

# Python(H) Project 1: 机器人仿真

May 19, 2020

## 概述

设计并实现一个模拟机器人运动的程序。请不要更改任何给定的函数。

### 1. 注意

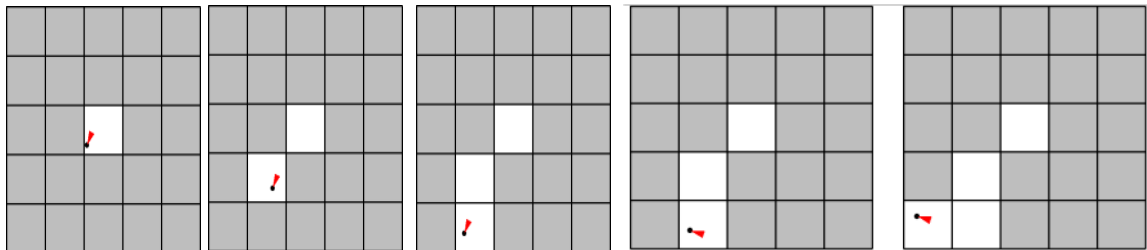
对违规行为将扣分(例如非描述性变量名和未注释代码)。

### 2. Using Python's Random Module

包括: `random.randint(a, b)` `random.random()` `random.seed(0)`

### 3. 仿真

机器人在地板上移动,清理它们经过的区域。下面是一个机器人在5x5平方米的房间移动的简化模型:机器人开始在房间里的某个随机位置。它的方向由从“北”开始以顺时针度测量的运动角度确定。它的位置左下角原点(0.0,0.0)。下图显示了机器人的位置(用黑点表示)和方向(用红色箭头表示)。



## Time

t=0: 机器人从位置(2.1、2.2)开始,角度为205度(从“北”开始顺时针测量)。它上面的瓷砖现在是干净的。

t=1 机器人已经朝它面对的方向移动了一个单元,移动到位置(1.7,1.3),清洁另一块瓷砖。

t=2 机器人向同一方向移动了一个单元(205度),移动到(1.2, 0.4)的位置,清洁另一块瓷砖。

T=3 机器人不可能在同一方向移动另一个单元而不撞到墙壁,所以它转向一个新的、随机的方向,287度。

T=4 机器人沿着它的新方向移动到位置(0.3,0.7),清洁另一块瓷砖

### 4. 仿真组件:

1. 房间: 方形的, 正方形瓷砖。开始时每个瓷砖都覆盖了一些污垢, 所有瓷砖上的污垢都是相同的。在Problem 1中实现抽象类RectangularRoom, 然后在Problem 2中实现子类EmptyRoom和FurnishedRoom。

2. 机器人: 房间里可以有多个机器人。Problem 1中实现抽象类Robot, 然后Problem 3和Problem 4中实现继承类StandardRobot和FaultyRobot。

### 5. 辅助代码

有两个额外的文件:pj1\_visualize.py和pj1\_verify\_movement27.py。这些Python文件包含用于测试代码和可视化机器人模拟的辅助代码。

## Problem 1: Implementing the RectangularRoom and Robot classes

请仔细阅读pj1.py: 仔细阅读每个函数的DocStrings, 了解它应该做什么和需要返回什么。

实现两个抽象类RectangularRoom和Robot。因为抽象类被实例化, 仅用作从它继承的其他类的模板。抽象类定义由其子类共享的方法: 这些方法可是在子类中实现。

在提供的框架代码中, 抽象类包含一些只能在子类中实现的方法。如果方法的注释说“不要更改”, 请不要更改它。

pj1.py中, 提供了类 Position 的完整实现。

### 类描述:

- 矩形房间-表示要清洁的空间，并记录已清洁的瓷砖。
- robot—存储机器人的位置、方向和清洁能力。
- position—表示x坐标和y坐标中的位置。x和y是满足 $0 \leq x < w$ 和 $0 \leq y < h$ 的浮点数

### RectangularRoom实现细节:

#### ● 表示:

需要记录机器人清理过地板的哪些部分。当机器人的位置位于某个特定瓷砖(砖)内的任何位置时，将认为整个瓷砖上的污垢会被机器人清扫后数量所减少。瓷砖上的污垢为0，就认为瓷砖是“干净的”。

使用整数的有序对(0,0)、(0,1)、...、(0,h-1)、(1,0)、(1,1)、...、(w-1, h-1)来引用瓷砖块。

瓷砖上的灰尘永远不会是负数。

#### ● 起始条件:

一开始，整层地板都是脏的。每个瓷砖应该以一个整数的污垢量开始，由dirt\_amount指定。

### 机器人实现细节:

#### ● 表示

每个机器人在房间里都有一个位置。使用position类的实例来表示位置。记住位置坐标是浮点数。机器人有运动的方向。使用一个满足 $0 \leq \text{direction} < 360$ 的浮动方向来表示该方向，该方向给出了一个以北向为单位的角度。机器人具有清洁能力，即每次清洁每块瓷砖上的污垢的能力。

#### ● 起始条件

每个机器人在房间里应该从一个随机的位置开始(提示:机器人的房间属性有一个可以使用的方法)

#### ● 运动策略

机器人根据它的运动策略移动，你将在update\_position\_and\_clean中实现。

注意，房间瓷砖使用整数(0,0)、(0,1)、...、(0,h-1)、(1,0)、(1,1)、...、(w-1, h-1)的有序对表示。但是机器人的位置被指定为浮点数(x, y)。小心在这两者之间转换!

如果在上面的任何地方发现模拟动力学的规范似乎不明确，那你就需要对程序/模型的行为做出合理的决定，并在代码中用文档记录该决定。

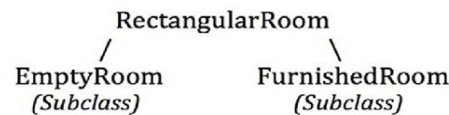
根据pj1.py中的规范实现RectangularRoom 和Robot 抽象类的方法。记住，这些类永远不会被实例化;我们只实例化它们的子类。

### 提示:

- 确保仔细考虑要使用哪种类型的数据来存储RectangularRoom类中有关地砖的信息。
- 大多数方法应该只需要一行代码。
- Robot类和RectangularRoom类是抽象类，这意味着永远不能创建它们的实例。相反，应该将实例化从抽象类继承的子类。
- 在这些抽象类的最终实现中，并不是所有的方法都会被实现，由子类们将实现它们。例如，Robot的子类将实现update\_position\_and\_clean方法。
- 记住，tile是用整数(0,0)、(0,1)的有序对来表示的，...，(0,h-1)，(1, 0)，(1, 1)，...，(w-1, h-1)。给定一个指定为浮动(x, y)的位置，如何确定机器人正在清理哪个块?
- 记住给机器人一个初始的随机位置和方向。机器人的位置应该是Position类的，并且应该是房间中有效的位置。注意，RectangularRoom 抽象类有一个get\_random\_position方法，它可能在这方面很有用。
- 建议使用math.floor(x)而不是int(x)，这样数字总是舍去，确保点在房间内

## Problem 2: Implementing EmptyRoom and FurnishedRoom

考虑增加房间的种类: 有家具的房间(FurnishedRoom)和没有家具的房间(EmptyRoom)。这些房间在它们自己的类中实现, 并且具有许多与RectangularRoom相同的方法。可通过继承来减少重复代码的数量, 将FurnishedRoom和EmptyRoom实现为RectangularRoom的子类, 如下图所示:



请考虑需要实现的方法在这两个类中有何不同, 以及如何使用已经在父类 RectangularRoom 中实现的方法。注意: 未能利用继承的优势将导致扣分。

此外, 要决定一个位置是否有效: 在有家具的房间中, 机器人不能在有家具的位置(在瓷砖上)。

在FurnishedRoom类中, 已经实现了add\_furniture\_to\_room方法, 为房间中添加了一件矩形家具。不需要调用这个方法: 这是测试代码所用, 不要改变这个方法。

请通过在pj1.py中实现它们的方法来完成EmptyRoom和FurnishedRoom类。

提示:

- ✓ 仔细阅读提供的代码, 了解家具房与空房间和矩形房间的区别。瓷砖是如何储存的?
- ✓ 请记住, tiles是(0,0)、(0,1)、...、(0,h-1)、(1,0)、(1,1)、...、(w-1, h-1)的有序对来表示的。但机器人的位置被指定为浮点数(x, y)。转换时使用math.floor(x)以确保位置总是在房间中。

## Problem 3: StandardRobot and Simulating a Timestep

**每个机器人还必须有一些代码来告诉它如何在房间中移动, 这些代码将在一个名为update\_and\_position\_and\_clean的方法中执行。**

通常应将机器人的所有方法放在一个类中。但本题考虑到具有可选移动策略的机器人, 它们将被实现为具有相同接口的不同类。这些类将有update\_and\_position\_and\_clean的不同实现, 但多与最初的Robot相同。应利用继承来减少重复代码的数量。

我们已经将Robot代码重构为两个类: 如上的抽象Robot类(包含一般的robot代码)和继承自它的StandardRobot类(其中包含它的移动策略)。

StandardRobot 的移动策略如下: 在每次步 (time-step) 中:

- 计算机器人以给定速度沿当前方向直线移动时的新位置。
- 如果这是一个有效的位置, 移动到那里, 机器人的能力清洁对应该位置瓷砖 (减少污垢)。如果该位置在房间内且没有家具, 那么该位置有效。不要担心机器人在新旧位置间的路径, 以及在这条路径上是否有家具。
- 否则, 将机器人转移到随机的新点。不要清洁当前的瓷砖或移动到其他瓷砖。

已提供的Position类的get\_new\_position方法, 在一个时钟滴答通过给定的角度和速度参数之后, 它计算并返回当前位置对象的新位置。请阅读此方法的docstrings以获取更多信息。

完成standardRobot的update\_position\_and\_clean方法, 以模拟机器人在单个time-step的运动(如上面的time-step动态中所述)

测试代码:

在解决Problem4前，检查 StandardRobot实现是否工作正常： StandardRobot的这一行代码前取消注释符号: `test_robot_movement(StandardRobot, EmptyRoom)`，这将测试机器人是否能在空房间里正确移动。当机器人能够正确移动，请确保注释掉`test_robot_movement`行。

测试将显示一个5 \* 5的房间(在EmptyRoom中实现)和一个机器人(在StandardRobot中实现)。最初，所有脏的瓷砖都标为黑色。当机器人访问每个瓷砖并根据其给定的容量对瓷砖进行清洁时，瓷砖的颜色从黑色变为灰色，再变为白色，白色表示完全清洁。

确保机器人在房间里移动时，每次机器人穿过房间时，瓷砖会变亮(从黑色到灰色、白色)。当机器人完成打扫整个房间时（或完成预定的比例），模拟就结束了。

确保机器人没有违反任何模拟规则（例如，机器人不应该移动到房间外的位置，如果它还必须选择一个新的方向，它不应该清洁瓷砖，等等）。

也应该测试一下机器人在FurnishedRoom里的移动是否正确，办法是取消下面这行的注释符号：`test_robot_movement (StandardRobot FurnishedRoom)`

要使 `update_position_and_clean`工作，不要更改它的实现。当完成测试时，请记住注释掉这一行。如果机器人在清洁时出现“偷工减料”的现象，也不要担心，只要它在每个time-step的最后位置永远不会在家具(红色)瓷砖上或房间外面。再次提醒：测试机器人是否正确移动时，请确保注释掉`test_robot_movement`行。

## Problem 4: Implementing FaultyRobot

设计一个模拟程序，以确定这对机器人打扫房间的时间有多大的影响。注意:每个time-step都可能存在缺陷。如果一个机器人在一个time-step出现故障，那么它在下一个time-step可能出现故障，也可能不出现故障。

编写一个新类FaultyRobot，它继承自Robot(就像继承standard Robot一样)，但是实现了一个新的移动策略。FaultyRobot应该有自己的`update_position_and_clean`实现

FaultyRobot的移动策略如下:

0. 检查这个time-step 机器人是否有故障。
1. 如果机器人出现故障，它不会清理当前正在使用的瓷砖，并且会随机更新其方向。
2. 如果机器人没有故障，就像对待StandardRobot: ——让它移动到一个新的位置，如果可以的话就进行清洁。如果它不能有效地移动到下一个位置，那么就改变它的方向。

在FaultyRobot中有个`gets_fault`方法，使用该方法来确定机器人是否出现故障。一开始机器人出现故障的概率是 $p = 0.15$ 。如standardRobot一样，`get_new_position`方法非常有用。

**测试：** 运行FaultyRobot，通过观察可视化，以确保它在做正确的事情。

```
test_robot_movement (FaultyRobot, EmptyRoom)
```

## Problem 5: Creating the Simulator

在这里中，将要编写代码:

1. 模拟机器人打扫房间直到房间的指定部分;和
2. 输出打扫房间平均需要多少time-step长。

编写完这段代码后，在Problem 6中，将对模拟的结果进行解释。

```
Implement run_simulation(num_robots, speed, capacity, width, height,
dirt_amount, min_coverage, num_trials, robot_type) according to its
specification. Use an EmptyRoom for this problem.
```

- **仿真初始条件:**

1. 每个机器人应该在房间里随机位置开始。
2. 每个房间开始时，每个瓷砖上都应该有相同数量的污垢，由dirt\_amount给出。当指定的房间瓷砖已清洁完(即，瓷砖上的污垢量为0)，仿真终止。

- **模拟动画:**

如果看模拟的可视化效果，类似于调用test\_robot\_movement时弹出的可视化效果，请参看本文档最后的说明。

- **模拟器代码:**

1. 模拟指定次数的机器人扫地清洗过程(num\_trials)。
2. 模拟机器人打扫房间，直到房间指定瓷砖被清洁(min\_coverage)。min\_coverage是房间中干净瓷砖与所有瓷砖的比例。
3. 记录每次试验达到min\_coverage所需的time-step数(时钟滴答声)。
4. 输出打扫房间所需的平均time-step数。

run\_simulation的前六个参数应该是不言自明的。目前应在StandardRobot中传递robot\_type参数:

```
avg = run_simulation(10, 1.0, 1, 15, 20, 5, 0.8, 30, StandardRobot)
```

然后，在run\_simulation中，无论何时希望实例化一个机器人，都应该使用robot\_type(...)而不是standardRobot(...)。(这能够很容易地调整模拟，以运行不同的机器人实现，Problem 6中将遇到这种情况)。可随意编写函数，同样有可用的Position的get\_new\_position方法。

提示:不要忘记在每次试验结束时重置必要的变量。

## Problem 6: Running the Simulator

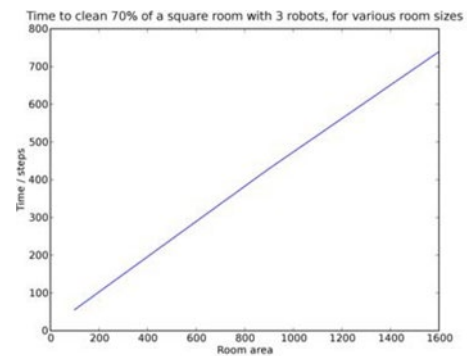
现在，用所编写模拟代码来回答一些关于机器人性能的问题。与此相关的是，将要使用pylab(又名matplotlib)的Python包。

对下面的问题，取消注释提供的函数调用(在project最后)，并使用pylab生成一个图，然后回答相应问题。

1. 检查pj1中的show\_plot\_compare\_strategies，它接受参数title、x\_label和y\_label。它输出一个图，比较两种类型的机器人在一个20x20的空房间中的性能，每个瓷砖上有3个单位的灰尘，80%的最小覆盖范围，不同数量的机器人的速度为1.0，清洁能力为1。取消对show\_plot\_compare\_strategies的注释，并回答question #1 (in the code)。根据计算机性能不同，画图时间可能需要几秒钟。

2. 在pj1中检查show\_plot\_room\_shape，它接受与show\_plot\_compare\_strategies相同的参数。这个图比较了两种类型的机器人打扫80%大小为10x30、20x15、25x12和50x6的空房间所需要的时间(注意，这些房间的面积相同)。取消对show\_plot\_room\_shape的调用的注释，并回答问题#2 (in the code)。

如右所示的图没有使用实际显示的图相同的坐标，只是作为pylab生成的图像的一个例子。如图所示，当机器人的数量固定时，打扫一个正方形房间的时间基本上与这个房间的面积成正比。



## Problem 7: Visualizing Robot Simulation

这里有一些代码来生成机器人打扫房间时的动画。这些动画能够直观地确定什么时候出现问题，有助于调试机器人模拟代码。

### Running the Visualization:

1. 在机器人模拟中，试验开始时，启动动画：

```
anim = pj1_visualize.RobotVisualization(num_robots,
width, height, is_furnished, delay)
```

这将打开一个新窗口来显示动画和绘制房间的图片。

当然，要传递适合试验的参数。`is_furnished`是一个布尔值，如果房间有家具，则为真，否则为假。`delay`是一个可选参数。

2. 然后，在每个time-step，机器人移动后，做以下事情来绘制一个新的动画帧：

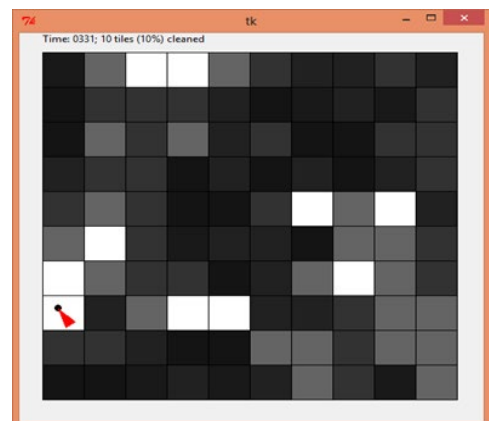
```
anim.update(room, robots)
```

其中`room`是一个矩形房间对象，`robots`是一个表示房间当前状态和房间中的机器人的Robot对象列表。

3. 试验结束后，调用以下方法：

```
anim.done()
```

The resulting animation will look like this:



提交Project:

1. Save your code in a single file, named `pj1.py`, 源代码文件`pj1.py`的开始部分:

```
#####
# Python(H) Project 1: Robot Simulator:
# Writer info.
#   Name:
#   Student-ID:
# Collaborators (Discussion):
# Date and Time:
#####
```

2. Self-Test以确保没有语法错误（有语法错误意味着Project没有完成，得分为0）。

测试 `run_simulation` (当然，可以不必按照这种测试过程，可以自行设计测试，确保代码能够正确运行)，以确保它仍然可以在`StandardRobot`和`FaultyRobot`类中正常工作。注意：重构过程中偶尔会破坏代码是很常见的，这也是测试非常重要的原因之一！



确保在运行问题5中的两个函数时生成图，并验证结果是否有意义。确保所有测试运行。你编写的代码必须要有注释。还要确保删除了任何未使用或注释掉的代码。

3. 撰写Project报告，使用PDF文档与代码打包提交。

报告主要内容：

开始部分与代码前部的注释内容相同，给出合作者的名字（如果有的话），以及各自设计和代码编写内容，贡献比例（如果是50:50），最高分不超过90。如果有讨论者，请给出讨论者的姓名和讨论的内容，通常与人讨论不会影响评分。

- （1） 如何运行程序（各种不同的方式）：一个非本课程的读者，能够据此完整运行程序的全部功能（例如：去掉哪些注释，运行得到何种功能）
- （2） 实现方法。
- （3） 解释各种运行结果。
- （4） 总结：如何进一步改进设计，使得机器人不那么蠢，或者说更聪明。