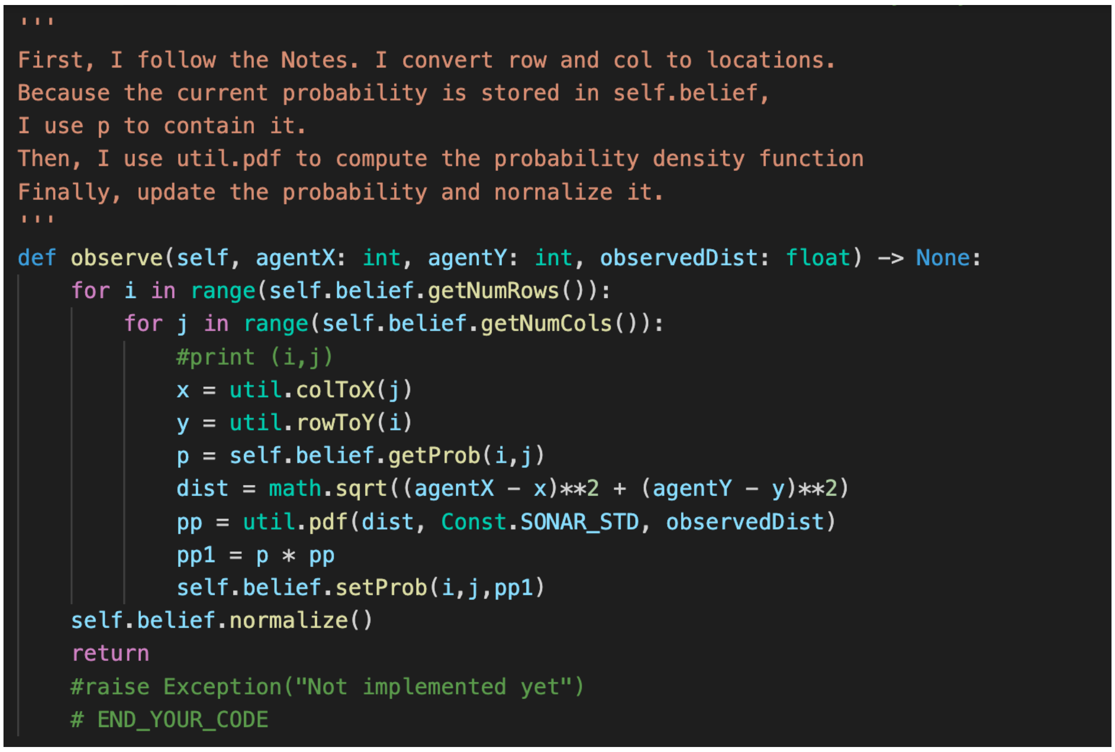
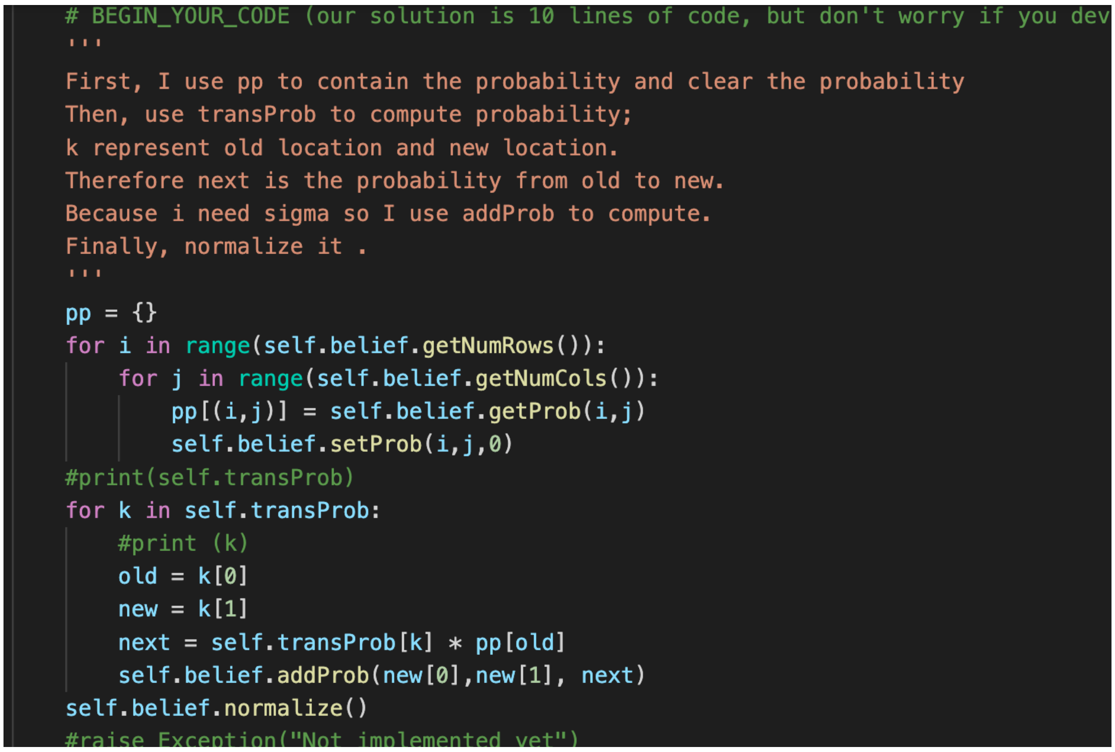
Homework 5: Car Tracking

109550116 楊傑宇

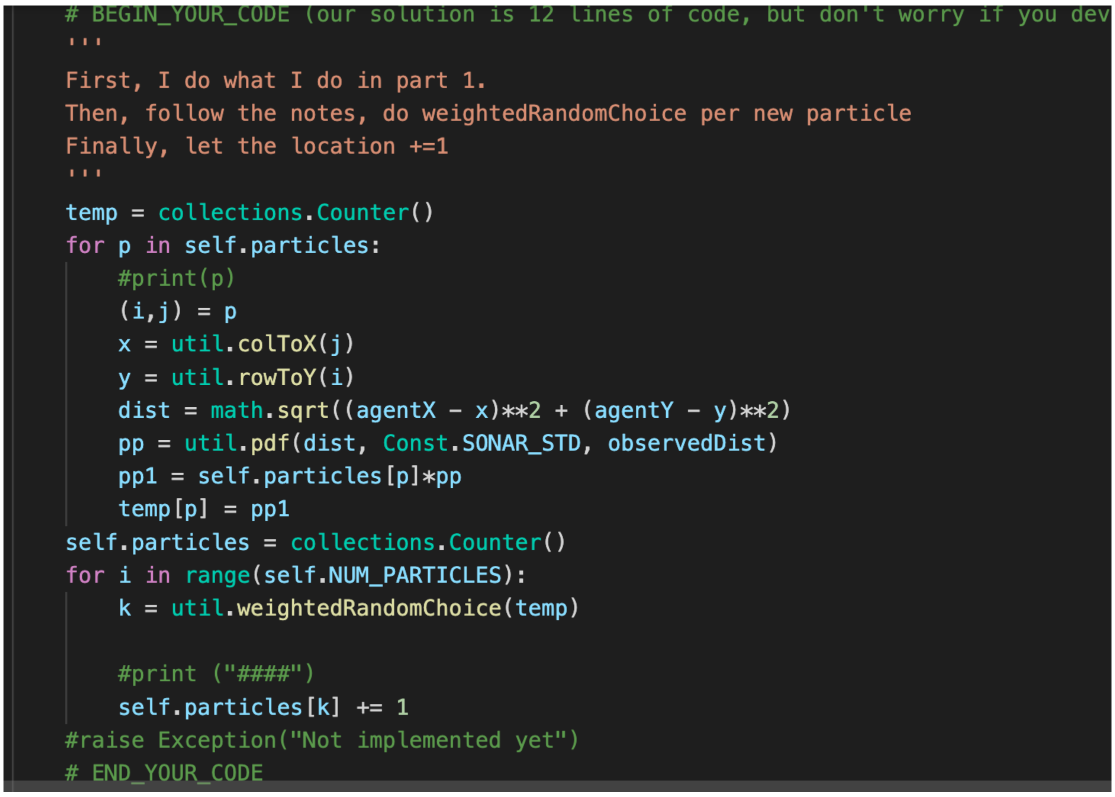
Part I. Implementation (20%):



```python
'''
First, I follow the Notes. I convert row and col to locations.
Because the current probability is stored in self.belief,
I use p to contain it.
Then, I use util.pdf to compute the probability density function
Finally, update the probability and nornalize it.
'''
def observe(self, agentX: int, agentY: int, observedDist: float) -> None:
    for i in range(self.belief.getNumRows()):
        for j in range(self.belief.getNumCols()):
            #print (i,j)
            x = util.colToX(j)
            y = util.rowToY(i)
            p = self.belief.getProb(i,j)
            dist = math.sqrt((agentX - x)**2 + (agentY - y)**2)
            pp = util.pdf(dist, Const.SONAR_STD, observedDist)
            pp1 = p * pp
            self.belief.setProb(i,j,pp1)
    self.belief.normalize()
    return
    #raise Exception("Not implemented yet")
    # END_YOUR_CODE
```

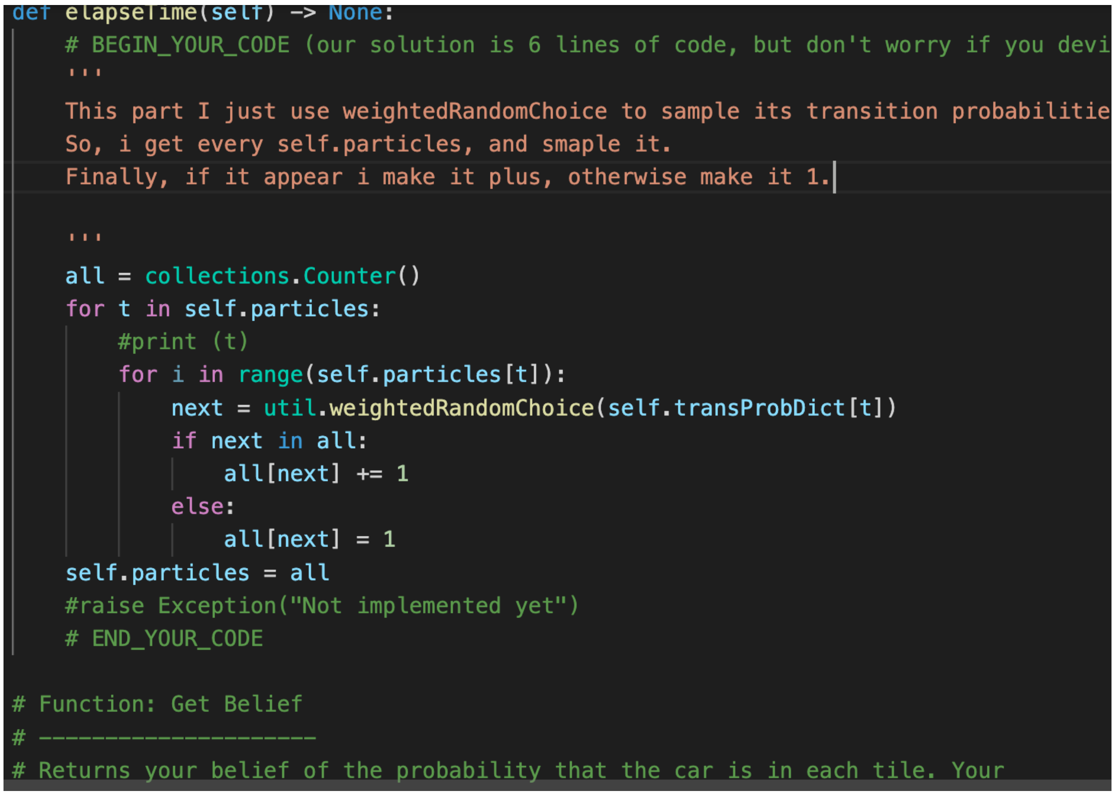


```python
    # BEGIN_YOUR_CODE (our solution is 10 lines of code, but don't worry if you dev
    '''
    First, I use pp to contain the probability and clear the probability
    Then, use transProb to compute probability;
    k represent old location and new location.
    Therefore next is the probability from old to new.
    Because i need sigma so I use addProb to compute.
    Finally, normalize it .
    '''
    pp = {}
    for i in range(self.belief.getNumRows()):
        for j in range(self.belief.getNumCols()):
            pp[(i,j)] = self.belief.getProb(i,j)
            self.belief.setProb(i,j,0)
    #print(self.transProb)
    for k in self.transProb:
        #print (k)
        old = k[0]
        new = k[1]
        next = self.transProb[k] * pp[old]
        self.belief.addProb(new[0],new[1], next)
    self.belief.normalize()
    #raise Exception("Not implemented yet")
```



```
    # BEGIN_YOUR_CODE (our solution is 12 lines of code, but don't worry if you dev
    '''
    First, I do what I do in part 1.
    Then, follow the notes, do weightedRandomChoice per new particle
    Finally, let the location +=1
    '''
    temp = collections.Counter()
    for p in self.particles:
        #print(p)
        (i,j) = p
        x = util.colToX(j)
        y = util.rowToY(i)
        dist = math.sqrt((agentX - x)**2 + (agentY - y)**2)
        pp = util.pdf(dist, Const.SONAR_STD, observedDist)
        pp1 = self.particles[p]*pp
        temp[p] = pp1
    self.particles = collections.Counter()
    for i in range(self.NUM_PARTICLES):
        k = util.weightedRandomChoice(temp)

        #print ("####")
        self.particles[k] += 1
    #raise Exception("Not implemented yet")
    # END_YOUR_CODE
```



```
def elapseTime(self) -> None:
    # BEGIN_YOUR_CODE (our solution is 6 lines of code, but don't worry if you devi
    '''
    This part I just use weightedRandomChoice to sample its transition probabilitie
    So, i get every self.particles, and smaple it.
    Finally, if it appear i make it plus, otherwise make it 1.

    '''
    all = collections.Counter()
    for t in self.particles:
        #print (t)
        for i in range(self.particles[t]):
            next = util.weightedRandomChoice(self.transProbDict[t])
            if next in all:
                all[next] += 1
            else:
                all[next] = 1
    self.particles = all
    #raise Exception("Not implemented yet")
    # END_YOUR_CODE

# Function: Get Belief
# ---------------------
# Returns your belief of the probability that the car is in each tile. Your
```