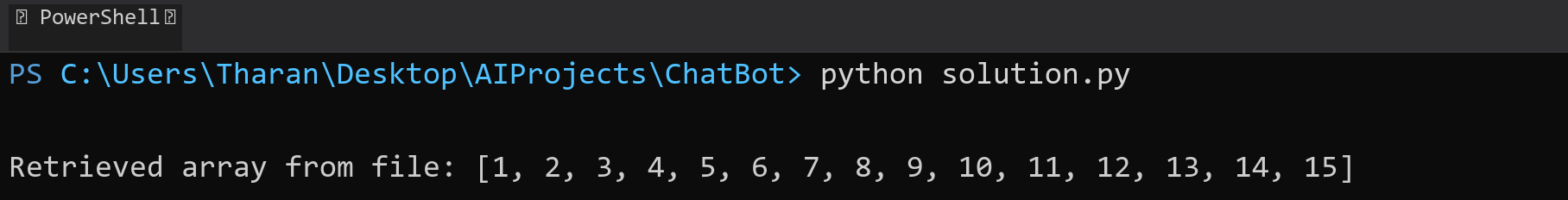
## **QUESTION 1**

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**

import numpy as np  
  
array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])  
  
with open('array\_data.txt', 'w') as file:  
 for number in array:  
 file.write(str(number) + '\n')  
  
stored\_array = []  
with open('array\_data.txt', 'r') as file:  
 for line in file:  
 stored\_array.append(int(line.strip()))  
  
print("Retrieved array from file:", stored\_array)

### **FINAL Output**



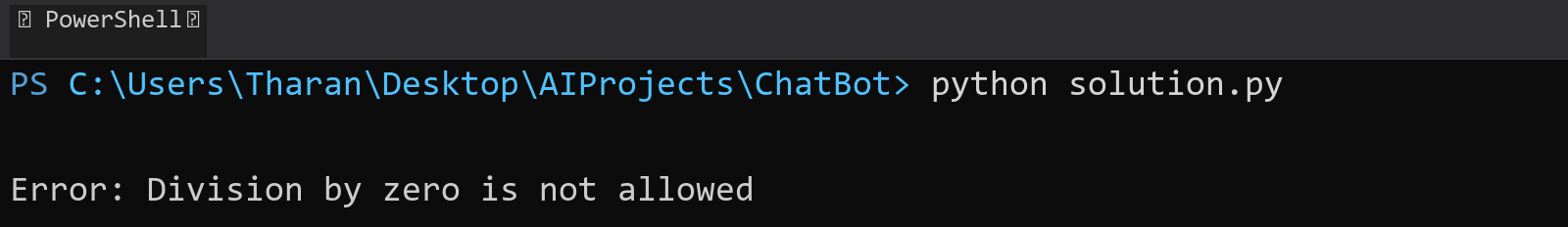
## **QUESTION 2**

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 = 2  
 b = 0  
 if not isinstance(a, int) or not isinstance(b, int):  
 raise TypeError("Both numbers must be integers")  
 result = a / b  
 print(f"Result of division: {result}")  
except ZeroDivisionError:  
 print("Error: Division by zero is not allowed")  
except TypeError as type\_error:  
 print(f"Error: {type\_error}")

### **FINAL Output**



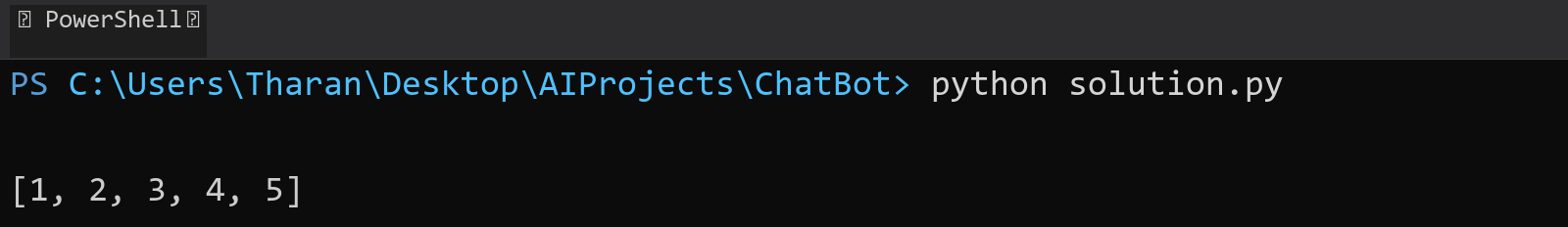
## **QUESTION 3**

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**

numbers = [1, 2, 3, 4, 5]  
  
with open('numbers.txt', 'w') as file:  
 for number in numbers:  
 file.write(str(number) + '\n')  
  
stored\_numbers = []  
with open('numbers.txt', 'r') as file:  
 for line in file:  
 stored\_numbers.append(int(line.strip()))  
  
print(stored\_numbers)

### **FINAL Output**



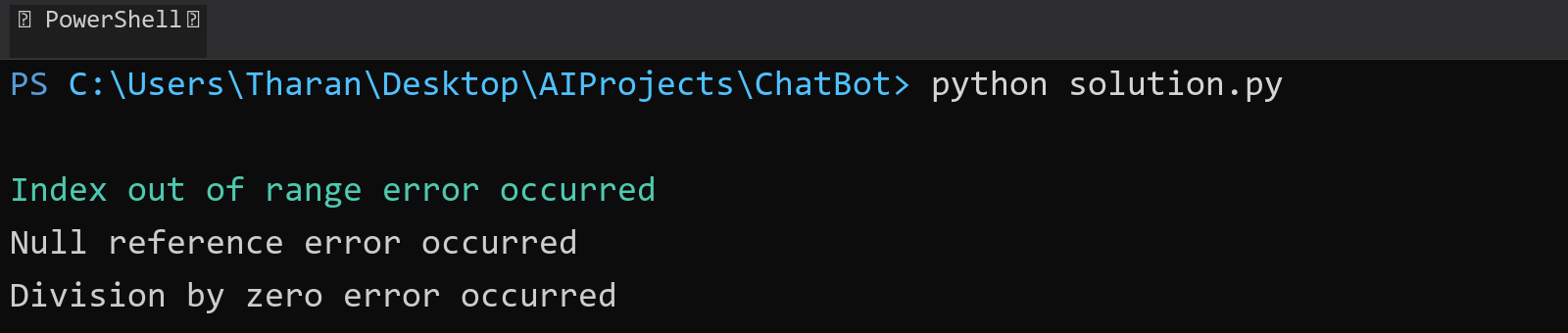
## **QUESTION 4**

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

### **Code Solution**

try:  
 my\_list = [1, 2, 3]  
 print(my\_list[5])  
except IndexError:  
 print("Index out of range error occurred")  
  
try:  
 my\_dict = None  
 print(my\_dict['key'])  
except TypeError:  
 print("Null reference error occurred")  
  
try:  
 result = 10 / 0  
except ZeroDivisionError:  
 print("Division by zero error occurred")  
except Exception as e:  
 print(f"An unexpected error occurred: {e}")

### **FINAL Output**



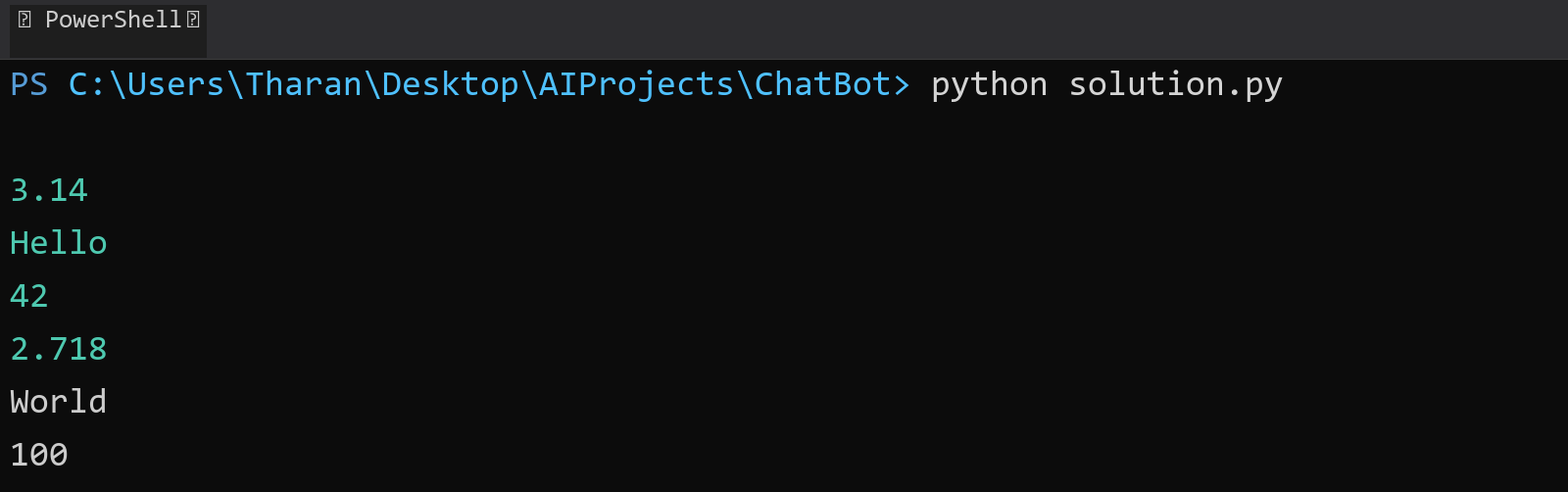
## **QUESTION 5**

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**

my\_list = []  
my\_list.append(3.14)  
my\_list.append("Hello")  
my\_list.append(42)  
my\_list.append(2.718)  
my\_list.append("World")  
my\_list.append(100)  
for item in my\_list:  
 print(item)

### **FINAL Output**



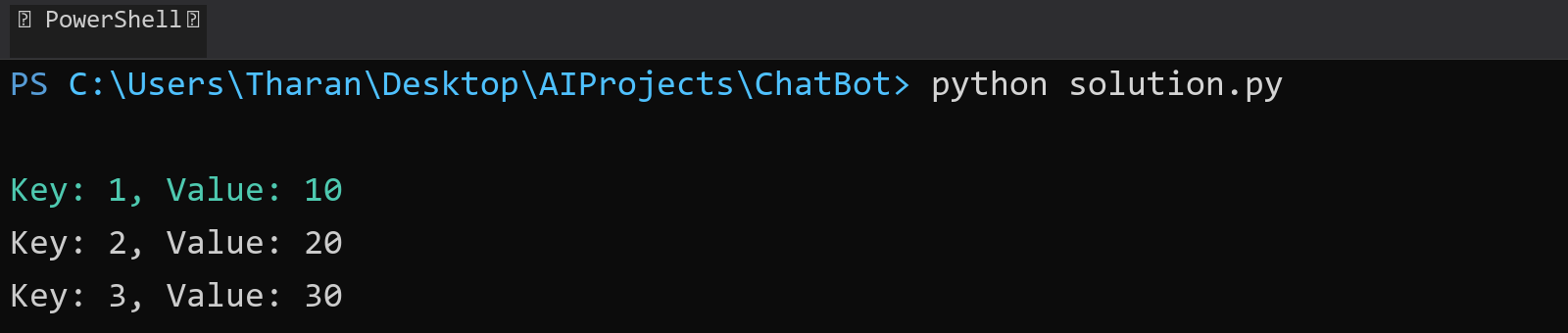
## **QUESTION 6**

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**

hashtable = {}  
hashtable[1] = 10  
hashtable[2] = 20  
hashtable[3] = 30  
for key, value in hashtable.items():  
 print(f"Key: {key}, Value: {value}")

### **FINAL Output**



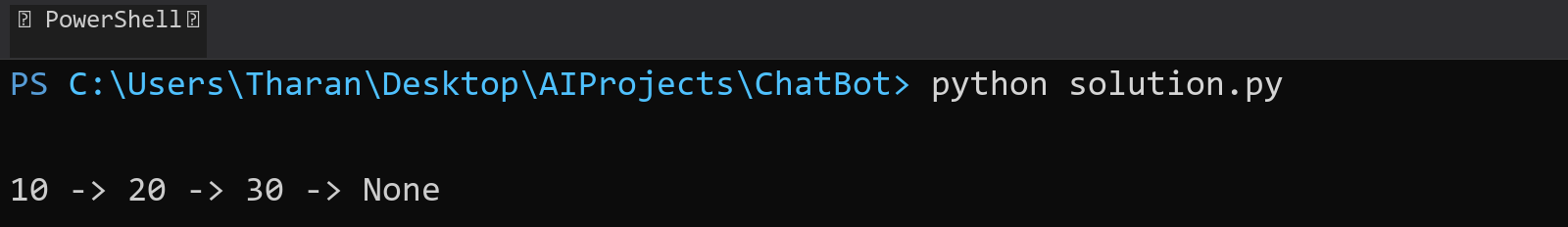
## **QUESTION 7**

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 print\_list(self):  
 current = self.head  
 while current:  
 print(current.data, end=" -> ")  
 current = current.next  
 print("None")  
  
linked\_list = LinkedList()  
linked\_list.insert\_at\_beginning(30)  
linked\_list.insert\_at\_beginning(20)  
linked\_list.insert\_at\_beginning(10)  
linked\_list.print\_list()

### **FINAL Output**



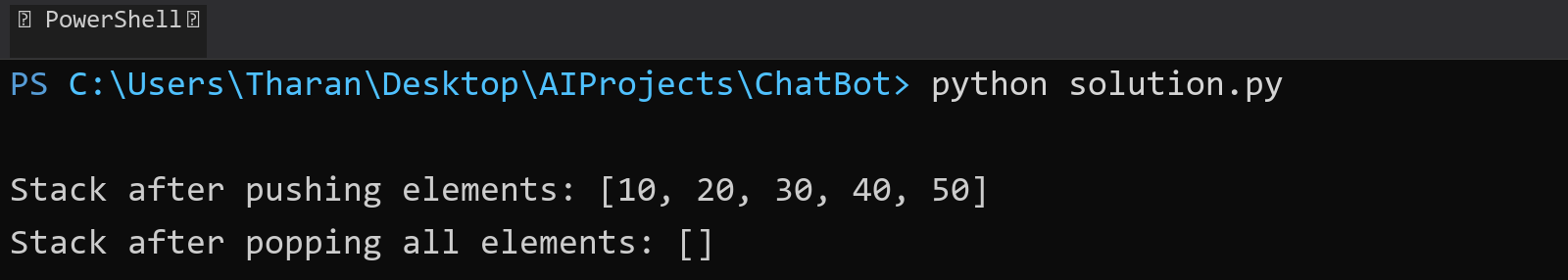
## **QUESTION 8**

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()  
   
 def is\_empty(self):  
 return len(self.items) == 0  
   
 def peek(self):  
 if not self.is\_empty():  
 return self.items[-1]  
   
 def size(self):  
 return len(self.items)  
  
stack = Stack()  
  
stack.push(10)  
stack.push(20)  
stack.push(30)  
stack.push(40)  
stack.push(50)  
  
print("Stack after pushing elements:", stack.items)  
  
stack.pop()  
stack.pop()  
stack.pop()  
stack.pop()  
stack.pop()  
  
print("Stack after popping all elements:", stack.items)

### **FINAL Output**



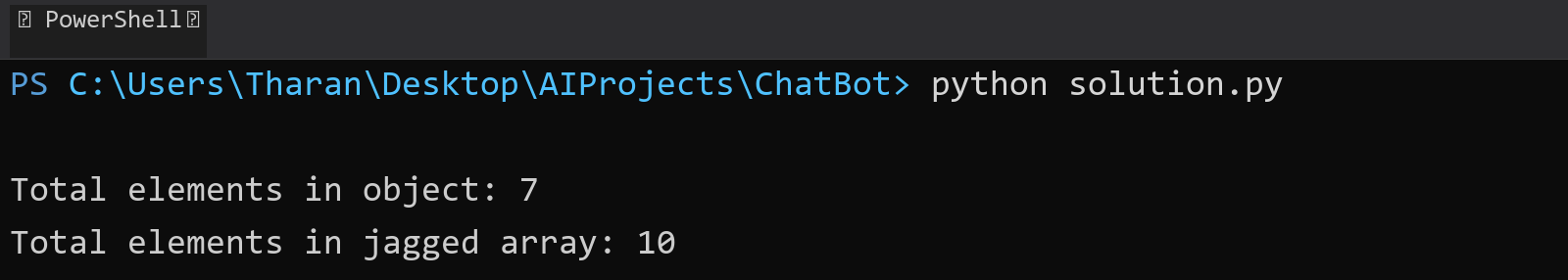
## **QUESTION 9**

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

### **Code Solution**

object = {'a': 1, 'b': 2, 'c': [1, 2, 3], 'd': {'x': 1, 'y': 2}}  
jagged\_array = [[1, 2, 3], [4, 5], [6, 7, 8, 9], [10]]  
  
def count\_elements\_object(obj):  
 count = 0  
 if isinstance(obj, dict):  
 for value in obj.values():  
 if isinstance(value, (dict, list)):  
 count += count\_elements\_object(value)  
 else:  
 count += 1  
 elif isinstance(obj, list):  
 for item in obj:  
 if isinstance(item, (dict, list)):  
 count += count\_elements\_object(item)  
 else:  
 count += 1  
 return count  
  
def count\_elements\_jagged(arr):  
 count = 0  
 for row in arr:  
 count += len(row)  
 return count  
  
total\_object = count\_elements\_object(object)  
total\_jagged = count\_elements\_jagged(jagged\_array)  
  
print(f"Total elements in object: {total\_object}")  
print(f"Total elements in jagged array: {total\_jagged}")

### **FINAL Output**



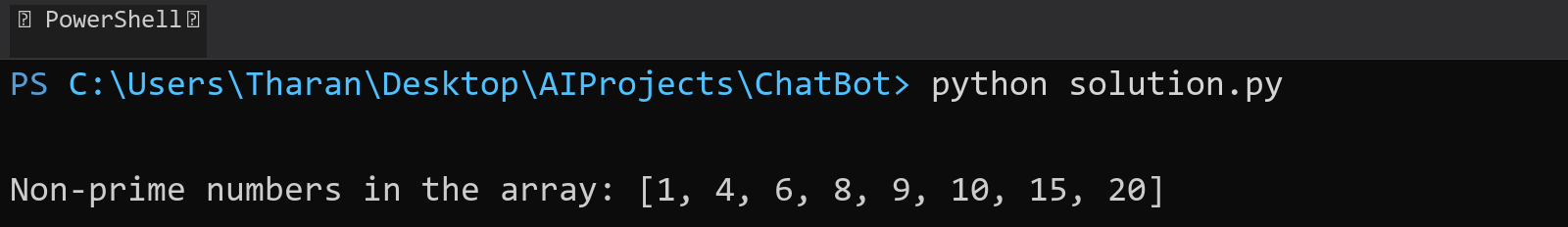
## **QUESTION 10**

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

### **Code Solution**

def is\_not\_prime(n):  
 if n < 2:  
 return True  
 for i in range(2, int(n \*\* 0.5) + 1):  
 if n % i == 0:  
 return True  
 return False  
  
def find\_non\_prime\_numbers(arr):  
 non\_primes = []  
 for num in arr:  
 if is\_not\_prime(num):  
 non\_primes.append(num)  
 return non\_primes  
  
array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 17, 20, 23]  
result = find\_non\_prime\_numbers(array)  
print("Non-prime numbers in the array:", result)

### **FINAL Output**



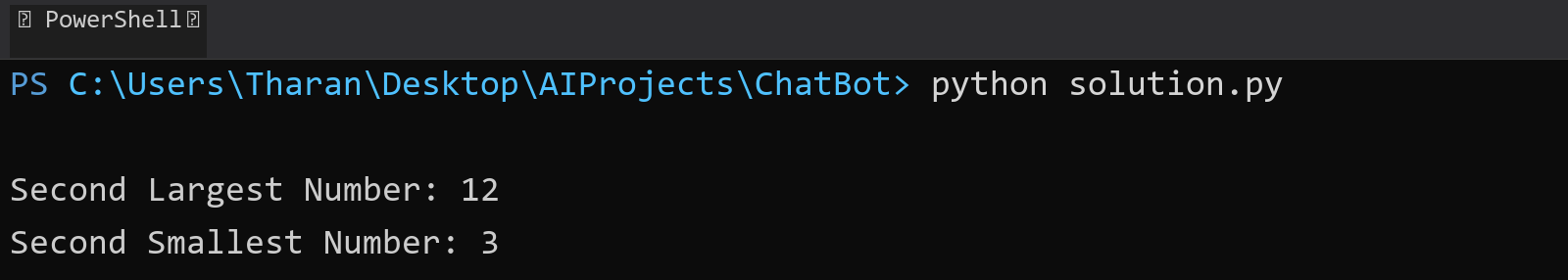
## **QUESTION 11**

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

### **Code Solution**

def find\_second\_largest\_smallest(arr):  
 if len(arr) < 2:  
 return None, None  
 sorted\_arr = sorted(arr)  
 second\_smallest = sorted\_arr[1]  
 second\_largest = sorted\_arr[-2]  
 return second\_largest, second\_smallest  
  
array = [10, 5, 8, 12, 3, 7, 9, 15, 1, 6]  
second\_largest, second\_smallest = find\_second\_largest\_smallest(array)  
print(f"Second Largest Number: {second\_largest}")  
print(f"Second Smallest Number: {second\_smallest}")

### **FINAL Output**



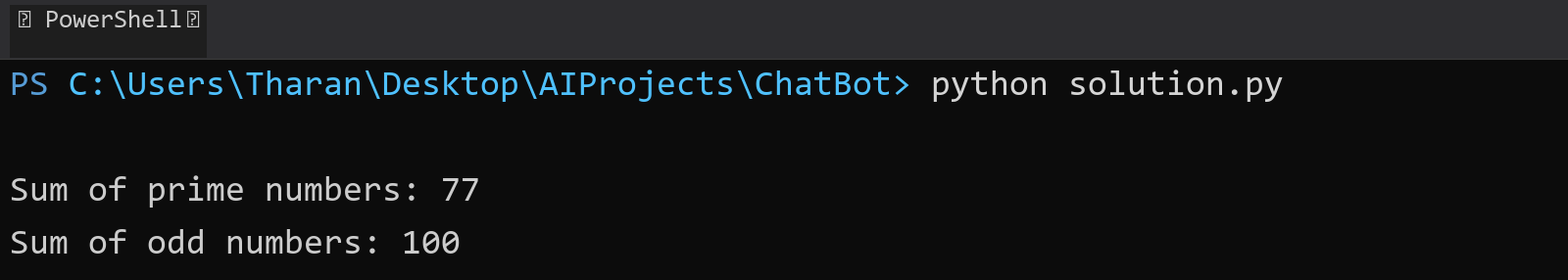
## **QUESTION 12**

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 < 2:  
 return False  
 for i in range(2, int(n \*\* 0.5) + 1):  
 if n % i == 0:  
 return False  
 return True  
  
def is\_odd(n):  
 return n % 2 != 0  
  
array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 17, 19]  
  
prime\_sum = sum(num for num in array if is\_prime(num))  
odd\_sum = sum(num for num in array if is\_odd(num))  
  
print(f"Sum of prime numbers: {prime\_sum}")  
print(f"Sum of odd numbers: {odd\_sum}")

### **FINAL Output**



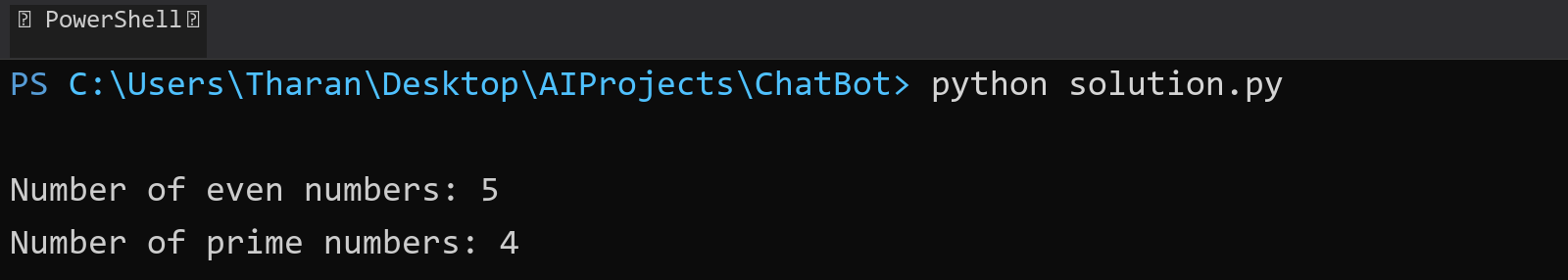
## **QUESTION 13**

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

### **Code Solution**

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  
  
def count\_even\_and\_prime(arr):  
 even\_count = 0  
 prime\_count = 0  
   
 for num in arr:  
 if num % 2 == 0:  
 even\_count += 1  
 if is\_prime(num):  
 prime\_count += 1  
   
 return even\_count, prime\_count  
  
array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
even\_numbers, prime\_numbers = count\_even\_and\_prime(array)  
  
print(f"Number of even numbers: {even\_numbers}")  
print(f"Number of prime numbers: {prime\_numbers}")

### **FINAL Output**



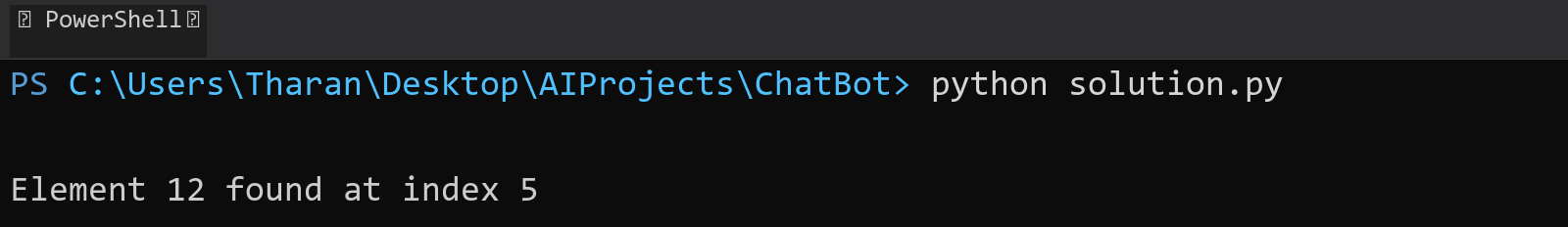
## **QUESTION 14**

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

### **Code Solution**

def binary\_search(array, target):  
 left = 0  
 right = len(array) - 1  
   
 while left <= right:  
 mid = (left + right) // 2  
 if array[mid] == target:  
 return mid  
 elif array[mid] < target:  
 left = mid + 1  
 else:  
 right = mid - 1  
 return -1  
  
arr = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]  
target = 12  
result = binary\_search(arr, target)  
  
if result != -1:  
 print(f"Element {target} found at index {result}")  
else:  
 print(f"Element {target} not found in array")

### **FINAL Output**



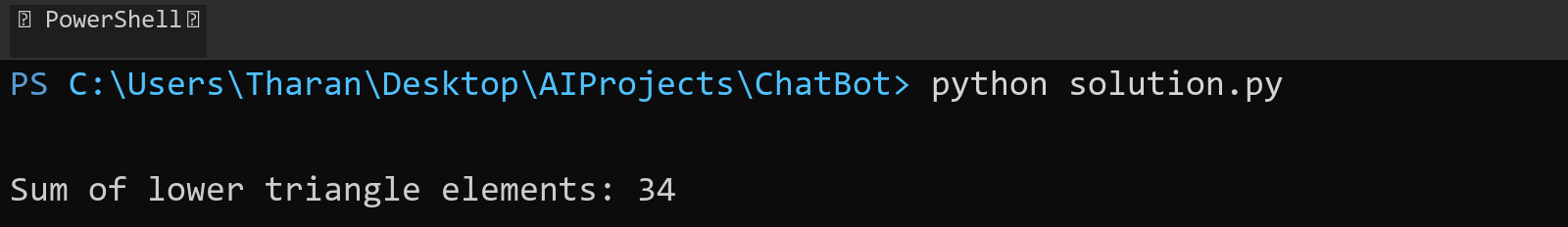
## **QUESTION 15**

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

### **Code Solution**

def calculate\_lower\_triangle\_sum(matrix):  
 rows = len(matrix)  
 cols = len(matrix[0])  
 sum = 0  
 for i in range(rows):  
 for j in range(cols):  
 if i >= j:  
 sum += matrix[i][j]  
 return sum  
  
matrix = [[1, 2, 3],  
 [4, 5, 6],  
 [7, 8, 9]]  
  
result = calculate\_lower\_triangle\_sum(matrix)  
print("Sum of lower triangle elements:", result)

### **FINAL Output**



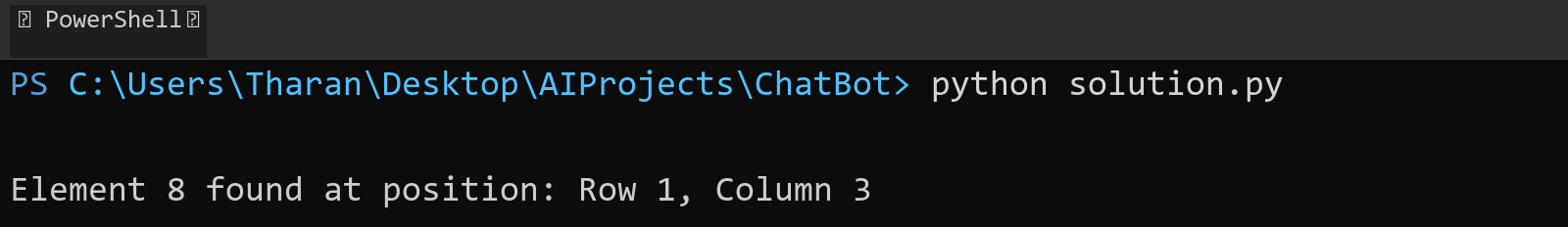
## **QUESTION 16**

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

### **Code Solution**

def linear\_search(arr, target):  
 for i in range(len(arr)):  
 for j in range(len(arr[i])):  
 if arr[i][j] == target:  
 return i, j  
 return -1, -1  
  
jagged\_array = [  
 [1, 3, 5],  
 [2, 4, 6, 8],  
 [7, 9],  
 [10, 11, 12, 13, 14]  
]  
  
target = 8  
row, col = linear\_search(jagged\_array, target)  
  
if row != -1:  
 print(f"Element {target} found at position: Row {row}, Column {col}")  
else:  
 print(f"Element {target} not found in the array")

### **FINAL Output**



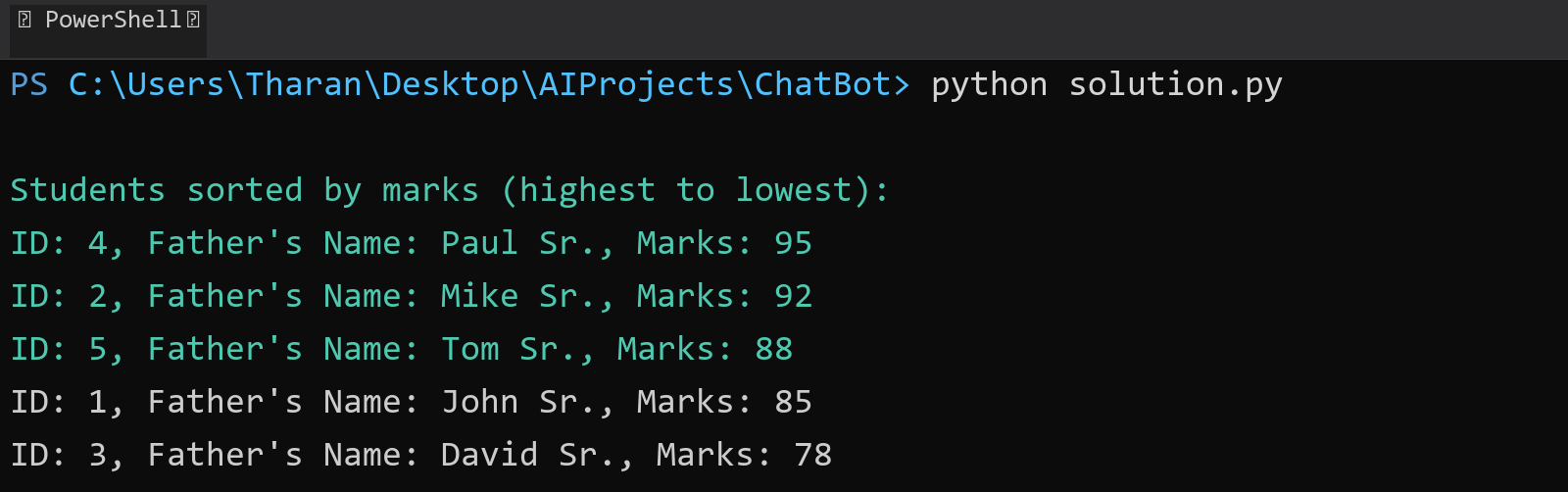
## **QUESTION 17**

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  
  
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 not self.head:  
 self.head = new\_node  
 return  
 current = self.head  
 while current.next:  
 current = current.next  
 current.next = new\_node  
  
 def display(self):  
 current = self.head  
 while current:  
 print(f"ID: {current.data.ID}, Father's Name: {current.data.Father\_Name}, Marks: {current.data.Marks}")  
 current = current.next  
  
students = [  
 Student(1, "John Sr.", 85),  
 Student(2, "Mike Sr.", 92),  
 Student(3, "David Sr.", 78),  
 Student(4, "Paul Sr.", 95),  
 Student(5, "Tom Sr.", 88)  
]  
  
sorted\_students = sorted(students, key=lambda x: x.Marks, reverse=True)  
  
linked\_list = LinkedList()  
for student in sorted\_students:  
 linked\_list.append(student)  
  
print("Students sorted by marks (highest to lowest):")  
linked\_list.display()

### **FINAL Output**



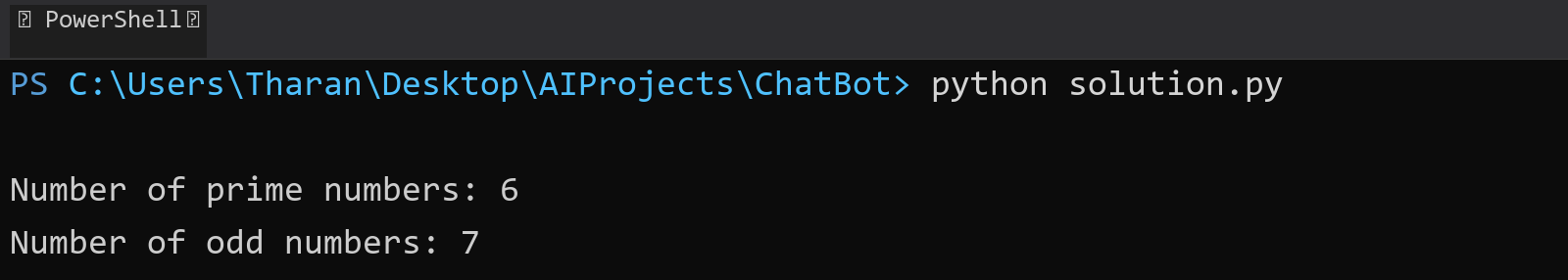
## **QUESTION 18**

18. Write a program to count the number of prime and odd numbers in a one -dimensional 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  
  
def count\_prime\_odd(arr):  
 prime\_count = 0  
 odd\_count = 0  
 for num in arr:  
 if is\_prime(num):  
 prime\_count += 1  
 if num % 2 != 0:  
 odd\_count += 1  
 return prime\_count, odd\_count  
  
array = [2, 3, 4, 5, 6, 7, 8, 9, 11, 13, 15]  
prime\_numbers, odd\_numbers = count\_prime\_odd(array)  
print(f"Number of prime numbers: {prime\_numbers}")  
print(f"Number of odd numbers: {odd\_numbers}")

### **FINAL Output**



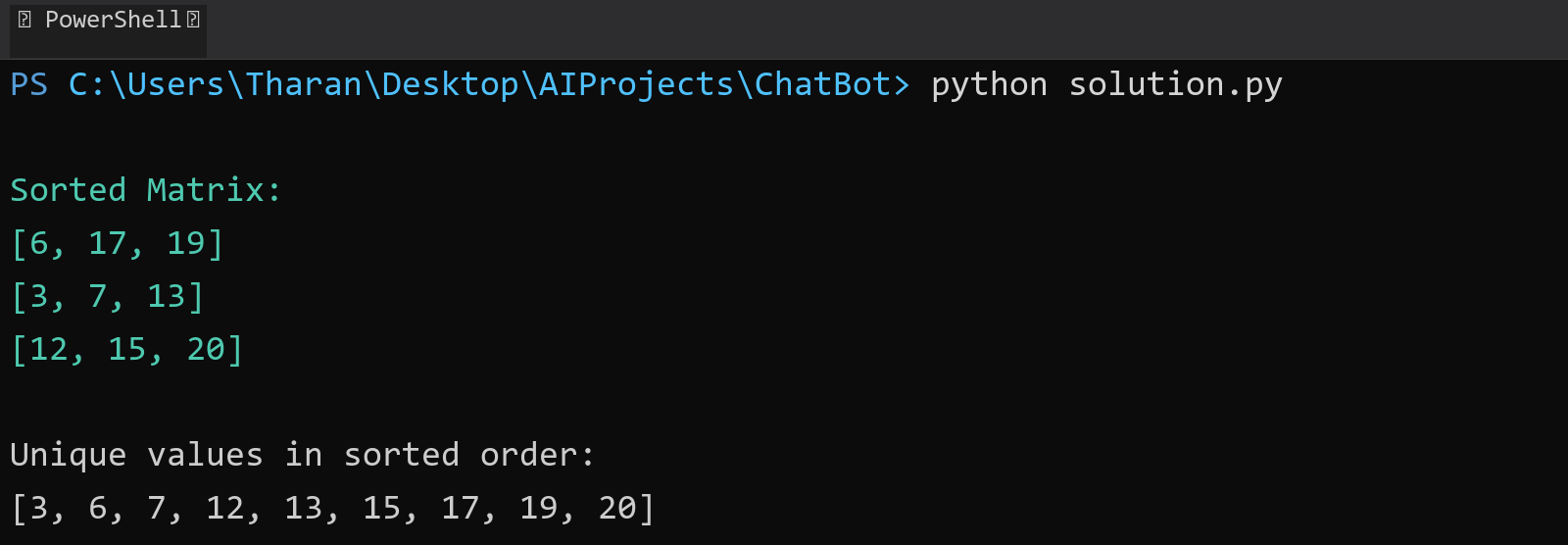
## **QUESTION 19**

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, 20) for \_ in range(3)] for \_ in range(3)]  
  
for row in matrix:  
 row.sort()  
  
print("Sorted Matrix:")  
for row in matrix:  
 print(row)  
  
unique\_values = sorted(list(set([num for row in matrix for num in row])))  
  
print("\nUnique values in sorted order:")  
print(unique\_values)

### **FINAL Output**



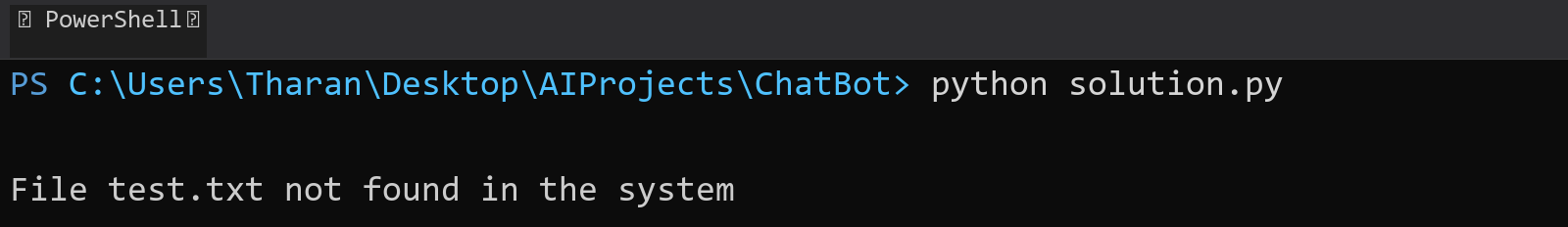
## **QUESTION 20**

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 os  
from pathlib import Path  
  
filenames = ["test.txt", "sample.doc", "data.pdf"]  
search\_file = "test.txt"  
directory = {}  
  
for filename in filenames:  
 file\_path = Path(filename)  
 if file\_path.exists():  
 file\_stats = os.stat(filename)  
 directory[filename] = {  
 "size": file\_stats.st\_size,  
 "created": file\_stats.st\_ctime,  
 "modified": file\_stats.st\_mtime,  
 "path": str(file\_path.absolute())  
 }  
  
if search\_file in directory:  
 file\_details = directory[search\_file]  
 print(f"File found: {search\_file}")  
 print(f"Size: {file\_details['size']} bytes")  
 print(f"Created: {file\_details['created']}")  
 print(f"Modified: {file\_details['modified']}")  
 print(f"Path: {file\_details['path']}")  
else:  
 print(f"File {search\_file} not found in the system")

### **FINAL Output**



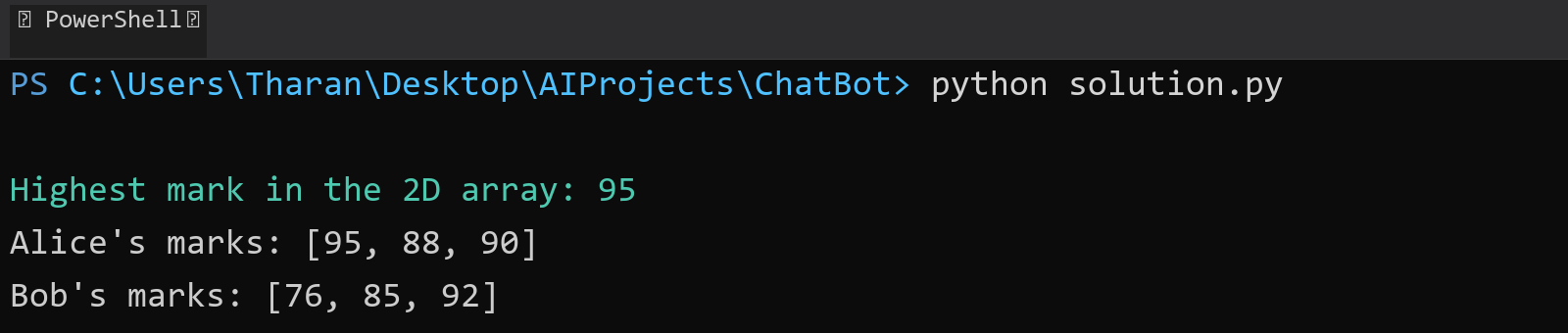
## **QUESTION 21**

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**

student\_marks = [[85, 92, 78], [95, 88, 90], [76, 85, 92], [89, 94, 87]]  
  
highest\_mark = max(max(row) for row in student\_marks)  
  
student\_data = {  
 "John": [85, 92, 78],  
 "Alice": [95, 88, 90],  
 "Bob": [76, 85, 92],  
 "Emma": [89, 94, 87]  
}  
  
def search\_by\_name(name):  
 if name in student\_data:  
 return student\_data[name]  
 return None  
  
print(f"Highest mark in the 2D array: {highest\_mark}")  
print(f"Alice's marks: {search\_by\_name('Alice')}")  
print(f"Bob's marks: {search\_by\_name('Bob')}")

### **FINAL Output**



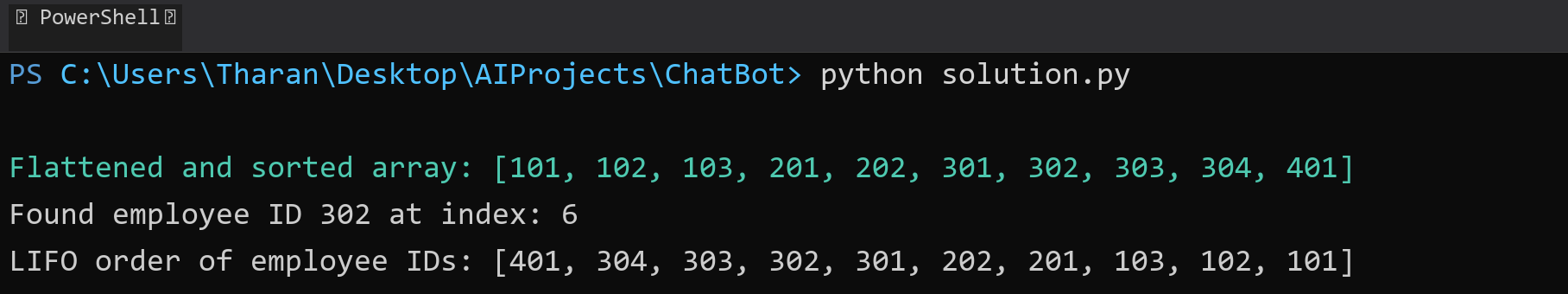
## **QUESTION 22**

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**

def binary\_search(arr, target):  
 left = 0  
 right = len(arr) - 1  
   
 while left <= right:  
 mid = (left + right) // 2  
 if arr[mid] == target:  
 return mid  
 elif arr[mid] < target:  
 left = mid + 1  
 else:  
 right = mid - 1  
 return -1  
  
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()  
   
 def is\_empty(self):  
 return len(self.items) == 0  
   
 def peek(self):  
 if not self.is\_empty():  
 return self.items[-1]  
   
 def size(self):  
 return len(self.items)  
  
jagged\_array = [  
 [101, 102, 103],  
 [201, 202],  
 [301, 302, 303, 304],  
 [401]  
]  
  
flattened\_array = []  
for subarray in jagged\_array:  
 flattened\_array.extend(sorted(subarray))  
  
employee\_stack = Stack()  
for emp\_id in flattened\_array:  
 employee\_stack.push(emp\_id)  
  
target\_id = 302  
search\_result = binary\_search(flattened\_array, target\_id)  
  
print(f"Flattened and sorted array: {flattened\_array}")  
print(f"Found employee ID {target\_id} at index: {search\_result}")  
  
popped\_ids = []  
while not employee\_stack.is\_empty():  
 popped\_ids.append(employee\_stack.pop())  
  
print(f"LIFO order of employee IDs: {popped\_ids}")

### **FINAL Output**



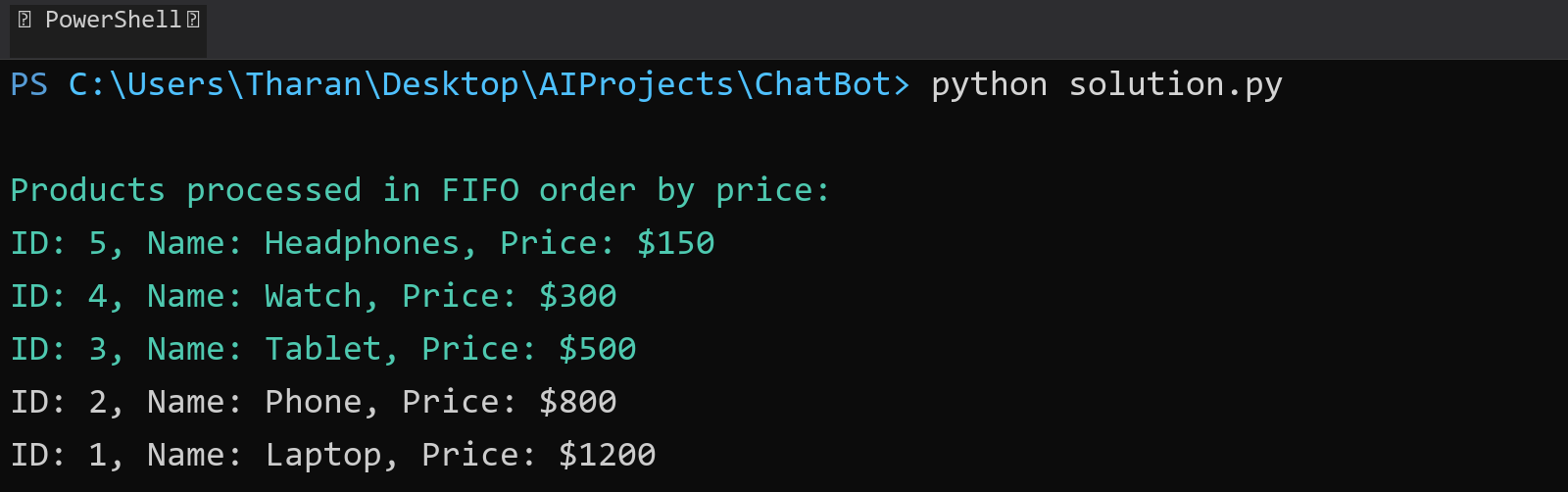
## **QUESTION 23**

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  
  
class Queue:  
 def \_\_init\_\_(self):  
 self.items = []  
  
 def enqueue(self, item):  
 self.items.append(item)  
  
 def dequeue(self):  
 if not self.is\_empty():  
 return self.items.pop(0)  
 return None  
  
 def is\_empty(self):  
 return len(self.items) == 0  
  
products = [  
 Product(1, "Laptop", 1200),  
 Product(2, "Phone", 800),  
 Product(3, "Tablet", 500),  
 Product(4, "Watch", 300),  
 Product(5, "Headphones", 150)  
]  
  
sorted\_products = sorted(products, key=lambda x: x.price)  
  
product\_queue = Queue()  
for product in sorted\_products:  
 product\_queue.enqueue(product)  
  
print("Products processed in FIFO order by price:")  
while not product\_queue.is\_empty():  
 product = product\_queue.dequeue()  
 print(f"ID: {product.id}, Name: {product.name}, Price: ${product.price}")

### **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]]  
  
diagonal\_sum = 0  
for i in range(len(matrix)):  
 diagonal\_sum += matrix[i][i]  
  
print(f"Sum of diagonal elements: {diagonal\_sum}")

### **FINAL Output**

