Task 2 - Loading A Maze from a File

Introduction

For this task, we will build on some of the things from the previous task and use them to read a maze in from a file. This time the task will be completed in Python. To complete this task you will need to create simple versions of the Location and GameMap classes from the previous task. To be clear about the (small) distinction, they will be call Position and Maze. You will also need to create a function called read\_maze which will be the main unit tested.

There is also a \_main\_ section in which you can write your own testing code, but this is not assessed.

Position

**You will need to complete the class Position**. This class will not be directly tested, and you may implement it in any manner you see fit as long as it has the following two methods:

has\_direction which is an instance method and takes a str as a parameter. It should return True if the Position has a path in the direction indicated by the parameter and False if it doesn't.

is\_exit which is an instance method and takes no other parameters. It should return True if the Position is an exit and False otherwise. A Position is an exit if it is on the edge of the map and there is a path leading off that edge. This should be determined at the point the Position is created and stored, rather than attempting to compute it when the method is called.

The class also comes with a list called symbols that contains the symbols the input will be expressed in. A line leading to an edge indicates a path in that direction, where "north" is up.

Maze

You will also need to complete the class Maze. As with Position this class will not be directly tested, and you are free to implement the class as you see fit as long as it has the following methods:

get\_position which is an instance method that takes two additional parameters that are numbers representing coordinates in the maze. The coordinates are in the order row then column and (0, 0) is the top left. The method should return the Position at those coordinates in the Maze, if such a Position exists and None otherwise.

get\_height which is an instance method that returns the "height" of the maze (the number of rows).

get\_width which is an instance method that returns the "width" of the maze (the number of columns).

read\_maze

The read\_maze function is the core function of the task. It is not associated with a class, and takes a single parameter which will be a str containing the filename to be read.

The filename will contain a rectangular maze. Each line in the file is a row in the maze, and is terminated by \n. A line may include whitespace, so be cautious about aggressively stripping whitespace from lines (looking up the documentation for the strip() function may be informative, but it is not the only way to deal with this problem). Each line will have the same length, which is the number of columns.

Each individual symbol represents a position. If the symbol has a line leading to an edge, there is a path in that direction, where "north" is up. A position is an exit if it is at the edge of the maze and there is a path to the edge.

The read\_maze function should read this information in from the file, and construct a Maze and suitable Positions that correctly represent the input, and return this Maze.

General Advice and Answers

If you have completed the previous task, you will have a good example of how to construct Position and Maze, though here they can be a bit simpler if you desire.

Possibly the trickiest part of this efficiently converting the symbols to positions. A big if or match will solve it, but some careful observation of the list may reveal a different approach (they're in the list in that order for a reason - the same is true in the previous task).

Có thể phần khó nhất của việc này là chuyển đổi các ký hiệu sang vị trí một cách hiệu quả. Một if hoặc match lớn sẽ giải quyết được nó, nhưng một số quan sát cẩn thận về danh sách có thể cho thấy một cách tiếp cận khác (chúng nằm trong danh sách theo thứ tự đó là có lý do - điều này cũng đúng trong nhiệm vụ trước).

You may import anything you like from the Python standard library if you think it is useful.

I also recommend implementing some of the other methods that you would expect from a class such as \_str\_, and \_init\_ for the two classes to help with your own debugging.

