Ardya Dipta Nandaviri Giri
agiri@andrew.cmu.edu

16662 Robot Autonomy
Homework 0

1. Python Tutorial:

```
#!/usr/bin/env python
import numpy as np

def print_list(l):
    print l

def sort_manual(shops):
    shops_sorted = []
    source = shops.values()
    for i in range(len(source)):
        maxi = max(source[i:]) #find maximum element
        max_index = source[i:].index(maxi) #find index of maximum element
        source[i + max_index] = source[i]  #replace element at max_index with first element
        source[i] = maxi                   #replace first element with max element

    #print source
    shops_sorted_notlist = [] #making a variable for temporary.
    for i in range(len(source)):
        for name, number in shops.iteritems():
            if number == source[i]:
                shops_sorted_notlist += [(name, number)]

    #making the list of list with key and value in the string type as seen in the question paper
    for i in range(len(source)):
        shops_sorted += [((shops_sorted_notlist[i][0]), str(shops_sorted_notlist[i][1]))]

    #print shops
    print 'Manual sorting manual result: '
    print_list(shops_sorted)

def sort_python(shops):

    shops_sorted = []
    shops_sorted_notlist = []
    shops_sorted_notlist = sorted(shops.items(), key=lambda x:x[1], reverse = True)

    #making the list of list with key and value in the string type as seen in the question paper
    for i in range(len(shops.values())):
        shops_sorted += [((shops_sorted_notlist[i][0]), str(shops_sorted_notlist[i][1]))]
```

```
    print 'Python sorting result: '
    print_list(shops_sorted)

def sort_numpy(shops):

    shops_sorted = []
    shops_sorted_notlist = []
    dtype = [('road','S20'),('no', float)]
    values = shops.items()
    a = np.array(values, dtype = dtype)
    a_sort = np.sort(a, order = 'no')
    shops_sorted_notlist  = a_sort[::-1]

    #making the list of list with key and value in the string type as seen in the question paper
    for i in range(len(shops.values())):
        shops_sorted += [((shops_sorted_notlist[i][0]), str(shops_sorted_notlist[i][1]))]

    #shops_sorted = np.sort(np.array(shops), order=)

    # TODO: Here implement sorting using numpys build-in sorting function

    print 'Numpy sorting result: '
    print_list(shops_sorted)

def main():

    shops = {}
    shops['21st Street'] = 0.9
    shops['Voluto'] = 0.6
    shops['Coffee Tree'] = 0.45
    shops['Tazza D\' Oro'] = 0.75
    shops['Espresso a Mano'] = 0.95
    shops['Crazy Mocha'] = 0.35
    shops['Commonplace'] = 0.5

    sort_manual(shops)
    sort_python(shops)
    sort_numpy(shops)



if __name__ == "__main__":
    main()
```

2. Openrave Tutorial

2.1. Move Straight.

I used the methodology that I have learned in MMC (Manipulation and Mobility Control) course. Variable 'a' is the position matrix of the robot, hence I used GetTransform to get the position matrix of the robot. And then put the input 'dist' as the distance of translation in the matrix 'b', then do a dot product to form the final matrix position of the robot.

```
def move_straight(self, dist):
  #TODO Fill in, remove sleepls
  with self.env:
    a = self.robot.GetTransform() #get the first position of the robot
    b = np.array([ [1, 0, 0, dist], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1] ]) # transformation matrix
    t = np.dot(a, b)
    self.robot.SetTransform(t)
```

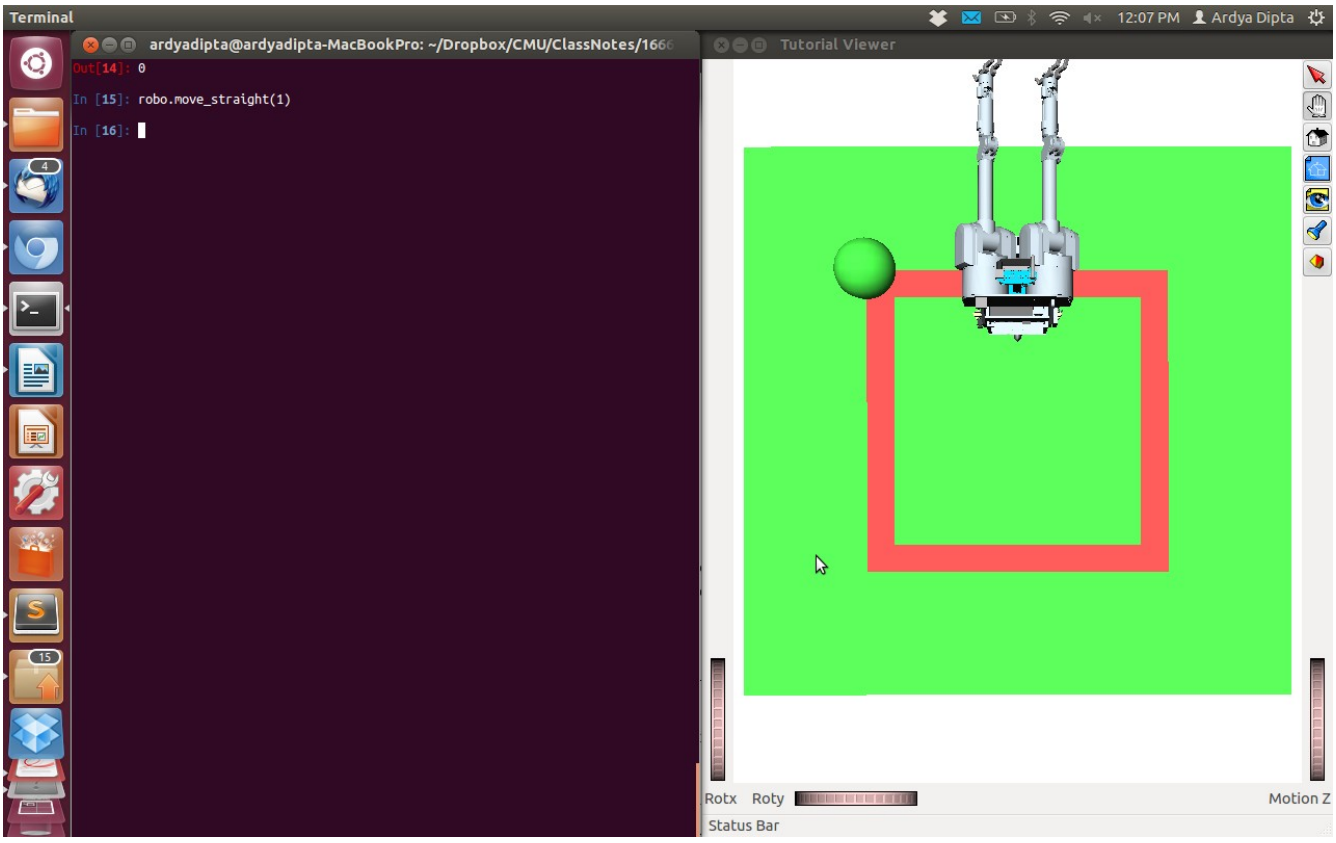


figure 1. Robot moves straight by distance 1

2.2. Rotate_by

The same with move_straight, the only difference is the transformation matrix for the rotation, which contains rotation operations.

```
def rotate_by(self, ang):
    with self.env:
      a = self.robot.GetTransform() #get the first position of the robot
      b = np.array([ [math.cos(ang), -math.sin(ang), 0, 0], [math.sin(ang), math.cos(ang), 0, 0], [0, 0, 1, 0], [0, 0, 0, 1.0] ])
      t = np.dot(a, b)
      self.robot.SetTransform(t)
```

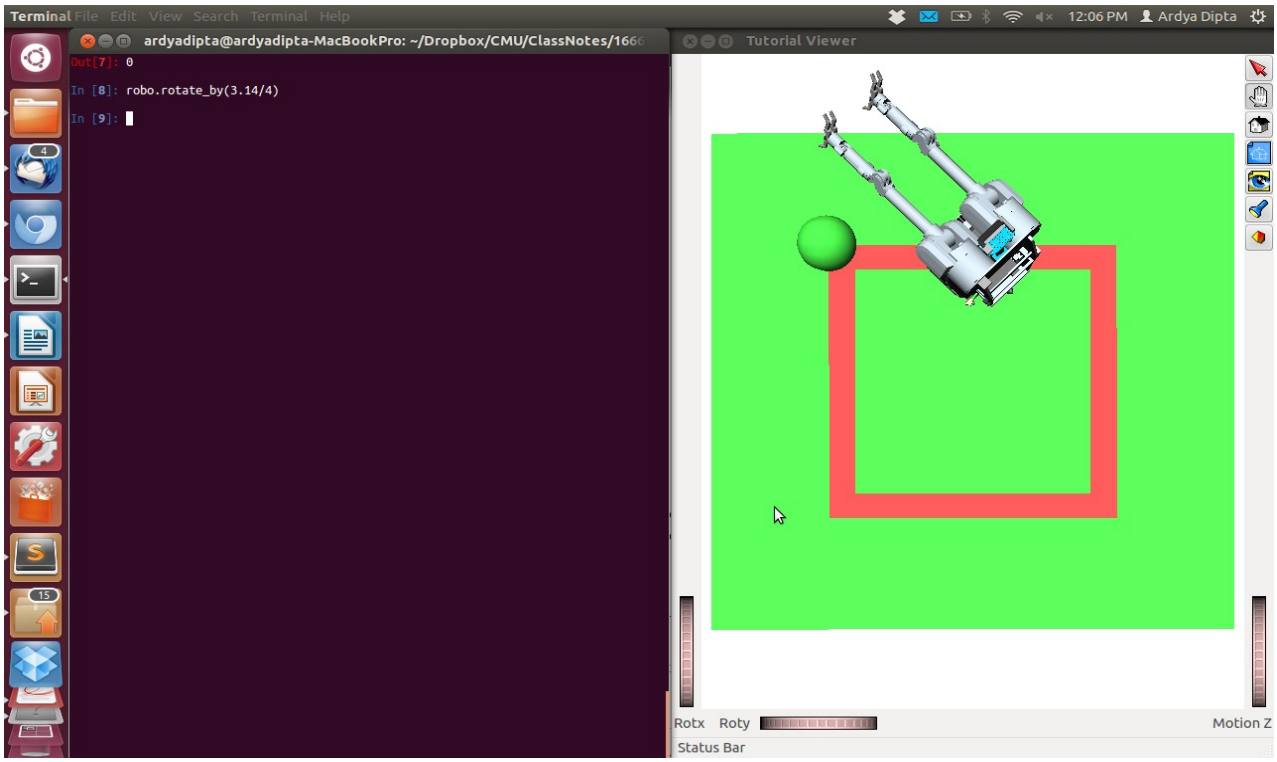


Figure 2. Robot Rotates by 3.14/4 rad (after move straight from the previous)

2.3. Go Around Square

Hard coded using move_straight and rotate_by. The screen shot is taken 4 times, so I ran the program 4 times because the robot moves too quickly.

```
def go_around_square(self):
  #TODO Fill in
  robo.move_straight(1)
  robo.rotate_by(3.14/2)
  robo.move_straight(1)
  robo.rotate_by(3.14/2)
  robo.rotate_by(3.14/4)
  time.sleep(1)

  robo.rotate_by(-3.14/4)
  robo.move_straight(2)
  robo.rotate_by(3*3.14/4)
  time.sleep(1)

  robo.rotate_by(-3.14/4)
  robo.move_straight(2)
  robo.rotate_by(3*3.14/4)
  time.sleep(1)

  robo.rotate_by(-3.14/4)
  robo.move_straight(2)
  robo.rotate_by(3*3.14/4)
  time.sleep(1)

  robo.rotate_by(-3.14/4)
  robo.move_straight(1)
  robo.rotate_by(3.14/2)
  robo.move_straight(1)
  robo.rotate_by(3.14)
  time.sleep(1)
  # set the robot back to the initialize position after
  with self.env:
    self.robot.SetTransform(np.identity(4));
```

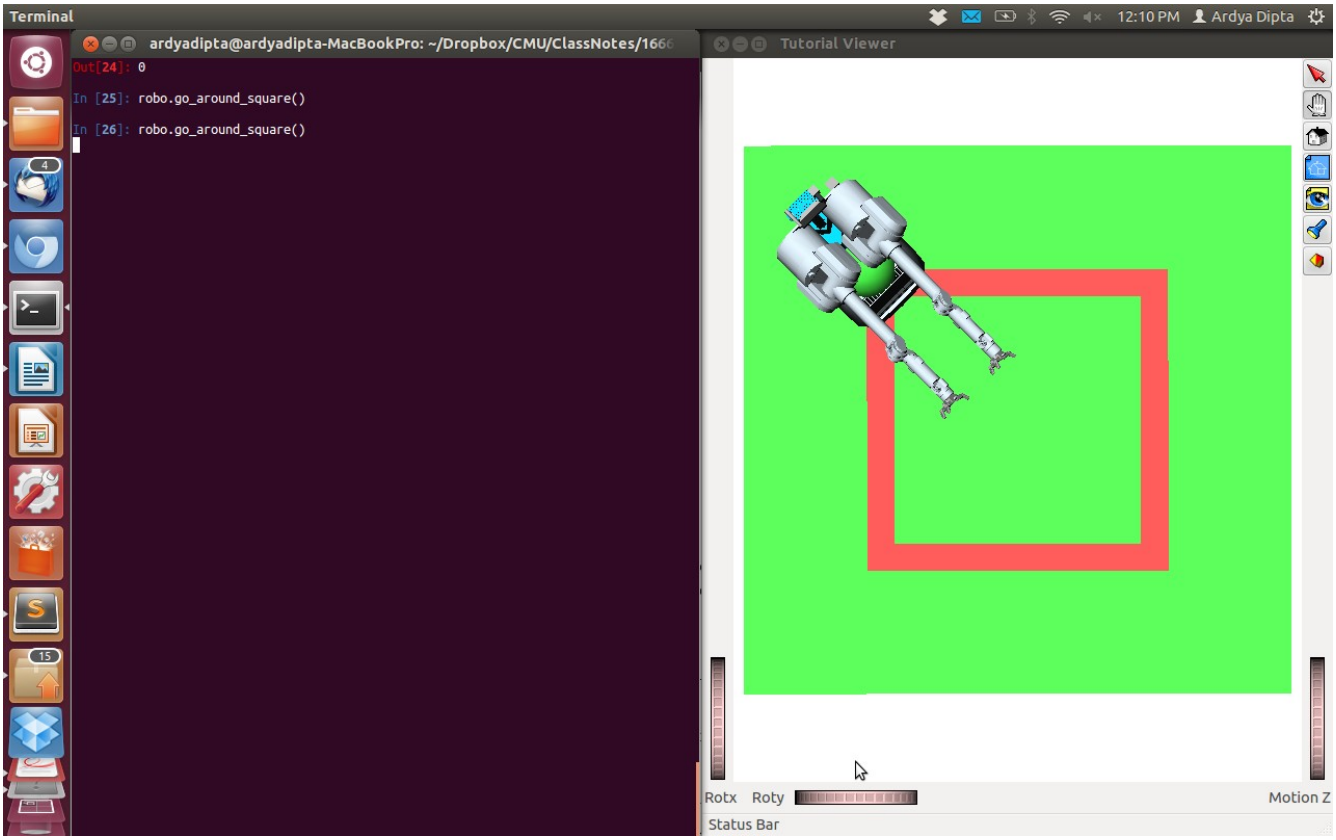


Figure 3. Robot goes to first corner

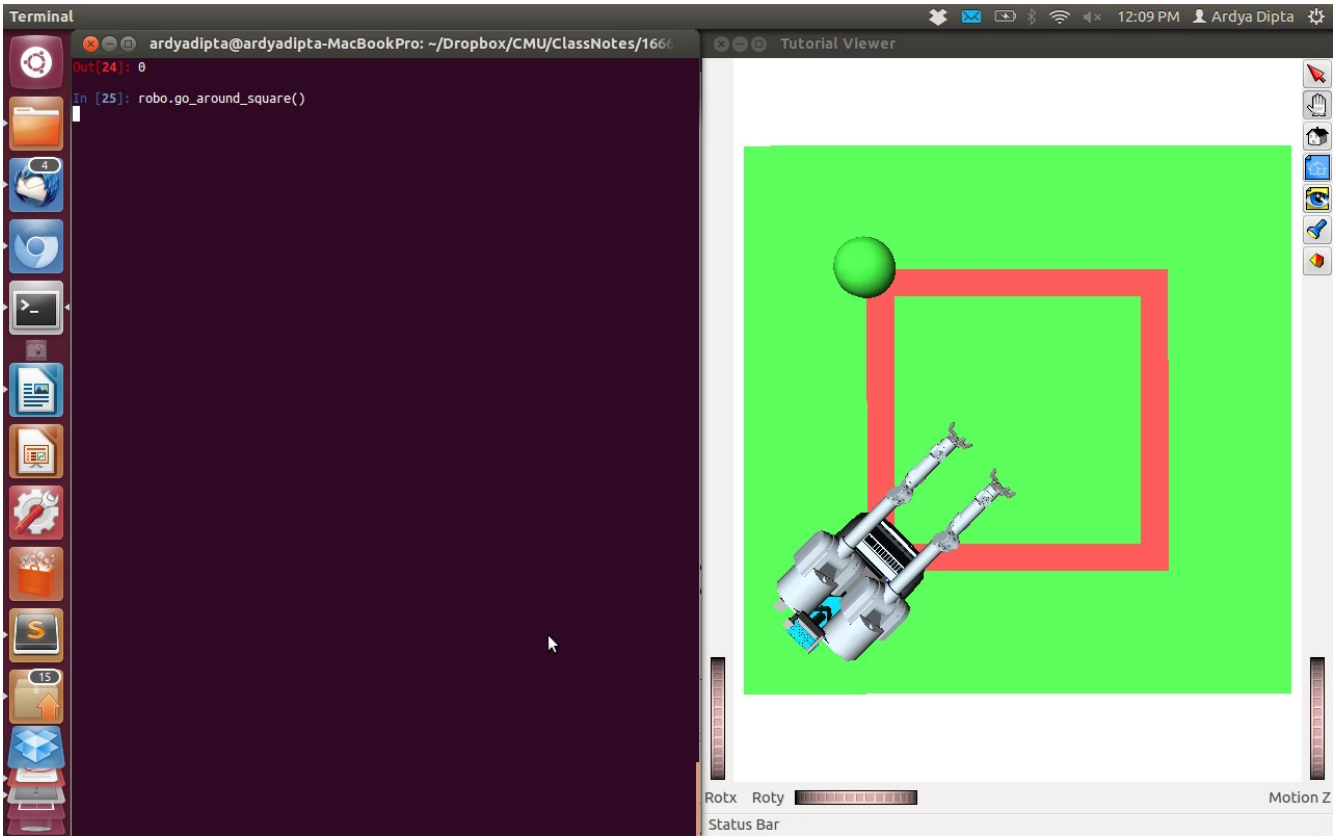


Figure 4. Robot goes to the second corner

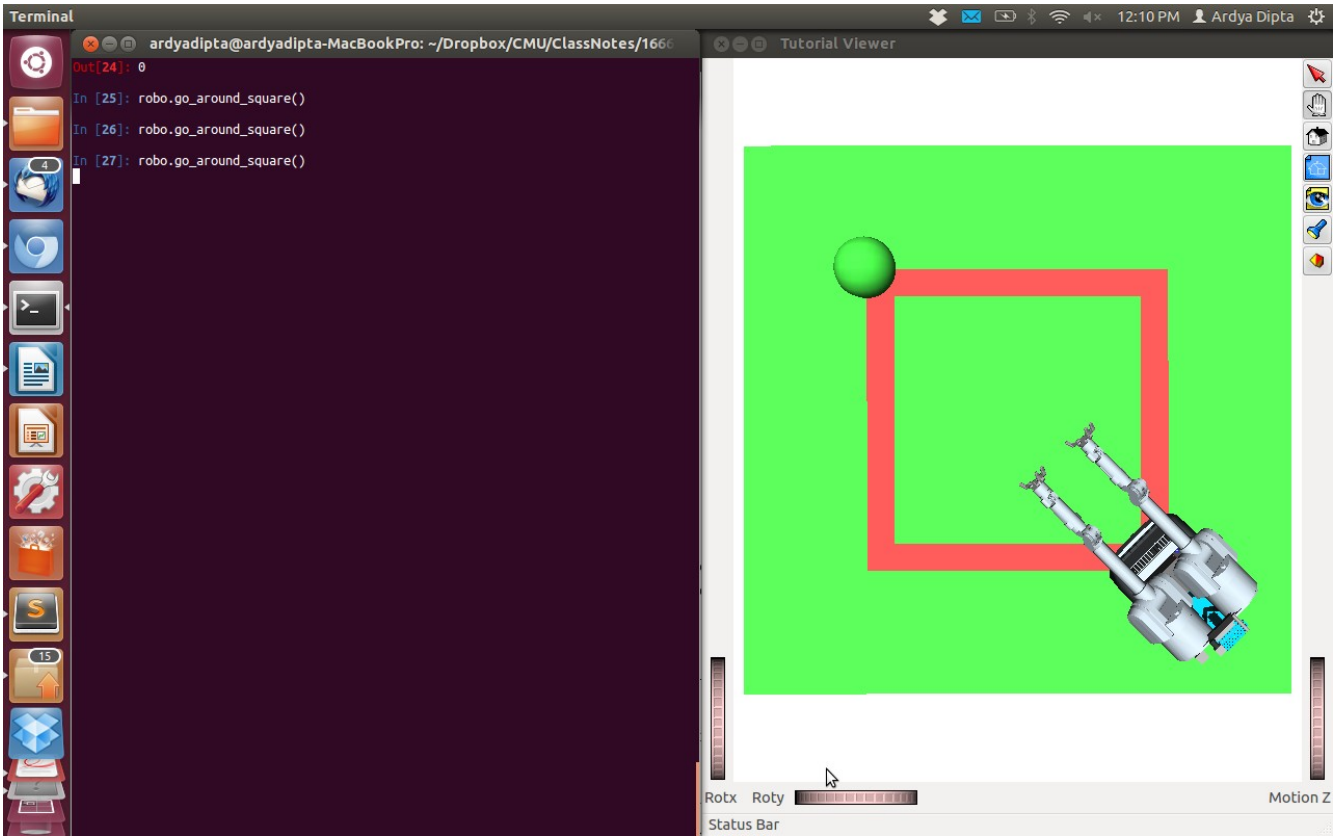


Figure 5. Robot goes to the third corner

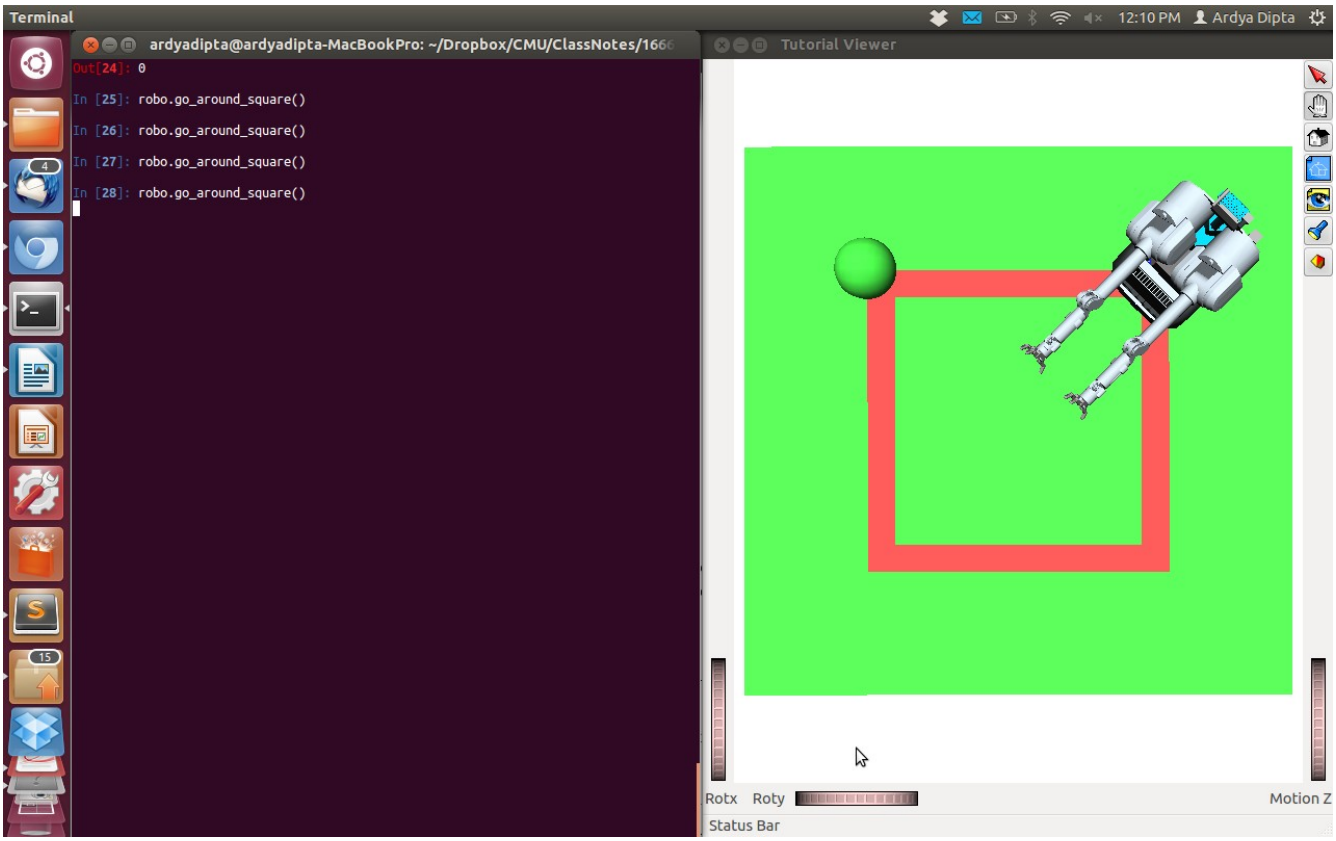


Figure 6. Robot goes to the fourth corner

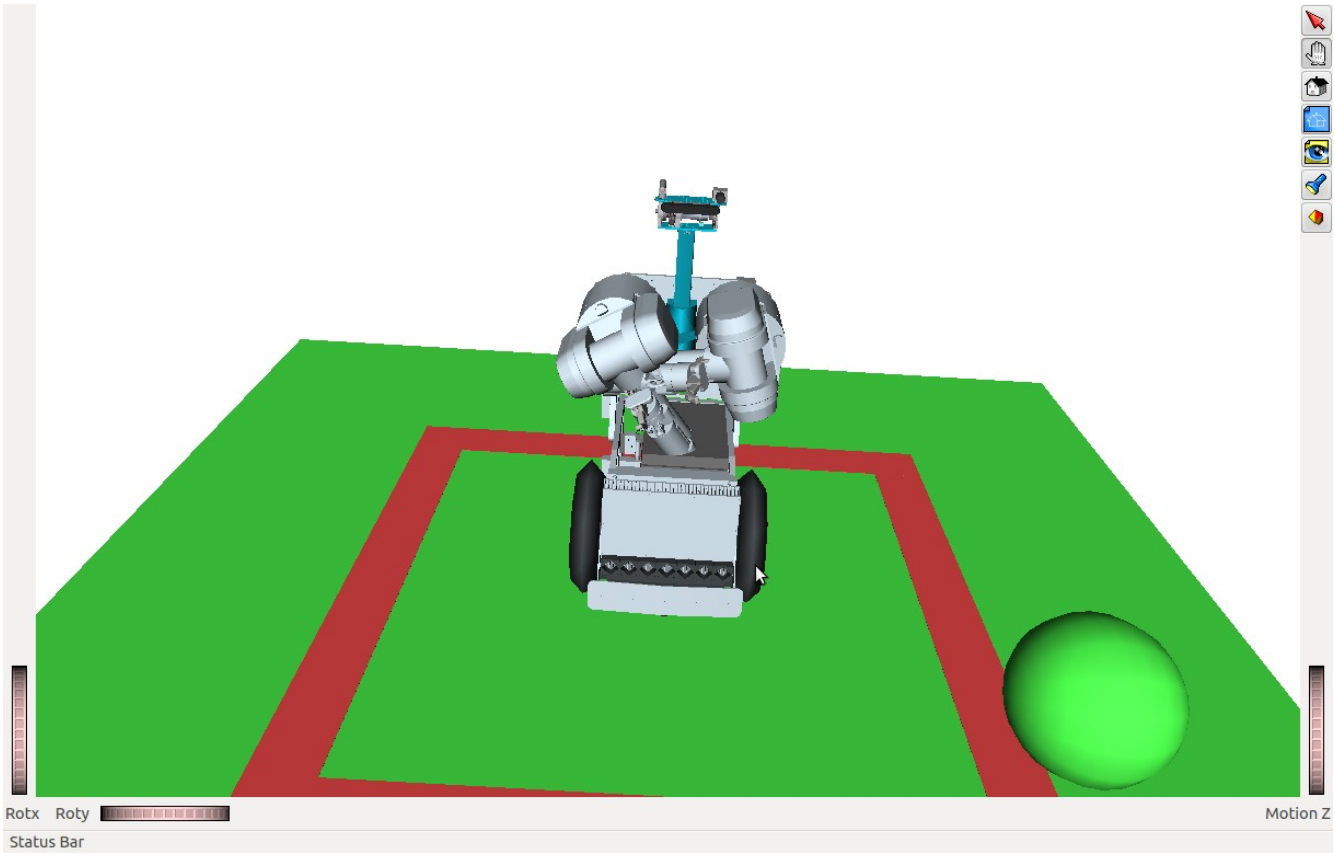
3. Specify the index ranges:

Right Arm : 0-3
Right Hand: 4-10
Left Arm: 11-14
Left Hand : 15-21
Head: 22-23

```
def figure_out_DOFS(self):
    #TODO Fill in, remove sleep
    with self.env:
      i = 0
      print "Right Arm: "
      while i<=3:
        print str(i), " ", robo.robot.GetJointFromDOFIndex(i).GetName()
        i += 1

      i = 4
      print "Right Hand: "
      while i<=10:
        print str(i), " ", robo.robot.GetJointFromDOFIndex(i).GetName()
        i += 1
      i = 11
      print "Left Arm: "
      while i<=14:
        print str(i), " ", robo.robot.GetJointFromDOFIndex(i).GetName()
        i += 1

      i = 15
      print "Left Hand: "
      while i<=21:
        print str(i), " ", robo.robot.GetJointFromDOFIndex(i).GetName()
        i += 1

      i = 22
      print "Head: "
      while i<=23:
        print str(i), " ", robo.robot.GetJointFromDOFIndex(i).GetName()
        i += 1
```

```
In [33]: robo.figure_out_DOFS()
Right Arm:
0   R_Shoulder_Yaw
1   R_Shoulder_Pitch
2   R_Shoulder_Roll
3   R_Elbow
Right Hand:
4   R_Wrist_Yaw
5   R_Wrist_Pitch
6   R_Wrist_Roll
7   RJF1
8   RJF2
9   RJF3
10   RJF4
Left Arm:
11   L_Shoulder_Yaw
12   L_Shoulder_Pitch
13   L_Shoulder_Roll
14   L_Elbow
Left Hand:
15   L_Wrist_Yaw
16   L_Wrist_Pitch
17   L_Wrist_Roll
18   LJF1
19   LJF2
20   LJF3
21   LJF4
Head:
22   Joint_Pan
23   Joint_Tilt
```

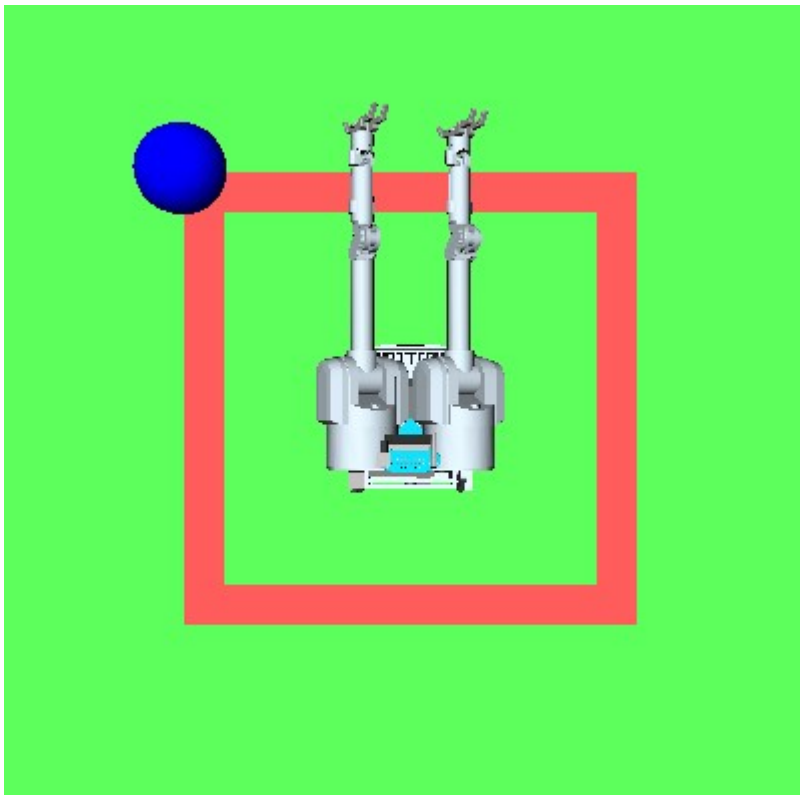
4. HERB in Self Collision




```
def put_in_self_collision(self):
  with self.env:
  #TODO Fill in, remove sleep
    robo.robot.SetJointValues([5], [11])
    robo.robot.SetJointValues([-5], [12])
    robo.robot.SetJointValues([-5], [1])
    robo.robot.SetJointValues([5], [14])
    robo.robot.SetJointValues([5], [3])
```

5. Putting the ball

```
<KinBody name="ball">
  <Body type="static">
    <Translation>1 1 0.5</Translation>
    <Geom type="sphere">
      <radius>0.2 </radius>
      <diffuseColor>0 0 1</diffuseColor>
      <ambientColor>0 0 1</ambientColor>
    </Geom>
  </Body>
</KinBody>
```



6. How Long this homework took you?

Learning python : 12 hours
Learning openrave : 6 hours
doing homework : 5 hours

total 23 hours.