In [1]:
```python
import torch
import pygame

import numpy as np
import gymnasium as gym
```

```
pygame 2.1.0 (SDL 2.0.16, Python 3.9.12)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

In [2]:
```python
# Run the code below to test the environment with keyboard input
env_name = "MountainCarContinuous-v0"

actions = {
    pygame.K_LEFT: -1.0,  # reverse
    pygame.K_RIGHT: 1.0,  # forward
}

pygame.init()
env = gym.make(env_name, render_mode="human")

# Reset the environment with a seed
env.reset()

# Game Loop
done = False
while not done:
    # Render the environment
    env.render()

    # Check for events (keyboard input)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:  # If the window close button is pressed
            done = True
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:  # If ESC is pressed
                done = True

    # Default action (no key pressed)
    action = 0

    # Check for key press and map it to action
    keys = pygame.key.get_pressed()
```

```python
    for key, action_value in actions.items():
        if keys[key]:
            action = action_value
            break

    action = np.array([action])

    # Step through the environment with the chosen action
    observation, reward, terminated, truncated, info = env.step(action)

    # Check if the episode is over
    if terminated or truncated:
        observation, info = env.reset()

# Close the environment after the game loop
env.close()
pygame.quit()
```

In [3]:

```python
class PolicyNet(torch.nn.Module):
    def __init__(self, state_dim, action_dim):
        super(PolicyNet, self).__init__()
        NotImplemented

    def forward(self, x):
        NotImplemented

class ValueNet(torch.nn.Module):
    def __init__(self, state_dim, action_dim):
        super(ValueNet, self).__init__()
        NotImplemented

    def forward(self, x):
        NotImplemented

class ActorCritic:
    def __init__(self, states, actions, device):
        self.device = device
        self.action = actions
        self.actor = PolicyNet(states, actions).to(device)
        self.critic = ValueNet(states, actions).to(device)
        NotImplemented

    def take_action(self, state):
```

```python
        NotImplemented

    def update(self):
        NotImplemented
```

In [4]:
```python
device = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")

env = gym.make(env_name, render_mode="rgb_array")

obs_space = env.observation_space.shape[0]  # 2, continuous
action_space = env.action_space.shape[0]   # 1, continuous

state, _ = env.reset()

episodes = 500

agent = ActorCritic(obs_space, action_space, device)

# Training Code Below
```

In [ ]:
```python
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
from tqdm import tqdm
import sklearn.preprocessing

env = gym.make("MountainCarContinuous-v0")
state_dim = env.observation_space.shape[0]
action_dim = env.action_space.shape[0]

state_space_samples = np.array([env.observation_space.sample() for _ in range(10000)])
scaler = sklearn.preprocessing.StandardScaler()
scaler.fit(state_space_samples)

def scale_state(state):
    state = np.array(state, dtype=np.float32).reshape(1, -1)
    scaled = scaler.transform(state)
    return torch.tensor(scaled, dtype=torch.float32)

class PolicyNet(nn.Module):
    def __init__(self, state_dim, action_dim):
        super(PolicyNet, self).__init__()
        self.fc1 = nn.Linear(state_dim, 40)
```

```python
        self.fc2 = nn.Linear(40, 40)
        self.mu = nn.Linear(40, action_dim)
        self.sigma = nn.Linear(40, action_dim)

    def forward(self, state):
        x = torch.relu(self.fc1(state))
        x = torch.relu(self.fc2(x))
        mu = self.mu(x)
        sigma = torch.nn.functional.softplus(self.sigma(x)) + 0.1  # 调整sigma最小值
        return mu, sigma

class ValueNet(nn.Module):
    def __init__(self, state_dim):
        super(ValueNet, self).__init__()
        self.fc1 = nn.Linear(state_dim, 400)
        self.fc2 = nn.Linear(400, 400)
        self.value = nn.Linear(400, 1)

    def forward(self, state):
        x = torch.relu(self.fc1(state))
        x = torch.relu(self.fc2(x))
        return self.value(x)

class ActorCriticAgent:
    def __init__(self, state_dim, action_dim, lr_actor=1e-3, lr_critic=5e-3):
        self.policy_net = PolicyNet(state_dim, action_dim)
        self.value_net = ValueNet(state_dim)
        self.optimizer_actor = optim.Adam(self.policy_net.parameters(), lr=lr_actor)
        self.optimizer_critic = optim.Adam(self.value_net.parameters(), lr=lr_critic)
        self.gamma = 0.99
        self.rewards = []
        self.states = []
        self.actions = []

    def take_action(self, state, exploration_noise=0.1):
        mu, sigma = self.policy_net(state)
        dist = torch.distributions.Normal(mu, sigma)
        action = dist.sample()
        action = action + exploration_noise * torch.randn_like(action)  # 动态探索
        action = torch.clamp(action, env.action_space.low[0], env.action_space.high[0])
        return action.detach().numpy()

    def update(self):
        if len(self.rewards) == 0:
```

```
            #print("Warning: No rewards to update. Skipping this step.")
            return 0, 0

        rewards = self.rewards
        next_state_value = 0
        returns = []

        for r in rewards[::-1]:
            next_state_value = r + self.gamma * next_state_value
            returns.insert(0, next_state_value)

        returns = torch.tensor(returns, dtype=torch.float32)
        states = torch.cat(self.states)
        actions = torch.cat(self.actions)

        returns = (returns - returns.mean()) / (returns.std() + 1e-8)

        state_values = self.value_net(states).squeeze(-1)  # 匹配 return
        critic_loss = torch.mean((returns - state_values) ** 2)

        mu, sigma = self.policy_net(states)
        dist = torch.distributions.Normal(mu, sigma)
        log_probs = dist.log_prob(actions).sum(dim=-1)
        advantages = (returns - state_values).detach()  # Advantage
        actor_loss = -torch.mean(log_probs * advantages)

        # 更新 Critic
        self.optimizer_critic.zero_grad()
        critic_loss.backward()
        self.optimizer_critic.step()

        # 更新 Actor
        self.optimizer_actor.zero_grad()
        actor_loss.backward()
        self.optimizer_actor.step()

        self.rewards = []
        self.states = []
        self.actions = []

        return actor_loss.item(), critic_loss.item()


iterations = 100
```

```python
episodes_per_iteration = 100
agent = ActorCriticAgent(state_dim, action_dim)
return_list, length_list, actor_loss_list, critic_loss_list = [], [], [], []

for i_iter in range(iterations):
    iteration_rewards = []
    with tqdm(range(episodes_per_iteration), desc=f"Iteration {i_iter+1}", ncols=100) as pbar:
        for i_episode_in_iter in pbar:
            state, _ = env.reset(seed=i_iter * episodes_per_iteration + i_episode_in_iter)
            state = scale_state(state)
            done = False
            episode_reward = 0
            episode_length = 0

            while not done:
                action = agent.take_action(state, exploration_noise=max(0.1, 0.5 * (1 - i_iter / iterations)))
                next_state, reward, terminated, truncated, _ = env.step(action)
                done = terminated or truncated

                agent.rewards.append(reward)
                agent.states.append(state)
                agent.actions.append(torch.tensor(action, dtype=torch.float32))

                state = scale_state(next_state)
                episode_reward += reward
                episode_length += 1

            iteration_rewards.append(episode_reward)

        actor_loss, critic_loss = agent.update()
        return_list.extend(iteration_rewards)
        actor_loss_list.append(actor_loss)
        critic_loss_list.append(critic_loss)

    avg_reward = np.mean(iteration_rewards)
    print(f"Iteration {i_iter+1}: Average Reward = {avg_reward:.2f}")

fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# rewards
axes[0].plot(return_list)
axes[0].set_title("Episode Rewards")
axes[0].set_xlabel("Episode")
axes[0].set_ylabel("Reward")
```

```python
axes[0].grid(True)

# Actor Loss
axes[1].plot(actor_loss_list)
axes[1].set_title("Actor Loss")
axes[1].set_xlabel("Iteration")
axes[1].set_ylabel("Loss")
axes[1].grid(True)

# Critic Loss
axes[2].plot(critic_loss_list)
axes[2].set_title("Critic Loss")
axes[2].set_xlabel("Iteration")
axes[2].set_ylabel("Loss")
axes[2].grid(True)

plt.tight_layout()
plt.show()

torch.save(agent.policy_net.state_dict(), "actor_critic_mountaincar.pth")
print("模型已保存为 'actor_critic_mountaincar.pth'")
```

```
Iteration 1:   0%|                                        | 0/100 [00:00<?, ?it/s]e:\Anaconda\lib\site-package
s\gymnasium\utils\passive_env_checker.py:158: UserWarning: WARN: The obs returned by the `step()` method is not within the obser
vation space.
  logger.warn(f"{pre} is not within the observation space.")
Iteration 1: 100%|████████████████████████████████████████| 100/100 [00:36<00:00,  2.72it/s]
Iteration 1: Average Reward = -47.42
Iteration 2: 100%|████████████████████████████████████████| 100/100 [00:37<00:00,  2.68it/s]
Iteration 2: Average Reward = -45.33
Iteration 3: 100%|████████████████████████████████████████| 100/100 [00:36<00:00,  2.71it/s]
Iteration 3: Average Reward = -47.03
Iteration 4: 100%|████████████████████████████████████████| 100/100 [00:36<00:00,  2.70it/s]
Iteration 4: Average Reward = -46.90
Iteration 5: 100%|████████████████████████████████████████| 100/100 [00:36<00:00,  2.72it/s]
Iteration 5: Average Reward = -45.63
Iteration 6: 100%|████████████████████████████████████████| 100/100 [00:36<00:00,  2.74it/s]
Iteration 6: Average Reward = -38.42
Iteration 7: 100%|████████████████████████████████████████| 100/100 [00:36<00:00,  2.74it/s]
Iteration 7: Average Reward = -42.41
Iteration 8: 100%|████████████████████████████████████████| 100/100 [00:36<00:00,  2.72it/s]
Iteration 8: Average Reward = -41.15
```

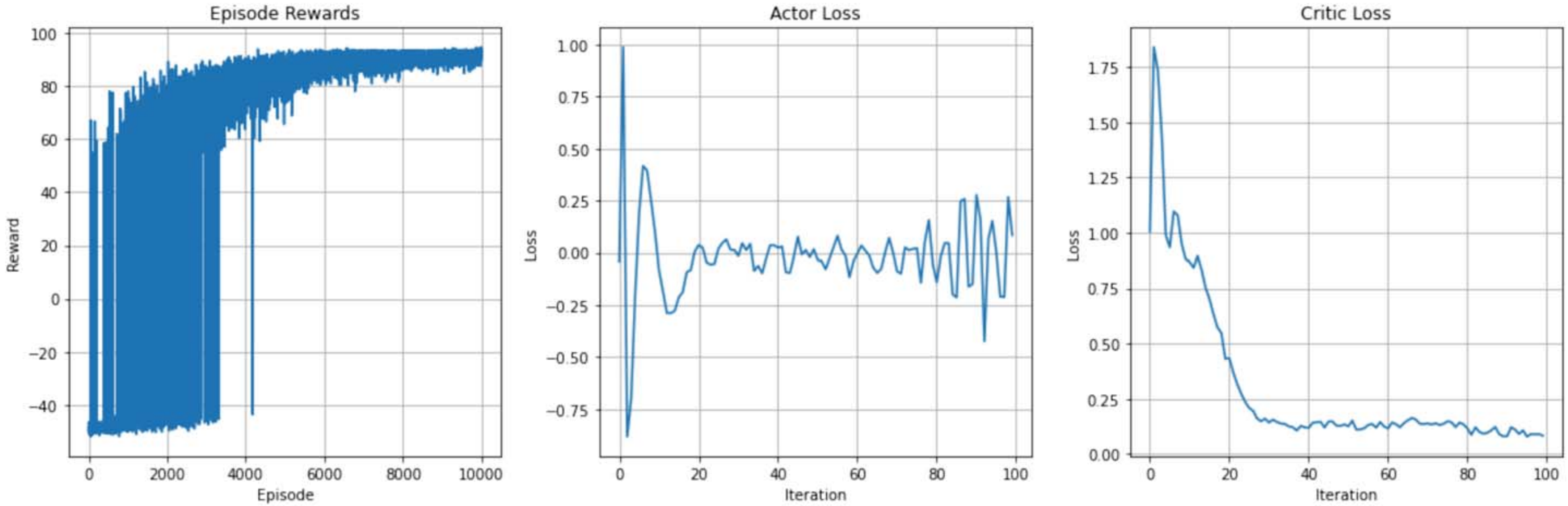
```
Iteration 9: 100%|██████████████████████████████████████████████| 100/100 [00:35<00:00,  2.79it/s]
Iteration 9: Average Reward = -34.60
Iteration 10: 100%|█████████████████████████████████████████████| 100/100 [00:35<00:00,  2.81it/s]
Iteration 10: Average Reward = -28.74
Iteration 11: 100%|█████████████████████████████████████████████| 100/100 [00:35<00:00,  2.79it/s]
Iteration 11: Average Reward = -29.83
Iteration 12: 100%|█████████████████████████████████████████████| 100/100 [00:34<00:00,  2.86it/s]
Iteration 12: Average Reward = -23.46
Iteration 13: 100%|█████████████████████████████████████████████| 100/100 [00:35<00:00,  2.84it/s]
Iteration 13: Average Reward = -22.43
Iteration 14: 100%|█████████████████████████████████████████████| 100/100 [00:34<00:00,  2.90it/s]
Iteration 14: Average Reward = -14.94
Iteration 15: 100%|█████████████████████████████████████████████| 100/100 [00:34<00:00,  2.93it/s]
Iteration 15: Average Reward = -7.29
Iteration 16: 100%|█████████████████████████████████████████████| 100/100 [00:33<00:00,  3.01it/s]
Iteration 16: Average Reward = -0.96
Iteration 17: 100%|█████████████████████████████████████████████| 100/100 [00:32<00:00,  3.09it/s]
Iteration 17: Average Reward = 8.08
Iteration 18: 100%|█████████████████████████████████████████████| 100/100 [00:31<00:00,  3.18it/s]
Iteration 18: Average Reward = 14.94
Iteration 19: 100%|█████████████████████████████████████████████| 100/100 [00:30<00:00,  3.28it/s]
Iteration 19: Average Reward = 24.53
Iteration 20: 100%|█████████████████████████████████████████████| 100/100 [00:29<00:00,  3.44it/s]
Iteration 20: Average Reward = 38.26
Iteration 21: 100%|█████████████████████████████████████████████| 100/100 [00:29<00:00,  3.43it/s]
Iteration 21: Average Reward = 33.14
Iteration 22: 100%|█████████████████████████████████████████████| 100/100 [00:28<00:00,  3.56it/s]
Iteration 22: Average Reward = 37.03
Iteration 23: 100%|█████████████████████████████████████████████| 100/100 [00:27<00:00,  3.64it/s]
Iteration 23: Average Reward = 49.42
Iteration 24: 100%|█████████████████████████████████████████████| 100/100 [00:25<00:00,  3.98it/s]
Iteration 24: Average Reward = 53.01
Iteration 25: 100%|█████████████████████████████████████████████| 100/100 [00:24<00:00,  4.00it/s]
Iteration 25: Average Reward = 56.46
Iteration 26: 100%|█████████████████████████████████████████████| 100/100 [00:24<00:00,  4.13it/s]
Iteration 26: Average Reward = 55.56
Iteration 27: 100%|█████████████████████████████████████████████| 100/100 [00:24<00:00,  4.12it/s]
Iteration 27: Average Reward = 60.13
```

```
Iteration 28: 100%|██████████████████████████████████████████████| 100/100 [00:24<00:00,  4.16it/s]
Iteration 28: Average Reward = 61.95
Iteration 29: 100%|██████████████████████████████████████████████| 100/100 [00:21<00:00,  4.69it/s]
Iteration 29: Average Reward = 69.97
Iteration 30: 100%|██████████████████████████████████████████████| 100/100 [00:21<00:00,  4.67it/s]
Iteration 30: Average Reward = 69.98
Iteration 31: 100%|██████████████████████████████████████████████| 100/100 [00:18<00:00,  5.35it/s]
Iteration 31: Average Reward = 74.29
Iteration 32: 100%|██████████████████████████████████████████████| 100/100 [00:19<00:00,  5.00it/s]
Iteration 32: Average Reward = 72.22
Iteration 33: 100%|██████████████████████████████████████████████| 100/100 [00:18<00:00,  5.37it/s]
Iteration 33: Average Reward = 74.01
Iteration 34: 100%|██████████████████████████████████████████████| 100/100 [00:18<00:00,  5.28it/s]
Iteration 34: Average Reward = 75.48
Iteration 35: 100%|██████████████████████████████████████████████| 100/100 [00:18<00:00,  5.46it/s]
Iteration 35: Average Reward = 77.50
Iteration 36: 100%|██████████████████████████████████████████████| 100/100 [00:16<00:00,  6.07it/s]
Iteration 36: Average Reward = 79.11
Iteration 37: 100%|██████████████████████████████████████████████| 100/100 [00:16<00:00,  6.07it/s]
Iteration 37: Average Reward = 79.62
Iteration 38: 100%|██████████████████████████████████████████████| 100/100 [00:14<00:00,  7.01it/s]
Iteration 38: Average Reward = 81.85
Iteration 39: 100%|██████████████████████████████████████████████| 100/100 [00:14<00:00,  6.68it/s]
Iteration 39: Average Reward = 81.00
Iteration 40: 100%|██████████████████████████████████████████████| 100/100 [00:14<00:00,  7.08it/s]
Iteration 40: Average Reward = 81.75
Iteration 41: 100%|██████████████████████████████████████████████| 100/100 [00:13<00:00,  7.68it/s]
Iteration 41: Average Reward = 83.64
Iteration 42: 100%|██████████████████████████████████████████████| 100/100 [00:14<00:00,  7.05it/s]
Iteration 42: Average Reward = 81.07
Iteration 43: 100%|██████████████████████████████████████████████| 100/100 [00:13<00:00,  7.33it/s]
Iteration 43: Average Reward = 82.79
Iteration 44: 100%|██████████████████████████████████████████████| 100/100 [00:12<00:00,  8.15it/s]
Iteration 44: Average Reward = 84.01
Iteration 45: 100%|██████████████████████████████████████████████| 100/100 [00:12<00:00,  8.07it/s]
Iteration 45: Average Reward = 83.83
Iteration 46: 100%|██████████████████████████████████████████████| 100/100 [00:12<00:00,  7.95it/s]
Iteration 46: Average Reward = 83.68
```

```
Iteration 47: 100%|██████████████████████████████████████████████████| 100/100 [00:11<00:00,  8.72it/s]
Iteration 47: Average Reward = 85.37
Iteration 48: 100%|██████████████████████████████████████████████████| 100/100 [00:11<00:00,  8.63it/s]
Iteration 48: Average Reward = 84.84
Iteration 49: 100%|██████████████████████████████████████████████████| 100/100 [00:11<00:00,  8.99it/s]
Iteration 49: Average Reward = 85.41
Iteration 50: 100%|██████████████████████████████████████████████████| 100/100 [00:11<00:00,  8.94it/s]
Iteration 50: Average Reward = 85.42
Iteration 51: 100%|██████████████████████████████████████████████████| 100/100 [00:11<00:00,  8.94it/s]
Iteration 51: Average Reward = 85.47
Iteration 52: 100%|██████████████████████████████████████████████████| 100/100 [00:10<00:00,  9.73it/s]
Iteration 52: Average Reward = 86.46
Iteration 53: 100%|██████████████████████████████████████████████████| 100/100 [00:10<00:00,  9.86it/s]
Iteration 53: Average Reward = 86.42
Iteration 54: 100%|██████████████████████████████████████████████████| 100/100 [00:09<00:00, 10.57it/s]
Iteration 54: Average Reward = 87.20
Iteration 55: 100%|██████████████████████████████████████████████████| 100/100 [00:09<00:00, 10.35it/s]
Iteration 55: Average Reward = 87.23
Iteration 56: 100%|██████████████████████████████████████████████████| 100/100 [00:09<00:00, 10.36it/s]
Iteration 56: Average Reward = 86.78
Iteration 57: 100%|██████████████████████████████████████████████████| 100/100 [00:09<00:00, 11.00it/s]
Iteration 57: Average Reward = 87.58
Iteration 58: 100%|██████████████████████████████████████████████████| 100/100 [00:08<00:00, 11.50it/s]
Iteration 58: Average Reward = 88.00
Iteration 59: 100%|██████████████████████████████████████████████████| 100/100 [00:08<00:00, 11.69it/s]
Iteration 59: Average Reward = 88.34
Iteration 60: 100%|██████████████████████████████████████████████████| 100/100 [00:07<00:00, 12.71it/s]
Iteration 60: Average Reward = 88.95
Iteration 61: 100%|██████████████████████████████████████████████████| 100/100 [00:08<00:00, 11.70it/s]
Iteration 61: Average Reward = 87.90
Iteration 62: 100%|██████████████████████████████████████████████████| 100/100 [00:08<00:00, 12.25it/s]
Iteration 62: Average Reward = 88.39
Iteration 63: 100%|██████████████████████████████████████████████████| 100/100 [00:07<00:00, 13.16it/s]
Iteration 63: Average Reward = 89.09
Iteration 64: 100%|██████████████████████████████████████████████████| 100/100 [00:07<00:00, 12.77it/s]
Iteration 64: Average Reward = 89.04
Iteration 65: 100%|██████████████████████████████████████████████████| 100/100 [00:07<00:00, 13.57it/s]
Iteration 65: Average Reward = 89.28
```

```
Iteration 66: 100%|██████████| 100/100 [00:07<00:00, 14.05it/s]
Iteration 66: Average Reward = 89.55
Iteration 67: 100%|██████████| 100/100 [00:06<00:00, 14.40it/s]
Iteration 67: Average Reward = 89.67
Iteration 68: 100%|██████████| 100/100 [00:06<00:00, 14.94it/s]
Iteration 68: Average Reward = 89.87
Iteration 69: 100%|██████████| 100/100 [00:06<00:00, 15.44it/s]
Iteration 69: Average Reward = 90.23
Iteration 70: 100%|██████████| 100/100 [00:06<00:00, 14.97it/s]
Iteration 70: Average Reward = 89.85
Iteration 71: 100%|██████████| 100/100 [00:06<00:00, 14.69it/s]
Iteration 71: Average Reward = 90.12
Iteration 72: 100%|██████████| 100/100 [00:06<00:00, 16.01it/s]
Iteration 72: Average Reward = 90.38
Iteration 73: 100%|██████████| 100/100 [00:06<00:00, 15.37it/s]
Iteration 73: Average Reward = 90.17
Iteration 74: 100%|██████████| 100/100 [00:06<00:00, 16.45it/s]
Iteration 74: Average Reward = 90.58
Iteration 75: 100%|██████████| 100/100 [00:06<00:00, 16.34it/s]
Iteration 75: Average Reward = 90.44
Iteration 76: 100%|██████████| 100/100 [00:06<00:00, 15.67it/s]
Iteration 76: Average Reward = 90.18
Iteration 77: 100%|██████████| 100/100 [00:05<00:00, 16.89it/s]
Iteration 77: Average Reward = 90.77
Iteration 78: 100%|██████████| 100/100 [00:05<00:00, 16.80it/s]
Iteration 78: Average Reward = 90.63
Iteration 79: 100%|██████████| 100/100 [00:06<00:00, 15.57it/s]
Iteration 79: Average Reward = 90.46
Iteration 80: 100%|██████████| 100/100 [00:05<00:00, 17.28it/s]
Iteration 80: Average Reward = 90.83
Iteration 81: 100%|██████████| 100/100 [00:05<00:00, 17.73it/s]
Iteration 81: Average Reward = 91.12
Iteration 82: 100%|██████████| 100/100 [00:05<00:00, 18.52it/s]
Iteration 82: Average Reward = 91.39
Iteration 83: 100%|██████████| 100/100 [00:05<00:00, 17.88it/s]
Iteration 83: Average Reward = 91.15
Iteration 84: 100%|██████████| 100/100 [00:05<00:00, 17.52it/s]
Iteration 84: Average Reward = 91.21
```

```
Iteration 85: 100%|██████████████████████████████| 100/100 [00:05<00:00, 17.80it/s]
Iteration 85: Average Reward = 91.21
Iteration 86: 100%|██████████████████████████████| 100/100 [00:05<00:00, 18.52it/s]
Iteration 86: Average Reward = 91.39
Iteration 87: 100%|██████████████████████████████| 100/100 [00:05<00:00, 17.71it/s]
Iteration 87: Average Reward = 91.43
Iteration 88: 100%|██████████████████████████████| 100/100 [00:05<00:00, 18.23it/s]
Iteration 88: Average Reward = 91.29
Iteration 89: 100%|██████████████████████████████| 100/100 [00:05<00:00, 18.44it/s]
Iteration 89: Average Reward = 91.55
Iteration 90: 100%|██████████████████████████████| 100/100 [00:05<00:00, 18.44it/s]
Iteration 90: Average Reward = 91.49
Iteration 91: 100%|██████████████████████████████| 100/100 [00:05<00:00, 18.45it/s]
Iteration 91: Average Reward = 91.49
Iteration 92: 100%|██████████████████████████████| 100/100 [00:05<00:00, 17.46it/s]
Iteration 92: Average Reward = 91.09
Iteration 93: 100%|██████████████████████████████| 100/100 [00:05<00:00, 18.50it/s]
Iteration 93: Average Reward = 91.67
Iteration 94: 100%|██████████████████████████████| 100/100 [00:05<00:00, 17.35it/s]
Iteration 94: Average Reward = 91.56
Iteration 95: 100%|██████████████████████████████| 100/100 [00:05<00:00, 19.02it/s]
Iteration 95: Average Reward = 91.55
Iteration 96: 100%|██████████████████████████████| 100/100 [00:05<00:00, 19.11it/s]
Iteration 96: Average Reward = 91.66
Iteration 97: 100%|██████████████████████████████| 100/100 [00:05<00:00, 18.11it/s]
Iteration 97: Average Reward = 91.32
Iteration 98: 100%|██████████████████████████████| 100/100 [00:05<00:00, 18.92it/s]
Iteration 98: Average Reward = 91.71
Iteration 99: 100%|██████████████████████████████| 100/100 [00:05<00:00, 19.04it/s]
Iteration 99: Average Reward = 91.74
Iteration 100: 100%|█████████████████████████████| 100/100 [00:05<00:00, 19.80it/s]
Iteration 100: Average Reward = 91.95
```



模型已保存为 'actor_critic_mountaincar.pth'