CS101 Homework #1

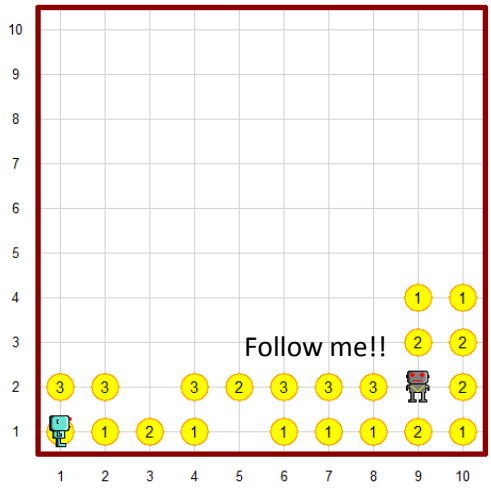
Follow, follow me

Due Date: Sun, Mar. 30, 2014 (Until 23:59)
Delayed Due Date: Wed, Apr. 2, 2014 (Until 23:59)

Please read the homework description carefully and make sure that your program meets all the requirements stated. The homework is an individual task. You can discuss the problem with your friends but you must not program together. **You will get F on entire course if your homework includes any plagiarism.**

Goal

Hubo wants **Ami** to follow the path of his(**Hubo's)** movement. Hubo tells Ami in which direction he moved by dropping beepers, so that Ami can find out the direction by counting the beepers and Ami can follow his trace.



Requirements

You can use built-in functions of python and functions for Hubo or Ami in the "cs1robots" library. You are required to complete the following functions whose functionality is guessable from their names. You

can finish your mission by calling the functions properly. You must complete the functions listed in the following box:

stepback_and_restore(), can_move(), move_one_step_or_stay(), follow_trace()

Lead & Follow

Hubo and **Ami** *start at the same position and with the same orientation*. **Hubo** leaves the sign of direction in order for **Ami** to follow him. In our homework, **Hubo** uses "the number of beepers" as a sign to tell **Ami** to which direction he moved. Hubo's signs are as follows: one beeper means to take a left turn and move one step forward, two beepers mean to take a move forward, and three beepers mean to take a right turn and move one step forward. Ami can identify each step by counting beepers. You have to implement the tracing function **follow_trace()** by taking the blue-arrow steps as shown in the example. While following the trace, Ami must pick all beepers to clean up the trace.

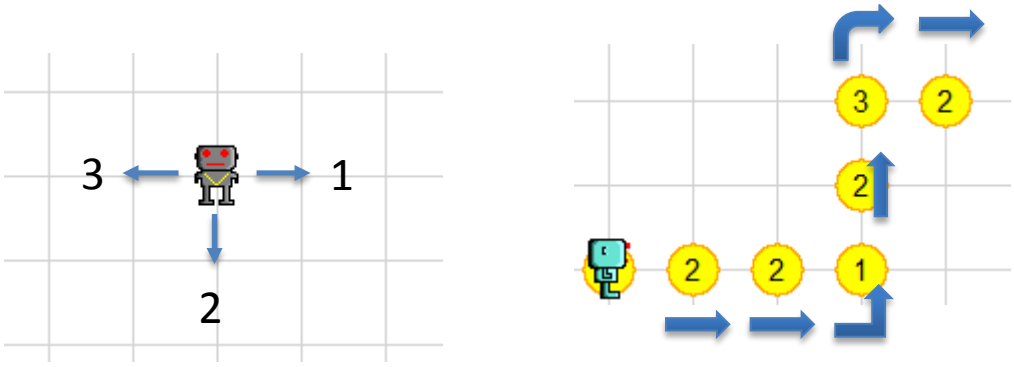


Figure 1. Hubo moves by randomly chosen number and Ami traces Hubo's movement

First**,** write the function **move_one_step_or_stay()** that makes a robot move only *one* or *zero step*. In this function, a robot moves in the *randomly chosen direction*. For random integer generation between 1 and 3 inclusive, you have to import **random** from python in-built library and use **randint()** function (http://docs.python.org/2/library/random.html). You can read **random.randint(a, b)** return a random integer N such that a <= N <= b.

Sample **randint** Code	

```
from random import *
a = randint(1,3)        # randomly generated 1≤a≤3.
print a                 # will print 1,2, or 3.
```

When moving to the chosen direction, **a robot** leaves a sign for the follower **Ami** to find out the direction to follow. If a randomly chosen direction is blocked by wall, the robot does not move but simply stays.

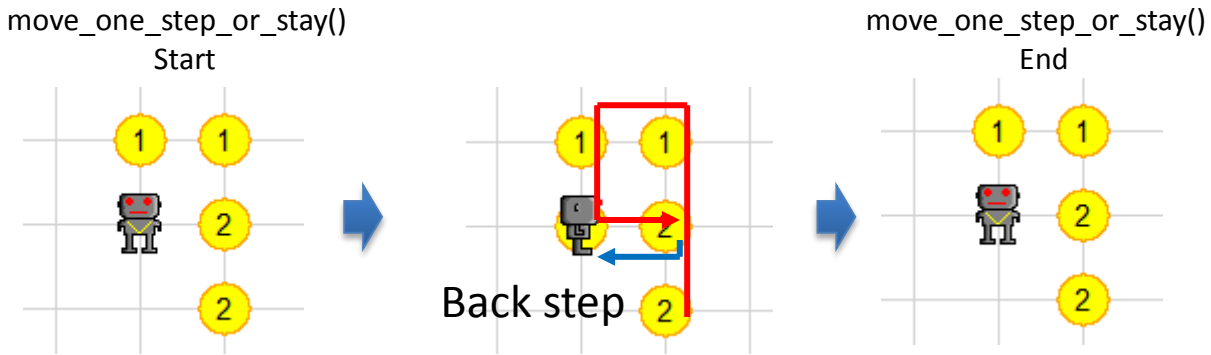


Figure 2. A trace overlap case, so there is no movement and stay original heading.

Here is *important constraint; trace line must not be overlapped*. So, after move, you have to check the new position was visited before. Visited places can be checked by testing beepers dropped. If there are beepers at the new position, you have to make the robot come back to the previous position; that is, ***at the same position and with the same orientation***. This role is handled by the function **stepback_and_restore()**. In that case, you need to count the number of beepers at the previous position to find out the original orientation (heading). In Figure 2 case, **Hubo** randomly chooses the number 1 and turn left and go. But there are beepers on the next place. So **Hubo** must come back and restore the previous orientation to south before next **move_one_step_or_stay()** function is executed.

Second, write a Boolean function **can_move()** to check whether a robot can move at least in one of the three directions. If all directions are blocked by either wall or trace, Hubo stops moving. Then Ami will start to follow Hubo's trace. This function checks three directions whether a robot can move forward; either left, front, or right. If all directions are blocked, **can_move()** function returns **False.** If at least one of the three directions is open, it returns **True**. Then you can call **move_one_step_or_stay()** function while **can_move()** function returns **True.** If **Hubo** can't move anymore, then **Hubo** stop.

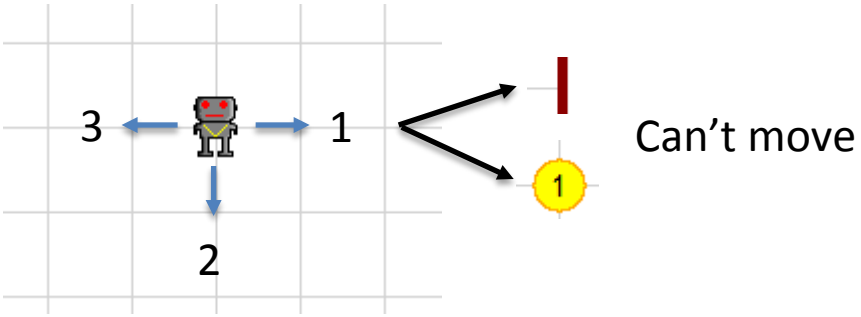


Figure 3. can_move() function check three directions

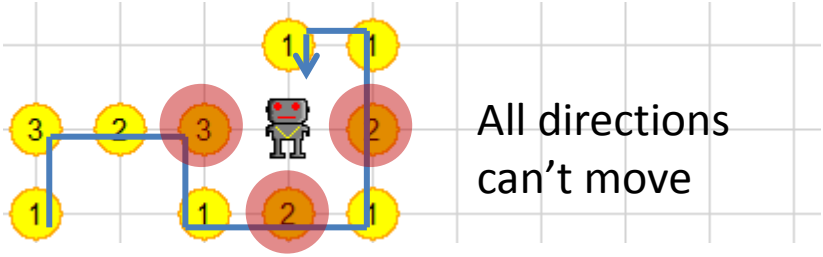
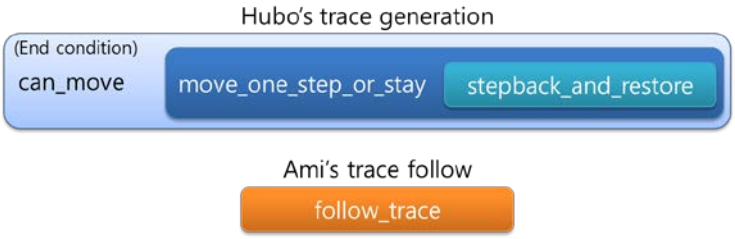


Figure 4. An example of all(three) directions are blocked by beeper.

Refer to the sample code template.

Sample
return Code
	def IsCS101Good():
	  if ReallyLkeCS101 : # Condition you want.
	  return True
	  else:
	  return False

Third, write the function **follow_trace()** for Ami. A robot picks and counts the beepers, and moves to the proper direction *while a robot is on beepers*. The **follow_trace()** function is very simple because it needs only to count beepers and move one step. Finally, you can implement trace generation code by repeating **move_one_step_or_stay()** while **can_move()** return True and it would like below:



Evaluation

Your Program will be tested with a world you generate. Please try many tests with an default world(10x10) and other worlds from the lab sessions. Test your program with your own worlds with 5 to 10 sizes of streets and avenues in order to verify your program.

Submission

You need to submit followings:

- The file "yourid.py" : the program that solves the problem (e.g.) 20141234.py
- The report "yourid.doc or docx or pdf" (e.g.) 20141234.doc, 20141234.docx, 20141234.pdf

You have to describe the way to solve the problem and show your testing images and the result (for example, before Ami starts and after Ami stops showing the trace with ami.set_trace()). If you have defined a function, you need to explain what this function does and so on. You may include two images shown above in the report. One of the images would be screenshot of Hubo's dropped beeper which trace generation result and other would be screenshot of Ami which followed result.

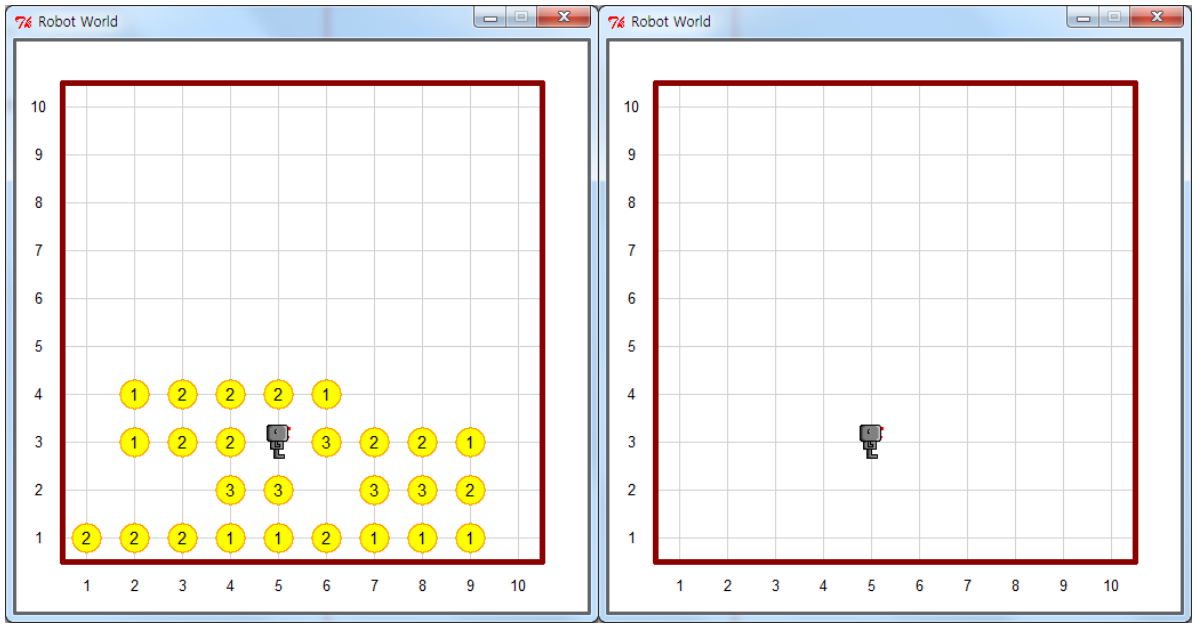


Figure 5. Hubo's trace generation result and Ami's following result

You must archive those files into "yourid.zip" and submit the archived file to the webpage for homework submission. (e.g.) 20141234.zip

If you don’t follow submission policy, you will get penalty.

Delayed homework will get 5% penalty per day from March 31 to April 2. After April 2, the server will be closed on that day and no more submission will be accepted. Please do it on time.