CSC213 project

Khalid Ebrahem Alhumaidan-391108165

Ebrahem Abdullah Alzain-391114405

Introduction

In this project we choose to do it in python, as it's known for its flexibility and simplicity and many other features. We modified the input file a little bit so it works but it's still the same concept, also it'll run even if you make changes to the input file. We used two python built-in modules: JSON and tkinter. We got the data from the input file to the source code by JSON, and to make the code more flexible we used a class from the module tkinter called "filedialog" and using the function "askopenfilename()" it makes you select the json data file from your file browser.

For the statistical charts, we used matplotlib. If you press the task number you want the chart will appear.

Language features we used

1- (tkinter) We imported tkinter to use the class “filedialog” to make the user select the file data and imported json to open and read the data file, and messagebox to display results.

```
from tkinter import filedialog
from tkinter import messagebox
from tkinter import *
```

2- (matplotlib) We imported matplotlib.pyplot to use it for making charts.

```
import matplotlib.pyplot as plt
```

3- (json) We imported json to use it to get the data.

```
import json
```

4- (tkinter) We created tkinter object to use the function “askopenfilename()”.

```
Tobj = Tk() #tkinter object
Tobj.withdraw()
file_path = filedialog.askopenfilename(title= "Select the data file(.Json)",filetypes = [("json data file
","*.json")]) #aks the user to select the json data file
Tobj.withdraw()
```

5- (json)Using json we open the data file and loaded the data into “data” variable.

```
with open(file_path) as i:  #here python opens the data file that was selected from the code above
    data = json.load(i)     #here assign the data file to val "data"
```

6- (number of members) We created a variable to extract and save the number of members from the data file.

```
number_of_members = len(data["Members"])    #calculating the number of members in the data file using the
 built in function "len()"
```

7- (member cycling) This function enables us to cycle through the members.

```
def MemberCycling(i):  #this function cycles between members in the data file
    Member = "Member1"
    memedit= list(Member)

    memedit[6]= str(i)
    Member = "".join(memedit)   #this function takes the word "Member1" and map it in to a list using "li
st()" built in function and then
    return Member               #modefies the last element from "1" to the last member (n) in the datafil
e
```

8- (book cycling) This function does exactly what the function above but for books.

```
def BookCycling(j):    #this function cycles between books in the data file
    bookstring = "Book1"
    bookedit= list(bookstring)

    bookedit[4] = str(j)
    bookstring = "".join(bookedit)  #this function takes the word "Member1" and map it in to a list using
 "list()" built in function and then
    return bookstring               #modefies the last element from "1" to the last book that is in the m
ember in the datafile
```

9- (number of books) This function counts the number of books using the function in #6 and a while loop, show the total number of books in a message box.

```
def NumberOfBooks(number_of_members): #Number of books read by the whole group members
    number_of_books = 0
    i = 1
    while i <= number_of_members:#cycle through members and add all books
        number_of_books += int(len(data["Members"][MemberCycling(i)]["Books"]))
        i += 1
    print_number = "Number of books:" + str(number_of_books)
    messagebox.showinfo("Task One", print_number)   #here to display the result in a popup window
```

10- (number of pages) Here we count the number of pages using a counter and the two functions in #6,7 and two nested loops and show the total number of pages in a message box.

```
def number_of_pages(number_of_members) : #Number of pages read by the whole group members
    number_of_pages = 0

    for f in range(1,number_of_members+1): #to cycle between members
        MemberCycling(f)
        for j in range(1,len(data["Members"][MemberCycling(f)]["Books"])+1): #to cycle between books
            number_of_pages += int(data["Members"][MemberCycling(f)]["Books"][BookCycling(j)]["Number of
pages"])
    print_pages = "Total number of pages:" + str(number_of_pages)
    messagebox.showinfo("Task two", print_pages)    #here to display the result in a popup window
```

11- (ranking by categories) This function appends all the Categories in a list using #6,7 and nested loops, and then creating a new dictionary using the built in “dict()” and then sort them using “sorted()”. For the chart, we used “.keys()” and “.values()” to get the contents of the dictionary “count_categories_sorted” to use them in the chart as x and y axis. We decided to go with bar chart because they’re simple and easy to look at, we gave the chart a title, legend, x-axis label, y-axis label and changed the window’s title.

```
def RankingOfCategories(number_of_members) : #Ranking of books categories mostly read by the group members
    plt.figure(figsize=(15,5))  #resizing the window
    categories = []
    count_categories = {}

    for f in range(1,number_of_members+1): #to cycle between members
        MemberCycling(f)
        for j in range(1,len(data["Members"][MemberCycling(f)]["Books"])+1): #to cycle between books
            categories.append(data["Members"][MemberCycling(f)]["Books"][BookCycling(j)]["Category"]) # add the category to the list categories

    for cat in categories:          #here the loop cycles every category that is in var "categories" and then if the category is not already in the dict "count_categories" the if statement makes
        if cat in count_categories: #the category a key value and initialize it by "1" after that if the category is already available it adds one to the key value
            count_categories[cat] += 1
        else:
            count_categories[cat] = 1

    count_categories_sorted = dict(sorted(count_categories.items(), key= lambda x:x[1],reverse= True)) #here we sorted the dict using "dict(sorted())" built in function and gave it "count_categories" to cycle through

    plt.bar(count_categories_sorted.keys(),count_categories_sorted.values(),label="Books")  #to implement a bar chart
    plt.title("Ranking of categories.")
    plt.legend()
    plt.xlabel('Categories')
    plt.ylabel('Number of books')
    plt.gcf().canvas.set_window_title("Task Three") #to change the window title
    plt.show()  #to show the chart
```

12- (ranking based on books read) Here we rank the members based on how many books they read, we cycle through the members and books using #6,7. Using two dictionaries, one loop, we created the second dictionary using “dict()”, sorted the dictionary by the function “sorted()”. As for the chart we made pretty much the same as before #10.

```
def RankingBasedOnBooksRead(number_of_members): #Ranking of group members based on number of books read
    plt.figure(figsize=(15,5))  #resizing the window
    count_books = {} #dict to store the total number books read by members
    m = 1
    while m <= number_of_members: #to cycle between members
        count_books[MemberCycling(m)] = len(data["Members"][MemberCycling(m)]["Books"]) #it puts the tota
l number of books in the current member in count_books
        m += 1

    countbooks_sorted = dict(sorted(count_books.items(), key= lambda x:x[1],reverse= True)) #sort the dic
t count_books using "dict(sorted())" built in function

    plt.bar(countbooks_sorted.keys(),countbooks_sorted.values(),color='#444444' ,label="Books")   #here w
e used the built-in func called Matplotlib to make the charts
    plt.title("Ranking based on books read.")
    plt.legend()
    plt.xlabel("Members")
    plt.ylabel("Number of books read")
    plt.gcf().canvas.set_window_title("Task four")  #to change the window title
    plt.show()  #to show the chart
```

13- (ranking based on pages read) Using two dictionaries and a counter to add the total pages read by each member (we used #6,7 to cycle through the members and books), we created the second dictionary by ”dict()” and then used “sorted()” function to sort. The chart is the same as before #10.

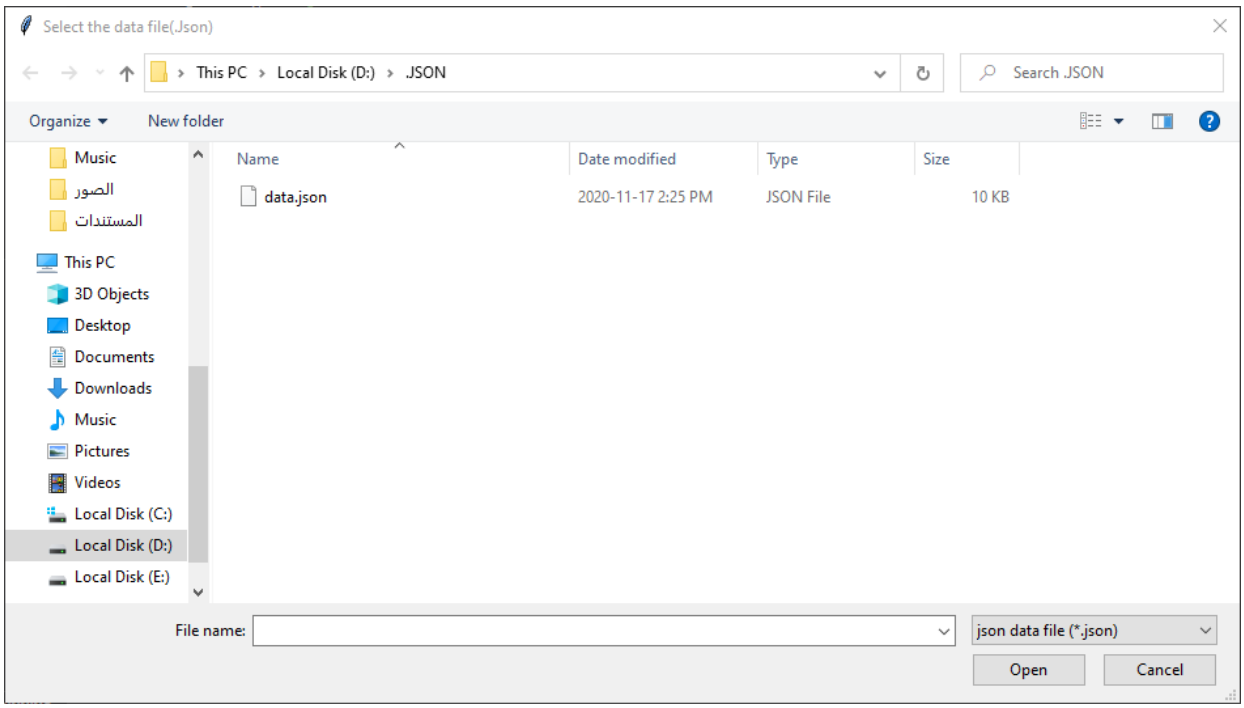
```
def RankingBasedOnPagesRead(number_of_members): #Ranking of group members based on number of pages read
    plt.figure(figsize=(15,5))  #resizing the window
    number_of_pages = 0
    countpages = {} #dict to store the total number of pages read by a member

    for m in range(1,number_of_members+1):  #to cycle between members
        MemberCycling(m)
        number_of_pages = 0  #reset the number of pages to 0, so they don't add up with other members
        for j in range(1,len(data["Members"][MemberCycling(m)]["Books"])+1): #to cycle between books to g
et number of pages

            number_of_pages += int(data["Members"][MemberCycling(m)]["Books"][BookCycling(j)]["Number of
pages"]) #adding number of pages
            countpages[MemberCycling(m)] = number_of_pages #putting it the dict

    countpages_sorted = dict(sorted(countpages.items(), key= lambda x:x[1],reverse= True)) #sort the dict
 countpages

    plt.bar(countpages_sorted.keys(),countpages_sorted.values(),color="g" ,label="Pages")   #here we used
 the built-in func called Matplotlib to make the charts
    plt.title("Ranking based on Pages read.")
    plt.legend()
    plt.xlabel("Members")
    plt.ylabel("Number of Pages read")
    plt.gcf().canvas.set_window_title("Task five")  #to change the window title
    plt.show()  #to show the chart
```

14- (tkinter GUI) Here we used the tkinter for the graphical user interface, first we put the tasks in functions, then we make the buttons and assign the tasks, placements.

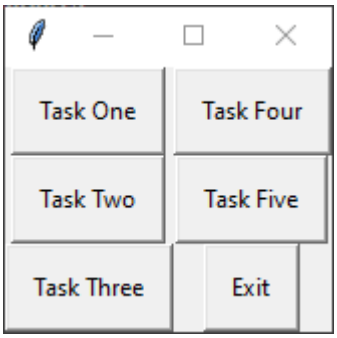
```
Tobj = Tk() #object for the tkinter
Tobj.title("CS213 Project")
def TaskOne():
    NumberOfBooks(number_of_members)

def TaskTwo():
    number_of_pages(number_of_members)

def TaskThree():
    RankingOfCategories(number_of_members)

def TaskFour():
    RankingBasedOnBooksRead(number_of_members)

def TaskFive():
    RankingBasedOnPagesRead(number_of_members)    #functions for the buttons in the gui

ButtonOne = Button(Tobj, text= "Task One",padx= 10,pady=10, command = TaskOne)
ButtonOne.grid(row=0, column=0)
ButtonTwo = Button(Tobj, text= "Task Two",padx= 10,pady=10,command = TaskTwo)
ButtonTwo.grid(row=1, column=0)
ButtonThree = Button(Tobj, text= "Task Three",padx= 10,pady=10,command = TaskThree)
ButtonThree.grid(row=2, column=0)
ButtonFour = Button(Tobj, text= "Task Four",padx= 10,pady=10,command = TaskFour)
ButtonFour.grid(row=0, column=4)
ButtonFive = Button(Tobj, text= "Task Five",padx= 10,pady=10,command = TaskFive)
ButtonFive.grid(row=1, column=4)
ButtonExit = Button(Tobj, text= "Exit",padx= 10,pady=10,command = Tobj.quit)
ButtonExit.grid(row=2, column=4)    #here's the gui buttons and there placement
Tobj.mainloop()
```

15- Output:

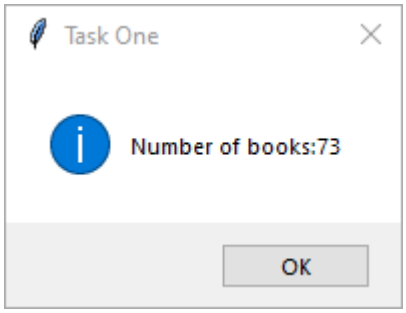
File select:



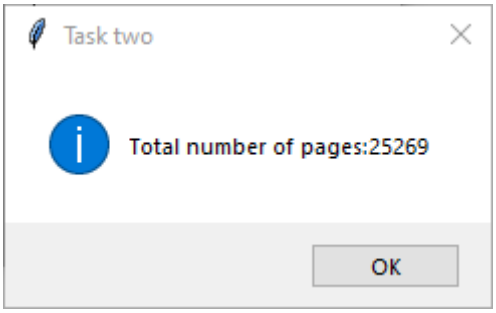
GUI:



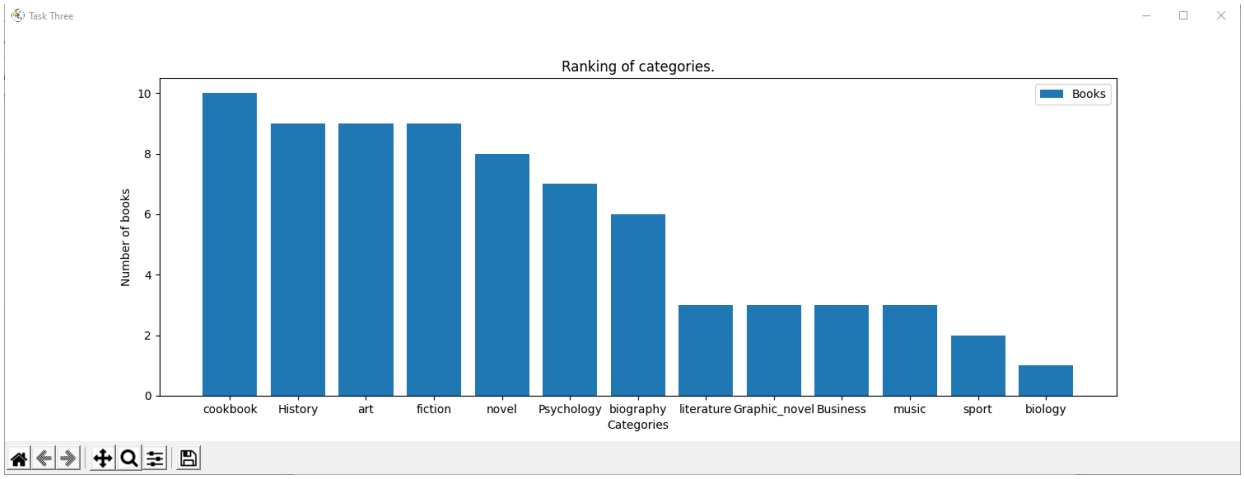
Task 1:



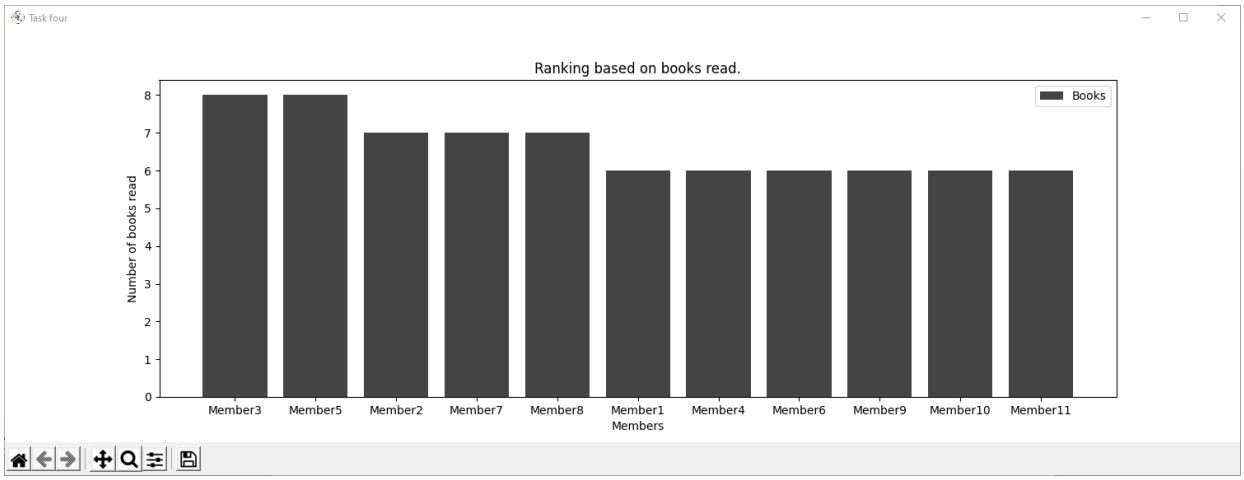
Task 2:



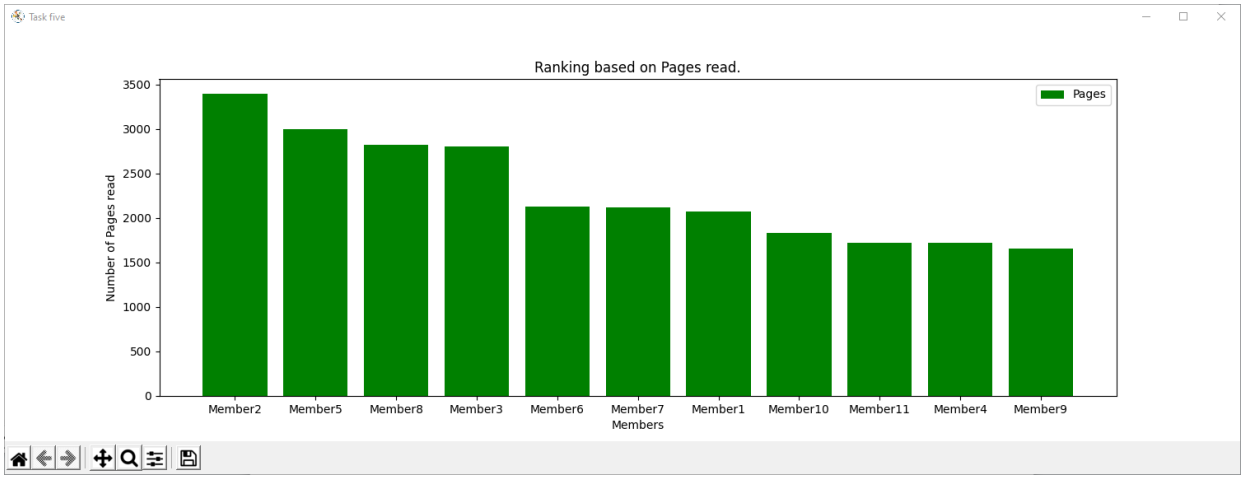
Task 3:



Task 4:



Task 5:



Language feature	Class/file name	Line
Tkinter.	Importing#1, object#3, GUI#13, in functions.	1-3 , 9-12 , 50 , 64 , 149 , 150 , 166-178.
Matplotlib.	Importing#2, in functions.	4 , 71 , 88-94 , 101 , 110-116 , 123 , 137-143.
Json.	Importing#3, object#5.	5 , 15 , 16.
Member cycling.	Function #7, in functions.	23-29 , 47 , 61 , 62 , 77 , 78 , 105 , 128 , 132 , 133.
Book cycling.	Function#8, in functions.	33-39 , 62 , 78 , 132.
Number of books.	Function#9.	43-50 , 152.
Number of pages.	Function#10.	56-64 , 155.
Ranking of categories.	Function#11.	70-94 , 158.
Ranking based on books read.	Function#12.	100-116 , 161.
Ranking based on pages read.	Function #13.	122 , 143 , 164.
If statement.	#11.	81-84.
Loops.	In functions.	46 , 59 , 61 , 75 , 77 , 80 , 104 , 127 , 130.