Assignment 01
Programming for scientists
CSM0120
Tob31

i. Executive summary

For this assignment I wrote a single python file split up into various functions that can be called in main, when uncommented, to be able to run the code for that task. I chose to layout the code like this because its easy for the reader to run/test the various task functions for the tasks set in this way.

Task 1, I wrote a simple function that calculated and returned the percentage of Birmingham and Manchester characters in the cities string entered by a user's input, and round the output to 2 decimal places.

This task was very easy as all I need to do was use python's inbuilt functions count and sum the total of Birmingham and Manchester characters divided by the length of the input string and then use the inbuilt round function to return the value which was be printed to the screen when the function was called in main.

Task 2, I wrote a simple function here calculated the total sale of the ticket by referencing the ticket table which I had put into a global dictionary and outputs the total price rounded to 2 decimal places after taking in the cities characters input string, This task was very simple as I only need to write a short function and have the table in a dictionary so that it could be parsed in to be referenced.

Task 3, This task was also very simple as I only needed to write one small function which took in the cities characters input string. The function had a premade dictionary if the cities key characters in and empty counts, the counts for each key characters increased given the number of key characters the cities characters input string contained. For example, if the input was this "BBBCL" the output would be {'B': 3, 'C': 1, 'H': 0, 'L': 1, 'M': 0, 'N': 0, 'S': 0}.

Task 4, This task was more complicated however had a simple process to follow as it required the use of code which I had written for the previous task. Before I begin giving an overview of this tasks logic I need to mention that in the python file there are 2 versions of this task, task4loopFinal is the final function which should be assessed however there is another function task4loop which was the original function I wrote that believe logistically works better than the final function I wrote.

However, it doesn't meet all of the task requirements since it doesn't read the file and take the input that way. It simply stores the inputs when the user inputs them and uses them to calculate the desired output and then writes it to a file. This is not what the task asked for however I believe it makes better sense to do it this way.

Task4loopFinal meets all of the requirements from task 4 it simply uses while loop that only ends with the user inputs quit meaning it keeps asking the user to input cities string characters, if an unrecognized symbol is present it will output/print the appropriate message to the screen.

It saves the input cities string characters to a text file named with the current date when quit and then the program reads the file and calculates the desired output ie. How many tickets have been sold for each city, total price of the tickets, and the number of hours in which the tickets of each city have been sold using sets to ensure that if a ticket entry has 2 of the same city, it should count as one.

ii. Technical overview

The main section: This is the main function, as you can see, I have included some example test inputs to try when running the code although any sufficient test input would suffice. The code for each task is labeled with comments here to make it easier to see which functions need to be called for each task.

```
if __name__ == '__main__':
    # uncomment the code below to run the code for the specific tasks

    # Task 1
    CitiesString = "BMMNSB" #Test Example
    print(percentOfBrumAndMan(input("Enter cities string Task 1\n")))

    #Task 2
    CitiesString = "BMMNSBK" #Test Example containing a wrong character k
    print(ticketTablePrice(input("Enter cities string Task 2\n"), ticketTableDict))

    # Task 3
    CitiesString = "BCCBBLNNLLBBMMMBCBCLLMSMBMBMBMNMNSSLSCS" #Test Example
    print(numberOfTicketsSold(CitiesString))

    # Task 4
    # BBMNH MMMBB #Test Example
    #task4loop(ticketTableDict) # this function is the old task 4 function which does not
    task4loopFinal(ticketTableDict)# this is the main task 4 function to be run which meet
```

Task 1

```
# Task 1
#CitiesString = "BMMNSB" #Test Example
print(percentOfBrumAndMan(input("Enter cities string Task 1\n")))
```

This is where the task 1 function is called from main it includes and input statement for the user to input the cities string on a new line “\n” when the prompt is printed to the screen.

```
from datetime import date
```

```
def percentOfBrumAndMan(string):
    string = string.upper()  #converts to upper
    percentage = (string.count("B") + string.count("M")) / int(len(string))*100
    # calculates the percentage with /int(len(string) on the count
    return round(percentage, 2)
```

This is the task 1 function called percentOfBrumAndMan it takes in the cities string from the user input, converts it to all uppercase using .Upper() then use the count functions to count the occurrences of B and M then sums then up together, then divides that sum by the length of the string multiplied by 100 to get the percentage. The function then returns that percentage rounded to 2 decimal places using the round function. Datetime is also imported here to be used later.

Task 2

```
#Task 2
# CitiesString = "BMMNSBK" #Test Example containing a wrong character k
print("£"+str(ticketTablePrice(input("Enter cities string Task 2\n"), ticketTableDict)))
```

This is where the task 2 function is called from main it includes and input statement for the user to input the cities string on a new line “\n” when the prompt is printed to the screen, it also parses in the ticket table dictionary for referencing.

```
ticketTableDict = {
        'B': {'City': 'Birmingham', 'Ticket Price': 29.2},# the t
        'C': {'City': 'Cardiff', 'Ticket Price': 16.6},
        'H': {'City': 'Hereford', 'Ticket Price': 34.7},
        'L': {'City': 'Liverpool', 'Ticket Price': 31.7},
        'M': {'City': 'Manchester', 'Ticket Price': 38.0},
        'N': {'City': 'Newport', 'Ticket Price': 19.8},
        'S': {'City': 'Shrewsbury', 'Ticket Price': 15.2}
    }
```

This is the dictionary I created from the table provided in the it’s a dictionary of a dictionary with the cities characters keys and the values of those are a dictionary of city’s and ticket price as the keys and the names of the cities and the prices as the values.

```
def ticketTablePrice(string, ticketTableDict):
    string = string.upper()  # converts to upper
    price = 0.0
    for i in string:
        if i in ticketTableDict:
            price += ticketTableDict[i]['Ticket Price']
        elif i == ' ':
            pass
        elif not i in ticketTableDict:
            print("Unrecognised city symbol detected " + str(i))
    return round(price, 2)
```

This is the task 2 function called ticketTablePrice. It parses in the cities string and the ticket table dictionary for referencing. It converts the cities string to all upper case to avoid a case sensitivity error. A float is also initialized for price to build up the total price.

The for loop starts by iterating through the string and then uses the if statement to check if it is contained within the dictionary, if it is it increments the total price by the ticket prices reference from that dictionary and that key. I have other elif statements in here error catching. For example if it reads a blank space ‘ ‘ it ignores it and passes, also if it detects a character that’s not a key in the dictionary it prints an appropriate error message.

Task 3

```
# Task 3
CitiesString = "BCCBBLNNLLBBMMMBCBCLLMSMBMBMBMNMNSSLSCS" #Test Example
print(numberOfTicketsSold(CitiesString))
```

This is where the task 3 function is called from main it includes a reference to string CitiesString as a test example, this can be changed, the input string here is long and so it is easier to have it prewritten.

```
def numberOfTicketsSold(string):
    string = string.upper() # conver
    outputDict = { # dictionary of t
        'B': 0,
        'C': 0,
        'H': 0,
        'L': 0,
        'M': 0,
        'N': 0,
        'S': 0,
    }
    for i in string:
        if i in outputDict:
            outputDict[i] += 1 # inc
    return outputDict
```

This is the task 3 function called numberOfTicketsSold. It parses in the cities string. It converts the cities string to all upper case to avoid a case sensitivity error. A dictionary is also initialized here it's a smaller simpler version of the table dictionary from the previous task.

The for loop starts by iterating through the string and then uses the if statement to check if it is contained within the output dictionary, if it is it increments the count in the dictionary by one. After its finished iterative through the string it outputs the dictionary to be printed to the screen.

Task 4

Task4loopFinal (the actual answer for the task 4 requirements)

```
# Task 4
# BBMNH MMMBB #Test Example
#task4loop(ticketTableDict) # this function is the old task 4 function which do
task4loopFinal(ticketTableDict)# this is the main task 4 function to be run whi
```

This is where the task 4 function is called from main it parses in the ticket table dictionary for referencing within the function.

```
def task4loopFinal(ticketTableDict):
    totalLetters = "" # initialises a string to increament the city key characters

    filename = str(date.today()) # creates the filename from the current date using
    filename += ".txt"
    try:
        file = open(filename, "w")

        quit = False # creates a boolean quit, so it can be used to exit out of the
        while quit == False: # while loop that keeps going while quit = false
            string = input("Enter cities string Task 4\n") # asks the users to inpu
            if string == "quit": # runs once the user inputs quit
                file.write(totalLetters) #writes the cities ticket string to the fi
                file.close()# closes the file
                quit = True # ends the loop
                quitFunction(filename)# calls the quit function to calculate the va

            else:
                Currentletters = "" # initialises a string to increament the city k
                for i in string:
                    if i in ticketTableDict:
                        Currentletters += i #for each element in the current cities

                    elif not i in ticketTableDict: # if not output this text and co
                        print("Error. Unrecognized city symbol is detected. Please
                        continue
                totalLetters = totalLetters + " " + Currentletters #adds the curren
    except:
        print("file cannot open")
```

This is the task 4 function called task4loopFinal it parses in the ticket table dictionary ticketTabledict. I initialize a string called totalLetters this is for the buildup of collected ticket characters.

This next section is for creating the file, I create a string variable that's equal to the string of the current date and add .txt on the end to create the filename I then use this to open a file in write mode. With try and except for added defensiveness.

I then initialize a Boolean variable quit to false so that the while will run continuously until it is updated. It then uses the input function to ask the user to input the cities string on a newline "\n" which will be assigned to the variable string.

The if statement then starts, if the string equals quit, meaning the user input quit then it runs the quiting code which writes the totalLetters to the file, closes the file, changes the Boolean quit value to true so the while loop ends, and calls the quit function to calculate and display the values.

The else in the if statement is meant to run when ever the user hasn't entered quit yet, and in this section I initialize a string called current letters which is meant to represent a single current ticket string, I then used a for loop and if statement to loop through the strings characters and checking that exist in the dictionary, if they do the index character is added to the current string, if its not then

the appropriate error message is displayed. Finally the single ticket string is added to the total string with a separation.

```
def quitFunction(filename):
    totalsetdict = {# dictionary of the city's and their counts
        'B': 0,
        'C': 0,
        'H': 0,
        'L': 0,
        'M': 0,
        'N': 0,
        'S': 0,
    }
    try:
        with open(filename, "r+") as file:# this opens and reads the file as read and write
            print("This is file: " + filename)
            totalLetters = file.read().split()# makes the txt file into a list
            totalLettersStr = ' '.join(totalLetters)# makes the list into a string
            for i in totalLetters:
                ticketset = set(SetOfTicketsSold(i)) # loops through the list and adds the strings to a set making f
                for j in ticketset:
        print(str(totalLetters))# below prints the calculated info to the screen calling previous functions
        print("Number of tickets sold: "+str(numberOfTicketsSold(totalLettersStr)))
        print("Number of hours in hours of each city: "+str(totalsetdict))
        print("The total price of sold tickets is: £" + str(ticketTablePrice(totalLettersStr, ticketTableDict)))
        file.close()# closes the file
            print("The total price of sold tickets is: £" + str(ticketTablePrice(totalLettersStr, ticketTableDict)))
            file.close()# closes the file
    except:
        print("file cannot open")
```

This is the quit function that is called with in the task4loopfinal function when the user enters quit. I have initialized a dictionary same as the one from task 3 it has the city characters as keys and their counts as values.

This is where the file that was written and saved from the previous function is opened as a read and write file. It displays the name of the file, and parses the text within the file into a list totalLetters. The next line then makes a string variable totallettersStr that is equal list from the previous line. Both variables are important. With try and except for added defensiveness.

A for loop is used to iterate through the elements of the list and make them into sets using the setofTicketsSold function I created to make a set of the characters in the string by simply looping through the string and adding them to the set then returning the set.

```
def SetOfTicketsSold(string):
    ticketset = set() # initic
    for i in string:
        ticketset.add(i) # add
    return ticketset
```

Finally, this last section is for printing the results to the screen. I used the print function and concatenated the strings of the results calculated using the functions from previous tasks to make the output easier to read.

iii. Software testing

For each task I tested each function using expected inputs and unexpected inputs to obtain appropriate expected outputs.

Task 1 Testing:

	input	Expected output	Actual output pass/fail
Expected input	BMMNSB	66.67	66.67
Expected input	MMMBB	100.0	100.0
Unexpected input	KNCE8	0.0	0.0
Unexpected input	kdmM 9Bb	50.0	50.0

Task 2 Testing:

	input	Expected output	Actual output pass/fail
Expected input	BMMNSB	£169.4	£169.4
Expected input	MMMBB	£172.4	£172.4
Unexpected input	KNCE8	Unrecognised city symbol detected K Unrecognised city symbol detected E Unrecognised city symbol detected 8 £36.4	pass
Unexpected input	kdmM 9Bb	Unrecognised city symbol detected K Unrecognised city symbol detected D Unrecognised city symbol detected 9 £134.4	pass

Task 3 Testing:

	input	Expected output	Actual output pass/fail
Expected input	BCCBBLNNLLBBMMMBCBC LLMSMBMBMBMNMNSSLSCS	{'B': 10, 'C': 5, 'H': 0, 'L': 6, 'M': 9, 'N': 4, 'S': 5}	pass
Expected input	MMMBB	{'B': 2, 'C': 0, 'H': 0, 'L': 0, 'M': 3, 'N': 0, 'S': 0}	pass
Unexpected input	KNCE8	{'B': 0, 'C': 1, 'H': 0, 'L': 0, 'M': 0, 'N': 1, 'S': 0}	pass
Unexpected input	kdmM 9Bb	{'B': 2, 'C': 0, 'H': 0, 'L': 0, 'M': 2, 'N': 0, 'S': 0}	pass

Task 4 Testing:

	input	Expected output This is file: 2023-10-17.txt Number of tickets sold hours of each city: The total price of sold tickets is:	Actual output pass/fail
Expected input	BBMNH MMMBB	['BBMNH', 'MMMBB'] {'B': 4, 'C': 0, 'H': 1, 'L': 0, 'M': 4, 'N': 1, 'S': 0} {'B': 2, 'C': 0, 'H': 1, 'L': 0, 'M': 2, 'N': 1, 'S': 0} £323.3	pass
Expected input	BBMNH BBMNH	['BBMNH', 'BBMNH'] {'B': 4, 'C': 0, 'H': 2, 'L': 0, 'M': 2, 'N': 2, 'S': 0} {'B': 2, 'C': 0, 'H': 2, 'L': 0, 'M': 2, 'N': 2, 'S': 0} £301.8	pass
Unexpected input	KNCE8	['NC'] {'B': 0, 'C': 1, 'H': 0, 'L': 0, 'M': 0, 'N': 1, 'S': 0} {'B': 0, 'C': 1, 'H': 0, 'L': 0, 'M': 0, 'N': 1, 'S': 0} £36.4	pass
Unexpected input	kdmM 9Bb	['MB'] : {'B': 1, 'C': 0, 'H': 0, 'L': 0, 'M': 1, 'N': 0, 'S': 0} {'B': 1, 'C': 0, 'H': 0, 'L': 0, 'M': 1, 'N': 0, 'S': 0} £67.2	pass

iv. Reflections and future work

Overall, I believe that I completed the tasks for the assignment extremely well, all of the functions for the task output the desired one given the inputs and I used clean and effective code to do so, so implement defensive measures when necessary to return error information back to the user or prevent case sensitivity errors. The only thing I believe I could do better is that I repeated the dictionary creation throughout the assignment to simply sections, this in future could be changed by only parsing one dictionary into each function, However the dictionary must be re-initialized each use. This assignment took me around 12 hours to complete.