[CS 11 25.1] Mock HOPE 2h – Maize Maze

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

Problem Statement

Mr. Cornetto has a maize maze, and he wants a nice diagram illustrating it!

You are given the top view of the maize maze in a format like the following:

```
##################
#   #   #       #
### ### ### #####
#   # #         #
# ### # # #######
# # # # #       #
# # # ### #######
# # # # # #     #
# # # # # # #####
# #     # #     #
# ### # # # ### #
#     #       # #
##################
```

Mr. Cornetto asks you to redraw it using *box-drawing characters* that can form natural segments and bends.

For example, for the given maze above, the expected output is the following:

```
┌─────┬─────┬─────────┐
│     │     │         │
├──   ├─┐   └──   ─────┤
│     │ │             │
│  ┌──┤ │  ┌  ────────┤
│  │  │ │  │          │
│  │  │ ├──┤  ┌───────┤
│  │  │ │  │  │       │
│  │  ' '  │  │  ─────┤
│  │       │  │       │
│  └──   ┌  '  '  ──┐  │
│        │          │  │
└────────┴──────────┴──┘
```

Note that the maze has resized a bit, and natural bends and branches in the walls are represented by the appropriate box-drawing characters.

Here are all the characters that you will need to draw the output maze:

```
┼

┌ ┐

└ ┘

─ │

├ ┤

┬ ┴

╴ ╵

╶ ╷
```

The last four characters are used as the "tail end" of a wall that stops somewhere inside the maze. Also, `┼` is for corners where four walls meet.

Note that the vertical bar character above (`│`) is not the same as the pipe character in your keyboard!

Task Details

Your task is to implement a function called `redraw_maze_in_a_nice_way`. This function has a single parameter: a `tuple` of `str`s representing an $r \times c$ grid, which illustrates a maze.

The function must not return anything. Instead, it must `print` several lines illustrating the output grid. The function must not output leading or trailing spaces, or blank lines.

Restrictions

Note that some names are banned.

For this problem:

- Loops and lists are allowed.
- Sets and dictionaries are allowed.
- Generators and comprehensions are allowed.
- Recursion is **disallowed**. (The recursion limit has been greatly reduced.)
- Up to 32 function definitions are allowed.
- `print` is allowed.
- The source code limit is 5000.

Example Calls

Example 1 Function Call

```
redraw_maze_in_a_nice_way((
    "##################",
    "#   #   #       #",
    "### ### ### #####",
    "#   # #         #",
    "# ### # # #######",
    "# # # # #       #",
    "# # # ### #######",
    "# # # # # #     #",
    "# # # # # # #####",
    "# #     # #     #",
    "# ### # # # ### #",
    "#     #       # #",
    "##################",
))
```

Example 1 Output

```
┌─────┬─────┬─────────┐
│     │     │         │
├──   ├─┐   └──   ─────┤
│     │ │             │
│  ┌──┤ │  ┌  ────────┤
│  │  │ │  │          │
│  │  │ ├──┤  ┌───────┤
│  │  │ │  │  │       │
│  │  ' '  │  │  ─────┤
│  │       │  │       │
│  └──   ┌  '  '  ──┐  │
│        │          │  │
└────────┴──────────┴──┘
```

Constraints

- The function `redraw_maze_in_a_nice_way` will be called at most 20 times.
- $3 \leq r, c \leq 75$
- The input is a valid maze.

Scoring

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

- You get 160 🔴 points if you solve all test cases.

Submit solution [CS 11 25.1] Mock HOPE 2

My submissions

Points: 160 (partial)
Time limit: 4.0s
Memory limit: 1G

Author: dlko ()

Problem type

Allowed languages ~~NONE~~, py3

Clarifications

Report an issue

No clarifications have been made at this time.