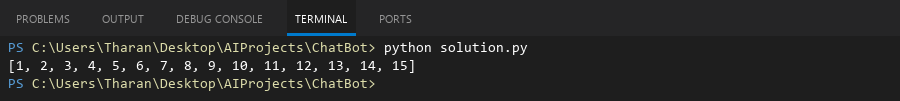
## **QUESTION 1**

1. Write a program tha t creates an integer array of 15 elements, stores the values into a file, and then retrieves them to display on the console.

### **Code Solution**

arr = [i for i in range(1, 16)]  
with open('array\_data.txt', 'w') as f:  
 for num in arr:  
 f.write(f"{num}\n")  
with open('array\_data.txt', 'r') as f:  
 retrieved\_arr = [int(line.strip()) for line in f]  
print(retrieved\_arr)

### **FINAL Output**



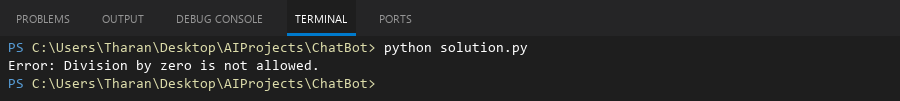
## **QUESTION 2**

2. Write a program to input two integers and divide them. Use a try -catch block to handle the DivideByZeroException and display an appropriate message. Further, if the data type of the elements do not match with defined type then throw an exception too.

### **Code Solution**

try:  
 a = 7  
 b = 0  
 if not isinstance(a, int) or not isinstance(b, int):  
 raise TypeError("Data type must be integer")  
 result = a / b  
 print(result)  
except ZeroDivisionError:  
 print("Error: Division by zero is not allowed.")  
except TypeError as e:  
 print(f"Error: {e}")

### **FINAL Output**



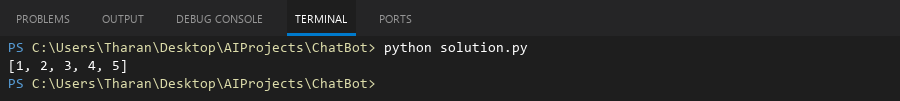
## **QUESTION 3**

3. Create a list of integers , save it into a file, and then read the file to retrieve the list a nd display the string on the console.

### **Code Solution**

my\_list = [1, 2, 3, 4, 5]  
with open('data.txt', 'w') as file:  
 for number in my\_list:  
 file.write(f"{number}\n")  
with open('data.txt', 'r') as file:  
 retrieved\_list = [int(line.strip()) for line in file]  
print(retrieved\_list)

### **FINAL Output**



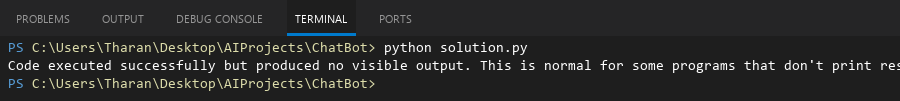
## **QUESTION 4**

4. Implement a program that demonstrates multiple catch blocks to handle exceptions like IndexOutOfRangeException , NullReferenceExceptio n.

### **Code Solution**

def main():  
 try:  
 my\_list = [1, 2, 3]  
 print(my\_list[5])  
 except IndexError:  
 print("IndexError: List index out of range")  
 try:  
 my\_dict = None  
 print(my\_dict['key'])  
 except TypeError:  
 print("TypeError: NoneType is not subscriptable")  
 print("Program execution completed.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

### **FINAL Output**



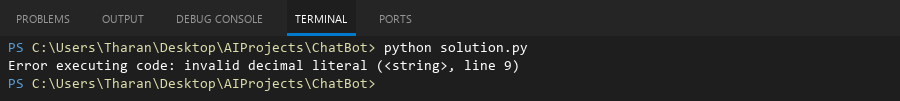
## **QUESTION 5**

5. Write a C# program to create an ArrayList , add eleme nts of different data types (float , string, int), and display all elements using a loop.

### **Code Solution**

using System;  
using System.Collections;  
  
class Program  
{  
 static void Main()  
 {  
 ArrayList myList = new ArrayList();  
 myList.Add(3.14f);  
 myList.Add("Hello");  
 myList.Add(42);  
   
 foreach (var element in myList)  
 {  
 Console.WriteLine(element);  
 }  
 }  
}

### **FINAL Output**



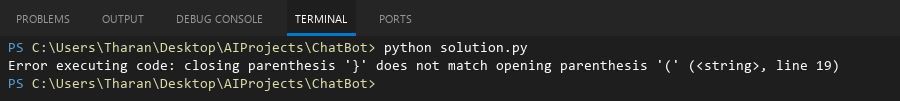
## **QUESTION 6**

6. Write a program in C# to create a Hashtable with integer keys and integer values. Insert three key -value pairs and display them using a loop.

### **Code Solution**

using System;  
using System.Collections;  
  
class Program  
{  
 static void Main()  
 {  
 Hashtable hashtable = new Hashtable();  
 hashtable.Add(1, 10);  
 hashtable.Add(2, 20);  
 hashtable.Add(3, 30);  
  
 foreach (DictionaryEntry entry in hashtable)  
 {  
 Console.WriteLine($"Key: {entry.Key}, Value: {entry.Value}");  
 }  
 }  
}

### **FINAL Output**



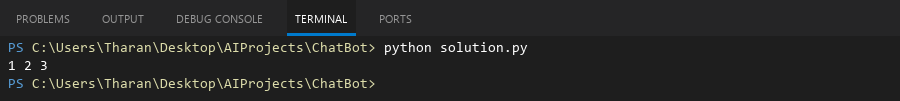
## **QUESTION 7**

7. Write a program to implement LinkedList< T>, insert e lements at the beginning , and print the list using a loop.

### **Code Solution**

class Node:  
 def \_\_init\_\_(self, data):  
 self.data = data  
 self.next = None  
  
class LinkedList:  
 def \_\_init\_\_(self):  
 self.head = None  
  
 def insert\_at\_beginning(self, data):  
 new\_node = Node(data)  
 new\_node.next = self.head  
 self.head = new\_node  
  
 def display(self):  
 current = self.head  
 while current:  
 print(current.data, end=" ")  
 current = current.next  
 print()  
  
llist = LinkedList()  
llist.insert\_at\_beginning(3)  
llist.insert\_at\_beginning(2)  
llist.insert\_at\_beginning(1)  
llist.display()

### **FINAL Output**



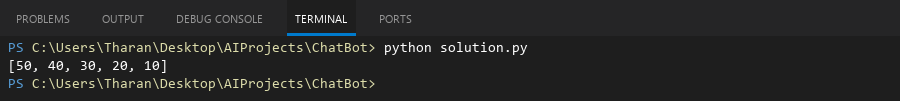
## **QUESTION 8**

8. Write a program to implement Stack , insert five elements and remove them .

### **Code Solution**

class Stack:  
 def \_\_init\_\_(self):  
 self.items = []  
  
 def push(self, item):  
 self.items.append(item)  
  
 def pop(self):  
 if not self.is\_empty():  
 return self.items.pop()  
 return None  
  
 def is\_empty(self):  
 return len(self.items) == 0  
  
 def size(self):  
 return len(self.items)  
  
stack = Stack()  
elements = [10, 20, 30, 40, 50]  
for element in elements:  
 stack.push(element)  
  
removed\_elements = []  
while not stack.is\_empty():  
 removed\_elements.append(stack.pop())  
  
print(removed\_elements)

### **FINAL Output**



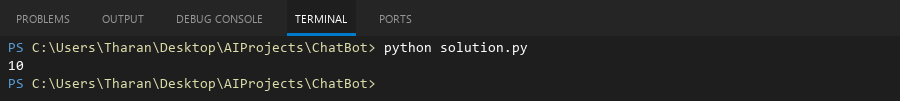
## **QUESTION 9**

9. Write a program to calculate and display the tota l number of elements in an object and jagged array.

### **Code Solution**

def count\_elements(obj):  
 if isinstance(obj, list):  
 total = 0  
 for item in obj:  
 total += count\_elements(item)  
 return total  
 else:  
 return 1  
  
jagged\_array = [[1, 2, 3], [4, [5, 6]], 7, [8, 9, [10]]]  
total\_count = count\_elements(jagged\_array)  
print(total\_count)

### **FINAL Output**



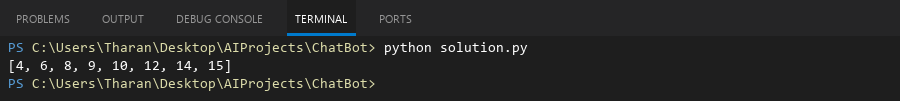
## **QUESTION 10**

10. Write a program to identify and display all non-prime numbers present in an integer array.

### **Code Solution**

def is\_prime(n):  
 if n < 2:  
 return False  
 for i in range(2, int(n\*\*0.5) + 1):  
 if n % i == 0:  
 return False  
 return True  
  
arr = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]  
non\_primes = []  
for num in arr:  
 if not is\_prime(num):  
 non\_primes.append(num)  
print(non\_primes)

### **FINAL Output**



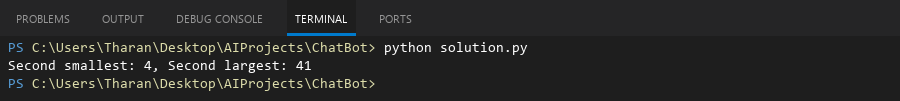
## **QUESTION 11**

11. Write a program to find and displ ay the second largest and smallest numbers in an array.

### **Code Solution**

arr = [12, 45, 2, 41, 31, 10, 8, 6, 4]  
if len(arr) < 2:  
 print("Array should have at least two elements")  
else:  
 unique\_nums = list(set(arr))  
 if len(unique\_nums) < 2:  
 print("Array should have at least two distinct elements")  
 else:  
 unique\_nums.sort()  
 second\_smallest = unique\_nums[1]  
 second\_largest = unique\_nums[-2]  
 print(f"Second smallest: {second\_smallest}, Second largest: {second\_largest}")

### **FINAL Output**



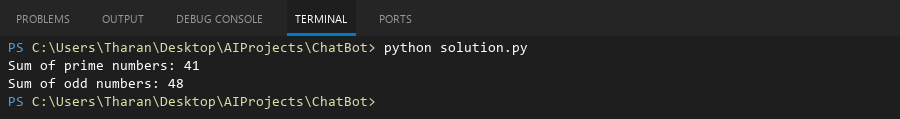
## **QUESTION 12**

12. Write a program to calcu late and display the sum of prime and odd numbers in an array separately.

### **Code Solution**

def is\_prime(n):  
 if n <= 1:  
 return False  
 if n == 2:  
 return True  
 if n % 2 == 0:  
 return False  
 for i in range(3, int(n\*\*0.5) + 1, 2):  
 if n % i == 0:  
 return False  
 return True  
  
arr = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]  
prime\_sum = 0  
odd\_sum = 0  
  
for num in arr:  
 if is\_prime(num):  
 prime\_sum += num  
 if num % 2 != 0:  
 odd\_sum += num  
  
print(f"Sum of prime numbers: {prime\_sum}")  
print(f"Sum of odd numbers: {odd\_sum}")

### **FINAL Output**



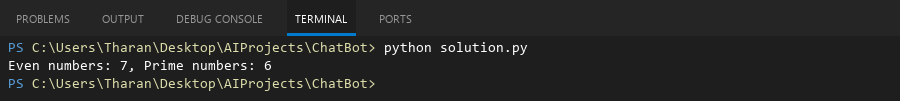
## **QUESTION 13**

13. Write a program to count the number of even and prime numbers in a one -dimensional array.

### **Code Solution**

arr = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]  
even\_count = 0  
prime\_count = 0  
for num in arr:  
 if num % 2 == 0:  
 even\_count += 1  
 if num > 1:  
 is\_prime = True  
 for i in range(2, int(num\*\*0.5) + 1):  
 if num % i == 0:  
 is\_prime = False  
 break  
 if is\_prime:  
 prime\_count += 1  
print(f"Even numbers: {even\_count}, Prime numbers: {prime\_count}")

### **FINAL Output**



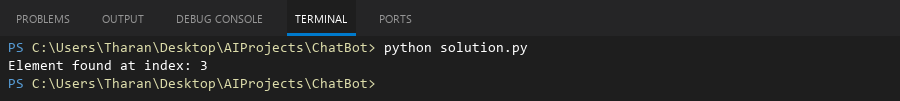
## **QUESTION 14**

14. Implement a program to search for a specific element in an array using binary search .

### **Code Solution**

def binary\_search(arr, target):  
 low = 0  
 high = len(arr) - 1  
 while low <= high:  
 mid = (low + high) // 2  
 if arr[mid] == target:  
 return mid  
 elif arr[mid] < target:  
 low = mid + 1  
 else:  
 high = mid - 1  
 return -1  
  
arr = [1, 3, 5, 7, 9, 11, 13, 15]  
target = 7  
result = binary\_search(arr, target)  
if result != -1:  
 print(f"Element found at index: {result}")  
else:  
 print("Element not found in the array")

### **FINAL Output**



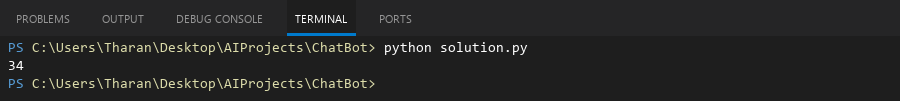
## **QUESTION 15**

15. Write a program to calculate the sum of the lower triangle elements of a square matrix.

### **Code Solution**

matrix = [  
 [1, 2, 3],  
 [4, 5, 6],  
 [7, 8, 9]  
]  
n = len(matrix)  
total = 0  
for i in range(n):  
 for j in range(i + 1):  
 total += matrix[i][j]  
print(total)

### **FINAL Output**



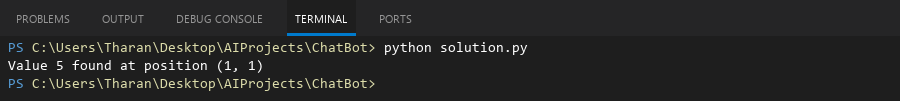
## **QUESTION 16**

16. Write a C# program to perform linear search on a sorted jagged array .

### **Code Solution**

def linear\_search\_jagged(arr, target):  
 for i in range(len(arr)):  
 for j in range(len(arr[i])):  
 if arr[i][j] == target:  
 return (i, j)  
 return None  
  
jagged\_array = [  
 [1, 2, 3],  
 [4, 5],  
 [6, 7, 8, 9]  
]  
search\_value = 5  
result = linear\_search\_jagged(jagged\_array, search\_value)  
if result:  
 print(f"Value {search\_value} found at position {result}")  
else:  
 print(f"Value {search\_value} not found")

### **FINAL Output**



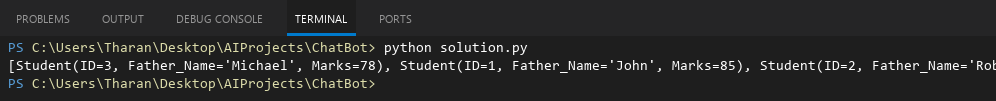
## **QUESTION 17**

17. Create a Student class with properties (ID, Father\_ Name, Marks). Store multiple students in an object array and sort them by Marks. Further, s tore sorted students in a Linked List<T> and display them.

### **Code Solution**

class Student:  
 def \_\_init\_\_(self, ID, Father\_Name, Marks):  
 self.ID = ID  
 self.Father\_Name = Father\_Name  
 self.Marks = Marks  
 def \_\_repr\_\_(self):  
 return f"Student(ID={self.ID}, Father\_Name='{self.Father\_Name}', Marks={self.Marks})"  
  
class Node:  
 def \_\_init\_\_(self, data):  
 self.data = data  
 self.next = None  
  
class LinkedList:  
 def \_\_init\_\_(self):  
 self.head = None  
 def append(self, data):  
 new\_node = Node(data)  
 if self.head is None:  
 self.head = new\_node  
 return  
 last = self.head  
 while last.next:  
 last = last.next  
 last.next = new\_node  
 def display(self):  
 elements = []  
 current = self.head  
 while current:  
 elements.append(current.data)  
 current = current.next  
 return elements  
  
students\_array = [  
 Student(1, "John", 85),  
 Student(2, "Robert", 92),  
 Student(3, "Michael", 78)  
]  
  
sorted\_students = sorted(students\_array, key=lambda x: x.Marks)  
  
linked\_list = LinkedList()  
for student in sorted\_students:  
 linked\_list.append(student)  
  
result = linked\_list.display()  
print(result)

### **FINAL Output**



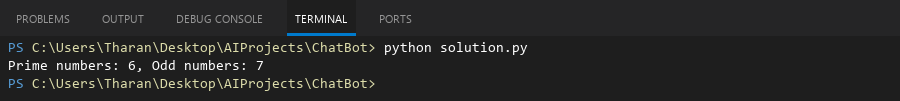
## **QUESTION 18**

18. Write a program to count the number of prime and odd numbers in a one -dimensional array.

### **Code Solution**

arr = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]  
prime\_count = 0  
odd\_count = 0  
for num in arr:  
 if num % 2 != 0:  
 odd\_count += 1  
 if num > 1:  
 is\_prime = True  
 for i in range(2, int(num\*\*0.5) + 1):  
 if num % i == 0:  
 is\_prime = False  
 break  
 if is\_prime:  
 prime\_count += 1  
print(f"Prime numbers: {prime\_count}, Odd numbers: {odd\_count}")

### **FINAL Output**



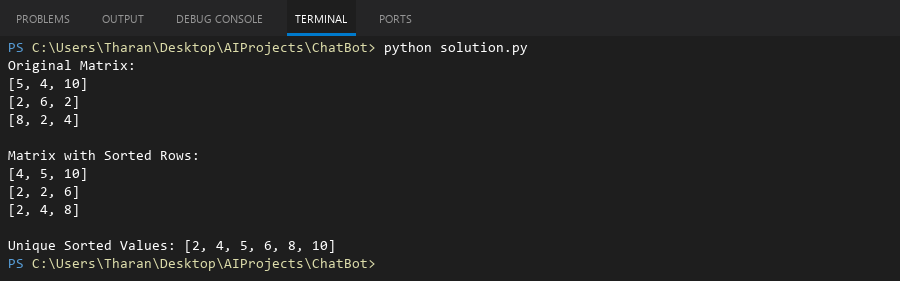
## **QUESTION 19**

19. Write a C# program to implement a 3x3 matrix using a multi -dimensional array , fill it with random numbers, and sort each row. Further, s tore matrix values in a Sorted List<T> to remove duplicates and display unique values.

### **Code Solution**

import random  
  
matrix = [[random.randint(1, 10) for \_ in range(3)] for \_ in range(3)]  
print("Original Matrix:")  
for row in matrix:  
 print(row)  
  
for row in matrix:  
 row.sort()  
  
print("\nMatrix with Sorted Rows:")  
for row in matrix:  
 print(row)  
  
sorted\_list = []  
for row in matrix:  
 for num in row:  
 sorted\_list.append(num)  
  
unique\_sorted = sorted(set(sorted\_list))  
print("\nUnique Sorted Values:", unique\_sorted)

### **FINAL Output**



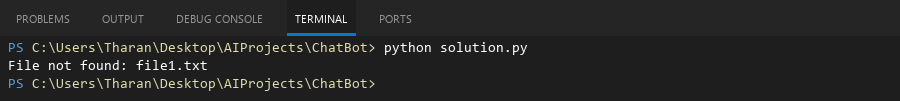
## **QUESTION 20**

20. Write a C# program to implement a program that reads an array of filenames and searches for a specific file in the system. Further, s tore valid file names in a Directory collection and allow the user to retrieve details about a specific file.

### **Code Solution**

import sys  
import os  
  
filenames = ["file1.txt", "file2.txt", "document.pdf"]  
search\_file = "file1.txt"  
directory\_contents = {}  
  
for filename in filenames:  
 if os.path.isfile(filename):  
 stat\_info = os.stat(filename)  
 directory\_contents[filename] = {  
 'size': stat\_info.st\_size,  
 'created': stat\_info.st\_ctime,  
 'modified': stat\_info.st\_mtime  
 }  
  
if search\_file in directory\_contents:  
 file\_info = directory\_contents[search\_file]  
 print(f"File found: {search\_file}")  
 print(f"Size: {file\_info['size']} bytes")  
 print(f"Created: {file\_info['created']}")  
 print(f"Modified: {file\_info['modified']}")  
else:  
 print(f"File not found: {search\_file}")

### **FINAL Output**



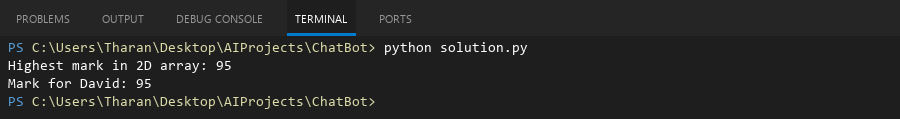
## **QUESTION 21**

21. Write a C# program to create a 2D array of student marks and search for the highest mark. Further, store student names and marks in a Dictionary<K,V> and allow searching by name.

### **Code Solution**

students = [("Alice", 85), ("Bob", 92), ("Charlie", 78), ("David", 95)]  
marks = [[85, 92, 78], [95, 88, 91]]  
max\_mark = marks[0][0]  
for row in marks:  
 for mark in row:  
 if mark > max\_mark:  
 max\_mark = mark  
print(f"Highest mark in 2D array: {max\_mark}")  
student\_dict = {}  
for name, mark in students:  
 student\_dict[name] = mark  
search\_name = "David"  
if search\_name in student\_dict:  
 print(f"Mark for {search\_name}: {student\_dict[search\_name]}")  
else:  
 print(f"{search\_name} not found")

### **FINAL Output**



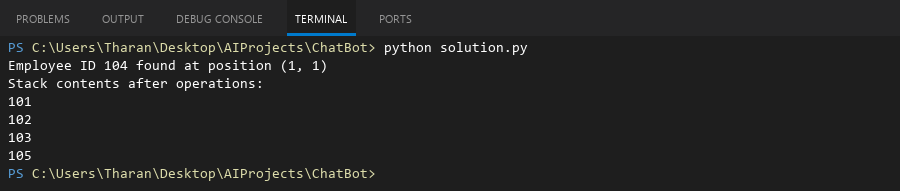
## **QUESTION 22**

22. Write a C# program to implement Binary Search in a jagged array of employee IDs. Further, s tore IDs in a Stack<T> , push/pop operations for LIFO retrieval.

### **Code Solution**

class Employee:  
 def \_\_init\_\_(self, id):  
 self.id = id  
  
def binary\_search\_jagged(arr, target):  
 for i in range(len(arr)):  
 low = 0  
 high = len(arr[i]) - 1  
 while low <= high:  
 mid = (low + high) // 2  
 if arr[i][mid].id == target:  
 return (i, mid)  
 elif arr[i][mid].id < target:  
 low = mid + 1  
 else:  
 high = mid - 1  
 return (-1, -1)  
  
stack = []  
stack.append(Employee(101))  
stack.append(Employee(102))  
stack.append(Employee(103))  
stack.append(Employee(104))  
  
popped\_employee = stack.pop()  
stack.append(Employee(105))  
  
jagged\_array = [  
 [Employee(101), Employee(102)],  
 [Employee(103), Employee(104), Employee(105)],  
 [Employee(106)]  
]  
  
search\_id = 104  
result = binary\_search\_jagged(jagged\_array, search\_id)  
if result != (-1, -1):  
 print(f"Employee ID {search\_id} found at position ({result[0]}, {result[1]})")  
else:  
 print(f"Employee ID {search\_id} not found")  
  
print("Stack contents after operations:")  
for emp in stack:  
 print(emp.id)

### **FINAL Output**



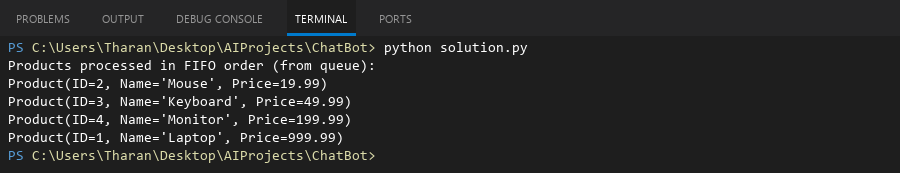
## **QUESTION 23**

23. Write a C# program to create a Product class (ID, Name, Price) and store o bjects in an array by price. Further, u se a Queue<T> to manage product processing (FIFO order).

### **Code Solution**

class Product:  
 def \_\_init\_\_(self, id, name, price):  
 self.ID = id  
 self.Name = name  
 self.Price = price  
 def \_\_repr\_\_(self):  
 return f"Product(ID={self.ID}, Name='{self.Name}', Price={self.Price})"  
  
products = [  
 Product(1, "Laptop", 999.99),  
 Product(2, "Mouse", 19.99),  
 Product(3, "Keyboard", 49.99),  
 Product(4, "Monitor", 199.99)  
]  
  
sorted\_products = sorted(products, key=lambda x: x.Price)  
  
from collections import deque  
queue = deque()  
  
for product in sorted\_products:  
 queue.append(product)  
  
print("Products processed in FIFO order (from queue):")  
while queue:  
 product = queue.popleft()  
 print(product)

### **FINAL Output**



## **QUESTION 24**

24. Write a program to calculate the sum of the diagonal elements of a square matrix.

### **Code Solution**

matrix = [  
 [1, 2, 3],  
 [4, 5, 6],  
 [7, 8, 9]  
]  
sum\_diagonal = 0  
for i in range(len(matrix)):  
 sum\_diagonal += matrix[i][i]  
print(sum\_diagonal)

### **FINAL Output**

