Task 1

1) In your report, mention what you see in the agent’s behavior. Does it eventually make it to the target location? Ans : Initially I have set the movement to be random. The agent behaviour is random. It chooses any of the direction from [None,'left','right','forward'] and work accordingly. Since there is no learning right now, so it takes a random step. It may or may not make it to the target location.

2) Justify why you picked these set of states, and how they model the agent and its environment. Ans : Here are the list of set of states I took a) Traffic light states : 2 = Red or green : Agent should learn about the traffic light b) Traffic states : 4 (left, right, forward, None) : (for each ongoing, left and right traffic) : Agent should be learning about the action it needs to take when traffic is coming from the direction mentioned c) waypoint : 3 (left, right, forward) : Agent should be learning to follow or unfollow the path of the planner to reach the destination.

Hence the matrix is of [4x2x4x4x4x3]

Modelling Agent get to know about states as it starts exploring the space. More and more interaction with the environment help it to learn about more and more states.

Task 2 What changes do you notice in the agent’s behavior? Ans : Here are the changes I observed in comparison to the random selection 1) Initially since the agent has not learned it took a random direction and therefore most of the time was unsuccessful in reaching the destination 2) Once the trial increases and it explores more space, it tends to learn more and follow the planner path. Also it learns to obey the rules of traffic light based on the rewards he gets.

Report what changes you made to your basic implementation of Q-Learning to achieve the final version of the agent. How well does it perform?

Initially I took the grid position as well in my set of states. This increased my states to many folds and hence reduced my learning. After that I realized there is no need for the grid positions as the destination is changing. Also I was not looking at the waypoints initially and taking the headings but later realized there is no need for heading. It performs really well. Slowly it learns from the rewards and then increases the success rate of reaching the destination I have plotted the graph below to show how the success rate increases with the increase in learning.

In [71]:

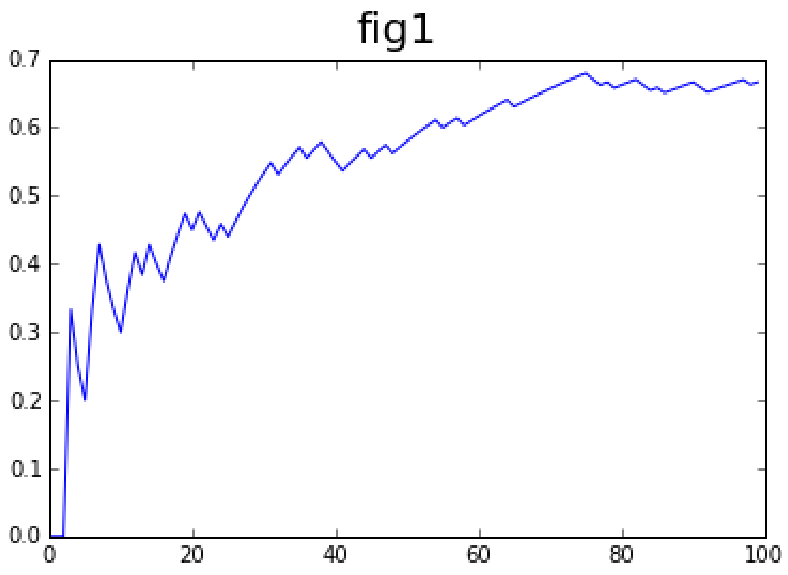
```
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

In [88]:

```
def plotSucessGraph(fileName, title):
    successAvg = np.zeros([1000], dtype=float)
    totalTrials = 0
    failedCount = 0
    successCount = 0



    with open('../' + fileName) as f:
        for line in f:
            if  line.find("Primary agent could not reach destination within deadli
ne!") != -1:
                totalTrials = totalTrials + 1
                failedCount = failedCount + 1

                successAvg[totalTrials] = successCount / (totalTrials * 1.0)
            elif line.find("Primary agent has reached destination!") != -1:
                totalTrials = totalTrials + 1
                successCount = successCount + 1
                successAvg[totalTrials] = successCount / (totalTrials  * 1.0)

    #print successAvg[:100]
    fig = plt.figure()
    plt.plot(successAvg[:100])
    fig.suptitle(title, fontsize=20)
```

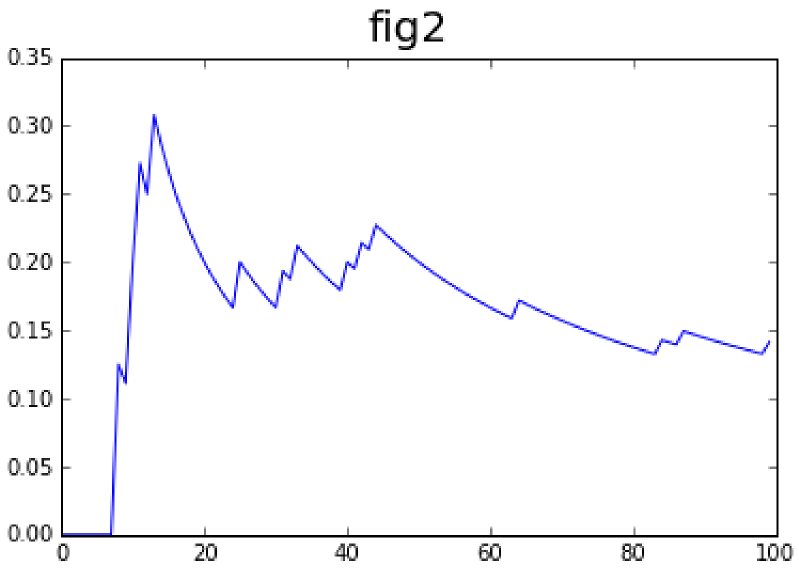
In [89]:

```
# Low learning rate and low discount factor
# alpha = .1, gamma = .1
plotSucessGraph("lowlow.log", "fig1")
```



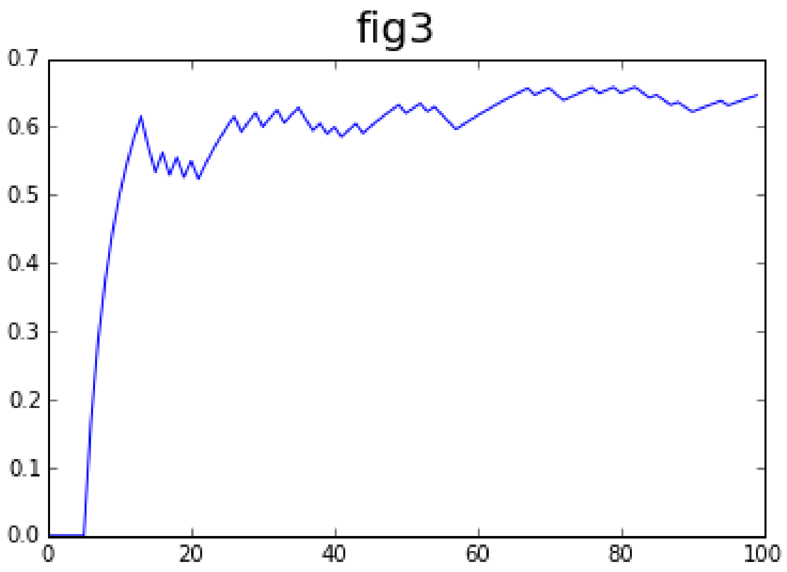
In [90]:

```
# high learning rate and high discount factor
# alpha = .9, gamma = .9
plotSucessGraph("highhigh.log", 'fig2')
```



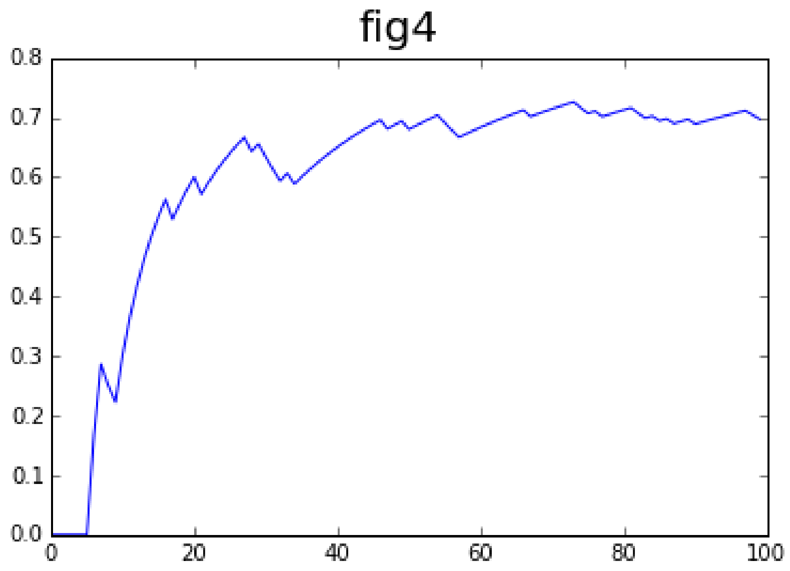
In [91]:

```
# low learning rate and high discount factor
# alpha = .1, gamma = .9
plotSucessGraph("lowhigh.log", 'fig3')
```



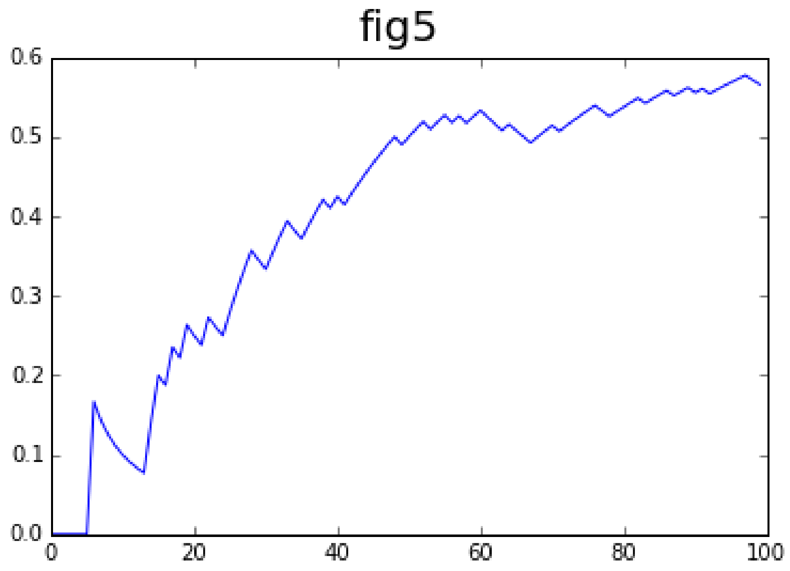
In [92]:

```
# high learning rate and low discount factor
# alpha = .9, gamma = .1
plotSucessGraph("highlow.log", 'fig4')
```



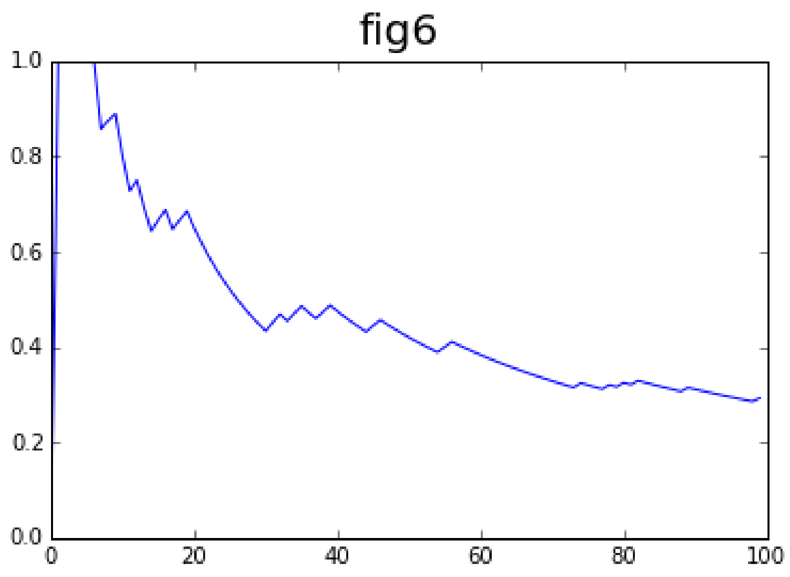
In [93]:

```
# one learning rate and zero discount factor
# alpha = 1, gamma = 0
plotSucessGraph("onezero.log", 'fig5')
```



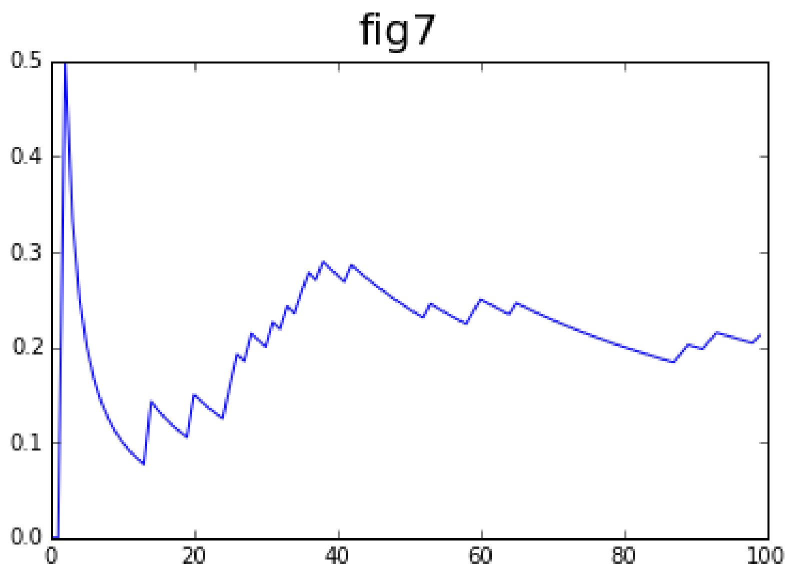
In [94]:

```
# one learning rate and 1 discount factor
# alpha = 1, gamma = 1
plotSucessGraph("oneone.log", 'fig6')
```



In [95]:

```
# 0 learning rate and 1 discount factor
# alpha = 0, gamma = 1
plotSucessGraph("zeroone.log", 'fig7')
```



Does your agent get close to finding an optimal policy, i.e. reach the destination in the minimum possible time, and not incur any penalties?

Yes, the agent learns and improve upon its success rate. It reaches the destination before deadlines without incurring penalities. So I tried to graph out with various combinations of learning rate and discount factor. So 1) when the learning rate is 0, no learning then the agent is pathetic in looking for the destination and hence you see a very bad success curve as in fig 7 2) when learning is high and a 0 discount factor will take the rewards into consideration where no future rewards. refer to fig5 for this. Since there are no future discount into consideration the learning is slow and it gradually increases. 3) So the best one is with low learning rate and high discount factor i.e fig 3. Since the environment is stochostic, low learing rate is ideal and high future rewards.

```
In [ ]:
```