In [1]:
```
import warnings ; warnings.filterwarnings('ignore')

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)
```

In [2]:
```
!pip install git+https://github.com/mimoralea/gym-walk#egg=gym-walk
!pip install gymnasium
```

```
Collecting gym-walk
  Cloning https://github.com/mimoralea/gym-walk to /tmp/pip-install-zus6n1ne/gy
m-walk_52ae546ff8b6490a8ad0fef2239a46f0
  Running command git clone --filter=blob:none --quiet https://github.com/mimor
alea/gym-walk /tmp/pip-install-zus6n1ne/gym-walk_52ae546ff8b6490a8ad0fef2239a46
f0
  Resolved https://github.com/mimoralea/gym-walk to commit b915b94cf2ad16f8833a
1ad92ea94e88159279f5
  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=537
7 sha256=bf9bece910fd72b594a5a104552a7b3792d3d15ec93dd08bcead6308b71351dd
  Stored in directory: /tmp/pip-ephem-wheel-cache-fbn6pewl/wheels/bf/23/e5/a94b
e4a90dd18f7ce958c21f192276cb01ef0daaf2bc66583b
Successfully built gym-walk
Installing collected packages: gym-walk
Successfully installed gym-walk-0.0.2
Requirement already satisfied: gymnasium in /usr/local/lib/python3.12/dist-pack
ages (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/pytho
n3.12/dist-packages (from gymnasium) (4.15.0)
Requirement already satisfied: farama-notifications>=0.0.1 in /usr/local/lib/py
thon3.12/dist-packages (from gymnasium) (0.0.4)
```

In [3]:
```
import gymnasium as gym, gym_walk # Moved imports here
import numpy as np # Keep numpy here
```

```
def print_policy(pi, P, action_symbols=('<', 'v', '>', '^'), n_cols=4, title='
    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 actic
            print("".rjust(9), end=" ")
        else:
            print(str(s).zfill(2), arrs[a].rjust(6), end=" ")
        if (s + 1) % n_cols == 0: print("|")
```

```
Gym has been unmaintained since 2022 and does not support NumPy 2.0 amongst oth
er critical functionality.
Please upgrade to Gymnasium, the maintained drop-in replacement of Gym, or cont
act the authors of your software and request that they upgrade.
See the migration guide at https://gymnasium.farama.org/introduction/migratio
n_guide/ for additional information.
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203: Deprecat
ionWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal
in a future version. Use timezone-aware objects to represent datetimes in UTC:
datetime.datetime.now(datetime.UTC).
  return datetime.utcnow().replace(tzinfo=utc)
```

In [4]:
```
def print_state_value_function(V, P, n_cols=4, prec=3, title='State-value func
    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 actic
            print("".rjust(9), end=" ")
        else:
            print(str(s).zfill(2), '{}'.format(np.round(v, prec)).rjust(6), er
        if (s + 1) % n_cols == 0: print("|")
```

In [5]:
```
def probability_success(env, pi, goal_state, n_episodes=100, max_steps=200):
    random.seed(123); np.random.seed(123)
    results = []
    for _ in range(n_episodes):
        state, _ = env.reset(seed=123) # Use reset with seed
        done, steps = False, 0
        while not done and steps < max_steps:
            next_state, reward, terminated, truncated, info = env.step(pi(stat
            done = terminated or truncated # Use terminated or truncated for d
            state = next_state # Update state
            steps += 1
        results.append(state == goal_state)
    return np.sum(results)/len(results)
```

In [6]:
```
def mean_return(env, pi, n_episodes=100, max_steps=200):
    random.seed(123); np.random.seed(123)
    results = []
    for _ in range(n_episodes):
```

```
        state, _ = env.reset(seed=123) # Use reset with seed
        done, steps = False, 0
        results.append(0.0)
        while not done and steps < max_steps:
            next_state, reward, terminated, truncated, info = env.step(pi(stat
            done = terminated or truncated # Use terminated or truncated for d
            results[-1] += reward
            state = next_state # Update state
            steps += 1
    return np.mean(results)
```

In [14]:
```
envdesc = envdesc = ['FSHF','FFFH','FFHF', 'GFFH']
env = gym.make('FrozenLake-v1',desc=envdesc)
init_state, _ = env.reset() # env.reset() now returns observation and info
goal_state = 12
# Access the unwrapped environment to get the P attribute
P = env.unwrapped.P
```

In [15]:
```
P
```

Out[15]:
```
{0: {0: [(0.3333333333333333, 0, 0.0, False),
   (0.3333333333333333, 0, 0.0, False),
   (0.3333333333333333, 4, 0.0, False)],
  1: [(0.3333333333333333, 0, 0.0, False),
   (0.3333333333333333, 4, 0.0, False),
   (0.3333333333333333, 1, 0.0, False)],
  2: [(0.3333333333333333, 4, 0.0, False),
   (0.3333333333333333, 1, 0.0, False),
   (0.3333333333333333, 0, 0.0, False)],
  3: [(0.3333333333333333, 1, 0.0, False),
   (0.3333333333333333, 0, 0.0, False),
   (0.3333333333333333, 0, 0.0, False)]},
 1: {0: [(0.3333333333333333, 1, 0.0, False),
   (0.3333333333333333, 0, 0.0, False),
   (0.3333333333333333, 5, 0.0, False)],
  1: [(0.3333333333333333, 0, 0.0, False),
   (0.3333333333333333, 5, 0.0, False),
   (0.3333333333333333, 2, 0.0, True)],
  2: [(0.3333333333333333, 5, 0.0, False),
   (0.3333333333333333, 2, 0.0, True),
   (0.3333333333333333, 1, 0.0, False)],
  3: [(0.3333333333333333, 2, 0.0, True),
   (0.3333333333333333, 1, 0.0, False),
   (0.3333333333333333, 0, 0.0, False)]},
 2: {0: [(1.0, 2, 0, True)],
  1: [(1.0, 2, 0, True)],
  2: [(1.0, 2, 0, True)],
  3: [(1.0, 2, 0, True)]},
 3: {0: [(0.3333333333333333, 3, 0.0, False),
   (0.3333333333333333, 2, 0.0, True),
   (0.3333333333333333, 7, 0.0, True)],
  1: [(0.3333333333333333, 2, 0.0, True),
   (0.3333333333333333, 7, 0.0, True),
   (0.3333333333333333, 3, 0.0, False)],
  2: [(0.3333333333333333, 7, 0.0, True),
   (0.3333333333333333, 3, 0.0, False),
   (0.3333333333333333, 3, 0.0, False)],
  3: [(0.3333333333333333, 3, 0.0, False),
   (0.3333333333333333, 3, 0.0, False),
   (0.3333333333333333, 2, 0.0, True)]},
 4: {0: [(0.3333333333333333, 0, 0.0, False),
   (0.3333333333333333, 4, 0.0, False),
   (0.3333333333333333, 8, 0.0, False)],
  1: [(0.3333333333333333, 4, 0.0, False),
   (0.3333333333333333, 8, 0.0, False),
   (0.3333333333333333, 5, 0.0, False)],
  2: [(0.3333333333333333, 8, 0.0, False),
   (0.3333333333333333, 5, 0.0, False),
   (0.3333333333333333, 0, 0.0, False)],
  3: [(0.3333333333333333, 5, 0.0, False),
   (0.3333333333333333, 0, 0.0, False),
   (0.3333333333333333, 4, 0.0, False)]},
 5: {0: [(0.3333333333333333, 1, 0.0, False),
   (0.3333333333333333, 4, 0.0, False),
```

```
  (0.3333333333333333, 9, 0.0, False)],
 1: [(0.3333333333333333, 4, 0.0, False),
  (0.3333333333333333, 9, 0.0, False),
  (0.3333333333333333, 6, 0.0, False)],
 2: [(0.3333333333333333, 9, 0.0, False),
  (0.3333333333333333, 6, 0.0, False),
  (0.3333333333333333, 1, 0.0, False)],
 3: [(0.3333333333333333, 6, 0.0, False),
  (0.3333333333333333, 1, 0.0, False),
  (0.3333333333333333, 4, 0.0, False)]},
6: {0: [(0.3333333333333333, 2, 0.0, True),
  (0.3333333333333333, 5, 0.0, False),
  (0.3333333333333333, 10, 0.0, True)],
 1: [(0.3333333333333333, 5, 0.0, False),
  (0.3333333333333333, 10, 0.0, True),
  (0.3333333333333333, 7, 0.0, True)],
 2: [(0.3333333333333333, 10, 0.0, True),
  (0.3333333333333333, 7, 0.0, True),
  (0.3333333333333333, 2, 0.0, True)],
 3: [(0.3333333333333333, 7, 0.0, True),
  (0.3333333333333333, 2, 0.0, True),
  (0.3333333333333333, 5, 0.0, False)]},
7: {0: [(1.0, 7, 0, True)],
 1: [(1.0, 7, 0, True)],
 2: [(1.0, 7, 0, True)],
 3: [(1.0, 7, 0, True)]},
8: {0: [(0.3333333333333333, 4, 0.0, False),
  (0.3333333333333333, 8, 0.0, False),
  (0.3333333333333333, 12, 1.0, True)],
 1: [(0.3333333333333333, 8, 0.0, False),
  (0.3333333333333333, 12, 1.0, True),
  (0.3333333333333333, 9, 0.0, False)],
 2: [(0.3333333333333333, 12, 1.0, True),
  (0.3333333333333333, 9, 0.0, False),
  (0.3333333333333333, 4, 0.0, False)],
 3: [(0.3333333333333333, 9, 0.0, False),
  (0.3333333333333333, 4, 0.0, False),
  (0.3333333333333333, 8, 0.0, False)]},
9: {0: [(0.3333333333333333, 5, 0.0, False),
  (0.3333333333333333, 8, 0.0, False),
  (0.3333333333333333, 13, 0.0, False)],
 1: [(0.3333333333333333, 8, 0.0, False),
  (0.3333333333333333, 13, 0.0, False),
  (0.3333333333333333, 10, 0.0, True)],
 2: [(0.3333333333333333, 13, 0.0, False),
  (0.3333333333333333, 10, 0.0, True),
  (0.3333333333333333, 5, 0.0, False)],
 3: [(0.3333333333333333, 10, 0.0, True),
  (0.3333333333333333, 5, 0.0, False),
  (0.3333333333333333, 8, 0.0, False)]},
10: {0: [(1.0, 10, 0, True)],
 1: [(1.0, 10, 0, True)],
 2: [(1.0, 10, 0, True)],
 3: [(1.0, 10, 0, True)]},
```

```
 11: {0: [(0.3333333333333333, 7, 0.0, True),
   (0.3333333333333333, 10, 0.0, True),
   (0.3333333333333333, 15, 0.0, True)],
  1: [(0.3333333333333333, 10, 0.0, True),
   (0.3333333333333333, 15, 0.0, True),
   (0.3333333333333333, 11, 0.0, False)],
  2: [(0.3333333333333333, 15, 0.0, True),
   (0.3333333333333333, 11, 0.0, False),
   (0.3333333333333333, 7, 0.0, True)],
  3: [(0.3333333333333333, 11, 0.0, False),
   (0.3333333333333333, 7, 0.0, True),
   (0.3333333333333333, 10, 0.0, 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, 1.0, True),
   (0.3333333333333333, 13, 0.0, False)],
  1: [(0.3333333333333333, 12, 1.0, 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, 1.0, True)]},
 14: {0: [(0.3333333333333333, 10, 0.0, True),
   (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, 0.0, True)],
  2: [(0.3333333333333333, 14, 0.0, False),
   (0.3333333333333333, 15, 0.0, True),
   (0.3333333333333333, 10, 0.0, True)],
  3: [(0.3333333333333333, 15, 0.0, True),
   (0.3333333333333333, 10, 0.0, True),
   (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)]}}
```

In [16]:

```python
def value_iteration(P, gamma=1.0, theta=1e-10):
    V = np.zeros(len(P), dtype=np.float64)
    while True:
        Q = np.zeros((len(P), len(P[0])), dtype=np.float64)
        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))
```

```
        if np.max(np.abs(V-np.max(Q, axis=1))) < theta:
            break
        V = np.max(Q, axis=1)
    pi= lambda s: {s:a for s, a in enumerate(np.argmax(Q, axis=1))}[s]

    return V, pi
```

In [17]:
```
# Finding the optimal policy
V_best_v, pi_best_v = value_iteration(P, gamma=0.99)
```

In [18]:
```
# Printing the policy
print('Optimal policy and state-value function (VI):')
print_policy(pi_best_v, P)
```

```
Optimal policy and state-value function (VI):
Policy:
| 00      < | 01      < |           | 03      < |
| 04      v | 05      < | 06      < |           |
| 08      v | 09      < |           | 11      < |
|           | 13      < | 14      < |           |
```

In [19]:
```
# printing the success rate and the mean return
print('Reaches goal {:.2f}%. Obtains an average undiscounted return of {:.4f}.
    probability_success(env, pi_best_v, goal_state=goal_state)*100,
    mean_return(env, pi_best_v)))
```

```
Reaches goal 100.00%. Obtains an average undiscounted return of 1.0000.
```

In [20]:
```
# printing the state value function
print_state_value_function(V_best_v, P, prec=4)
```

```
State-value function:
| 00 0.8822 | 01 0.8744 |           | 03    0.0 |
| 04 0.9089 | 05 0.8932 | 06 0.2947 |           |
| 08 0.9522 | 09 0.9232 |           | 11    0.0 |
|           | 13 0.9522 | 14  0.469 |           |
```