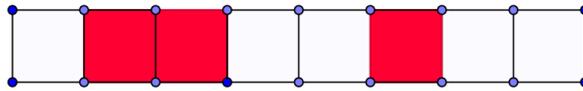


As you all know, I am taking a course to learn how to program a self driving car. Today we are going to learn a bit more about the math behind how a computer could do this on its own.

Last week, I showed you this problem:



The cells represent a simple map of possible locations of a robot in a one dimensional world. The robot is guaranteed to be located in one of the cells. The red cells represent places on the map where the robot knows objects are located.

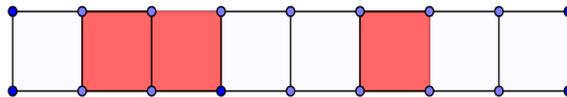
The robot can use a sensor to determine the color of the cell it is in. Thus, it will either read 'red' or 'white'.

The robot can also move from cell to cell. If the robot is all the way to the right, and moves one space to the right, it will end up back in the left most cell. This is called a **cyclic** world.

Suppose I tell you the following information:

- The robot is located in one of the eight cells.
- The sensor sees 'white'.
- The robot moves right one cell.
- The robot sees 'red'.

Where could the robot be located after these four steps? (Mark the possible locations.)

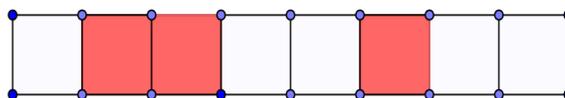


What is the probability that the robot is located in each cell? Write the probability inside the box.

We can't be absolutely sure in this case where the robot is located. What if the robot instead does the following:

- The sensor sees 'red'.
- The robot moves right one cell.
- The robot sees 'red'.

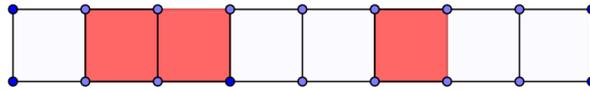
Mark the possible locations of the robot after these steps, and write the probability that the robot is located inside the cell, in the cell.



Notice that we now have 100% certainty in where the robot is located. This is fairly simple for us as humans to reason out. How do you teach a computer to do this?

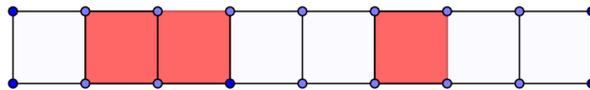
This is what we are going to investigate today.

Let's start by ignoring the movement of the robot, and just look at what the robot can learn about its environment from its sensor reading.



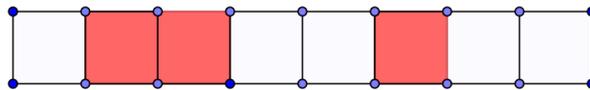
We will assume that the robot knows nothing about its environment at the start.

Problem 1. If the robot doesn't read its sensor at all, where might it be located? Fill in the boxes with the probability it is located in each cell.

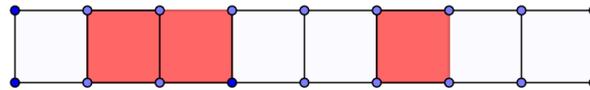


Problem 2. In each of the cases below, fill in the probability that the robot is located in each cell if the sensor reads the given color:

The sensor reads 'red':



The sensor reads 'white':



This way of seeing how probability is divided into different locations is called a **probability distribution**.

We want to come up with an automatic way for a computer program to generate the probability distribution for the cells above automatically.

For programming purposes, we are going to use Python. Before you go crazy - we're keeping it real easy to get programming this time. Go to <http://repl.it/> and select 'Python' as the language. Once it has finished loading, type the following in the left panel:

```
print "See how easy python can be when it runs inside the browser?"
```

Press  to run your code. Simple, right?

To work on the robot problem, we first want to make a program that generates the probability distribution in Problem 1.

Paste the following code into the browser:

```
p = []
n = 5
for i in range(n):
    p.append(1./n)
print(p)
```

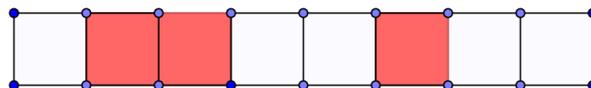
This program generates a probability distribution for a world with five cells.

Problem 4. Modify the program above to make it generate the same probability distribution as the one you generated in Problem 1. Have it also print out the text "This distribution sure is uniform. Uniformly BORING. Can we get some objects in this world?"

The next step is to teach the computer the map of the world it knows. To do this, we want to show the computer that certain cells are more important than the others. A good operation for this is multiplication.

We want to take the distribution from Problems 1 and 4 and multiply the probabilities in the red cells by 1 to show that they are a possible location of the robot. The white cells should be multiplied by 0 since they could not be the location of the robot.

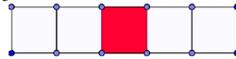
Problem 5. Multiply the red cell probabilities from Problem 1 by 1, and multiply the others by zero. Write the resulting probabilities in the cells below:



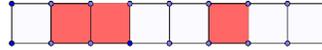
What is the total probability for the robot being in the eight cells?

Do you see any problem with this total probability?

Problem 6. The code below uses the matrix 'sense' to show where the red cells are located in the one dimensional world. It is currently set to show the following world:



Change this matrix (and the value of n below it) to match the **eight** cell world we are trying to model.



You must also fill in the multiplication values for when the robot senses 'red' or 'white'.

```
p = []
sense = [0,0,1,0,0]
n = 5
for i in range(n):
    p.append(1./n)
print(p)

p_multiply = 0
for i in range(n):
    if(sense[i]==1): #This line checks if the cell is 'colored' 1 (red).
        p_multiply = # What do we multiply by if it sees red? Enter the value .
    else:
        p_multiply = # What should we multiply by if the cell is white?
    p[i]=p[i]*p_multiply

print(p)
```

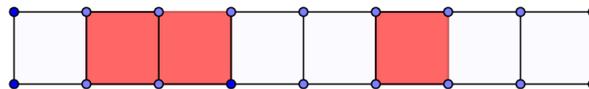


Press  to run your code. If the button doesn't appear, it means there is a mistake somewhere.

Your result should match your answer to Problem 5.

The fact that the probabilities do not add up to 1 shows that there is a problem with the current calculation. In order to make the sum equal 1, we need to divide by the sum of the numbers in all of the cells. This is the total probability you found in Problem 5.

Problem 7. Take your answers from Problem 5 and divide them by the total probability for all cells. Write the resulting probability in each cell below.



Problem 8. Cut and paste the following at the bottom of your program from Problem 6 and run the program. This should give you the same total value you calculated in Problem 5.

```
sum_p = sum(p)
print sum_p
```

Problem 9: Add the following to the bottom of your program:

```
for i in range(n):
    p[i] = p[i]/sum_p
print p
```

The full program will divide each cell of the probability distribution `p` that you created by the total probability you calculated.

You should now have this full program:

```
p = []
sense = [0,1,1,0,0,1,0,0]
n = 8
measurement = 1
for i in range(n):
    p.append(1./n)
#print(p)

p_multiply = 0
for i in range(n):
    if(sense[i]==measurement): #This line sees if the cell is 'colored' 1 (red).
        p_multiply = 1      # What do we multiply by if it sees red? Enter the
value .
    else:
        p_multiply = 0      # What should we multiply by to keep the number the
same if it                # sees white?
        p[i]=p[i]*p_multiply

#print(p)

sum_p = sum(p)
#print sum_p

for i in range(n):
    p[i] = p[i]/sum_p
print p
```

Paste the above code into the browser to make sure you are at the right point with your program.

The last step is to introduce an option for the robot to actually sense either 'red' or 'white'. This will automatically generate the probability distribution you made yourself way back in Problem 2. We can do this by introducing a line for the sensor reading.

Problem 10. Paste `measurement = 1` underneath the line with '`n = 8`'. This shows that the robot sees red.

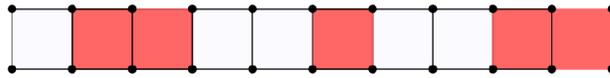
Change line 11 to read `if(sense[i]==measurement):` and leave the rest of the code the same.

Run the program. How does the output compare to your result in Problem 2?

Problem 11. Modify the program to give the probability distribution you had in problem 2 for if the sensor reads 'white'.

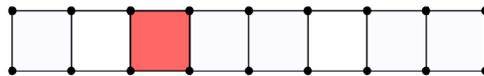
Congratulations! You've now calculated a probability distribution automatically using Python!

Problem 12. Modify the program to determine the probability distribution for the 1D world below if the robot reads red.



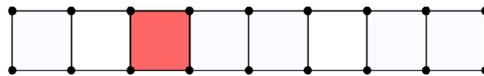
If the robot has a good idea where it is, it needs to also know how to figure out where it is if it moves.

Suppose the robot is in the following world:



The robot sensor reads red. What does this mean?

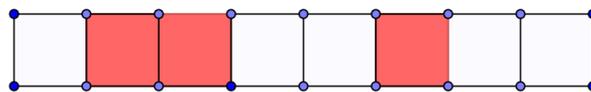
Problem 13. The robot sensor reads red and then moves right two cells in the world below.



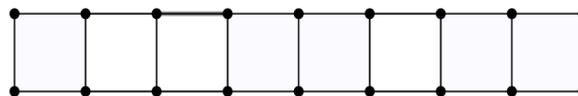
Where is it now located?



Problem 14. Suppose the robot sensor reads red, and then moves right two cells in this new world.



Where could it be located now?

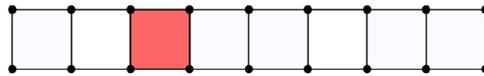


You have hopefully noticed that a move to the right of two units causes the possible locations for the robot to move right by two units.

In other words, if we know a probability distribution for the robot, and want to know what it will be after a move of 2 units to the right, we shift the entire distribution to the right 2 units.

We want the program to also be able to handle this sort of shift. This shouldn't be too hard, but there are a couple of tricks we need to be able to handle.

Problem 15. The robot sensor reads red and then moves right **seven** cells in the world below. (Remember that the world is cyclic.)



Where is it now located?

