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

class MultiArmedBanditEnv:
  def __init__(self,n_arms=10):
    self.n_arms = n_arms
    self.q_true = np.random.normal(0,1,n_arms)
    self.best_arm = np.argmax(self.q_true)
    self.reset()

  def reset(self):
    return None

  def step(self ,action):
    reward = np.random .normal(self.q_true[action],1)
    return None , reward,False,False,{}


def run_epsilon_greedy(env, episodes=500, epsilon=0.1):
 k = env.n_arms
 Q = np.zeros(k) # Initialize Q
 N = np.zeros(k) # Initialize N
 rewards = []


 for _ in range(episodes):
    # env.reser() # Typo, corrected to reset
    # This line seems incorrect, the reset should be outside the loop if you want to simulate episodes

    action = np.random.choice(k) if np.random.rand() < epsilon else np.argmax(Q)
    _,reward, _, _, _ =env.step(action)
    N[action] +=1
    Q[action] += (reward - Q[action])/N[action]
    rewards.append(reward)

 return Q, rewards


def run_softmax(env, episodes=500,  temperature=0.5):
  k =env.n_arms
  Q = np.zeros(k)
  N = np.zeros(k)
  rewards = []

  for _ in range(episodes):
    env.reset()

    exp_q = np.exp((Q - np.max(Q)) / temperature)
    probabilities = exp_q / np.sum(exp_q)
    action = np.random.choice(k, p=probabilities)

    _, reward, _, _, _ = env.step(action)
    N[action] += 1
    Q[action] += (reward - Q[action]) / N[action]
    rewards.append(reward)

  return Q, rewards


env = MultiArmedBanditEnv(n_arms=10)

q_eps, rewards_eps = run_epsilon_greedy(env, episodes=500,epsilon=0.1)
q_softg,rewards_softg = run_softmax(env,episodes=500, temperature=0.5) # Corrected function call


print("Epsilon-Greedy Q-values:", q_eps)
print("Total reward (Epsilon-Greedy):",round(sum(rewards_eps))+1)
print("\n Softmax Q-values:" , q_softg) # Corrected variable name
print("Total reward (Softmax):", round(sum(rewards_softg), 2)) # Corrected variable name


avg_rewards_eps  = np.cumsum(rewards_eps)/np.arange(1,len(rewards_eps)+1)
avg_rewards_soft = np.cumsum(rewards_softg)/np.arange(1,len(rewards_softg)+1) # Corrected variable name

plt.plot(avg_rewards_eps,label='Epsilon-Greedy') # Corrected variable name
plt.plot(avg_rewards_soft,label='Softmax')
plt.xlabel('Episodes')
plt.ylabel('Average Reward')
plt.title('Average Reward vs Episodes')
plt.legend()
plt.grid(True)
plt.show()
```

```
Epsilon-Greedy Q-values: [-0.34790954  0.67889042 -0.11289455 -0.66631973  0.52365345  0.90678116
 -2.73717217  0.50347616 -0.02427536  0.6149806 ]
Total reward (Epsilon-Greedy): 400

 Softmax Q-values: [-0.23737237 -0.08568551  0.23920629 -0.70346795  0.87398795  0.75535462
 -2.05918451  0.59036351 -0.09232409  0.48660194]
Total reward (Softmax): 248.45
```