Robot City Navigation System

Overview

Your task is develop a system (end-to-end in theory, and some parts concretely) for robots to navigate city streets. Route planners (who have no technical expertise) will devise routes from common starting points to common destinations. When a robot wants to go somewhere, it will need to request a route and follow the instructions to get to its destination.

Your Responsibilities

You need to develop a means for planners to specify and save routes (to a database) that can be interpreted by robots (whose logic you must develop). Try to make sure this process is not too burdensome or complicated for the planner. You will also need to develop the logic the robots use when interpreting the instructions and navigating the city (but need not control individual motions of the robots).

Sample Workflows

For a route planner:

1. Open some sort of document
2. Fill in route instructions (*this can be anything from natural language sentences to rows in a spreadsheet with predefined columns, to some machine-readable format like JSON or XML*)
3. Upload document to your system

For uploader/database:

1. Load and process instructions
2. Create database entries to represent the uploaded route
3. Provide route data when queried

For robots:

1. Request a route from the database
2. Follow route instructions

Challenge Instructions

The city will be represented by a grid of x-y coordinates, starting at (0,0) in the bottom left (Southwest) corner. Each vertex in the grid is a street corner at the end of a city block. Assume the grid is very large (you don't need to worry about hitting the Northern or Eastern edges), but you will never have negative coordinates. The city has landmarks, represented as string tags, at various coordinates throughout the grid.

Part 1 - Route Specification

A)

Develop a domain-specific language for representing the following:

- start coordinates
- instructions:
    - turn left
    - turn right
    - go distance x in direction N/S/E/W
    - go distance x (in whatever direction you are facing)
    - go until you reach a landmark

Sample instructions in plain English (*instructions may double-back along your route, so don't worry about efficiency of travel or wasted time - robots don't get tired or bored easily*):

- Start at (245, 161)
- Go North 5 blocks
- Turn right
- Go until you reach landmark "Statue of Old Man with Large Hat"
- Go West 25 blocks
- Turn left
- Go 3 blocks

B)

Create a sample route upload file. The file should have specifications for at least 2 routes, and should include all possible instruction types.

Part 2 - Database schema

Define database tables for representing route information (use whatever format is clear and works for you - this need not be precise MySQL or UML, although it could be).

You don't need to implement anything here, but make some notes about the following considerations:

- Would you change your design (and if so, how) for:

- A system with millions of routes vs. a system with <100 routes
- A system where routes have thousands of instructions each vs. a system where routes have 1-10 instructions
- A system with millions of simultaneous users vs. <10 simultaneous users
- A system where routes are frequently changed and updated vs. one where routes are permanent once initially devised

Part 3 - Implementation

Write some functions or class(es) to interpret instructions for the robots (you can skip support for landmarks so that you don't need to worry about representing the city).

Your code should handle data in the format it would be received from the database (sample data could be created dynamically, or read in from a file). You don't need to develop any sort of graphics or interface, just output the starting coordinates, each instruction as it is read and interpreted, and the ending coordinates.

Your code should be at least minimally documented (with comments and docstrings) and you should include some unit tests for at least the core functionality.

If you have time and inclination you're welcome to more fully develop any part of this system (route interpreter/uploader, database, simulation, testing, etc.), but **please do not feel in any way obligated** to go beyond what is requested in Parts 1-3