Setup

Install Dependencies

```
!pip install gymnasium
```

```
Requirement already satisfied: gymnasium in /usr/loc
Requirement already satisfied: numpy>=1.21.0 in /usr
Requirement already satisfied: cloudpickle>=1.2.0 in
Requirement already satisfied: typing-extensions>=4.
Requirement already satisfied: farama-notifications>
```



Import dependencies

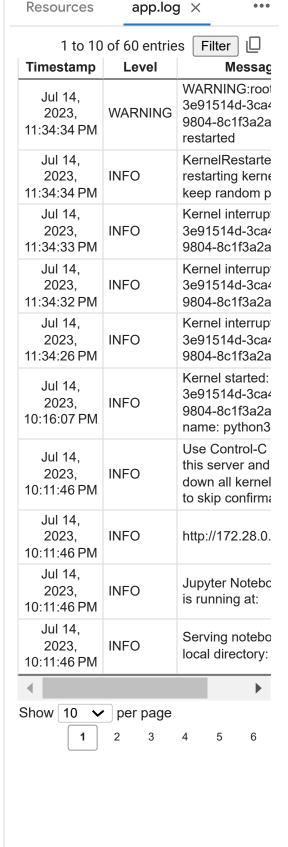
```
import gymnasium as gym
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from torch.distributions import Categorical
import time
import matplotlib.pyplot as plt
```

Preperation

Define the Policy Network

```
class Policy(nn.Module):
    def __init__(self, state_dim, action_dim):
        super(Policy, self).__init__()
        self.fc1 = nn.Linear(state_dim, 128)
        self.fc2 = nn.Linear(128, action_dim)

def forward(self, x):
    x = F.relu(self.fc1(x))
    x = self.fc2(x)
    return F.softmax(x, dim=1)
```



Create the environment, instantiate the policy network and define the optimizer

```
# Create the environment
env = gym.make('CartPole-v1',render_mode="rgb_array")
state_dim = env.observation_space.shape[0]
action_dim = env.action_space.n

# Initialize the policy network
policy = Policy(state_dim, action_dim)

# Define the optimizer
optimizer = optim.Adam(policy.parameters(), lr=0.01)
```

Algorithm

Pick an action based on policy

```
def select_action(state):
    state = np.array(state)
    state = torch.from_numpy(state).float().unsqueeze(0)
    probs = policy(state)
    m = Categorical(probs)
    action = m.sample()
    return action.item(), m.log_prob(action)
```

Policy Gradiant Algorithm, the actual training loop

```
def policy_gradient():
    num_episodes = 1000
    gamma = 0.99

    rewards_per_episode = [] # List to store rewards for

# for 1000 episodes
for episode in range(num_episodes):
    observations = env.reset()
    state = np.array(observations[0])
    episode_reward = 0
    log_probs = []
    rewards = []

# loop through each time step in one episode
    while True:
```

```
action, log prob = select action(state)
        next_state, reward, done, _, _ = env.step(act:
        log probs.append(log prob)
        rewards.append(reward)
        episode reward += reward
        if done:
            break
        state = next state
    # Compute the discounted rewards
    discounts = [gamma**i for i in range(len(rewards))
    discounted rewards = [discount * reward for discount
    # Convert the discounted rewards into a Tensor
    discounted rewards = torch.Tensor(discounted reward)
    # Normalize the discounted rewards
    discounted_rewards -= torch.mean(discounted_reward
    discounted_rewards /= torch.std(discounted_rewards
    # Calculate the loss
    policy loss = []
    for log_prob, reward in zip(log_probs, discounted)
        policy_loss.append(-log_prob * reward)
    policy loss = torch.cat(policy loss).sum()
    # Update the policy network
    optimizer.zero_grad()
    policy loss.backward()
    optimizer.step()
    # Print the episode statistics
    if episode % 10 == 0:
        print('Episode {}: reward = {}'.format(episode
# Plot the rewards per episode
plt.plot(rewards per episode)
plt.xlabel('Episode')
plt.ylabel('Reward')
plt.title('Reward per Episode')
plt.show()
```

▼ Run Trials

horrch Ri.anrelic()

```
Episode 0: reward = 11.0
Episode 10: reward = 31.0
Episode 20: reward = 11.0
Episode 30: reward = 72.0
Episode 40: reward = 31.0
Episode 50: reward = 124.0
Episode 60: reward = 50.0
Episode 70: reward = 36.0
Episode 80: reward = 42.0
Episode 90: reward = 203.0
Episode 100: reward = 112.0
Episode 110: reward = 53.0
Episode 120: reward = 106.0
Episode 130: reward = 93.0
Episode 140: reward = 65.0
Episode 150: reward = 41.0
Episode 160: reward = 87.0
Episode 170: reward = 97.0
Episode 180: reward = 44.0
Episode 190: reward = 62.0
Episode 200: reward = 114.0
Episode 210: reward = 70.0
Episode 220: reward = 83.0
Episode 230: reward = 51.0
Episode 240: reward = 49.0
Episode 250: reward = 39.0
Episode 260: reward = 89.0
Episode 270: reward = 70.0
Episode 280: reward = 46.0
Episode 290: reward = 36.0
Episode 300: reward = 46.0
Episode 310: reward = 61.0
Episode 320: reward = 68.0
Episode 330: reward = 110.0
Episode 340: reward = 135.0
Episode 350: reward = 134.0
Episode 360: reward = 175.0
Episode 370: reward = 115.0
Episode 380: reward = 78.0
Episode 390: reward = 160.0
Episode 400: reward = 388.0
Episode 410: reward = 144.0
Episode 420: reward = 98.0
Episode 430: reward = 63.0
Episode 440: reward = 58.0
Episode 450: reward = 47.0
Episode 460: reward = 64.0
Episode 470: reward = 119.0
Episode 480: reward = 114.0
Episode 490: reward = 115.0
Episode 500: reward = 102.0
Episode 510: reward = 114.0
Episode 520: reward = 63.0
Episode 530: reward = 110.0
```

```
Episode 540: reward = 121.0
Episode 550: reward = 114.0
Episode 560: reward = 138.0
   Colab paid products - Cancel contracts here
                     54m 16s
                                 completed at 11:34 PM
                                                                                     ×
```