Problem 3: Running the Simulation

courses.edx.org/courses/course-v1:MITx+6.00.2x_4+3T2015/courseware/d39541ec36564a88af34d319a2f16bd7/1067d0bb20374

In this problem you will write code that runs a complete robot simulation.

Recall that in each trial, the objective is to determine how many time-steps are on average needed before a specified fraction of the room has been cleaned. **Implement the following function:**

```
def runSimulation(num_robots, speed, width, height, min_coverage, num_trials,
                  robot_type):
    """
    Runs NUM_TRIALS trials of the simulation and returns the mean number of
    time-steps needed to clean the fraction MIN_COVERAGE of the room.

    The simulation is run with NUM_ROBOTS robots of type ROBOT_TYPE, each with
    speed SPEED, in a room of dimensions WIDTH x HEIGHT.
    """
```

The first six parameters should be self-explanatory. For the time being, you should pass in `StandardRobot` for the `robot_type` parameter, like so:

> *avg = runSimulation(10, 1.0, 15, 20, 0.8, 30,* ***StandardRobot****)*

Then, in `runSimulation` you should use `robot_type(...)` instead of `StandardRobot(...)` whenever you wish to instantiate a robot. (This will allow us to easily adapt the simulation to run with different robot implementations, which you'll encounter in Problem 5.)

Feel free to write whatever helper functions you wish.

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

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

For your reference, here are some approximate room cleaning times. These times are with a robot speed of 1.0.

- One robot takes around 150 clock ticks to completely clean a 5x5 room.
- One robot takes around 190 clock ticks to clean 75% of a 10x10 room.
- One robot takes around 310 clock ticks to clean 90% of a 10x10 room.
- One robot takes around 3322 clock ticks to completely clean a 20x20 room.
- Three robots take around 1105 clock ticks to completely clean a 20x20 room.

(These are only intended as guidelines. Depending on the exact details of your implementation, you may get times slightly different from ours.)

You should also check your simulation's output for speeds other than 1.0. One way to do this is to take the above test cases, change the speeds, and make sure the results are sensible.

For further testing, see the next page in this problem set about the optional way to use visualization methods. Visualization will help you see what's going on in the simulation and may assist you in debugging your code.

Enter your code for the definition of `runSimulation` below.

Test: Simulation1

```
        Testing 1 robot at 1.0 speed cleaning 78% of a 5x5 room.
        We use a new subclass of the Robot class, TestRobot, that sequentially
cleans one
         square at a time.
```

Output:

```
    19.0
```

Test: Simulation2

```
        Testing 1 robot at 1.0 speed cleaning 96% of a 10x12 room.
        We use a new subclass of the Robot class, TestRobot, that sequentially
cleans one
         square at a time.
```

Output:

```
    115.0
```

Test: Simulation3

```
        Testing 1 robot at 2.0 speed cleaning 96% of a 10x12 room.
        We use a new subclass of the Robot class, TestRobot, that sequentially
cleans one
         square at a time.
```

Output:

```
    58.0
```

Test: Simulation4

```
        Testing 2 robots at 1.0 speed cleaning 80% of a 8x8 room.
        We use a new subclass of the Robot class, TestRobot, that sequentially
cleans one
         square at a time.
```

Output:

```
25.0
```

Test: Simulation5

```
        Testing 2 robots at 3.0 speed cleaning 98% of a 15x13 room.
        We use a new subclass of the Robot class, TestRobot, that sequentially
cleans one
         square at a time.
```

Output:

```
64.0
```