```
!pip install gymnasium
!pip install gymnasium[box2d]
!pip install jupyterlab
!pip install tqdm
```

```
Collecting fqdn (from jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->jupyterlab)
  Downloading fqdn-1.5.1-py3-none-any.whl.metadata (1.4 kB)
Collecting isoduration (from jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->jupyterlab)
  Downloading isoduration-20.11.0-py3-none-any.whl.metadata (5.7 kB)
Requirement already satisfied: jsonpointer>1.13 in /usr/local/lib/python3.11/dist-packages (from jsonschema[format-nongpl]>=4.18.
Collecting uri-template (from jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->jupyterlab)
  Downloading uri_template-1.3.0-py3-none-any.whl.metadata (8.8 kB)
Requirement already satisfied: webcolors>=24.6.0 in /usr/local/lib/python3.11/dist-packages (from jsonschema[format-nongpl]>=4.18
Requirement already satisfied: wcwidth in /usr/local/lib/python3.11/dist-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->jupyter-client>=
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from argon2-cffi-bindings->argon2-cffi>=21
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4->nbconvert>=6.4.4->j
Requirement already satisfied: pycparser in /usr/local/lib/python3.11/dist-packages (from cffi>=1.0.1->argon2-cffi-bindings->argo
Collecting arrow>=0.15.0 (from isoduration->jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->
  Downloading arrow-1.3.0-py3-none-any.whl.metadata (7.5 kB)
Collecting types-python-dateutil>=2.8.10 (from arrow>=0.15.0->isoduration->jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.1
  Downloading types_python_dateutil-2.9.0.20250516-py3-none-any.whl.metadata (2.1 kB)
Downloading jupyterlab-4.4.2-py3-none-any.whl (12.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.3/12.3 MB 96.7 MB/s eta 0:00:00
Downloading async_lru-2.0.5-py3-none-any.whl (6.1 kB)
Downloading jupyter_lsp-2.2.5-py3-none-any.whl (69 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 69.1/69.1 kB 5.0 MB/s eta 0:00:00
Downloading jupyter_server-2.16.0-py3-none-any.whl (386 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 386.9/386.9 kB 22.2 MB/s eta 0:00:00
Downloading jupyterlab_server-2.27.3-py3-none-any.whl (59 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.7/59.7 kB 4.1 MB/s eta 0:00:00
Downloading json5-0.12.0-py3-none-any.whl (36 kB)
Downloading jupyter_client-8.6.3-py3-none-any.whl (106 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 106.1/106.1 kB 7.0 MB/s eta 0:00:00
Downloading jupyter_events-0.12.0-py3-none-any.whl (19 kB)
Downloading jupyter_server_terminals-0.5.3-py3-none-any.whl (13 kB)
Downloading overrides-7.7.0-py3-none-any.whl (17 kB)
Downloading jedi-0.19.2-py2.py3-none-any.whl (1.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 55.6 MB/s eta 0:00:00
Downloading python_json_logger-3.3.0-py3-none-any.whl (15 kB)
Downloading rfc3986_validator-0.1.1-py2.py3-none-any.whl (4.2 kB)
Downloading rfc3339_validator-0.1.4-py2.py3-none-any.whl (3.5 kB)
Downloading fqdn-1.5.1-py3-none-any.whl (9.1 kB)
Downloading isoduration-20.11.0-py3-none-any.whl (11 kB)
Downloading uri_template-1.3.0-py3-none-any.whl (11 kB)
Downloading arrow-1.3.0-py3-none-any.whl (66 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.4/66.4 kB 4.2 MB/s eta 0:00:00
Downloading types_python_dateutil-2.9.0.20250516-py3-none-any.whl (14 kB)
Installing collected packages: uri-template, types-python-dateutil, rfc3986-validator, rfc3339-validator, python-json-logger, ove
  Attempting uninstall: jupyter-client
    Found existing installation: jupyter-client 6.1.12
    Uninstalling jupyter-client-6.1.12:
      Successfully uninstalled jupyter-client-6.1.12
  Attempting uninstall: jupyter-server
    Found existing installation: jupyter-server 1.16.0
    Uninstalling jupyter-server-1.16.0:
      Successfully uninstalled jupyter-server-1.16.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the
jupyter-kernel-gateway 2.5.2 requires jupyter-client<8.0,>=5.2.0, but you have jupyter-client 8.6.3 which is incompatible.
notebook 6.5.7 requires jupyter-client<8,>=5.3.4, but you have jupyter-client 8.6.3 which is incompatible.
Successfully installed arrow-1.3.0 async-lru-2.0.5 fqdn-1.5.1 isoduration-20.11.0 jedi-0.19.2 json5-0.12.0 jupyter-client-8.6.3 j
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (4.67.1)
```

```
import gymnasium as gym
import numpy as np
from itertools import count
from tqdm import tqdm
import time
import matplotlib.pyplot as plt



g_bins = 10
Q_track = 0


def create_bins(n_bins=g_bins, n_dim=4):

    bins = [
        np.linspace(-4.8, 4.8, n_bins),
        np.linspace(-4, 4, n_bins),
        np.linspace(-0.418, 0.418, n_bins),
        np.linspace(-4, 4, n_bins)
    ]
```

```
    return bins


def discretize_state(observation, bins):

    binned_state = []

    for i in range(len(observation)):
        d = np.digitize(observation[i], bins[i])
        binned_state.append( d - 1)

    return tuple(binned_state)


def decay_schedule(
    init_value, min_value, decay_ratio,
    max_steps, log_start = -2, log_base=10):
    decay_steps = int(max_steps*decay_ratio)
    rem_steps = max_steps - decay_steps
    values = np.logspace(
      log_start, 0, decay_steps,
      base = log_base, endpoint = True)[::-1]
    values = (values -values.min())/(values.max() - values.min())
    values = (init_value - min_value)*values +min_value
    values = np.pad(values, (0, rem_steps), 'edge')

    return values


def generate_trajectory(
    select_action, Q, epsilon,
    env, max_steps=200):
    done, trajectory = False, []
    bins = create_bins(g_bins)

    observation,_ = env.reset()
    state = discretize_state(observation, bins)

    for t in count():
        action = select_action(state, Q, epsilon)
        observation, reward, done, _, _ = env.step(action)
        next_state = discretize_state(observation, bins)
        if not done:
            if t >= max_steps-1:
                break
            experience = (state, action,
                    reward, next_state, done)
            trajectory.append(experience)
        else:
            experience = (state, action,
                    -100, next_state, done)
            trajectory.append(experience)
            #time.sleep(2)
            break
        state = next_state

    return np.array(trajectory, dtype=object)




def mc_control (env,n_bins=g_bins, gamma = 1.0,
                init_alpha = 0.5,min_alpha = 0.01, alpha_decay_ratio = 0.5,
                init_epsilon = 1.0, min_epsilon = 0.1, epsilon_decay_ratio = 0.9,
                n_episodes = 3000, max_steps = 200, first_visit = True, init_Q=None):

    nA = env.action_space.n
    discounts = np.logspace(0, max_steps,
                            num = max_steps, base = gamma,
                            endpoint = False)
    alphas = decay_schedule(init_alpha, min_alpha,
                            0.9999, n_episodes)
    epsilons = decay_schedule(init_epsilon, min_epsilon,
                            0.99, n_episodes)
    pi_track = []
    global Q_track
    global Q


    if init_Q is None:
        Q = np.zeros([n_bins]*env.observation_space.shape[0] + [env.action_space.n],dtype =np.float64)
    else:
```

```
            Q = init_Q

        n_elements = Q.size
        n_nonzero_elements = 0

        Q_track = np.zeros([n_episodes] + [n_bins]*env.observation_space.shape[0] + [env.action_space.n],dtype =np.float64)
        select_action = lambda state, Q, epsilon: np.argmax(Q[tuple(state)]) if np.random.random() > epsilon else np.random.randint(len(Q[tu

        progress_bar = tqdm(range(n_episodes), leave=False)
        steps_balanced_total = 1
        mean_steps_balanced = 0
        for e in progress_bar:
            trajectory = generate_trajectory(select_action, Q, epsilons[e],
                                     env, max_steps)

            steps_balanced_total = steps_balanced_total + len(trajectory)
            mean_steps_balanced = 0

            visited = np.zeros([n_bins]*env.observation_space.shape[0] + [env.action_space.n],dtype =np.float64)
            for t, (state, action, reward, _, _) in enumerate(trajectory):
                #if visited[tuple(state)][action] and first_visit:
                #    continue
                visited[tuple(state)][action] = True
                n_steps = len(trajectory[t:])
                G = np.sum(discounts[:n_steps]*trajectory[t:, 2])
                Q[tuple(state)][action] = Q[tuple(state)][action]+alphas[e]*(G - Q[tuple(state)][action])
            Q_track[e] = Q
            n_nonzero_elements = np.count_nonzero(Q)
            pi_track.append(np.argmax(Q, axis=env.observation_space.shape[0]))
            if e != 0:
                mean_steps_balanced = steps_balanced_total/e
            #progress_bar.set_postfix(episode=e, Epsilon=epsilons[e], Steps=f"{len(trajectory)}" ,MeanStepsBalanced=f"{mean_steps_balanced:.
            progress_bar.set_postfix(episode=e, Epsilon=epsilons[e], StepsBalanced=f"{len(trajectory)}" ,MeanStepsBalanced=f"{mean_steps_ba

        print("mean_steps_balanced={0},steps_balanced_total={1}".format(mean_steps_balanced,steps_balanced_total))
        V = np.max(Q, axis=env.observation_space.shape[0])
        pi = lambda s:{s:a for s, a in enumerate(np.argmax(Q, axis=env.observation_space.shape[0]))}[s]

        return Q, V, pi


env = gym.make("CartPole-v1", render_mode="human")
observation, info = env.reset(seed=42)


observation, info = env.reset(seed=42)


observation, reward, done, _, _ = env.step(0)
print(done)
```

```
False
```

```
env.action_space.n
```

```
np.int64(2)
```

```
optimal_Q, optimal_V, optimal_pi = mc_control (env,n_episodes=200)
```

```
mean_steps_ba
```

```
optimal_Q, optimal_V, optimal_pi = mc_control (env,n_episodes=200,
                                    init_alpha = 0.5,min_alpha = 0.01, alpha_decay_ratio = 0.5,
                                    init_epsilon = 1.0, min_epsilon = 0.1, epsilon_decay_ratio = 0.9,
                                    max_steps=500, init_Q=Q)
```

```
mean_steps_b
```

```
optimal_Q, optimal_V, optimal_pi = mc_control (env,n_episodes=500,
                                    init_alpha = 0.01,min_alpha = 0.005, alpha_decay_ratio = 0.5,
                                    init_epsilon = 0.1 , min_epsilon = 0.08, epsilon_decay_ratio = 0.9,
                                    max_steps=500, init_Q=Q)
```

```
mean_steps_
```

```
Q = np.load("state_action_values.npy")
```

```
np.save("state_action_values.npy", Q)
```