[CS 11 25.1] HOPE 2e – Infiltration

Cheatsheet is available here: https://oj.dcs.upd.edu.ph/cs11cheatsheet/

Problem Statement

You are infiltrating a secret hideout! In the sky!

This hideout can be represented a grid of characters with r rows and c columns. The rows are numbered 0 to $r-1$ from top to bottom, and the columns are numbered 0 to $c-1$ from left to right. The cell at row i and column j is denoted by (i, j).

Each cell is one of the following characters:

- `<` - if you step on this tile, you move one tile to the left.
- `>` - if you step on this tile, you move one tile to the right.
- `^` - if you step on this tile, you move one tile up.
- `v` - if you step on this tile, you move one tile down.
- `.` - if you step on this tile, nothing happens.

Suppose you start at cell (s_i, s_j). After being moved (or not moved) by the tiles, which tile do you end up at?

If you keep cycling around tiles forever or fall out of the sky, please say so (see **Task Details**).

Task Details

Your task is to implement a function named `ending_location`, which should start like this:

```
def ending_location(hideout, s):
```

Here,

- `hideout` is a string denoting the hideout with rows being separated by newlines (`\n`). You may use `hideout.splitlines()` to obtain the lines of this string as a list.
- `s` is a tuple of two integers corresponding to s_i and s_j, respectively.

The function must return one of three things:

- (i, j), where $0 \le i < r$ and $0 \le j < c$, if you stop at some cell without getting into a cycle or falling out of the sky.
- `"(*.*)"`, if you get into a cycle.
- `"(x_x)"`, if you fall out of the sky.

Restrictions

- Loops and lists are allowed.
- Sets and dictionaries are allowed.
- Generators and comprehensions are allowed.
- Recursion is allowed.
- Your source code must have at most 950 bytes.

Examples

Example 1 Function Call

```
ending_location("""\
>>v
>.v
^<<""", (0, 0))
```

Example 1 Return Value

```
(1, 1)
```

Example 1 Explanation

The order in which you visit the cells is given below (with `1` being the first cell you visit):

```
123
894
765
```

Example 2 Function Call

```
ending_location("""\
>>v
^.v
^<<""", (0, 0))
```

Example 2 Return Value

```
"(*.*)"
```

Example 3 Function Call

```
ending_location("""\
>>v
<.v
^<<""", (0, 0))
```

Example 3 Return Value

```
"(x_x)"
```

Constraints

- The function `ending_location` will be called at most $70{,}000$ times.
- $1 \le r, c$
- $rc \le 350{,}000$
- The sum of rc across all test cases is at most $350{,}000$.
- Each cell is one of `<`, `>`, `^`, `v`, or `.`.

Scoring

Note: New tests may be added and all submissions may be rejudged at a later time. (All future tests will satisfy the constraints.)

- You get 100 ❤️ points if you solve all test cases where:
  - You will not end up in a cycle.
- You get 80 🔴 points if you solve all test cases where:
  - $rc \le 6{,}000$
  - The sum of rc across all test cases is at most $12{,}000$.
- You get 40 🔴 points if you solve all test cases.

Clarifications

Report an issue

No clarifications have been made at this time.

Submit solution [CS 11 25.1] HOPE 2

My submissions

Points: 220 (partial)
Time limit: 3.0s
Memory limit: 2G

Problem type

Allowed languages
py3