1. Write a python program using functions to take input from the user and calculate the factorial of the number.

**def factorial(*n*):**

**if *n*==0 or *n*==1:**

**return 1**

**else:**

**return *n*\*factorial(*n*-1)**

**try:**

**num=int(input("enter a number to find its factorial: - "))**

**if num<0:**

**print("please write a non negative number ")**

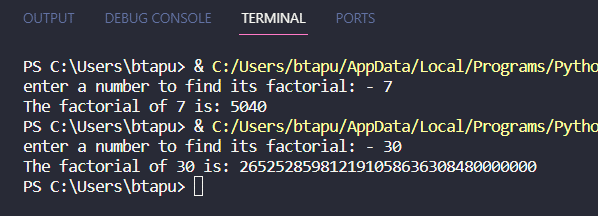
**else:**

**result=factorial(num)**

**print(f"The factorial of {num} is: {result}")**

**except ValueError:**

**print("invalid input.please enter a integer not a character")**



2. Write a python program to take numbers from the user and check if it is prime or not.

**p = int(input("Enter a number to check whether it's prime or composite: "))**

**if p < 2:**

**print("It's neither prime nor composite.")**

**elif p == 2:**

**print("It's a prime number.")**

**elif p % 2 == 0:**

**print("It's a composite number.")**

**else:**

**is\_prime = True**

**for i in range(3, int(p\*\*0.5) + 1, 2):**

**if p % i == 0:**

**is\_prime = False**

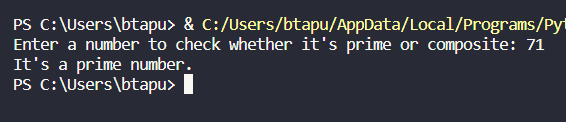
**break**

**if is\_prime:**

**print("It's a prime number.")**

**else:**

**print("It's a composite number.")**



3. Write a program to calculate the nth term of Fibonacci series.

**def fibonacci(*n*):**

**if *n* <= 0:**

**return "Invalid input. Please enter a positive integer."**

**fib\_sequence = [0, 1]**

**while len(fib\_sequence) <= *n*:**

**next\_term = fib\_sequence[-1] + fib\_sequence[-2]**

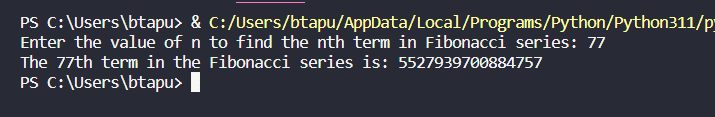
**fib\_sequence.append(next\_term)**

**return fib\_sequence[*n*]**

**user\_input = int(input("Enter the value of n to find the nth term in Fibonacci series: "))**

**result = fibonacci(user\_input)**

**print(f"The {user\_input}th term in the Fibonacci series is: {result}")**



4. Write a random number generator that generates random numbers between 1 and 6

**import random**

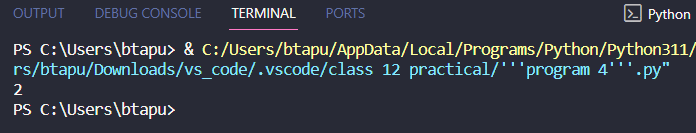
**def random\_no():**

**return print(random.randint(1,6))**

**while True:**

**random\_no()**

**break**



5. Write a Python program to demonstrate the concept of variable length argument to calculate the product and power of the ϐirst 10 numbers

**def calculate\_product\_and\_square(\**ar*):**

**product = 1**

**square = []**

**for num in *ar*:**

**product \*= num**

**square.append(num\*\*2)**

**return product, square**

**m=int(input("input no to the list: "))**

**n=int(input("input no to the list: "))**

**o=int(input("input no to the list: "))**

**p=int(input("input no to the list: "))**

**x=int(input("input no to the list: "))**

**y=int(input("input no to the list: "))**

**numbers = [m,n,o,p,x,y]**

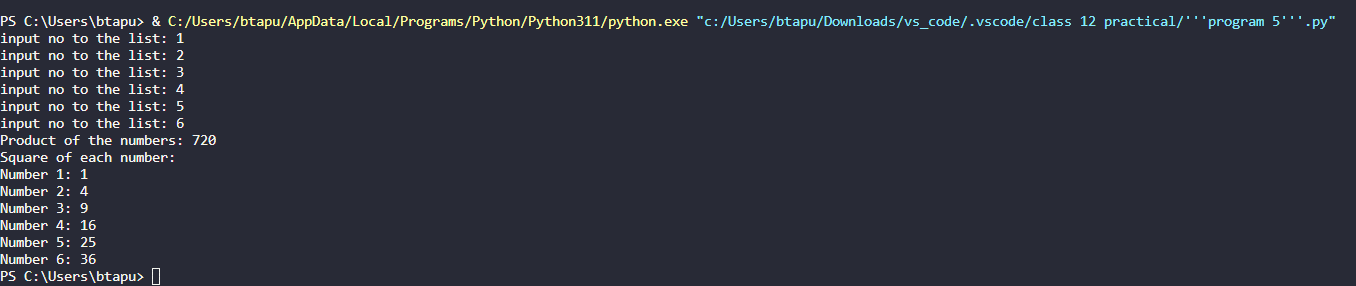
**result\_product, result\_squares = calculate\_product\_and\_square(\*numbers)**

**print(f"Product of the numbers: {result\_product}")**

**print("Square of each number:")**

**for i, square in enumerate(result\_squares):**

**print(f"Number {numbers[i]}: {square}")**



6. Write a program to accept the number of days and display appropriate weeks in an integer. Display an appropriate error message if input is not an integer. Handle the errors including the ϐinally clause.

**def days\_to\_weeks(*days*):**

**try:**

***days* = int(*days*)**

**weeks = *days* // 7**

**print(f"{*days*} days is approximately {weeks} weeks.")**

**except ValueError:**

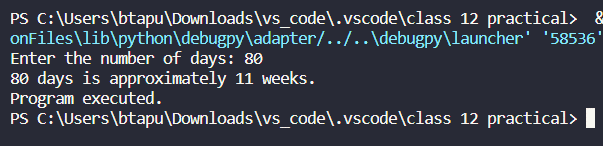
**print("Error: Input is not an integer.")**

**finally:**

**print("Program executed.")**

**days = input("Enter the number of days: ")**

**days\_to\_weeks(days)**



7. Write a python program to search any word in given string/sentence. User provides input string/sentence and the word to search.

**def search\_word\_in\_sentence(*sentence*, *search\_word*):**

**words = *sentence*.split()**

**if *search\_word* in words:**

**return f"The word '{*search\_word*}' is found in the sentence."**

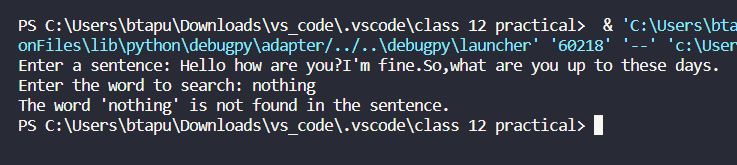
**else:**

**return f"The word '{*search\_word*}' is not found in the sentence."**

**user\_sentence = input("Enter a sentence: ")**

**user\_search\_word = input("Enter the word to search: ")**

**print(search\_word\_in\_sentence(user\_sentence, user\_search\_word))**



8. Write a python program to read and display ϐile content line by line with each word separated by ‘\*’

**def display(*file\_path*):**

**try:**

**with open("{0}.txt".format(*file\_path*), "r") as file:**

**for line in file:**

**words = line.strip().split()**

**formatted\_line = '\*'.join(words)**

**print(formatted\_line)**

**except FileNotFoundError:**

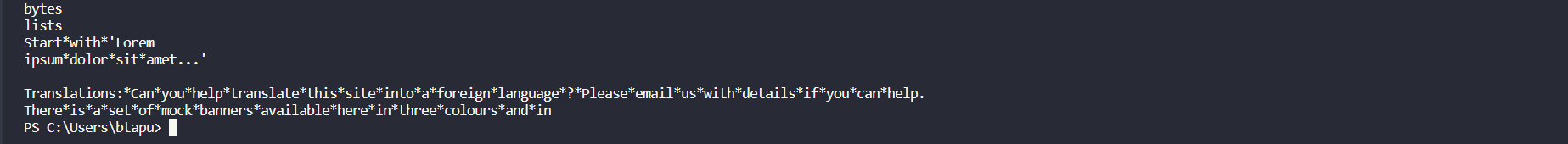
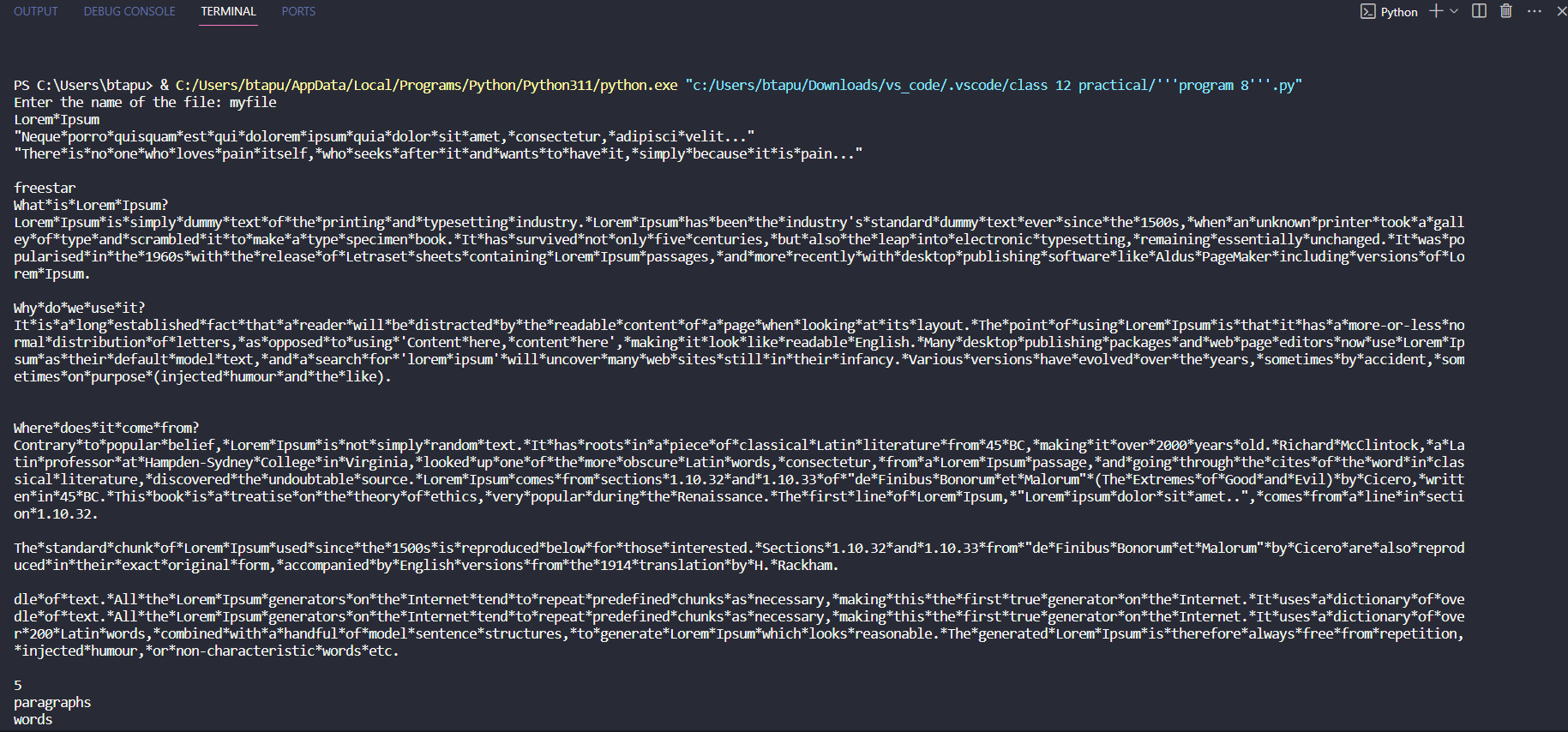
**print(f"Error: File '{0}' not found.")**

**except Exception as e:**

**print(f"An error occurred: {e}")**

**file\_PATH = input("Enter the name of the file: ")**

**display(file\_PATH)**



9. Write a python program to read the content of the ϐile and display total numbers of upper case and lower case characters

**def countul(*file\_path*):**

**try:**

**with open("{0}.txt".format(*file\_path*), 'r') as file:**

**content = file.read()**

**upper\_count = 0**

**lower\_count = 0**

**for char in content:**

**if char.isupper():**

**upper\_count += 1**

**elif char.islower():**

**lower\_count += 1**

**print(f"Total uppercase characters: {upper\_count}")**

**print(f"Total lowercase characters: {lower\_count}")**

**except FileNotFoundError:**

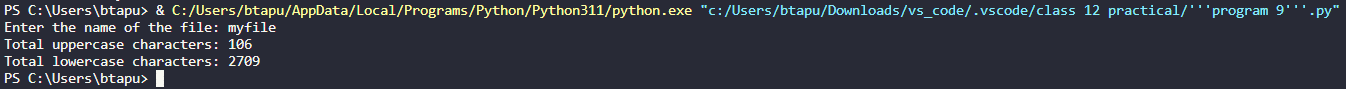
**print(f"Error: File '{*file\_path*}' not found.")**

**except Exception as e:**

**print(f"An error occurred: {e}")**

**file\_route = input("Enter the name of the file: ")**

**countul(file\_route)**



10.Write a python program to create binary ϐile to store Employee Name, Employee code and take input from user to display name of the employee if employee code is found in the ϐile otherwise display “Employee not found

**import pickle**

**def write\_employee\_data(*file\_data*, *employee\_data*):**

**with open(*file\_data*, 'wb') as file:**

**pickle.dump(*employee\_data*, file)**

**def find\_employee\_by\_code(*file\_data*, *search\_code*):**

**try:**

**with open(*file\_data*, 'rb') as file:**

**employee\_data = pickle.load(file)**

**for employee in employee\_data:**

**if employee['EmployeeCode'] == *search\_code*:**

**return employee['EmployeeName']**

**return None**

**except FileNotFoundError:**

**print(f"Error: File '{*file\_data*}' not found.")**

**except Exception as e:**

**print(f"An error occurred: {e}")**

**employee\_data = [**

**{'EmployeeCode': 101, 'EmployeeName': "Supratik"},**

**{'EmployeeCode': 102, 'EmployeeName': "Debojit"},**

**{'EmployeeCode': 103, 'EmployeeName': "Tapubrat"},**

**{'EmployeeCode': 104, 'EmployeeName': "Ronald"},**

**{'EmployeeCode': 105, 'EmployeeName': "Puneet Babua"}**

**]**

**file\_route = "employee\_data.pkl"**

**write\_employee\_data(file\_route, employee\_data)**

**search = int(input("Enter the employee code to search: "))**

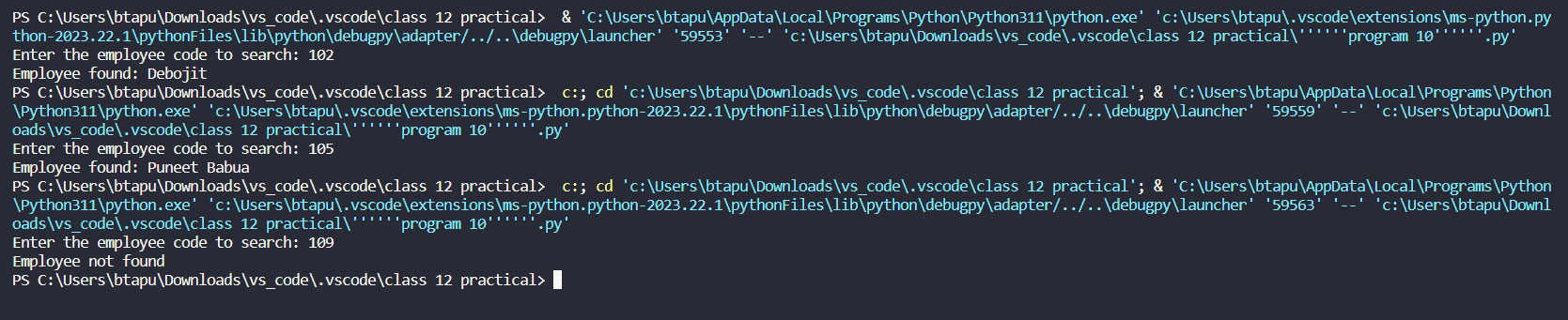
**result = find\_employee\_by\_code(file\_route, search)**

**if result:**

**print(f"Employee found: {result}")**

**else:**

**print("Employee not found")**



11.Write a python program to create a binary ϐile to store RollNo, Name and marks of the student. Take input from the user to update the marks of the entered Roll no.

**import pickle**

**def write\_student\_data(*file\_data*, *student\_data*):**

**with open(*file\_data*, 'wb') as file:**

**pickle.dump(*student\_data*, file)**

**def update\_marks(*file\_data*, *roll\_no*, *new\_marks*):**

**try:**

**with open(*file\_data*, 'rb') as file:**

**student\_data = pickle.load(file)**

**for student in student\_data:**

**if student['RollNo'] == *roll\_no*:**

**student['Marks'] = *new\_marks***

**with open(*file\_data*, 'wb') as file:**

**pickle.dump(student\_data, file)**

**except FileNotFoundError:**

**print(f"Error: File '{*file\_data*}' not found.")**

**except Exception as e:**

**print(f"An error occurred: {e}")**

**student\_data = [**

**{'RollNo': 1, 'Name': "Superman", 'Marks': 85},**

**{'RollNo': 2, 'Name': "Batman", 'Marks': 90},**

**{'RollNo': 3, 'Name': "The Flash", 'Marks': 78}**

**]**

**file\_route = "student\_data.pkl"**

**write\_student\_data(file\_route, student\_data)**

**roll\_no\_to\_update = int(input("Enter the RollNo to update marks: "))**

**new\_marks = float(input("Enter the new marks: "))**

**update\_marks(file\_route, roll\_no\_to\_update, new\_marks)**

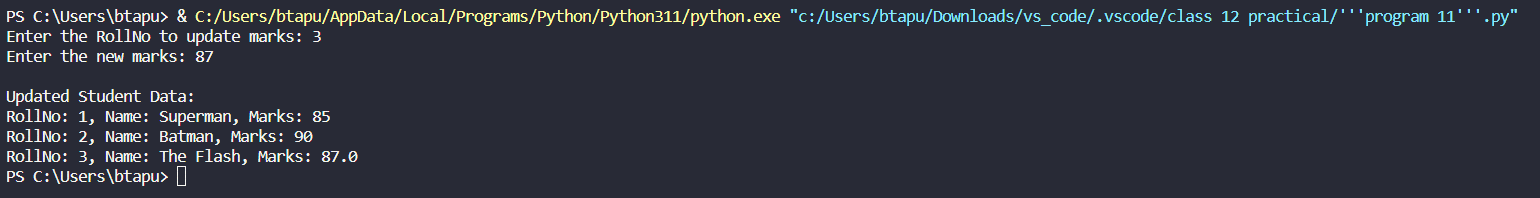
**with open(file\_route, 'rb') as file:**

**updated\_student\_data = pickle.load(file)**

**print("\nUpdated Student Data:")**

**for student in updated\_student\_data:**

**print(f"RollNo: {student['RollNo']}, Name: {student['Name']}, Marks: {student['Marks']}")**



12.Write a python program to read the content of the ϐile line by line and write it to the another ϐile except for the lines that contain ‘o’ letter in it

**input\_file\_name = 'input\_file.txt'**

**output\_file\_name = 'output\_file.txt'**

**try:**

**with open(input\_file\_name, 'r') as input\_file, open(output\_file\_name, 'w') as output\_file:**

**for line in input\_file:**

**if 'o' not in line.lower():**

**output\_file.write(line)**

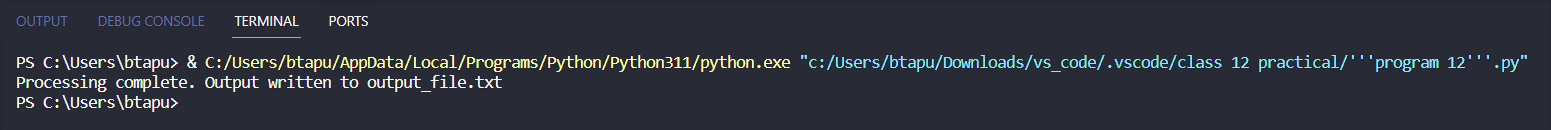
**print("Processing complete. Output written to", output\_file\_name)**

**except FileNotFoundError:**

**print(f"Error: The file '{input\_file\_name}' does not exist.")**

**except Exception as e:**

**print("An error occurred:", str(e))**



13.Create a binary ϐile “Employee.dat” with emp id, name and salary. Write a function Update\_Sal(empid), receive an employee id and update the salary.

**import pickle**

**def write\_employee\_data(*employee\_data*, *filename*):**

**with open(*filename*, 'wb') as file:**

**pickle.dump(*employee\_data*, file)**

**def read\_employee\_data(*filename*):**

**try:**

**with open(*filename*, 'rb') as file:**

**return pickle.load(file)**

**except FileNotFoundError:**

**return {}**

**def update\_salary(*emp\_id*, *new\_salary*, *employee\_data*):**

**if *emp\_id* in *employee\_data*:**

***employee\_data*[*emp\_id*]['salary'] = *new\_salary***

**print(f"Salary for Employee ID {*emp\_id*} updated successfully.")**

**else:**

**print(f"Employee ID {*emp\_id*} not found.")**

**employee\_data = {**

**1: {'name': 'Bhupendra Jogi', 'salary': 500000},**

**2: {'name': 'Ayeein Baigaan', 'salary': 60000},**

**3: {'name': 'Andrew Tate', 'salary': 100000000000000},**

**}**

**file\_name = 'Employee.dat'**

**write\_employee\_data(employee\_data, file\_name)**

**def update\_salary\_function():**

**try:**

**emp\_id = int(input("Enter Employee ID to update salary: "))**

**new\_salary = float(input("Enter the new salary: "))**

**update\_salary(emp\_id, new\_salary, employee\_data)**

**write\_employee\_data(employee\_data, file\_name)**

**except ValueError:**

**print("Invalid input. Please enter a valid employee ID and salary.")**

**update\_salary\_function()**



14.Write a program to know the cursor position and print the text according to the belowgiven speciϐications: 1. Print the initial position 2. Move the cursor to 4th position 3. Display next 5 characters 4. Move cursor to next 10 characters

**with open("myfile.txt","r") as fh:**

**fh.read()**

**initial\_position=fh.tell()**

**print(f"the initial position of the pointer is: {initial\_position}")**

**fh.seek(3)**

**position\_now=fh.tell()**

**print(f"the pointer after the cursor is moved to 4th position: {position\_now}")**

**text=fh.read(5)**

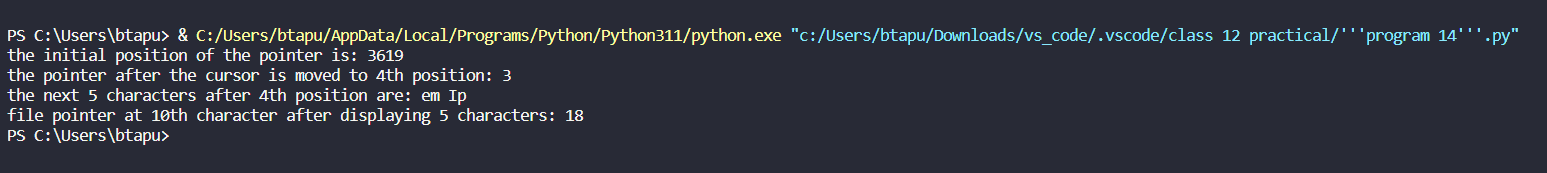
**print(f"the next 5 characters after 4th position are: {text}")**

**tell=fh.tell()**

**fh.seek(tell+10,0)**

**final=fh.tell()**

**print(f"file pointer at 10th character after displaying 5 characters: {final}")**



15.Write a program to implement stack in python using list.

**class Stack:**

**def \_\_init\_\_(*self*):**

***self*.stack = []**

**def push(*self*, *item*):**

***self*.stack.append(*item*)**

**print(f"Pushed: {*item*}")**

**def pop(*self*):**

**if not *self*.is\_empty():**

**item = *self*.stack.pop()**

**print(f"Popped: {item}")**

**return item**

**else:**

**print("Stack is empty. Cannot pop.")**

**def is\_empty(*self*):**

**return len(*self*.stack) == 0**

**def display(*self*):**

**if not *self*.is\_empty():**

**print("Stack contents:")**

**for item in reversed(*self*.stack):**

**print(item)**

**else:**

**print("Stack is empty.")**

**my\_stack = Stack()**

**while True:**

**print("\nStack Operations:")**

**print("1. Push")**

**print("2. Pop")**

**print("3. Display")**

**print("4. Quit")**

**choice = input("Enter your choice (1/2/3/4): ")**

**if choice == '1':**

**item = input("Enter the item to push: ")**

**my\_stack.push(item)**

**elif choice == '2':**

**my\_stack.pop()**

**elif choice == '3':**

**my\_stack.display()**

**elif choice == '4':**

**print("Exiting the program.")**

**break**

**else:**

**print("Invalid choice. Please enter 1, 2, 3, or 4.")**

