In [ ]:
```
import gym
import random

#Red – 0 , Green – 1, Yellow – 2, and Blue – 3 for pick up
streets = gym.make("Taxi-v3").env #New versions keep getting released; if -v3 doesn't work, try -v2 or -v4
streets.render()
```

In [ ]:
```
##Red – 0 , Green – 1, Yellow – 2, and Blue – 3 for pick up
#Each state is defined by a 4 entries tuple:  (taxi_row, taxi_col, passenger_location, destination)
initial_state = streets.encode(2, 1, 3, 1)

streets.s = initial_state

streets.render()

#State Space:  25 possible taxi positions, 5 possible locations of the passenger
# 25*5*4 = 500
```

In [ ]:
```
import numpy as np
#Action space:6 --> N,S,E,W, DROP-OFF, PICKUP
#Rewards: CORRECT FINAL DEST. +20, STEP -1, INCORRECT PICK/DROP -10

q_table = np.zeros([streets.observation_space.n, streets.action_space.n]) # 500 , 6
# a 2D array that represent every possible state and action in the virtual space and initialize all of them to 0
G = 0
learning_rate = 0.1
discount_factor = 0.5
exploration = 0.1
epochs = 500

for taxi_run in range(epochs): #Start training (the agent plays the number of epochs)
    state = streets.reset()
    done = False
    G = 0
    steps=0
    while not done:#each epoch/play contains this number of actions, starting from pickup a passenger until drop-off
        steps +=1
        random_value = random.uniform(0, 1)
```

```
        if (random_value < exploration):
            action = streets.action_space.sample() # Explore a random action
        else:
            action = np.argmax(q_table[state]) # Return the action with the highest q-value

        next_state, reward, done, info = streets.step(action) # Do the above action

        prev_q = q_table[state, action]
        next_max_q = np.max(q_table[next_state])
        # see RL-2 PPT file --- slide# 5
        new_q = (1 - learning_rate) * prev_q + learning_rate * (reward + discount_factor * next_max_q)
        G += reward
        q_table[state, action] = new_q
        #streets.render()
        state = next_state
```

In [ ]:

```
from IPython.display import clear_output
from time import sleep
lengths=[]
for tripnum in range(1, 11):
    state = streets.reset()

    done = False
    trip_length = 0

    while not done and trip_length < 25:
        action = np.argmax(q_table[state])
        next_state, reward, done, info = streets.step(action)
        clear_output(wait=True)
        print("Trip number " + str(tripnum) + " Step " + str(trip_length))
        print(streets.render(mode='ansi'))
        sleep(.2)
        state = next_state
        trip_length += 1
    lengths.append(trip_length)

    sleep(.2)
avg_len=sum(lengths)/10
print(avg_len)
```

In [ ]: