Problem 1: Robot Class

For the `Robot` class, decide what fields you will use and decide how the following operations are to be performed:

- Initializing the object
- Accessing the robot's position
- Accessing the robot's direction
- Setting the robot's position
- Setting the robot's direction

Complete the `Robot` class by implementing its methods in `ps2.py`.

Note: When a `Robot` is initialized, it should clean the first tile it is initialized on. Generally the model these Robots will follow is that after a robot lands on a given tile, we will mark the entire tile as clean. This might not make sense if you're thinking about really large tiles, but as we make the size of the tiles smaller and smaller, this does actually become a pretty good approximation.

Although this problem has many parts, it should not take long once you have chosen how you wish to represent your data. For reasonable representations, *a majority of the methods will require only a couple of lines of code*.)

Note:
The `Robot` class is an *abstract* class, which means that we will never make an instance of it. Read up on the Python docs on abstract classes at this link and if you want more examples on abstract classes, follow this link. If you took edX course 6.00.1x already, you've seen an abstract class - the `Trigger` class from the final problem set!

In the final implementation of `Robot`, not all methods will be implemented. Not to worry -- its subclass(es) will implement the method `updatePositionAndClean()` (this is similar to the `evaluate` method of the `Trigger` class from 6.00.1x).

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

Test: 1 class creation

```
Although Robot is an abstract class, we create instances of it for
the purposes of testing your code's correctness.
robot = Robot(RectangularRoom(1,2), 1.0)
```

Output:

Test: 2 test getRobotPosition

```
robot = Robot(RectangularRoom(5,8), 1.0)
robot.getRobotPosition()
```

Output:

```
Test passed
```

Test: 3 test getRobotDirection

```
robot = Robot(RectangularRoom(5,8), 1.0)
robot.getRobotDirection()
```

Output:

```
Test passed
```

Test: 4 test setRobotPosition

```
robot = Robot(RectangularRoom(5,8), 1.0)
robot.getRobotPosition()
loop 10 times:
    * Generate random x, y values
    * Check if Position(x,y) is in the room
        * If so, robot.setRobotPosition(Position(x, y))
        * robot.getRobotPosition()
```

Output:

```
Random position 0: (1.00, 9.00)
Random position 1: (5.00, 9.00)
Random position 2: (4.00, 8.00)
Random position 3: (0.00, 2.00)
(0.00, 2.00)
Random position 4: (2.00, 8.00)
Random position 5: (1.00, 1.00)
(1.00, 1.00)
Random position 6: (0.00, 9.00)
Random position 7: (1.00, 2.00)
(1.00, 2.00)
Random position 8: (2.00, 5.00)
(2.00, 5.00)
Random position 9: (6.00, 9.00)
```

Test: 5 test setRobotDirection

```
robot = Robot(RectangularRoom(5,8), 1.0)
robot.getRobotDirection()
loop 10 times:
    * Generate random direction value
    * robot.setRobotDirection(randDirection)
    * robot.getRobotDirection()
```

Output:

```
Test passed
Random direction: 273
```

```
273
Random direction: 320
320
Random direction: 159
159
Random direction: 94
94
Random direction: 25
25
Random direction: 136
136
Random direction: 185
185
Random direction: 82
82
Random direction: 85
85
Random direction: 122
122
```

Test: 6 test updatePositionAndClean

```
The abstract class Robot should not implement updatePositionAndClean.
```

Output:

```
NotImplementedError successfully raised.
```