Qn1. Program1:

#1. Write a program to count the number of lines, words, and characters in a text file.

#opening file in read mode

f= open("program1.txt", 'r')

#counting the length of lines

lines = len(f.readlines())

print('Total number of line:', lines)

#the readline moved the pointer to end of file

f.seek(0) #to move pointer at first

word\_count= 0

line = f.read() #reading the entire content and stores in variable data

words = line.split() #splits words at every whitespace

#iterating over every element of words

for word in words:

  word\_count+=1

print(f"The total number of word:{word\_count}")

#to print count of character

ch\_count = 0

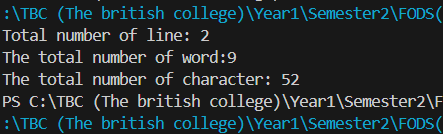
for ch in line: #iterate over each character

  ch\_count+=1

print(f"The total number of character: {ch\_count}")

f.close()

Output:



Qn2. Program2:

#2. Write a program to copy the contents of one file to another.

source= input("Enter the name of source file: ")

destination= input("Enter the name of destination file: ")

try:

  with open (source, 'r') as source\_file: #open file and read it as source file

    content = source\_file.read()

  with open (destination, 'w') as destination\_file: #writes the content in destination filr

    destination\_file.write(content)

  print ("You have copied the content of the file successfully")

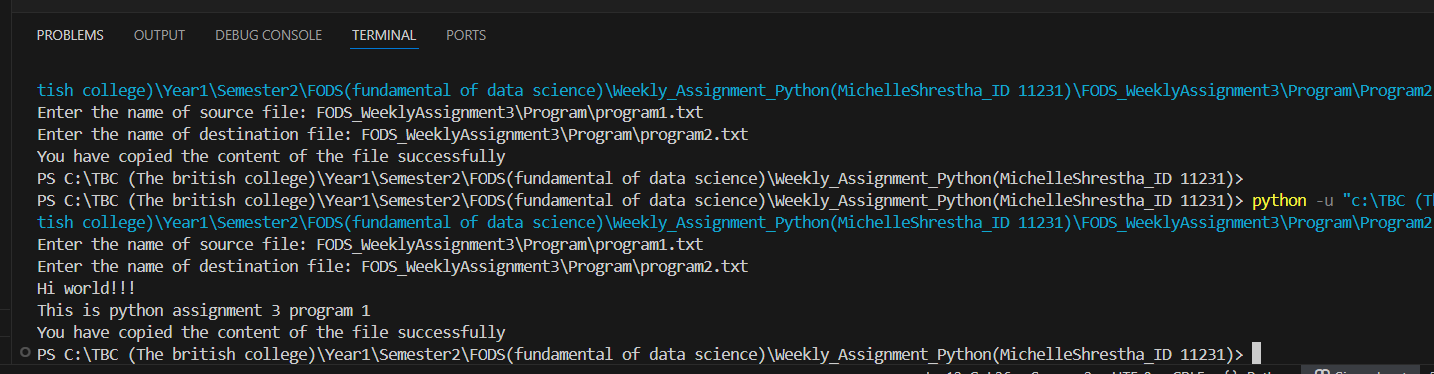
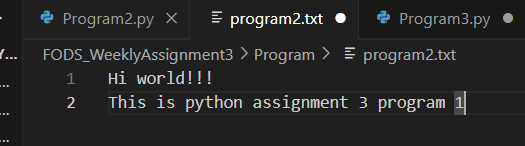
except FileNotFoundError: #if file not found it prints ...

  print("The file doesnot exist")

except Exception as e: #shows what are the other errors

  print(f"The problem is: {e}")

Output:



Qn3. Program3:

#3. Write a program to find and replace a specific word in a file with another word.

file\_name= input('Enter the name of your file: ')

old\_word= input('Enter the word that should be replaced: ')

new\_word= input('Enter the new word: ')

try:

  with open (file\_name, "r") as file:#opening file in read mode

    content = file.read()

  replaced\_word = content.replace(old\_word , new\_word)#replacing old word by new

  with open (file\_name,"w" ) as file: #opening file in write mode

    file.write(replaced\_word)

    print("The word replaced successfully")

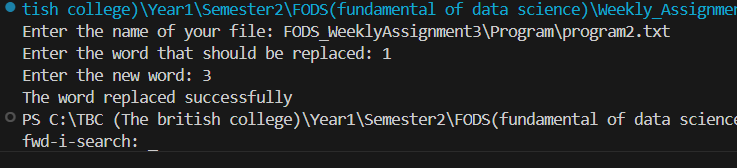
except FileNotFoundError:

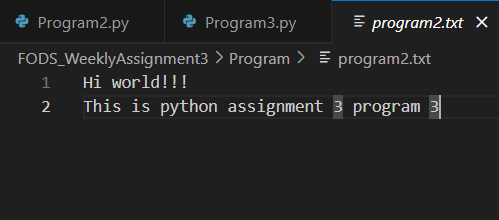
  print("The file doesnot exist")

except Exception as e:

  print (f"The problem is {e}")

Output:





Qn4. Program 4:

#4. Implement a program to read a CSV file and display its contents in a tabular format

import csv

file\_name = input('Enter the name of your file: ')

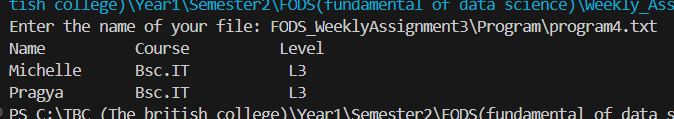
with open(file\_name,'r')as file: #opens the file in read mode

    readFile = csv.reader(file)  #reads the file and iterates over the row in the CSV file

    for row in readFile:

         #for tabular formatting

         print('\t'.join(row)) #joins the elements of row and separates them in tabs

Output:

Qn5. Program5:

#5. Develop a program that counts the occurrence of each word in a file

file\_name = input("Enter the name of your file: ")

try:

    # Open the file in read mode

    with open(file\_name, "r") as file:  # Opens the file in read mode

        content = file.read()

    content = content.lower()  # Converts the content to lowercase

    words = content.split()  # Splits the content into words

    word\_count = {}  #empty dictionary

    # Iterate over each word

    for word in words:

        if word not in word\_count:  # Checks if the word is not already in the dictionary

            word\_count[word] = 1  # Adds the word with an initial count of 1

        else:

            word\_count[word] += 1  # Increments the count for repeated words

    # Print the word counts

    print("\nWords and its occurrence:")

     # Each word ans its count iterates through the dictionary

    for word, count in word\_count.items():  #.items() returns key-value pairs as tuples

        fileWord = word

        occurrence = count

        print(f"{fileWord}: {occurrence}")

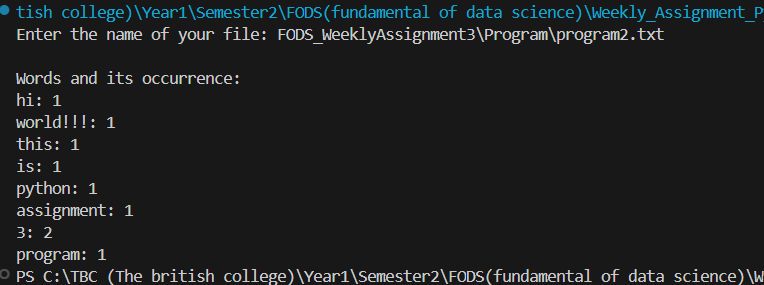
except FileNotFoundError:

    print("File does not exist.")

except Exception as e:

    print(f"The problem is {e}")

Output:



Qn6. Program6:

# 6. Create a class Student with the attributes such as id, name, address, admission year, level,

# section. Instantiate the object of class to take input for all the attributes and display the output. [10 marks]

class Student:

    def \_\_init\_\_(self):

        #initializing tbea attributes of the object

        self.student\_id= input("Enter student ID: ")

        self.student\_name= input("Enter the name of student: ")

        self.student\_address=input("Enter the address of student: ")

        self.student\_admission\_year=input("Enter the admission year of student: ")

        self.student\_level=input("Enter the level of student: ")

        self.student\_section=input("Enter the section of student: ")

    def display(self):

        # Prints all the student details

        print("\n ---Student details--- \n")

        print("Student ID:", self.student\_id)

        print("Student Name:", self.student\_name)

        print("Student Address:", self.student\_address)

        print("Student Admission Year:", self.student\_admission\_year)

        print("Student Level:", self.student\_level)

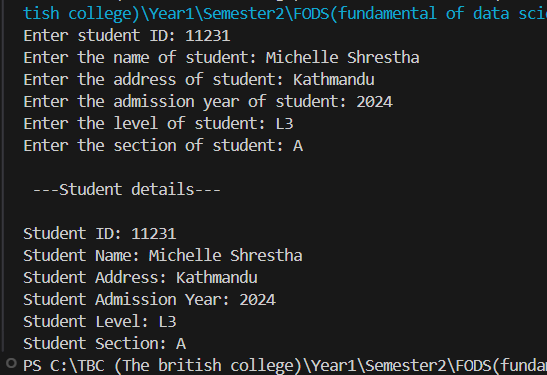
        print("Student Section:", self.student\_section)

student1= Student()# Creating an object of student class

#and calls the constructor (\_\_init\_\_) automatically

student1.display()

Output:



Qn7. Program7:

#7.Write a program to implement a class called employee with attributes such as

# empid, name, address, contact\_number, spouse name, number\_of\_child, salary.

# Instantiate this class to input the values for multiple employees and write it in a file “employees.csv”.

# Allow the user of your program to see the list of employees and their details as well.

# Try to use the concept of try/except too in the program.

import csv

class Employee:

    #initializing employee attributes

    def \_\_init\_\_(self, empid, name, address, contact\_number, spouse\_name, child\_count, salary):

        self.empid = empid

        self.name = name

        self.address = address

        self.contact\_number = contact\_number

        self.spouse\_name = spouse\_name

        self.child\_count = child\_count

        self.salary = salary

def employee\_data():#reuturns the instance of employee

    try:

        empid = int(input("Enter the Employee ID: "))

        name = input("Enter the Name: ")

        address = input("Enter the Address: ")

        contact\_number = int(input("Enter the Contact Number: "))

        spouse\_name = input("Enter the Spouse Name: ")

        child\_count = int(input("Enter the Number of Children: "))

        salary = float(input("Enter the Salary: "))

        return Employee(empid, name, address, contact\_number, spouse\_name, child\_count, salary)

    except Exception as e:

        print(f"The Error is {e}")

        return None

def write\_employee\_csv(employees, filename="employees.csv"): #writes the list of employee to csv file

    try:

        with open(filename, 'w', newline='') as file:#open file in write mode

            writer = csv.writer(file)

            #writing the row to the file

            writer.writerow(["Employee ID", "Name", "Address", "Contact Number", "Spouse Name", "Number of Children", "Salary"])

            for employ in employees:

                writer.writerow([employ.empid, employ.name, employ.address, employ.contact\_number, employ.spouse\_name, employ.child\_count, employ.salary])

        print("Employee details written successfully.")

    except Exception as e:

        print(f"The problem is {e}")

def display\_employess(employees):

    if not employees:

        print("There is no employee data to display.")

    else:

        for employ in employees:#iterate over the list and displays the details

            print(f"Employee ID: {employ.empid}, Name: {employ.name}, Address: {employ.address}, "

                  f"Contact: {employ.contact\_number}, Spouse: {employ.spouse\_name}, Children: {employ.child\_count}, Salary: {employ.salary}")

def main():

    employees = [] #initializing empty list

    while True:

        print("\n--- Employee Details ---\n")

        print("1. Add an employee")

        print("2. Display Employee Details")

        print("3. Save details to the CSV")

        print("4. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 1:

            employ = employee\_data()

            if employ: #adds employ if theres no error

                employees.append(employ)

        elif choice == 2:

            display\_employess(employees)

        elif choice == 3:

            write\_employee\_csv(employees)

        elif choice == 4:

            print("You are exiting the program...")

            break

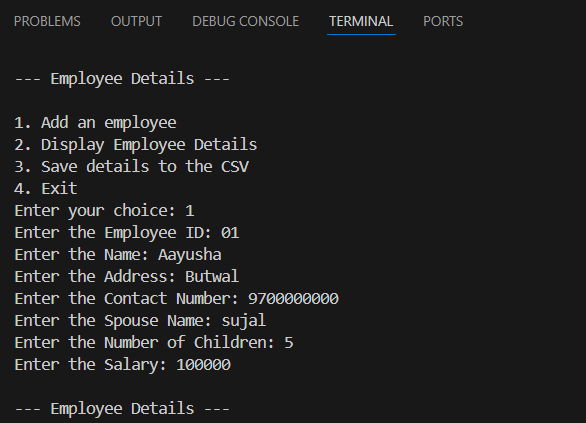
        else:

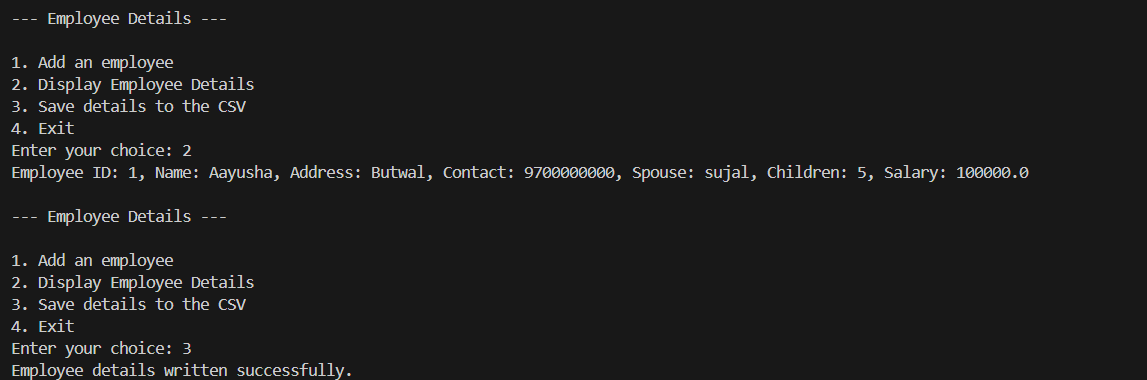
            print("Invalid choice. Please choose a valid number (1-4).")

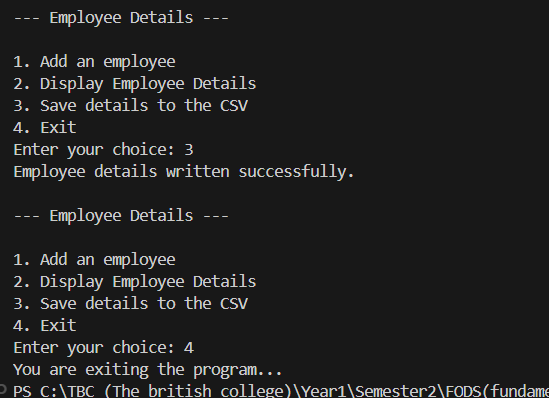
if \_\_name\_\_ == '\_\_main\_\_':

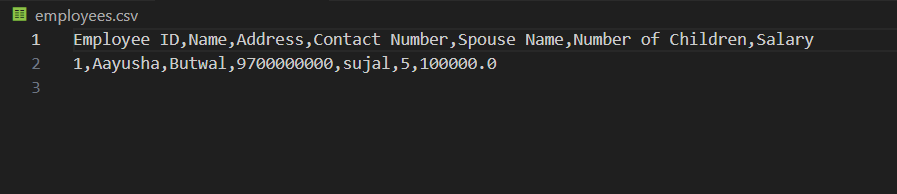
    main() #main function is executed when all the data are valid

Output:









Qn8. Program 8:

#8.Write a program to implement a basic library book management with the functionalities such as

# issue the book, return the book and search the book.

# Use the concept of OOP to create the necessary classes on your own and implement the concept of other OOP features.

# For the storage of book details, use the file handling along with the exception handling.

# Book class to represent individual book objects

import os

class Book:

    def \_\_init\_\_ (self, book\_id, title, author):

        self.book\_id = book\_id #books unique id

        self.title = title #book title

        self.author = author #author name

class Library:

    def \_\_init\_\_(self):

        self.books = [] #empty list to hold book objects

        self.book\_data() #loads the book data from the file

    def book\_data(self):

        try: #if file exists

            with open ("books.txt", "r") as file:#opening the books.txt in read mode

                for line in file:

                    book\_id, title, author = line.strip().split(",") #removes white space and splits by comma and makes list

                    #appends the class book info in empty list

                    self.books.append(Book(book\_id, title, author))#book class append

        except FileNotFoundError:

            print ("The file doesnot exist")

        except Exception as e:

            print (f"The error is {e}")

    def save\_data (self): #saves the book data in the file

        with open ("books.txt", "a+") as file: #opening the books.txt in write mode

            for book in self.books:

                file.write (f"{book.book\_id}, {book.title}, {book.author} \n")

    def display\_books(self): #displays all the book

        try:

            if  self.books: #checks in books list

                print ("Book is in the library")

                for book in self.books: #books available in library

                    print (f"ID: {book.book\_id}, Title: {book.title}, Author: {book.author}")

        except Exception:

            print ("Books are not available in the library")

    def issue\_book(self, book\_id):

        try:

            for book in self.books:

                if book.book\_id == book\_id:

                    self.books.remove(book) #if the book is issued it removes the book from books list

                    print (f"The book {book.title} {book.book\_id} has been issued")

                    self.save\_data() #saves the data(book issued) in save\_data

                    return

        except Exception:

            print ("Book not found")

    def return\_book (self, book\_id, title, author):

        self.books.append(Book(book\_id, title, author))# again adds the book after being returned

        print (f"The book {title} {book\_id} has been returned")

        self.save\_data()#saves the data

    #to search book by title

    def search\_book (self,title):

        for book in self.books:

            if book.title.lower() in book.title.lower():

                print (f"{book.title},{book.book\_id} has been found")

                return

        print ("Book not found!") #if not in books

def main():

    library = Library()  # Creating an instance of the Library class

    while True:

        print("\n--- Library Menu: --- \n")

        print("1. Display Books")

        print("2. Issue Book")

        print("3. Return Book")

        print("4. Search Book")

        print("5. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":

            library.display\_books()  # Displays all books

        elif choice == "2":

            book\_id = input("Enter the Book ID to issue: ")

            library.issue\_book(book\_id)  # Issue a book

        elif choice == "3":

            book\_id = input("Enter the Book ID to return: ")

            title = input("Enter the Title of the book: ")

            author = input("Enter the Author of the book: ")

            library.return\_book(book\_id, title, author)  # Return a book

        elif choice == "4":

            title = input("Enter the Title of the book to search: ")

            library.search\_book(title)  # Search for a book

        elif choice == "5":

            print("Exiting the program...")

            break  # Exit the loop

        else:

            print("Invalid choice. Please try again.")

# Run the main function when the script is executed

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Output:

