In [114]:

```
pip install gym
```

```
Requirement already satisfied: gym in c:\users\rohith\anaconda3\lib\site-packages (0.18.0
)
Requirement already satisfied: numpy>=1.10.4 in c:\users\rohith\anaconda3\lib\site-packag
es (from gym) (1.18.1)
Requirement already satisfied: scipy in c:\users\rohith\anaconda3\lib\site-packages (from
gym) (1.4.1)
Requirement already satisfied: Pillow<=7.2.0 in c:\users\rohith\anaconda3\lib\site-packag
es (from gym) (7.0.0)
Requirement already satisfied: pyglet<=1.5.0,>=1.4.0 in c:\users\rohith\anaconda3\lib\sit
e-packages (from gym) (1.5.0)
Requirement already satisfied: cloudpickle<1.7.0,>=1.2.0 in c:\users\rohith\anaconda3\lib
\site-packages (from gym) (1.3.0)
Requirement already satisfied: future in c:\users\rohith\anaconda3\lib\site-packages (fro
m pyglet<=1.5.0,>=1.4.0->gym) (0.18.2)
Note: you may need to restart the kernel to use updated packages.
```

In [115]:

```
import gym
enviro = gym.make('Taxi-v3').env
enviro.render()
```

```
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
```

In [116]:

```
enviro.reset()
enviro.render()
```

```
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
```

In [117]:

```
state = enviro.encode(2,2,3,0)
print("State:",state)
enviro.s = state
enviro.render()
```

```
State: 252
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
```

In [118]:

```
print("Action {}".format(enviro.action_space))
```

```
print("State {}".format(enviro.observation_space))
```

```
Action Discrete(6)
State Discrete(500)
```

In [119]:

```
enviro.P[252]
```

Out[119]:

```
{0: [(1.0, 352, -1, False)],
 1: [(1.0, 152, -1, False)],
 2: [(1.0, 272, -1, False)],
 3: [(1.0, 232, -1, False)],
 4: [(1.0, 252, -10, False)],
 5: [(1.0, 252, -10, False)]}
```

In [121]:

```
epochs = 0
penalities,rewards = 0,0
frames = []
completed = False
while not completed:
    action = enviro.action_space.sample()
    state,reward,completed,info = enviro.step(action)

    if reward == -10:
        penalities += 1
    frames.append({'frame':enviro.render(mode='ansi'),
                   'state':state,'action':action,
                   'reward':reward})

    epochs += 1

print('Steps taken:{}'.format(epochs))
print('Penalities received are:{}'.format(penalities))
```

```
Steps taken:194
Penalities received are:60
```

In [122]:

```
from IPython.display import clear_output
from time import sleep

def display(frames):
    for i,frame in enumerate(frames):
        clear_output(wait = True)
        print(frame['frame'])
        print(f"step:{i+1}")
        print(f"State:{frame['state']}")
        print(f"Action:{frame['action']}")
        print(f"Reward:{frame['reward']}")
        sleep(.1)
display(frames)
```

```
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (Dropoff)

step:194
State:0
Action:5
Reward:20
```

Implementing the Q-Learning Policy for the environment

In [123]:

```
import numpy as np
q_table = np.zeros([enviro.observation_space.n,enviro.action_space.n])
```

In [124]:

```
import random
alpha = 0.1
gamma = 0.6
epsilon = 0.1

tot_epochs = []
tot_penalties = []

for i in range(1, 100001):
    state = enviro.reset()

    epochs, penalties, reward, = 0, 0, 0
    done = False

    while not done:
        if random.uniform(0, 1) < epsilon:
            action = enviro.action_space.sample()
        else:
            action = np.argmax(q_table[state])

        next_state, reward, done, info = enviro.step(action)

        old_value = q_table[state, action]
        next_max = np.max(q_table[next_state])

        new_value = (1 - alpha) * old_value + alpha * (reward + gamma * next_max)
        q_table[state, action] = new_value

        if reward == -10:
            penalties += 1

        state = next_state
        epochs += 1

    if i % 100 == 0:
        clear_output(wait=True)
        print(f"Episode: {i}")

print("Training finished.\n")
```

```
Episode: 100000
Training finished.
```

In [125]:

```
total_epochs, total_penalties = 0, 0
episodes = 10000

for _ in range(episodes):
    state = enviro.reset()
    epochs, penalties, reward = 0, 0, 0

    done = False

    while not done:
        action = np.argmax(q_table[state])
        state, reward, done, info = enviro.step(action)

        if reward == -10:
            penalties += 1
```

```
        epochs += 1

    total_penalties += penalties
    total_epochs += epochs

print(f"Results after {episodes} episodes:")
print(f"Average timesteps per episode: {total_epochs / episodes}")
print(f"Average penalties per episode: {total_penalties / episodes}")
```

```
Results after 10000 episodes:
Average timesteps per episode: 13.0658
Average penalties per episode: 0.0
```

In [ ]:

```
```