Mark as done

Description Submission view

Available from: Thursday, 25 September 2025, 9:20 AM
Due date: Thursday, 25 September 2025, 10:50 AM
Requested files: p1.py, p2.py, p3.py, p4.py (Download)
Type of work: Individual work

Problem 1: Numbers Equal to Their Index

Description
Write a program that counts how many numbers in a list are equal to their index position..

Concept
This checks your understanding of list indexing and conditional comparisons.

Task

1. Prompt the user: `Enter numbers:`
2. Read space separated integers into a list.
3. Traverse the list with indices.
4. Count how many satisfy `list[i] == i`.
5. Print the count.

Input format

- A line of space–separated integers.

Output format

```
[value]
```

Example 1

```
Enter numbers: 1 1 2 3 5
3
```

Explanation:

- Index 0: value 1 != 0 → no
- Index 1: value 1 == 1 → count=1
- Index 2: value 2 == 2 → count=2
- Index 3: value 3 == 3 → count=3
- Index 4: value 5 != 4 → no
- So, total Count: 3

Example 2

```
Enter numbers: 2 0 2 4 4
2
```

Explanation:

- Index 0: value 2 != 0 → no
- Index 1: value 0 != 1 → no
- Index 2: value 2 == 2 → count=1
- Index 3: value 4 != 3 → no
- Index 4: value 4 == 4 → count=2
- So, total Count: 2

?

Restrictions

- Input list length ≥ 1.
- Use 0-based indexing (i.e. Indexing starts from 0.)
- No external libraries or predefined functions allowed.

Problem 2 - Product of Column Minima

Description:
Write a program that finds the minimum value in each column of a matrix, and then calculates the product of all these minimums.

Concept:
This task requires processing a matrix column-by-column. For each column, you must find its minimum value. The final result is an aggregation (the product) of these individual minimums, which can be calculated using an accumulator variable.

Task:
Your program must produce output that exactly matches the format specified.
1. Prompt for the number of rows with: `Enter number of rows: `.
2. Prompt for the matrix data with: `Enter matrix rows with each row on a new line:`. The user will then enter each row on a separate line with elements separated by spaces.
3. Check if the matrix is rectangular. If not, print `-1` and stop.
4. Find the minimum value in each column of the matrix.
5. Compute the product of these minimums and print the result.

Example:
*(Text in **bold** is what the user types.)*

```
Enter number of rows: 3
Enter matrix rows with each row on a new line:
8 4 9
3 5 2
6 7 4
24
```

(Another example)

```
Enter number of rows: 2
Enter matrix rows with each row on a new line:
10 20
50 60 70
-1
```

Restrictions:
All matrix elements will be integers.
No external libraries are necessary.

Problem 3: Magic Rows in a Matrix

Description:
You are given a matrix of integers. A **magic row** is defined as a row whose sum equals the sum of the corresponding column at the same index. For example, row `i` is a magic row if the sum of all elements in row `i` equals the sum of all elements in column `i`.
Your task is to count how many magic rows exist in the given matrix.

Concept:
This problem tests the ability to:

- Traverse a matrix by rows and columns.
- Compute row sums and column sums efficiently.
- Compare values across rows and columns.

Task:

1. Prompt the user to input the number of rows and columns.
2. Take matrix elements as input in row-major order.
3. Make sure that the number of matrix elements = r*c, if not, print "INVALID INPUT"

4. Compute the sum of each row and its corresponding column.

5. Count how many rows are magic rows.

6. Print the count.

Input Format:

```
Enter number of rows and columns: <Two space separated integers>
Enter matrix elements: <space-separated integers>
```

Output Format:

```
<value>(if number of matrix elements = r*c)
INVALID INPUT(if number of matrix elements != r*c)
```

Example 1:

Input:

```
Enter number of rows and columns: 3 3
Enter matrix elements: 2 7 6 9 5 1 4 3 8
```

Output:

```
3
```

Explanation:
Matrix:
2 7 6
9 5 1
4 3 8

Row sums = [15, 15, 15] Column sums = [15, 15, 15]
Each row sum matches its corresponding column sum → All 3 rows are magic.

Example 2:

Input:

```
Enter number of rows and columns: 3 3
Enter matrix elements: 1 2 3 4 5 6 7 8 9
```

Output:

```
1
```

Explanation:
Matrix:
1 2 3
4 5 6
7 8 9
Row sums = [6, 15, 24] Column sums = [12, 15, 18]
Only the second row sum matches the second column sum → Count = 1

Problem 4 - Custom Strip Function

Description:
Write a program that mimics the behavior of the `.strip()` string method. Given a main string and a second string of characters to strip, remove all leading and trailing characters from the main string that are present in the strip-characters string.

Concept:
This problem tests your understanding of the character-set nature of stripping, as opposed to substring removal. The logic involves identifying the boundaries of the core string by finding the first and last characters that are *not* part of the set of characters to be stripped.

Task:
Your program must produce output that exactly matches the format specified.
1. Prompt the user for the main string with: `Enter main string: `.
2. Prompt the user for the strip characters with: `Enter characters to strip: `.
3. Create and print a new string by removing all leading and trailing characters from the main string that are present in the strip-characters string.
4. Your implementation must correctly handle the case where the entire string is stripped away, resulting in an empty string.

Example:

*(Text in **bold** is what the user types.)*

```
Enter main string: ___hello!!!__
Enter characters to strip: _!
hello
```

(Another example demonstrating character-set logic)

```
Enter main string: lolhello
Enter characters to strip: lo
he
```

(Explanation: The function checks from the left and from the right for any characters in the set {"l", "o"}. From the left, it removes 'l', 'o', 'l' and stops at 'h'. From the right, it removes 'o', 'l','l' and stops at 'e'.)

(Another example demonstrating character-set logic)

```
Enter main string: lolhelpolol
Enter characters to strip: ol
help
```

(Explanation: The function checks from the left and from the right for any characters in the set {"l", "o"}. From the left, it removes 'l', 'o', 'l' and stops at 'h'. From the right, it removes 'l', 'o', 'l', 'o' and stops at 'p'.)

Restrictions:

You **must not** use the built-in `.strip()`, `.lstrip()`, or `.rstrip()` string methods.
Both the main string and the strip-characters string are valid strings of **length ≥ 0**.

Requested files

p1.py

```
# write your code here below
```

p2.py

```
# write your code here below
```

p3.py

```
# write your code here below
```

p4.py

```
# write your code here below
```

VPL