Mark as done

Description Submission view

Available from: Wednesday, 24 September 2025, 9:15 AM
Due date: Wednesday, 24 September 2025, 10:45 AM
Requested files: p1.py, p2.py, p3.py, p4.py (Download)
Type of work: Individual work

Problem 1: Numbers Greater than the First Element

Description
Count how many numbers in a list are **strictly greater** than the first element.

Concept
Tests comparison with a fixed reference (the first element).

Task

1. Prompt:
   `Enter numbers:`
2. Store input into a list.
3. Compare each number with the first element.
4. Count how many elements after the first are strictly greater than first number and print the total.

Input format

- User enters space-separated integers in one line.

Output format

- `[value]`

Example 1

```
Enter numbers: 4 5 2 7 8
3
```

Explanation:
- First element: 4
- Compare each subsequent number to the first element (4):
  - 5 > 4 → count = 1
  - 2 > 4? → no
  - 7 > 4 → count = 2
  - 8 > 4 → count = 3
- Numbers strictly greater than the first element (4): 5, 7, 8 → total 3

Example 2

```
Enter numbers: 2 6 3 2 7 2
3
```

Explanation:
- First element: 2
- Compare each subsequent number to the first element (2):
  - 6 > 2 → count = 1
  - 3 > 2 → count = 2
  - 2 > 2? → no (must be strictly greater)

?

- 7 > 2 → count = 3
- 2 > 2? → no

- Numbers strictly greater than the first element (2): 6, 3, 7 → total 3

Restrictions

- No external libraries or predefined functions allowed. Use of map() and split() is allowed for handling input.
- The input list will always have atleast 1 element.

Problem 2 - Maximum Column Span

Description:
Write a program to find the "span" for each column in a matrix. The span of a column is defined as the difference between the maximum and minimum values in that column. Your program should then find and print the largest span among all columns.

Concept:
This problem requires processing matrix data column-by-column. For each column, you must identify its minimum and maximum values to calculate its span. The final result is the maximum of these individual column spans.

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 (all rows have the same length). If not, print `-1` and stop.
4. Calculate the span for each column by finding the difference between its maximum and minimum value.
5. Determine the maximum of these spans and print it.

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

```
Enter number of rows: 3
Enter matrix rows with each row on a new line:
1 9 3
4 -2 6
7 5 -1
11
```

Explanation:
The program calculates the span for each of the three columns:

- **Column 1:** Contains `[1, 4, 7]`. The maximum is `7` and the minimum is `1`. The span is `7 - 1 = 6`.
- **Column 2:** Contains `[9, -2, 5]`. The maximum is `9` and the minimum is `-2`. The span is `9 - (-2) = 11`.
- **Column 3:** Contains `[3, 6, -1]`. The maximum is `6` and the minimum is `-1`. The span is `6 - (-1) = 7`.

The calculated spans are `6`, `11`, and `7`. The largest among these is `11`, which is the final output.

(Another example)

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

Restrictions:
You may not use any external libraries.
All matrix elements will be integers.
The matrix will be non-empty (i.e., no. of rows > 0 and each row has atleast one integer)

Problem 3: Diagonal-Constant Matrix Check

Description

A **diagonal-constant matrix** (also called a **Toeplitz matrix**) is a special type of matrix in which each descending diagonal from left to right contains the same element.

- The **main diagonal** starts at the top-left corner and moves down-right.

- Every other diagonal parallel to the main diagonal must also contain equal values.

Concept

This problem tests the ability to traverse matrices and compare elements along diagonals. Each element should be equal to the element at its top-left position for the matrix to be diagonal-constant.

Task

Implement a program that:

1. Prompt the user to enter the number of rows(r) and columns(c).
2. Take the 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. Checks whether the matrix is diagonal-constant.
5. Prints **YES** if the condition holds, otherwise **NO**.

Input format

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

Output format

```
YES or NO(if number of matrix elements = r*c)
INVALID INPUT(if number of matrix elements != r*c)
```

Examples 1:

Input:

```
Enter number of rows and columns: 3 3
Enter matrix elements: 1 2 3 4 1 2 5 4 1
```

Output:

```
YES
```

Explanation:
Matrix is given as:
1 2 3
4 1 2
5 4 1
All diagonals: [5], [4,4], [1,1,1], [2,2], [3] have same elements.

Example 2:

Input:

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

Output:

```
NO
```

Explanation:
Matrix is given as:
1 2 3
4 1 9
5 4 1

The diagonal [2, 9] does not have the same values, so the matrix is not diagonal-constant

Restrictions

- All values should be integers. Row, column >1, matrix elements can be any integers
- No external libraries or predefined functions allowed

Problem 4 - Custom Replace Function

Description:
Write a program that mimics the behavior of the `replace()` string method. Given a main string `s`, a pattern `old`, a replacement `new`, and an integer `k`, you must replace the first `k` occurrences of `old` in `s` with `new`.

Concept:
Since strings are immutable, this problem requires you to build a new result string. The logic involves systematically finding occurrences of the `old` substring and constructing the new string by combining slices of the original string with the `new` substring.

Task:
Your program must produce output that exactly matches the format specified.
1. Prompt the user with the following messages on separate lines:
- `Enter main string:`
- `Enter old pattern:`
- `Enter new pattern:`
- `Enter k:`
2. If `k` is -1, it means all occurrences should be replaced. If `k` is greater than the number of occurrences, all occurrences are replaced.
3. Construct and print a new string where up to `k` occurrences of `old` have been replaced with `new`.

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

```
Enter main string: abracadabra
Enter old pattern: abra
Enter new pattern: xyz
Enter k: 1
xyzcadabra
```

(Another Example):

```
Enter main string: abracadabra
Enter old pattern: abra
Enter new pattern: xyz
Enter k: -1
xyzcadxyz
```

Restrictions:
You **must not** use the built-in `.replace()` string method.
The value for 'k' will either be -1 (to signify all occurrences) or a non-negative integer (k >= 0)

Requested files

p1.py

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

p2.py

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

p3.py

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

p4.py

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

VPL