```
In [141]: import gymnasium as gym
import random
import numpy as np
import matplotlib.pyplot as plt
env = gym.make("CartPole-v1", render_mode="human")
```

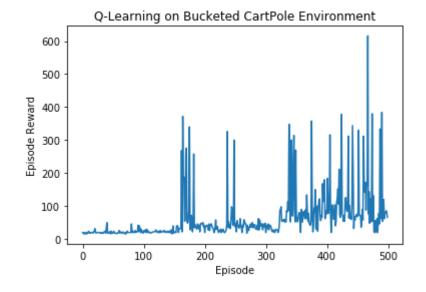
1 of 3 11/22/23, 23:10

```
In [143]: class QLearnerSolver:
              def init (self,env):
                  self.q table = {}
                  self.alpha = 0.1
                  self.gamma = 0.95
                  self.epsilon = 0.1
                  self.env = env
                  self.action space = list(range(env.action space.n))
                  self.state space size = [3,3,6,6]
                  self.q table = np.zeros(self.state space size + [env.action spa
              def convert cont to discrete space(self, state):
                  if isinstance(state, tuple):
                      state = state[0]
                  bucket edgesv = np.linspace(-4.8, 4.8, 3)
                  bucket edgesvdot = np.linspace(-5, 5, 3)
                  bucket edgestheta = np.linspace(-0.418, 0.418, 6)
                  bucket edgesthetadot = np.linspace(-5, 5, 6)
                  bucket indexv = np.digitize(state[0], bucket edgesv) - 1
                  bucket indexvtheta = np.digitize(state[1], bucket edgesvdot) -
                  bucket indexdot = np.digitize(state[2], bucket edgestheta) - 1
                  bucket indexdottheta = np.digitize(state[3], bucket edgesthetad
                  return tuple([bucket indexv,bucket indexvtheta,bucket indexdot,
              def choose action(self, state):
                  # if the random is more than epsilon
                  if random.uniform(0,1) < self.epsilon:</pre>
                      return self.env.action space.sample()
                  else:
                      discretized state = self.convert cont to discrete space(sta
                      q values = self.q table[discretized state]
                      return np.argmax(q values)
              def learn(self, num episodes):
                  total rewards = []
                  for learning epoch in range(num episodes):
                      state = env.reset()
                      total reward = 0
                                                        #every episode, reset the
                      for time step in range(500):
                          action = self.choose action(state) #learner chooses one
                          next state,reward,done, , =env.step(action) #the actio
                          discretized state = self.convert cont to discrete space
                          # Use Q-learning update rule
                          max q next = np.max(self.q table[self.convert cont to d
                          self.q table[discretized state + (action,)] += self.alp
                          state = next state
                          total reward += reward
                          # update total reward
```

2 of 3 11/22/23, 23:10

```
total_reward = total_reward + reward
    total_rewards.append(total_reward)
return total_rewards
```

```
In [144]: env = gym.make('CartPole-v1')
    learner1=QLearnerSolver(env)
    total_rewards = learner1.learn(500)
    plt.plot(range(0, 500), total_rewards)
    plt.xlabel("Episode")
    plt.ylabel("Episode Reward")
    plt.title("Q-Learning on Bucketed CartPole Environment")
    plt.show()
```



3 of 3 11/22/23, 23:10