Problem 3

1/1 point (ungraded)

Each robot must also have some code that tells it how to move about a room, which will go in a method called `updatePositionAndClean`.

Ordinarily we would consider putting all the robot's methods in a single class. However, later in this problem set we'll consider robots with alternate movement strategies, to be implemented as different classes with the same interface. These classes will have a different implementation of `updatePositionAndClean` but are for the most part the same as the original robots. Therefore, we'd like to use inheritance to reduce the amount of duplicated code.

We have already refactored the robot code for you into two classes: the `Robot` class you completed in Problem 2 (which contains general robot code), and a `StandardRobot` class that inherits from it (which contains its own movement strategy).

Complete the `updatePositionAndClean` method of `StandardRobot` to simulate the motion of the robot after a single time-step (as described on the Simulation Overview page).

```
class StandardRobot(Robot):
    """
    A StandardRobot is a Robot with the standard movement strategy.

    At each time-step, a StandardRobot attempts to move in its current direction; when
    it hits a wall, it chooses a new direction randomly.
    """
    def updatePositionAndClean(self):
        """
        Simulate the passage of a single time-step.

        Move the robot to a new position and mark the tile it is on as having
        been cleaned.
        """
```

We have provided the `getNewPosition` method of `Position`, which you may find helpful:

```
class Position(object):

    def getNewPosition(self, angle, speed):
        """
        Computes and returns the new Position after a single clock-tick has
        passed, with this object as the current position, and with the
        specified angle and speed.

        Does NOT test whether the returned position fits inside the room.

        angle: number representing angle in degrees, 0 <= angle < 360
        speed: positive float representing speed

        Returns: a Position object representing the new position.
        """
```

Note: You can pass in an integer or a float for the `angle` parameter.

Before moving on to Problem 4, check that your implementation of `StandardRobot` works by uncommenting the following line under your implementation of `StandardRobot`. Make sure that as your robot moves around the room, the tiles it traverses switch colors from gray to white. It should take about a minute for it to clean all the tiles.

```
testRobotMovement(StandardRobot, RectangularRoom)
```

Enter your code for classes `Robot` (from the previous problem) and `StandardRobot` below.

```
# Enter your code for Robot (from the previous problem)
#  and StandardRobot in this box

class Robot(object):
    """
    Represents a robot cleaning a particular room.
```

```
    At all times the robot has a particular position and direction in the room.
    The robot also has a fixed speed.
    Subclasses of Robot should provide movement strategies by implementing
    updatePositionAndClean(), which simulates a single time-step.
    """
    def __init__(self, room, speed):
        """
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Test results

See full output

CORRECT

See full output

Submit You have used 1 of 30 attempts

✔ Correct (1/1 point)

Problem Set 2: Problem 3

Hide Discussion

Topic: Sandbox / Problem Set 2: Problem 3

Show all posts by recent activity

There are no posts in this topic yet.

✖