Python(H) Project 1： 扫地机器人仿真

6/9/2021

概述

设计视窗模式下运行的模拟扫地机器人运动的程序。本文档解释了已有代码，请不要更改任何给定的函数名称，但可以重写其代码。你为机器人增加的功能，请编写必要的代码。其运行代码不必照搬已经给出的那些代码，应该符合视窗软件的“面向按钮、菜单”的设计要求。

1. 注意

对违规行为将扣分（如非描述性变量名和未注释代码，建议使用中文注释。描述性变量是指变量名使用底划线连接英文单词，这些英文单词可以描述类、方法、变量的作用，也可以使用Camel命名法）。给定的代码可以实现了文档中介绍的功能，但可能有bugs或exceptions。

2. Using Python’s Random Module

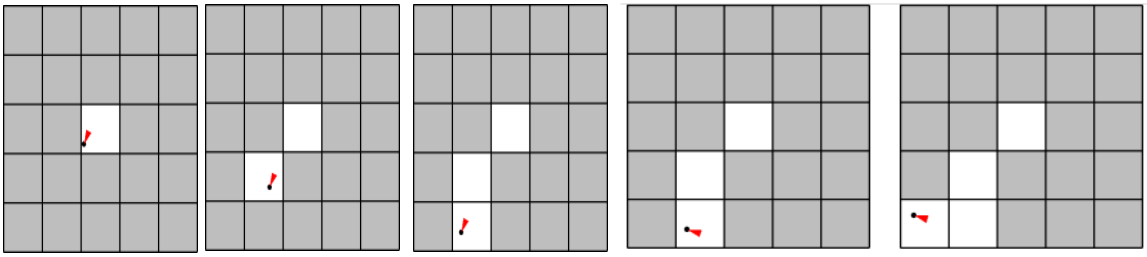
包括：`random.randint(a, b)` `random.random()` `random.seed(0)`。

如果设计其他运动(清扫)模式，则可能需要使用其他模块，或自行设计代码。

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平方米的房间里移动的简化模型：机器人开始在房间里的某个随机位置。图中，瓷片（Tile）为6×6。

机器人的方向由从“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, 你可以将下列文字中的相对坐标值用瓷片位置代替。但机器人经过一个瓷片，就意味着这个瓷片被清扫过一次。扫地机器人经过瓷片，如果瓷片是脏的，其灰尘值减1，如果瓷片是干净的，则灰尘值仍然为0)

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

如果你认为有更好的确定位置的方法，也可以重写相关代码。

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，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（num_robots>=1）。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。*注意机器人数目的意义。*

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。你可以使用、也可以无视它们。

Problem 1: Implementing the RectangularRoom and Robot classes

请仔细阅读pj1.py代码中每个函数的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（也许，给定的代码中并非如此）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。

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≤x < w和0≤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。dirt_amount>=0，0值意味着瓷砖是干净的。

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≤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算法似乎那么不明确，请自行决定是否改进算法。请将你改进的算法在Project报告中加以说明，代码中也要加上注释。

可以将RectangularRoom和Robot定义抽象类。如果你重新设定它们为抽象类，请记住这些类永远不会被实例化：只能实例化它们的子类。换言之，抽象类只能被用于继承，且继承抽象类的子类要有实现父类方法的代码。

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。

●大多数方法可能只需一行代码。

●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。

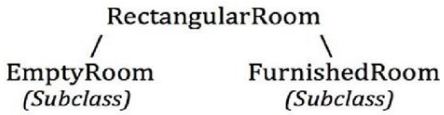
●记住，tile是用整数(0,0)、(0,1)的有序对来表示的，…，(0,h-1)，(1, 0)，(1, 1)，…，(w-1, h-1)。给定一个指定为浮动(x, y)的位置，如何确定机器人正在清理哪个块?

●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:



请考虑实现的方法在这两个类中有何不同，以及怎样使用了已在父类 RectangularRoom 中实现的方法。**如果给出的代码中没有利用继承，你应该改写源代码。否则可能导致扣分。**

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。 另外（不作硬性要求）如果在房间中增加其他形状的家具，可以自行设计。

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

- ✓ 仔细阅读代码，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

机器人还必须有一些代码来告诉它如何在房间中移动，代码将在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，如上述，你在研究给出的代码的基础上，决定是否重写代码。。

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)。

StandardRrobot 的移动策略可视为一种随机策略。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。

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（这是程序设计中一个有意义的工作）：

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(StandardRrobot, 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

这是一个模拟程序，以确定对机器人打扫房间的时间有多大的影响。注意:每个time-step都可能有缺陷。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。

FaultyRobot继承自Robot(像继承standardRobot一样)，但是实现了一个新的移动策略。FaultyRobot有自己的update_position_and_clean实现。注意：请思考FaultyRobot的意义。

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

代码完成如下两个任务：：

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。

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:**

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的实现。Problem 6中将遇到这种情况。你可以改写函数，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

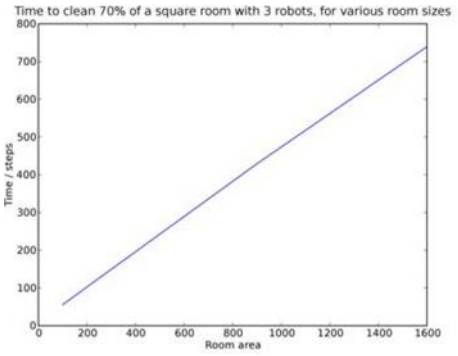
现在，用所编写模拟代码来回答一些关于机器人性能的问题。与此相关的是，将要使用pylab(又名matplotlib)的Python包。

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）。

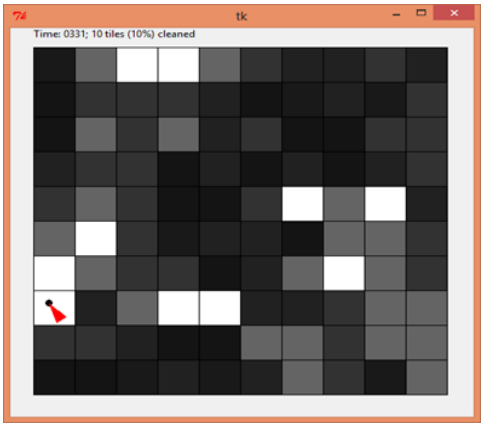


如右所示的图没有使用实际显示的图相同的坐标，只是作为pylab生成的图像的一个例子。如图所示，当机器人的数量固定时，打扫一个正方形房间的时间基本上与这个房间的面积成正比。

验收时需要展示上述模拟过程和结果。

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)
```

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(as fig of right):

4．设计要求（含基本设计和加分选项）

在理解上述功能(代码)的基础上，必须改进其设计，使机器人稍微“聪明”点。在你的设计里，至少应该有包括Start/Stop和另外一种运动模式：至少有两个按钮，注意随机清扫模式是默认的模式。

（1）改进程序，在界面上添加4个按钮

Start/Stop

三个运动模式的按钮：**PlanClean、EdgeClean、 RandomClean**

Start/Stop 开始/停止。停止可以是结束清扫并提示退出程序，也可以是暂停功能。

注意：Start为默认随机模式。第一次点击Start即为启动，其后如果有暂停设计，奇数次按下本按

钮，默认为暂停前的模式继续运行。

PlanClean　规划式清扫模式，机器人沿着瓷片直线前进，遇到边缘或者障碍后回转再次按照直线方向前进。当然，再次前进应该是邻近瓷片。自行设计规划模式的清扫路线（通常，机器人的位置可能是随机的）。本按钮除了确定清扫模式外，也可以设计为启动本模式，即按下本按钮，扫地机器人直接开始启动（无需按下Start），当然也可以设计为模式设置，按下本按钮后再按Start启动机器人。

EdgeClean　沿边清扫模式。这种模式仅仅是对房间的沿墙边进行专门清扫的模式。可以设计沿边的次数，也可以用边缘瓷片的清洁度决定是否介绍。如果整个房间尚未清扫完成，沿边清扫完成和应该自动转入规划模式。同PlanClean，本按钮除了确定清扫模式外，也可以设计为启动本模式，即按下本按钮，扫地机器人直接开始启动（无需按下Start），当然也可以设计为模式设置，按下本按钮后再按Start启动机器人。

RandomClean 即为本项目中代码要求的清扫模式。也是系统默认的模式。也就是说，如果没有选择规划模式或者沿边模式，点击Start按钮后自动进入随机模式。

（2）可以设计一个帮助菜单，介绍如何使用你设计的机器人。

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Problem 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。

有任何值得加分的设计，如按照上述要求添加了清扫模式、帮助菜单等功能的，可以得到更多的分数，当然最高分小于100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。

（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，使得机器人更聪明。