```
!pip install gymnasium
```

```
Requirement already satisfied: gymnasium in /usr/local/lib/python3.12/dist-packages (1.2.0)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.12/dist-packages (from gymnasium) (2.0.2)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from gymnasium) (3.1.1)
Requirement already satisfied: typing-extensions>=4.3.0 in /usr/local/lib/python3.12/dist-packages (from gymnasium) (4.15.0)
Requirement already satisfied: farama-notifications>=0.0.1 in /usr/local/lib/python3.12/dist-packages (from gymnasium) (0.0.4)
```

```
pip install git+https://github.com/mimoralea/gym-walk#egg=gym-walk
```

```
Collecting gym-walk
  Cloning https://github.com/mimoralea/gym-walk to /tmp/pip-install-saudrrye/gym-walk_88175d5cabc746a8b8dabd0b7d18f915
  Running command git clone --filter=blob:none --quiet https://github.com/mimoralea/gym-walk /tmp/pip-install-saudrrye/gym-wal
  Resolved https://github.com/mimoralea/gym-walk to commit b915b94cf2ad16f8833a1ad92ea94e88159279f5
  Preparing metadata (setup.py) ... done
Requirement already satisfied: gym in /usr/local/lib/python3.12/dist-packages (from gym-walk) (0.25.2)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.12/dist-packages (from gym->gym-walk) (2.0.2)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from gym->gym-walk) (3.1.1)
Requirement already satisfied: gym-notices>=0.0.4 in /usr/local/lib/python3.12/dist-packages (from gym->gym-walk) (0.1.0)
Building wheels for collected packages: gym-walk
  Building wheel for gym-walk (setup.py) ... done
  Created wheel for gym-walk: filename=gym_walk-0.0.2-py3-none-any.whl size=5377 sha256=a9ca47f2bce0f9f7dc1ef3fec65a06e404fac9
  Stored in directory: /tmp/pip-ephem-wheel-cache-vp4ra7on/wheels/bf/23/e5/a94be4a90dd18f7ce958c21f192276cb01ef0daaf2bc66583b
Successfully built gym-walk
Installing collected packages: gym-walk
Successfully installed gym-walk-0.0.2
```

```python
import warnings ; warnings.filterwarnings('ignore')

import gym, gym_walk
import numpy as np

import random
import warnings

warnings.filterwarnings('ignore', category=DeprecationWarning)
np.set_printoptions(suppress=True)
random.seed(123); np.random.seed(123)

```

```
Gym has been unmaintained since 2022 and does not support NumPy 2.0 amongst other critical functionality.
Please upgrade to Gymnasium, the maintained drop-in replacement of Gym, or contact the authors of your software and request tha
See the migration guide at https://gymnasium.farama.org/introduction/migration_guide/ for additional information.
```

```python
def print_policy(pi, P, action_symbols=('<', 'v', '>', '^'), n_cols=4, title='Policy:'):
    print(title)
    arrs = {k:v for k,v in enumerate(action_symbols)}
    for s in range(len(P)):
        a = pi(s)
        print("| ", end="")
        if np.all([done for action in P[s].values() for _, _, _, done in action]):
            print("".rjust(9), end=" ")
        else:
            print(str(s).zfill(2), arrs[a].rjust(6), end=" ")
        if (s + 1) % n_cols == 0: print("|")
```

```python
def print_state_value_function(V, P, n_cols=4, prec=3, title='State-value function:'):
    print(title)
    for s in range(len(P)):
        v = V[s]
        print("| ", end="")
        if np.all([done for action in P[s].values() for _, _, _, done in action]):
            print("".rjust(9), end=" ")
        else:
            print(str(s).zfill(2), '{}'.format(np.round(v, prec)).rjust(6), end=" ")
        if (s + 1) % n_cols == 0: print("|")
```

```python
def probability_success(env, pi, goal_state, n_episodes=100, max_steps=200):
    random.seed(123); np.random.seed(123) ; env.seed(123)
    results = []
    for _ in range(n_episodes):
        state, done, steps = env.reset(), False, 0
        while not done and steps < max_steps:
            state, _, done, h = env.step(pi(state))
            steps += 1
        results.append(state == goal_state)
    return np.sum(results)/len(results)
```

```
def mean_return(env, pi, n_episodes=100, max_steps=200):
    random.seed(123); np.random.seed(123) ; env.seed(123)
    results = []
    for _ in range(n_episodes):
        state, done, steps = env.reset(), False, 0
        results.append(0.0)
        while not done and steps < max_steps:
            state, reward, done, _ = env.step(pi(state))
            results[-1] += reward
            steps += 1
    return np.mean(results)
```

```
env = gym.make('FrozenLake-v1')
P = env.env.P
init_state = env.reset()
goal_state = 15
LEFT, DOWN, RIGHT, UP = range(4)

```

```
P
```

```
   (0.3333333333333333, 13, 0.0, False)],
  1: [(0.3333333333333333, 8, 0.0, False),
   (0.3333333333333333, 13, 0.0, False),
   (0.3333333333333333, 10, 0.0, False)],
  2: [(0.3333333333333333, 13, 0.0, False),
   (0.3333333333333333, 10, 0.0, False),
   (0.3333333333333333, 5, 0.0, True)],
  3: [(0.3333333333333333, 10, 0.0, False),
   (0.3333333333333333, 5, 0.0, True),
   (0.3333333333333333, 8, 0.0, False)]},
 10: {0: [(0.3333333333333333, 6, 0.0, False),
   (0.3333333333333333, 9, 0.0, False),
   (0.3333333333333333, 14, 0.0, False)],
  1: [(0.3333333333333333, 9, 0.0, False),
   (0.3333333333333333, 14, 0.0, False),
   (0.3333333333333333, 11, 0.0, True)],
  2: [(0.3333333333333333, 14, 0.0, False),
   (0.3333333333333333, 11, 0.0, True),
   (0.3333333333333333, 6, 0.0, False)],
  3: [(0.3333333333333333, 11, 0.0, True),
   (0.3333333333333333, 6, 0.0, False),
   (0.3333333333333333, 9, 0.0, False)]},
 11: {0: [(1.0, 11, 0, True)],
  1: [(1.0, 11, 0, True)],
  2: [(1.0, 11, 0, True)],
  3: [(1.0, 11, 0, True)]},
 12: {0: [(1.0, 12, 0, True)],
  1: [(1.0, 12, 0, True)],
  2: [(1.0, 12, 0, True)],
  3: [(1.0, 12, 0, True)]},
 13: {0: [(0.3333333333333333, 9, 0.0, False),
   (0.3333333333333333, 12, 0.0, True),
   (0.3333333333333333, 13, 0.0, False)],
  1: [(0.3333333333333333, 12, 0.0, True),
   (0.3333333333333333, 13, 0.0, False),
   (0.3333333333333333, 14, 0.0, False)],
  2: [(0.3333333333333333, 13, 0.0, False),
   (0.3333333333333333, 14, 0.0, False),
   (0.3333333333333333, 9, 0.0, False)],
  3: [(0.3333333333333333, 14, 0.0, False),
   (0.3333333333333333, 9, 0.0, False),
   (0.3333333333333333, 12, 0.0, True)]},
 14: {0: [(0.3333333333333333, 10, 0.0, False),
   (0.3333333333333333, 13, 0.0, False),
   (0.3333333333333333, 14, 0.0, False)],
  1: [(0.3333333333333333, 13, 0.0, False),
   (0.3333333333333333, 14, 0.0, False),
   (0.3333333333333333, 15, 1.0, True)],
  2: [(0.3333333333333333, 14, 0.0, False),
   (0.3333333333333333, 15, 1.0, True),
   (0.3333333333333333, 10, 0.0, False)],
  3: [(0.3333333333333333, 15, 1.0, True),
   (0.3333333333333333, 10, 0.0, False),
   (0.3333333333333333, 13, 0.0, False)]},
 15: {0: [(1.0, 15, 0, True)],
  1: [(1.0, 15, 0, True)],
  2: [(1.0, 15, 0, True)],
  3: [(1.0, 15, 0, True)]}}
```

```
init_state
```

```
0
```

```
state, reward, done, info = env.step(RIGHT)
print("state:{0} - reward:{1} - done:{2} - info:{3}".format(state, reward, done, info))
```

```
state:4 - reward:0.0 - done:False - info:{'prob': 0.3333333333333333}
```

```
pi_frozenlake1 = lambda s: {
    0: RIGHT,
    1: RIGHT,
    2: RIGHT,
    3: RIGHT,
    4: RIGHT,
    5: RIGHT,
    6: RIGHT,
    7: RIGHT,
    8: RIGHT,
    9: RIGHT,
    10:RIGHT,
    11:RIGHT,
    12:RIGHT,
    13:RIGHT,
    14:RIGHT,
    15:RIGHT #Stop
}[s]
print("Name: JERUSHLIN JOSE JB ")
print("Register Number: 212222240039 ")
print_policy(pi_frozenlake1, P, action_symbols=('<', 'v', '>', '^'), n_cols=4)
```

```
Name: JERUSHLIN JOSE JB
Register Number: 212222240039
Policy:
| 00      > | 01      > | 02      > | 03      > |
| 04      > |           | 06      > |           |
| 08      > | 09      > | 10      > |           |
|           | 13      > | 14      > |           |
```

```
print('Reaches goal {:.2f}%. Obtains an average undiscounted return of {:.4f}.'.format(
    probability_success(env, pi_frozenlake1, goal_state=goal_state)*100,
    mean_return(env, pi_frozenlake1)))
```

```
Reaches goal 1.00%. Obtains an average undiscounted return of 0.0100.
```

```
def policy_evaluation(pi, P, gamma=1.0, theta=1e-10):
    prev_V = np.zeros(len(P), dtype=np.float64)
    while True:
        V = np.zeros(len(P), dtype=np.float64)
        for s in range(len(P)):
            for prob, next_state, reward, done in P[s][pi(s)]:
                V[s] += prob * (reward + gamma * prev_V[next_state] * (not done))
        if np.max(np.abs(prev_V - V)) < theta:
            break
        prev_V = V.copy()
    return V
```

```
V1 = policy_evaluation(pi_frozenlake1, P)
print("Name: JERUSHLIN JOSE JB ")
print("Register Number: 212222240039 ")
print_state_value_function(V1, P, n_cols=4, prec=5)
```

```
Name: JERUSHLIN JOSE JB
Register Number: 212222240039
State-value function:
| 00 0.0315 | 01 0.02381 | 02 0.04762 | 03    0.0 |
| 04 0.03919 |           | 06 0.09524 |           |
| 08 0.08608 | 09 0.21905 | 10 0.2381 |           |
|           | 13 0.41905 | 14 0.61905 |           |
```

```
def policy_improvement(V, P, gamma=1.0):
    Q = np.zeros((len(P), len(P[0])), dtype=np.float64)
    # Write your code here to improve the given policy
    for s in range(len(P)):
      for a in range(len(P[s])):
        for prob,next_state,reward,done in P[s][a]:
          Q[s][a]+=prob*(reward+gamma*V[next_state]*(not done))
    new_pi=lambda s:{s:a for s, a in enumerate(np.argmax(Q,axis=1))}[s]
    return new_pi
```

```
pi_2 = policy_improvement(V1, P)
print("Name: JERUSHLIN JOSE JB ")
print("Register Number: 212222240039 ")
print_policy(pi_2, P, action_symbols=('<', 'v', '>', '^'), n_cols=4)
```

```
Name: JERUSHLIN JOSE JB
Register Number: 212222240039
Policy:
| 00      < | 01      ^ | 02      < | 03      < |
| 04      < |           | 06      < |           |
| 08      ^ | 09      v | 10      < |           |
|           | 13      > | 14      v |           |
```

```
print('Reaches goal {:.2f}%. Obtains an average undiscounted return of {:.4f}.'.format(
    probability_success(env, pi_2, goal_state=goal_state)*100,
    mean_return(env, pi_2)))
```

```
Reaches goal 66.00%. Obtains an average undiscounted return of 0.6600.
```

```
V2 = policy_evaluation(pi_2, P)
print("Name: JERUSHLIN JOSE JB ")
print("Register Number: 212222240039 ")
print_state_value_function(V2, P, n_cols=4, prec=5)
```

```
Name: JERUSHLIN JOSE JB
Register Number: 212222240039
State-value function:
| 00 0.78049 | 01 0.65854 | 02 0.53659 | 03 0.26829 |
| 04 0.78049 |            | 06 0.41463 |            |
| 08 0.78049 | 09 0.78049 | 10 0.70732 |            |
|            | 13 0.85366 | 14 0.92683 |            |
```

```
if(np.sum(V1>=V2)==16):
  print("The Adversarial policy is the better policy")
elif(np.sum(V2>=V1)==16):
  print("The Improved policy is the better policy")
else:
  print("Both policies have their merits.")
```

```
The Improved policy is the better policy
```

```
def policy_iteration(P, gamma=1.0, theta=1e-10):
   random_actions=np.random.choice(tuple(P[0].keys()),len(P))
   pi = lambda s: {s:a for s, a in enumerate(random_actions)}[s]
   while True:
    old_pi = {s:pi(s) for s in range(len(P))}
    V = policy_evaluation(pi, P,gamma,theta)
    pi = policy_improvement(V,P,gamma)
    if old_pi == {s:pi(s) for s in range(len(P))}:
      break
   return V, pi
```

```
optimal_V, optimal_pi = policy_iteration(P)

```

```
print("Name: JERUSHLIN JOSE JB ")
print("Register Number: 212222240039 ")
print('Optimal policy and state-value function (PI):')
print_policy(optimal_pi, P, action_symbols=('<', '>', 'v', '^'), n_cols=4)
```

```
Name: JERUSHLIN JOSE JB
Register Number: 212222240039
Optimal policy and state-value function (PI):
Policy:
| 00      < | 01      ^ | 02      ^ | 03      ^ |
| 04      < |           | 06      < |           |
| 08      ^ | 09      > | 10      < |           |
|           | 13      v | 14      > |           |
```

```
print('Reaches goal {:.2f}%. Obtains an average undiscounted return of {:.4f}.'.format(
    probability_success(env, optimal_pi, goal_state=goal_state)*100,
    mean_return(env, optimal_pi)))
```

```
Reaches goal 69.00%. Obtains an average undiscounted return of 0.6900.
```

```
print("Name: JERUSHLIN JOSE JB ")
print("Register Number: 212222240039 ")
print_state_value_function(optimal_V, P, n_cols=4, prec=5)
```

```
Name: JERUSHLIN JOSE JB
Register Number: 212222240039
State-value function:
| 00 0.82353 | 01 0.82353 | 02 0.82353 | 03 0.82353 |
| 04 0.82353 |            | 06 0.52941 |            |
| 08 0.82353 | 09 0.82353 | 10 0.76471 |            |
|            | 13 0.88235 | 14 0.94118 |            |
```

```
Start coding or generate with AI.
```