[-2] == "-":
            sport_count = sport_count + 1
        elif book[1] == "action" and book[-2] == "-":
            action_count = action_count + 1
        elif book[1] == "sci-fi" and book[-2] == "-":
            sci_fi_count = sci_fi_count + 1
        elif book[1] == "biography" and book[-2] == "-":
            biography_count = biography_count + 1
        elif book[1] == "fantasy" and book[-2] == "-":
            fantasy_count = fantasy_count + 1
        elif book[1] == "mystery" and book[-2] == "-":
            mystery_count = mystery_count + 1
        elif book[1] == "short_story" and book[-2] == "-":
            short_story_count = short_story_count + 1
        elif book[1] == "fiction" and book[-2] == "-":
            fiction_count = fiction_count + 1
    count_list = {"autobiography":autobio_count, "horror":horror_count, "sport":sport_count,
                  "action":action_count, "sci-fi":sci_fi_count, "biography":biography_count,
                  "fantasy":fantasy_count, "mystery":mystery_count, "short_story":short_story_count,
"fiction":fiction_count}
    key1 = [key for key in count_list.keys()][0]
    key2 = [key for key in count_list.keys()][1]
    key3 = [key for key in count_list.keys()][2]
    key4 = [key for key in count_list.keys()][3]
    key5 = [key for key in count_list.keys()][4]
    key6 = [key for key in count_list.keys()][5]
    key7 = [key for key in count_list.keys()][6]
    key8 = [key for key in count_list.keys()][7]
    key9 = [key for key in count_list.keys()][8]
    key10 = [key for key in count_list.keys()][-1]
    import matplotlib.pyplot as plt
    genre = [key1,key2,key3,key4,key5,key6,key7,key8,key9,key10]
    quantity =
[autobio_count,horror_count,sport_count,action_count,sci_fi_count,biography_count,fantasy_count
,mystery_count,short_story_count,fiction_count]
    plt.bar(genre,quantity,color=('khaki'),edgecolor='olive')
    plt.title("Transaction Log")
    plt.xlabel("Book Genre")
    plt.ylabel("Qauntity")
    plt.savefig('tLog.png')
    plt.show()
    sorted_list = dict( sorted(count_list.items(), key=operator.itemgetter(1),reverse=True))
    order = sorted_list.keys()
    order1 = list(sorted_list.keys())
    print("The three most popular genre of books are "+order1[0]+", "+order1[1]+", "+order1[2]+
          " so it's recommended that the library purchases more of these genres, that are in high
demand!")
    print("")
    time.sleep(0.5)
    budget = input("please enter the budget you have in GBP: ")
    n1 = budget.isdecimal()
    n1 = str(n1)
    while n1 == 'False':
        print('error')
        budget = input("please enter the budget you have in GBP: ")
        n1 = budget.isdecimal()
        n1 = str(n1)
    budget = float(budget)
    avrgeForEachGenre = budget/3 #how much to spend on books for 3 most in-demand genres
    print("based on the libraries budget of £%4.2f it should look to spend £%4.2f on each
genre:"%(budget, avrgeForEachGenre))
    print("\n\t"+order1[0]+"\n\t"+order1[1]+"\n\t"+order1[2]+"\n")
    price1 = 0
    price2 = 0
    price3 = 0
    for book in shelf:
        if order1[0] in book:
            n1 = int(book[4])
            price1 = price1 + n1

        elif order1[1] in book:
            n2 = int(book[4])
            price2 = price2 + n2

        elif order1[2] in book:
            n3 = int(book[4])
            price3 = price3 + n3

    SLV = list(sorted_list.values()) #how many books of that genre have been taken out
    SLV1 = int(SLV[0])
    SLV2 = int(SLV[1])
    SLV3 = int(SLV[2])
    avrgForBook1 = price1/SLV1 #average price for books in each genre; price1/2/3 = price of all
    avrgForBook2 = price2/SLV2 #books in the genre that have been taken out and SLV1/2/3 is how
many of each genre was taken out
    avrgForBook3 = price3/SLV3
    n1 = avrgeForEachGenre/avrgForBook1 #n1/2/3 is how many books they can buy per genre
according to the budget
    n2 = avrgeForEachGenre/avrgForBook2
    n3 = avrgeForEachGenre/avrgForBook3
    print("")                                              #printing blank lines to space out the text
so its more user friendly
    print("From "+order1[0]+", you should look to purchase around %1d books for the future! "%(n1))
    print("")
    print("From "+order1[1]+", you should look to purchase around %1d books for the future! "%(n2))
    print("")
    print("From "+order1[2]+", you should look to purchase around %1d books for the future! "%(n3))
    print("(All figures based off prices of current books)")
    print("")



if __name__ == "__main__":
    print(MemberID()) #will prompt user to enter 4 digit member ID
    print(transaction_log()) #will print a diagram of most popular genres in a bar chart
```

Transaction Log:



Book Search:

```
#student ID:	f225694
#book search:
from database import *
def book_search():
   '''this function only takes the input from the user, the search_data function
      in database.py contains the bulk for searching through the files.'''
   search = input("please search for id number, genre, title, author or price: ")
   search_data(search) #this parameter is what the user inputs and is passed into
                  #database.py so it can be used in the search_data function





if __name__ == "__main__":
   print(book_search()) #will prompt user to search for something
```

Book Return:

```
#student ID:	f225694
#bookReturn
from database import *
import time
def Book_Return():
   '''Book_Return takes an input from the user which is then used as a parameter further down to be
      used in the database for criteria for the book, once the process in database has been completed
      then the programme outputs a statement.'''
   BookID = input("please enter the ID number of the book you wish to return: ")
   sortdata()
   for book in shelf:
      if book[0] == BookID:
            print("")
            print(book)
            print("")
            time.sleep(0.5)
            check = input("would you like to return this book? ")
            if check == 'no':
                book_Return()
            else:
                returns(BookID)
                break
    print("book has been returned succesfully")



if __name__ == "__main__":
    print(Book_Return()) #will ask user to enter the ID number of the book they wish to return
```

Book Checkout:

```
#student ID:      f225694
from database import *
import time
def memberID():
    '''this function takes an input from the user and manipulates it to assess whethehr it meets
       the criteria of length 4 and fro it to be only integers'''
    MID = input("Please enter your member ID in order to checkout a book: ")
    MID1 = MID.isdecimal() #checks whether the input given is made up of only integers
    MID1 = str(MID1)  #converts the booleon response from isdecimal function into string
    MID2 = len(MID)   #finds length of input
    if MID1 == "True" and MID2 == 4:
        print("Valid ID")
        time.sleep(0.5)
        checkout()#go to search for book
    else:
        print("Invalid ID \tPlease try again")
        time.sleep(0.5)
        memberID() #returns back to top of code for user to re-enter there member id




if __name__ == "__main__":
    print(memberID()) #will ask the user to input member ID
```

Books.txt:

```
001;biography;the_fight;norman_mailer;8;7/3/1975
002;autobiography;gloves_off:_the_autobiography;tyson_fury;9;4/7/2022
003;autobiography;the_first_half;gabby_logan;10;4/5/2015
004;autobiography;open;andre_agassi;7;19/12/2009
005;horror;the_shining;stephen_king;8;28/1/1977
006;horror;bird_box;josh_malerman;8;27/3/2014
007;sci-fi;dune;frank_herbert;10;20/8/1965
008;sci-fi;neuromancer;william_gibson;8;1/7/1984
009;sport;the_sweet_science;A.J_liebling;15;5/5/1949
010;fantasy;a_game_of_thrones;george_r.r_martin;7;1/8/1996
011;horror;beloved;toni_morrison;4;06/16/1987
012;horror;world_war_z;max_brooks;10;01/01/2006
013;action;jurassic_park;michael_crichton;18;09/11/1990
014;action;sahara;clive_cussler;10;19/10/1992
015;action;congo;michael_crichton;9;03/11/1980
016;action;desert_star;michael_connelly;8;09/09/2022
```

```
017;fiction;life_of_pi;yann_martel;14;16/05/2001

018;fiction;lesons_in_chemistry;bonnie_garmus;12;10/12/2022

019;short_story;fragile_things;neil_gaiman;6;03/09/2006

020;short_story;runaway;alice_munro;9;28/05/2004

021;short_story;exhalation;ted_chiang;14;06/06/2008

022;short_story;dubliners;james_joyce;2;27/05/1914

023;mystery;gone_girl;gillian_flynn;4;09/09/2012

024;mystery;the_paris_apartment;lucy_foley;19;04/05/2022

025;fantasy;jade_city;fonda_lee;11;11/11/2017

026;fantasy;six_of_crows;leigh_bardugo;5;18/09/2015

027;fantasy;the_fellowship_of_the_ring;J.R.R_tolkien;8;29/03/1954

028;fantasy;fairy_tale;stephen_king;11;05/05/2015

029;sci-fi;fahrenheit_451;ray_bradbury;20;19/10/1953

030;sci-fi;the_martian;andy_weir;8;04/05/2011

031;horror;bird_box;josh_malerman;8;27/3/2014

032;mystery;gone_girl;gillian_flynn;4;09/09/2012
```

Log File:

```
002;autobiography;gloves_off:_the_autobiography;2022-12-06;-;-

002;autobiography;gloves_off:_the_autobiography;2022-12-06;-;[RESERVED]

029;sci-fi;fahrenheit_451;2022-12-06;-;-

019;short_story;fragile_things;2022-12-06;-;-

012;horror;world_war_z;2022-12-06;-;-

030;sci-fi;the_martian;2022-12-06;-;-

004;autobiography;open;2022-12-06;-;-

026;fantasy;six_of_crows;2022-12-06;-;-

008;sci-fi;neuromancer;2022-12-06;-;-

019;short_story;fragile_things;2022-12-06;-;-

012;horror;world_war_z;2022-12-06;-;-

009;sport;the_sweet_science;2022-12-06;-;-

010;fantasy;a_game_of_thrones;2022-12-06;-;-
```

```
022;short_story;dubliners;2022-12-06;-;-
019;short_story;fragile_things;2022-12-06;-;-
002;autobiography;gloves_off:_the_autobiography;2022-12-06;-;[RESERVED]
026;fantasy;six_of_crows;2022-12-06;-;-
009;sport;the_sweet_science;2022-12-06;-;[RESERVED]
019;short_story;fragile_things;2022-12-06;2022-12-08
013;action;jurassic_park;2022-12-08;-;-
032;horror;bird_box;2022-12-11;-;-
012;horror;world_war_z;2022-12-06;-;[RESERVED]
030;sci-fi;the_martian;2022-12-06;-;[RESERVED]
019;short_story;fragile_things;2022-12-06;-;[RESERVED]
032;horror;bird_box;2022-12-11;-;[RESERVED]
032;horror;bird_box;2022-12-11;-;-
016;action;desert_star;2022-12-11;-;-
017;fiction;life_of_pi;2022-12-11;-;-
004;autobiography;open;2022-12-06;-;[RESERVED]
002;autobiography;gloves_off:_the_autobiography;2022-12-06;-;[RESERVED]
001;biography;the_fight;2022-12-11;-;-
009;sport;the_sweet_science;2022-12-06;-;[RESERVED]
009;sport;the_sweet_science;2022-12-06;2022-12-11
032;horror;bird_box;2022-12-11;-;[RESERVED]
002;autobiography;gloves_off:_the_autobiography;2022-12-11;-;-
027;fantasy;the_fellowship_of_the_ring;2022-12-11;-;-
017;fiction;life_of_pi;2022-12-11;-;[RESERVED]
005;horror;the_shining;2022-12-14;-;-
007;sci-fi;dune;2022-12-14;-;-
030;sci-fi;the_martian;2022-12-06;-;[RESERVED]
020;short_story;runaway;2022-12-14;-;-
021;short_story;exhalation;2022-12-14;-;-
023;mystery;gone_girl;2022-12-14;-;-
023;mystery;gone_girl;2022-12-14;-;[RESERVED]
```

```
023;mystery;gone_girl;2022-12-14;2022-12-14

015;action;congo;2022-12-15;-;-

019;short_story;fragile_things;2022-12-06;-;[RESERVED]

020;short_story;runaway;2022-12-14;-;[RESERVED]

024;mystery;the_paris_apartment;2022-12-15;-;-

026;fantasy;six_of_crows;2022-12-06;-;[RESERVED]

023;mystery;gone_girl;2022-12-14;2022-12-15

026;fantasy;six_of_crows;2022-12-06;2022-12-15

011;horror;beloved;2022-12-15;-;-

017;fiction;life_of_pi;2022-12-11;-;[RESERVED]

011;horror;beloved;2022-12-15;2022-12-15

010;fantasy;a_game_of_thrones;2022-12-06;-;[RESERVED]

019;short_story;fragile_things;2022-12-06;-;[RESERVED]

031;horror;bird_box;2022-12-15;-;-

028;fantasy;fairy_tale;2022-12-15;-;-

032;horror;bird_box;2022-12-11;2022-12-15

001;biography;the_fight;2022-12-11;2022-12-15

001;biography;the_fight;2022-12-11;2022-12-15

001;biography;the_fight;2022-12-11;2022-12-15

001;biography;the_fight;2022-12-11;2022-12-15
```

student ID: f225694

What i struggled with:

- using the files and filtering out certain books under some circumstances through if statements

- writing the test code because i wasnt sure how to get it to work with my code as i dont use many parameters

- the book search function and getting it to show what books are available or not.

what im proud of:

- im proud of the book select function that creates the bar chart for all the genres and how it returns the three most popular genres