

**Name: Shamsa Nawaz**

**Inter ID : TN/IN02/PY/039**

**Task No : 02**

### **Question 01**

- 1. Store 5 student names & print each.**
- 2. Reverse list without reverse().**

#### **Description:**

#### **Store and Print Student Names:**

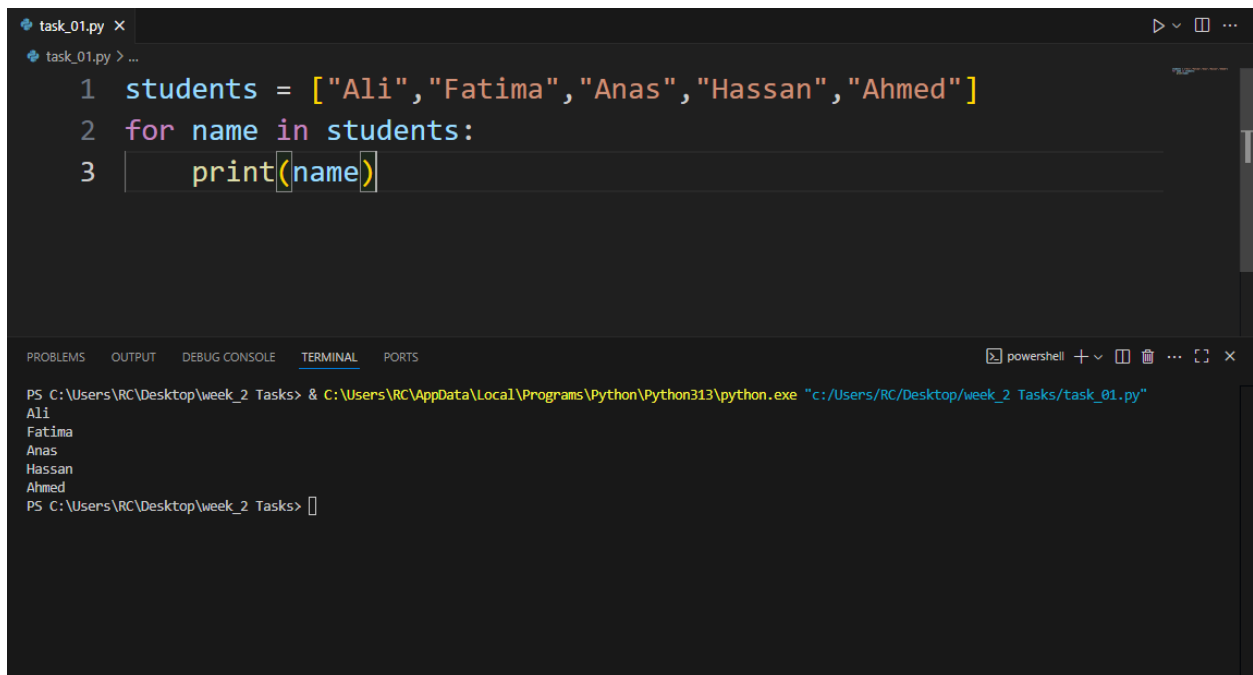
Create a list to store the names of five students.

Use a loop to print each name from the list one by one.

**Reverse list without reverse().**

Print the list of student names in reverse order without using the `.reverse()` function or slicing (`[::-1]`).

Use a loop with indexing to access elements from the end to the beginning.

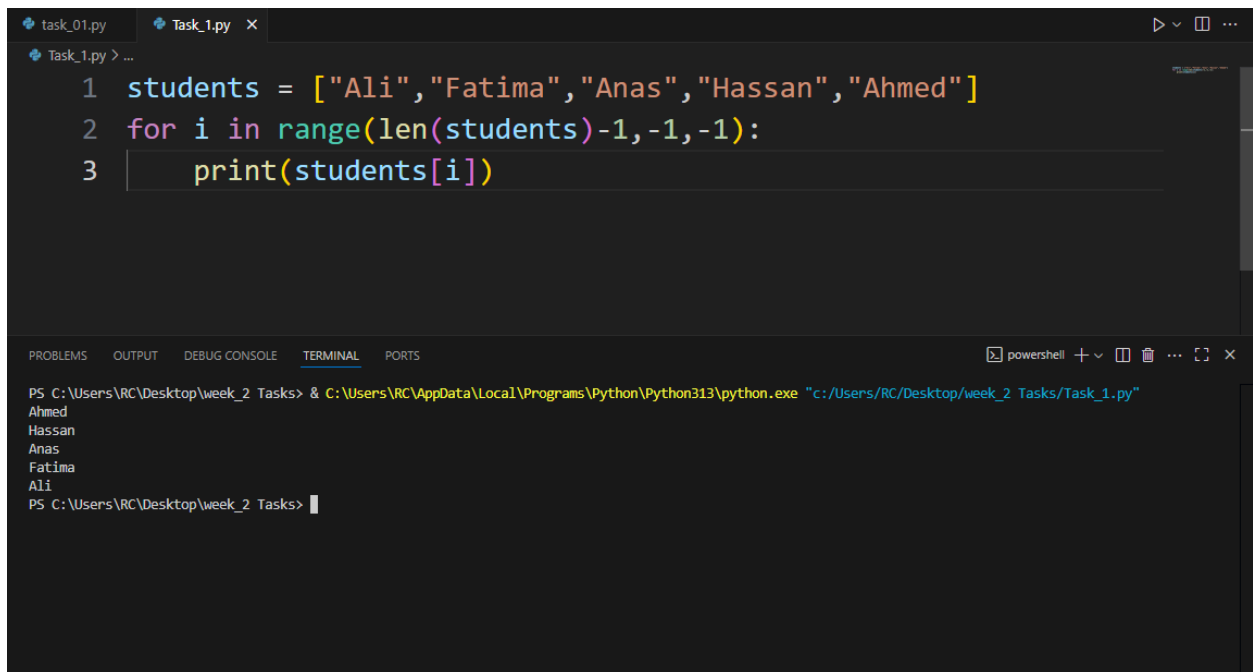


The screenshot shows a code editor window with a file named `task_01.py`. The code in the editor is as follows:

```
1 students = ["Ali", "Fatima", "Anas", "Hassan", "Ahmed"]
2 for name in students:
3     print(name)
```

Below the code editor is a terminal window. The terminal shows the command to run the script and its output:

```
PS C:\Users\RC\Desktop\week_2 Tasks> & C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week_2 Tasks/task_01.py"
Ali
Fatima
Anas
Hassan
Ahmed
PS C:\Users\RC\Desktop\week_2 Tasks>
```



```
task_01.py Task_1.py x
Task_1.py > ...
1 students = ["Ali", "Fatima", "Anas", "Hassan", "Ahmed"]
2 for i in range(len(students)-1, -1, -1):
3     print(students[i])

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\RC\Desktop\week_2 Tasks> C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week_2 Tasks/Task_1.py"
Ahmed
Hassan
Anas
Fatima
Ali
PS C:\Users\RC\Desktop\week_2 Tasks>
```

## QUESTION: 02

1. Store 3 coordinates & unpack.
2. Swap vars using tuple assignment

### Description:

#### Store 3 Coordinates & Unpack:

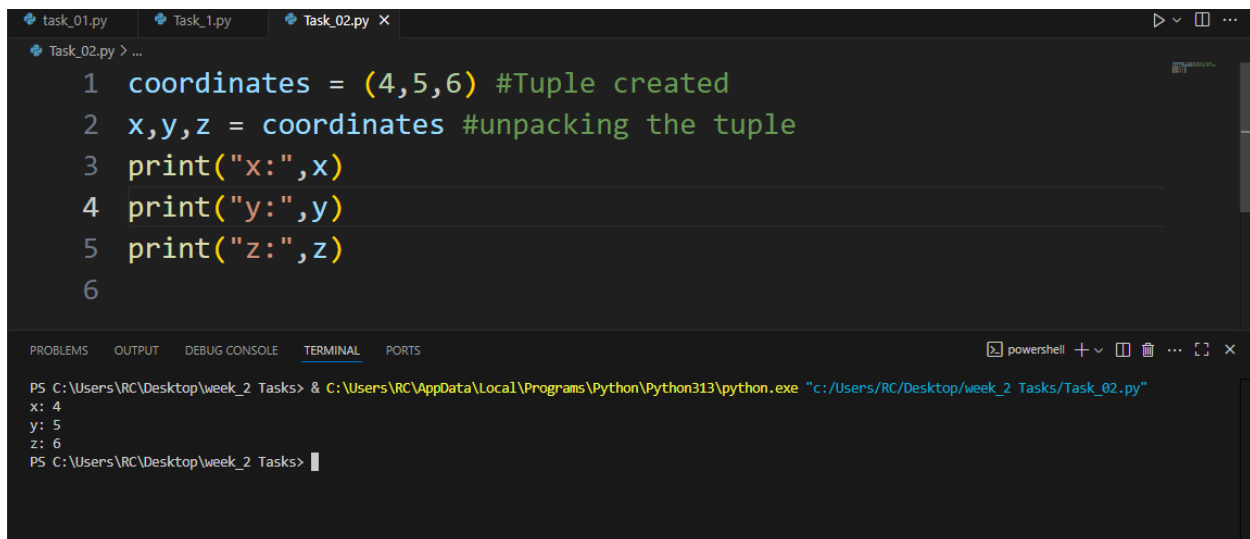
Create a tuple that stores three coordinate values.

Then unpack those values into individual variables (x, y, z) and print them.

## Swap Variables Using Tuple Assignment

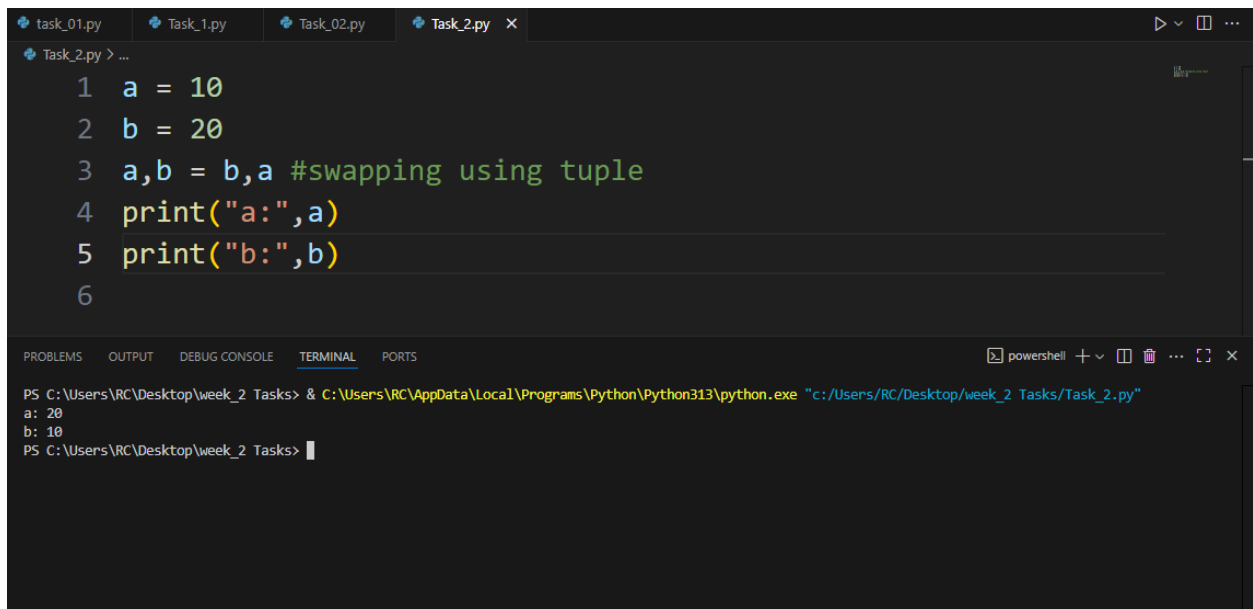
Create two variables with values.

Swap their values using tuple assignment without using a third (temporary) variable.



```
task_01.py Task_1.py Task_02.py x
Task_02.py > ...
1 coordinates = (4,5,6) #Tuple created
2 x,y,z = coordinates #unpacking the tuple
3 print("x:",x)
4 print("y:",y)
5 print("z:",z)
6

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\RC\Desktop\week_2 Tasks> & C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week_2 Tasks/Task_02.py"
x: 4
y: 5
z: 6
PS C:\Users\RC\Desktop\week_2 Tasks> |
```



```
task_01.py Task_1.py Task_02.py Task_2.py x
Task_2.py > ...
1 a = 10
2 b = 20
3 a,b = b,a #swapping using tuple
4 print("a:",a)
5 print("b:",b)
6

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\RC\Desktop\week_2 Tasks> & C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week_2 Tasks/Task_2.py"
a: 20
b: 10
PS C:\Users\RC\Desktop\week_2 Tasks> |
```

## QUESTION: 03

1. Remove duplicates from list.
2. Find intersection of two sets.

### Description:

#### Remove Duplicates from List:

In this task, we take a list that contains repeated (duplicate) values.

We use a set in Python to remove these duplicates, because a set only keeps unique items.

This way, we get a list with no repeated values.

### **Find Intersection of Two Sets:**

This task is about finding common elements between two sets.

In Python, we can use the `intersection()` method or `&` operator to find which items appear in both sets.

This is helpful when we want to compare two sets and get only shared values

```
task_01.py Task_1.py Task_02.py Task_2.py Task_03.py X
Task_03.py > ...
1 numbers = [ 1,2,2,3,4,4,5]
2 unique_numbers = list(set(numbers))
3 print(unique_numbers)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\RC\Desktop\week_2 Tasks> & C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week_2 Tasks/Task_03.py"
[1, 2, 3, 4, 5]
PS C:\Users\RC\Desktop\week_2 Tasks> |
```

```
task_01.py Task_1.py Task_02.py Task_2.py Task_03.py Task_3.py X
Task_3.py > ...
1 set1 = {1,2,3,4}
2 set2 = {3,4,5,6}
3 intersection = set1 & set2
4 print("Common elements:",intersection)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\RC\Desktop\week_2 Tasks> & C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week_2 Tasks/Task_3.py"
Common elements: {3, 4}
PS C:\Users\RC\Desktop\week_2 Tasks> |
```

## QUESTION: 04

1. Student record CRUD in dict.
2. Count word frequency in sentence

## **Description:**

### **Student Record CRUD using Dictionary:**

In this task, we use a Python dictionary to store student records. Each student has a roll number as the key and a name as the value. We perform CRUD operations:

Create: Add a new student

Read: View all students

Update: Change the name of an existing student

Delete: Remove a student from the record



## Count Word Frequency in a Sentence:

This task counts how many times each word appears in a sentence. The sentence is split into words, and a dictionary is used to store the word counts. If a word appears again, its count increases.

```
Task_04.py > ...
1 # create : Adding student records
2 students = {
3
4     "101": "Ali",
5     "102": "sara",
6
7 }
8 # read : print all student records
9 print("Student Records:")
10 for roll, name in students.items():
11     print(f"Roll No :{roll},Name : {name}")
12 # update : change name of student with roll 101
13 students["101"] = "Ahmed"
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

powershell + - [ ] ... [ ] [ ] X

```
PS C:\Users\RC\Desktop\week_2 Tasks> & C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week_2 Tasks/Task_04.py"
Student Records:
Roll No :101,Name : Ali
Roll No :102,Name : sara

After update and delete:
Roll No:101, Name:Ahmed
PS C:\Users\RC\Desktop\week_2 Tasks> |
```

```
Task_4.py > ...
1  sentece = input("Enter a sentence :")
2  word = sentece.split()
3  freq = {}
4  for word in word :
5      if word in freq :
6          freq[word] += 1
7      else:
8          freq[word] = 1
9      print(freq)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\RC\Desktop\week_2 Tasks> & C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week_2 Tasks/Task_4.py"
Enter a sentence :This is a cat and that is also cat
{'This': 1}
{'This': 1, 'is': 1}
{'This': 1, 'is': 1, 'a': 1}
{'This': 1, 'is': 1, 'a': 1, 'cat': 1}
{'This': 1, 'is': 1, 'a': 1, 'cat': 1, 'and': 1}
{'This': 1, 'is': 1, 'a': 1, 'cat': 1, 'and': 1, 'that': 1}
{'This': 1, 'is': 2, 'a': 1, 'cat': 1, 'and': 1, 'that': 1, 'also': 1}
PS C:\Users\RC\Desktop\week_2 Tasks>
```

## QUESTION: 05

1. Write `calc(a,b,op)`.
2. Write `factorial(n)` recursive.

### Description:

This program performs basic calculations using a function `calc(a, b, op)`.

The user enters two numbers and an operator (+, -, \*, /).

The function returns the result based on the operator.

The main() function takes input and displays the result

## **Recursive Factorial Function:**

This program calculates the factorial of a number using a recursive function called factorial(n).

If n is 0 or 1, the function returns 1 (because  $0! = 1$  and  $1! = 1$ ).

Otherwise, it returns  $n * \text{factorial}(n - 1)$ , which means it calls itself repeatedly until it reaches 1.

In the end, it prints the result of factorial(5), which is 120.

```
Task_05.py > calc
1 def calc(a,b,op):
2     if op == '+':
3         return a+b
4     elif op == '-':
5         return a-b
6     elif op == '*':
7         return a*b
8     elif op == '/':
9         if b != 0:
10            return a/b
11        else:
12            return "cannot divide by zero"
13    else:
14        return "invalid operator"
15
16 def main():
17     a = int(input("Enter First number :"))
18     b = int(input("Enter second number :"))
19     op = input("Enter operator(+,-,*,/) :")
20     result = calc(a,b,op)
21
22 if __name__ == '__main__':
23     main()
24
25 # Test cases
26 # calc(12,20, '/') == 0.6
27 # calc(12,20, '+') == 32
28 # calc(12,20, '-') == -8
29 # calc(12,20, '*') == 240
30 # calc(12,20, '/') == 0.6
31 # calc(12,0, '/') == "cannot divide by zero"
32 # calc(12,20, '^') == "invalid operator"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\RC\Desktop\week\_2 Tasks> & C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week\_2 Tasks/Task\_05.py"

Enter First number :12  
Enter second number :20  
Enter operator(+,-,\*,/) :/  
Result: 0.6  
PS C:\Users\RC\Desktop\week\_2 Tasks>

Ln 9, Col 20 Spaces: 3 UTF-8 CRLF {} Python Python 3.13 (64-bit)

```
task_5.py > ...
1 def factorial(n):
2     if n == 0 or n == 1:
3         return 1
4     else:
5         return n * factorial(n-1)
6 print("factorial:",factorial(5))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\RC\Desktop\week\_2 Tasks> & C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week\_2 Tasks/Task\_5.py"

factorial: 120  
PS C:\Users\RC\Desktop\week\_2 Tasks>

## **QUESTION: 06**

- 1. Use random & datetime in script.**
- 2. Create math\_utils module & import**

### **Description:**

This script uses two built-in Python modules: random and datetime.

`random.randint(1, 100)` generates a random number between 1 and 100.

`datetime.datetime.now()` returns the current date and time.

The program prints both the random number and the current time.

**2.Create math\_utils module & import**

In this task, we created a custom module named `math_utils` which contains two functions:

`add(a, b)` — returns the sum of two numbers.

`square(n)` — returns the square of a number.

We then imported this module into another file (`main.py`) using the `import` keyword, and called both functions to display their results.

This task helped us understand how to create, import, and reuse our own modules in Python.

```
Task_06.py > main
1 import random
2 import datetime
3 def main():
4     # Generate a random number between 1 and 100
5     num = random.randint(1,100)
6     print("Random number:",num)
7
8     # Get current date and time
9     now = datetime.datetime.now()
10    print("Current date and time:",now)
11    main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\RC\Desktop\week\_2 Tasks> & C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week\_2 Tasks/Task\_06.py"

Random number: 49  
Current date and time: 2025-08-03 17:11:51.993822  
PS C:\Users\RC\Desktop\week\_2 Tasks>

```
main.py > ...
1 import math_utils
2 def main():
3     result1 = math_utils.add(4,5)
4     result2 = math_utils.square(6)
5     print("Addition:",result1)
6     print("square:",result2)
7    main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\RC\Desktop\week\_2 Tasks> & C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week\_2 Tasks/main.py"

Addition: 9  
square: 36  
PS C:\Users\RC\Desktop\week\_2 Tasks>

## QUESTION: 07

1. Safe int input loop.
2. File open with error message

**Description:**

In this task, we take integer input from the user inside a while loop using try-except.

If the user enters something that's not an integer (like text), the program shows an error message and asks again.

It keeps repeating until the user gives a valid number.

In this task, we try to open a file using try-except.

If the file does not exist or there's a mistake in file name, it catches the error and shows a message like "File not found."



```
Task_07.py > ...
1 while True:
2     try:
3         number = int(input("Please enter an integer:"))
4         print("You entered:",number)
5         break
6     except ValueError:
7         print("Invalid input! Please enter a valid integer .")
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\RC\Desktop\week\_2 Tasks> & C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week\_2 Tasks/Task\_07.py"  
Please enter an integer:7  
You entered: 7  
PS C:\Users\RC\Desktop\week\_2 Tasks> |

```
Task_7.py > ...
1 try:
2     file = open("date.txt", "r")
3     content = file.read()
4     print(content)
5     file.close()
6 except FileNotFoundError:
7     print("Error: File not found .Please check the file name.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\RC\Desktop\week\_2 Tasks> & C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week\_2 Tasks/Task\_7.py"  
Error: File not found .Please check the file name.  
PS C:\Users\RC\Desktop\week\_2 Tasks> |

## QUESTION: 08

Phonebook App: CRUD contacts dict <->  
JSON file storage

Description:

This program is a simple phonebook application using Python. It allows the user to:

Add new contacts

View all contacts

Update a contact's phone number

Delete a contact

Exit the program

All contacts are stored in a JSON file, so the data is saved even after the program ends.

The program uses functions to handle each task and includes exception handling to prevent errors (like invalid input or missing files). It also uses the `os` module to check if the contacts file exists before reading it.

Each contact is stored as a dictionary with a name and phone number, and all contacts are stored in a list.

```

1 import json
2 import os
3
4 filename = "contacts.json"
5
6 # Load existing contacts from file
7 def load_contacts():
8     if not os.path.exists(filename):
9         return []
10    try:
11        with open(filename, "r") as f:
12            return json.load(f)
13    except json.JSONDecodeError:
14        return []
15
16 # Save contacts to file
17 def save_contacts(contacts):
18    try:
19        with open(filename, "w") as f:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\RC\Desktop\week_2 Tasks> & C:\Users\RC\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/RC/Desktop/week_2 Tasks/Task_8.py"
```

```
1. Add
2. View
3. Update
4. Delete
5. Exit
Choose: 1
Enter name: Ali
Enter phone: 03001234567
Contact added successfully!
```

```
1. Add
2. View
3. Update
4. Delete
5. Exit
Choose: 
```

Ln 27, Col 35 Spaces: 4 UTF-8 CRLF {} Python Python 3.13 (64-bit)

8:12 PM