

112-1 (Fall 2023) Semester

# Reinforcement Learning

## Assignment #3

TA: Chen-Yu Lin (林辰宇)

TA: Wei-Hsu Lee (李威緒)

---

Department of Electrical Engineering  
National Taiwan University

# Outline

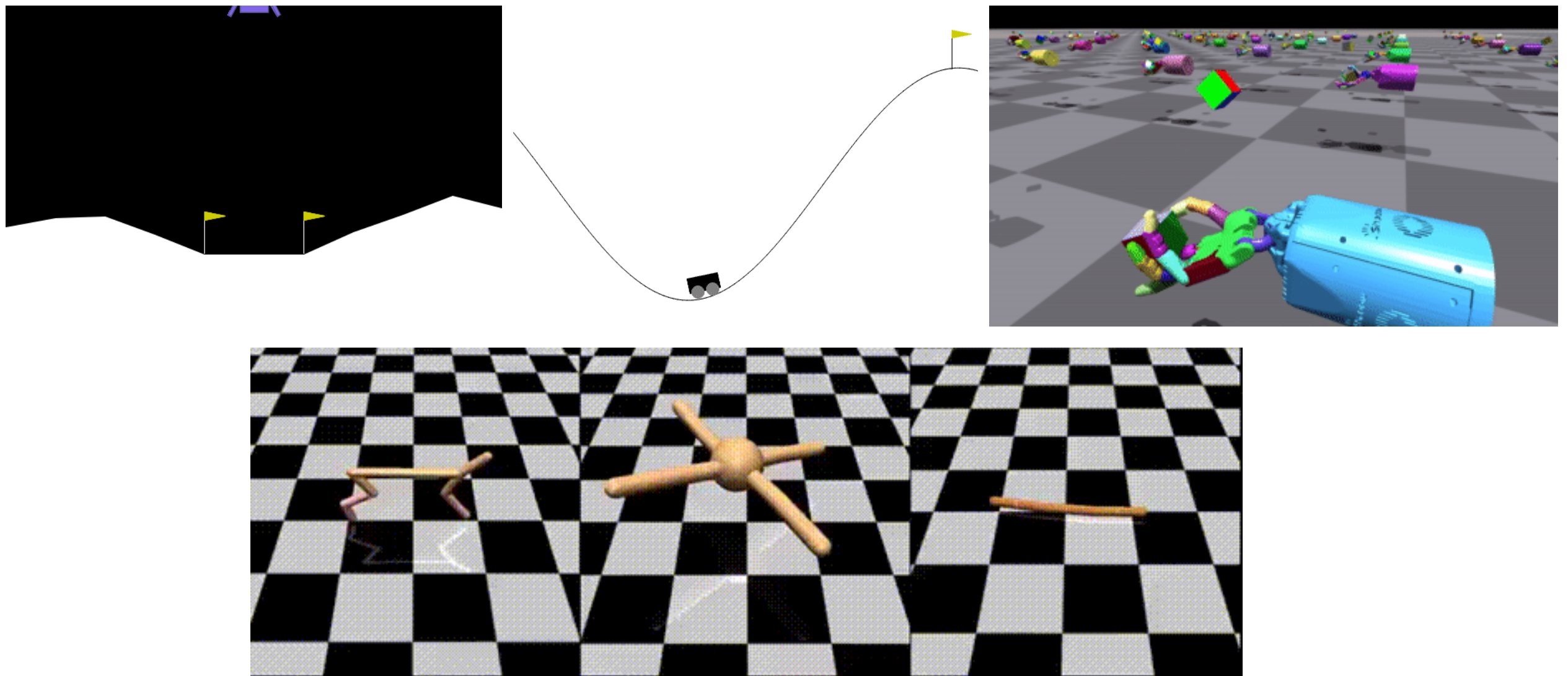
---

- Stable baseline 3
- Tasks
  - Implement environment dynamic
  - Speed up environment
- Environment
- Code structure
- Report
- Grading
- Submission
- Policy
- Contact

# Stable Baselines 3

# Environment Packages

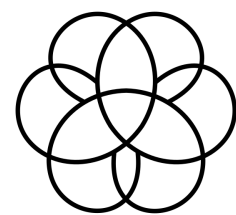
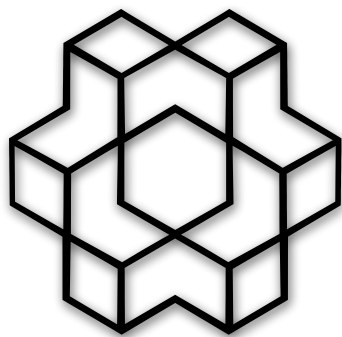
- There are many open-source environment packages for RL training.
- Examples: “OpenAI Gym”, “Deepmind Control”, “Isaac Gym”, etc.
- For this assignment, we will mainly focus on “OpenAI Gym” for its compatibility with “Stable Baselines 3”.



# OpenAI Gym / Gymnasium

---

- Gym is an open source package developed by OpenAI to provide a standard API to communicate between learning algorithms and environments.
- The Farama Foundation announced the release of Gymnasium by October 2022 and took place in the future maintenance of OpenAI Gym. ([Announcing The Farama Foundation](#))
- For this assignment, we will be using Gymnasium.
- [OpenAI Gym](#) / [Gymnasium](#)



Gymnasium

# Environment Methods and Tips

---

- Methods: (For Gymnasium or Gym version after v0.26.0)
  - step(action): Interacts with the environment. Returns obs, rewards, termination, truncation, info.
    - “True” if episode ends because it reaches max episode length.
    - “True” if the episode ends due to termination conditions. (ex: Agent made illegal moves or finished a task)
  - reset(): Resets the environment. Returns obs, info.
    - A dictionary of information that might be useful. (ex: {“success”: False, “score”: 42})
- Note: For a customized Gym environment, you need to first register your environment with `gym.envs.register()` to use `gym.make(“ENV_NAME”)` for returning env object.

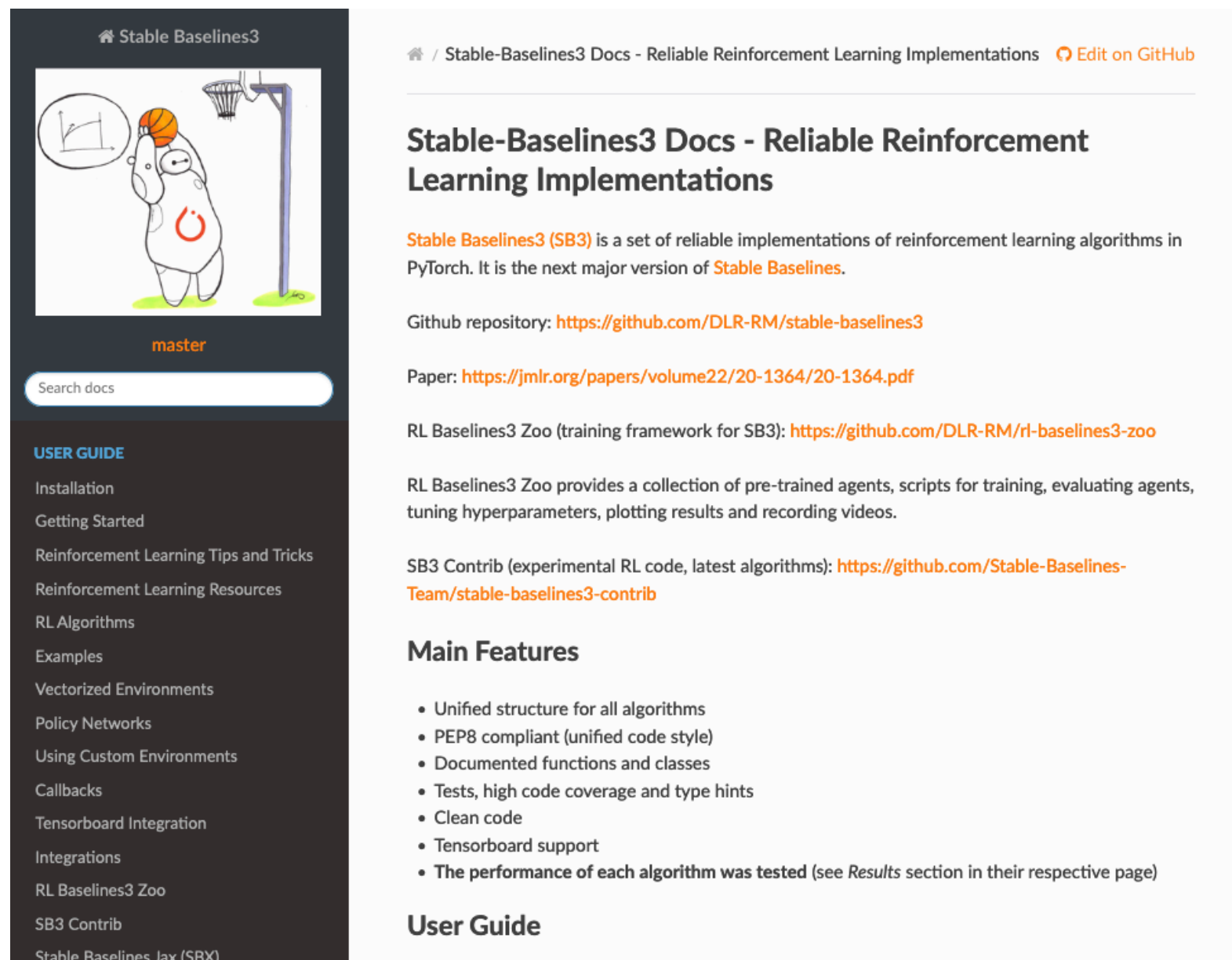
# Environment Methods and Tips

---

- There was a major update for the Gym API after version 0.26.0
- If you're using environments from older versions of Gym, you might want to watch out for compatibility issues.
- [Release notes for v0.26.0](#)

# Stable Baselines 3

- Stable Baselines is a package that provides training APIs for RL training.
- Useful for training Gym environment on basic RL algorithms.
- [Documentation](#)



Stable Baselines3

Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations [Edit on GitHub](#)

## Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations

**Stable Baselines3 (SB3)** is a set of reliable implementations of reinforcement learning algorithms in PyTorch. It is the next major version of **Stable Baselines**.

Github repository: <https://github.com/DLR-RM/stable-baselines3>

Paper: <https://jmlr.org/papers/volume22/20-1364/20-1364.pdf>

RL Baselines3 Zoo (training framework for SB3): <https://github.com/DLR-RM/rl-baselines3-zoo>

RL Baselines3 Zoo provides a collection of pre-trained agents, scripts for training, evaluating agents, tuning hyperparameters, plotting results and recording videos.

SB3 Contrib (experimental RL code, latest algorithms): <https://github.com/Stable-Baselines-Team/stable-baselines3-contrib>

### Main Features

- Unified structure for all algorithms
- PEP8 compliant (unified code style)
- Documented functions and classes
- Tests, high code coverage and type hints
- Clean code
- Tensorboard support
- The performance of each algorithm was tested (see *Results* section in their respective page)

### User Guide



# Vectorized Environments

---

- [Vectorized Environments](#) are used to stack multiple independent environments into a single environment.
- Training the agent on a parallel environment helps speed up the training process.
- You can use VecEnv for training and still use the original Env for evaluation.
- Note: VecEnv wraps the environment with Gym 0.21 API logic, so:
  - Output of `VecEnv.step()` would be obs, reward, termination, info instead.
  - Output of `VecEnv.reset()` would be obs only.(See [VecEnv API vs Gym API](#))

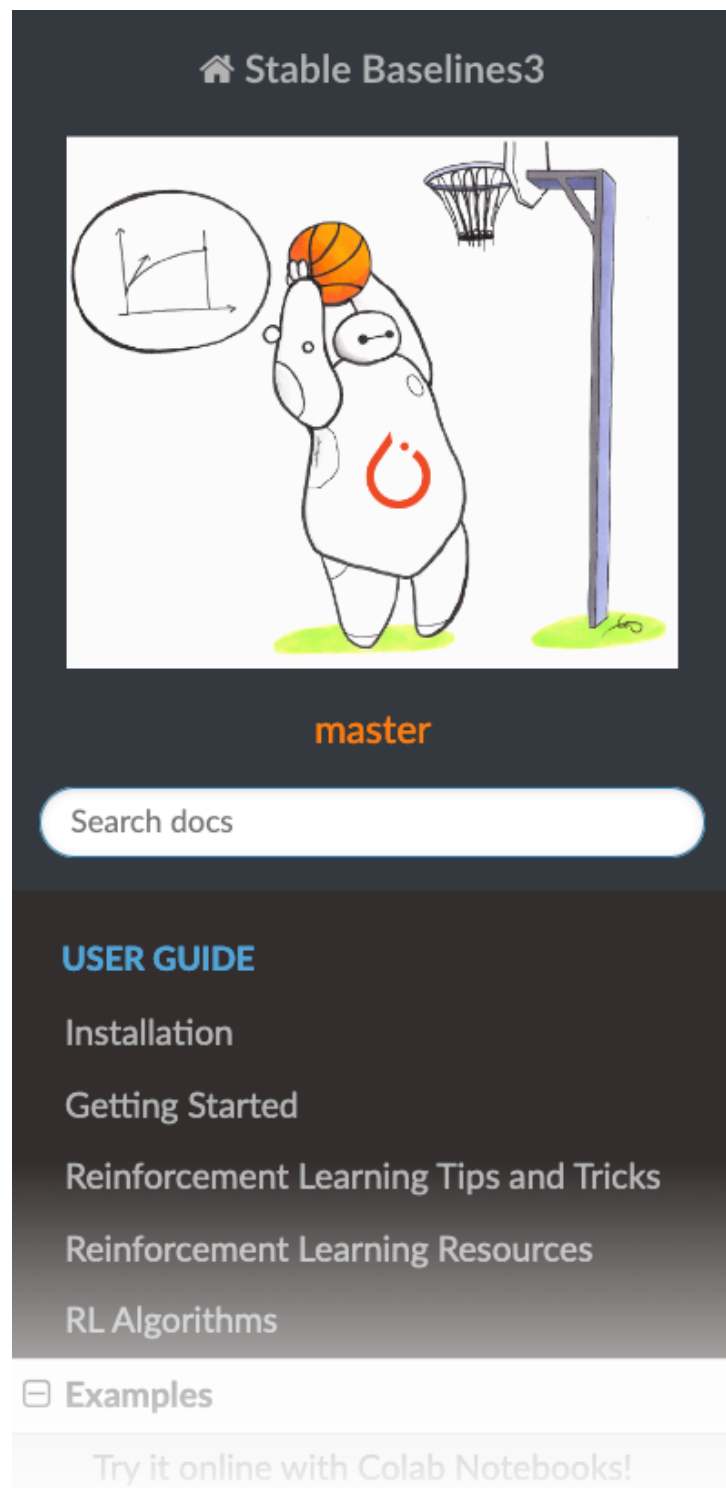
# Supported Algorithms

---

- [A2C](#) (Asynchronous Advantage Actor Critic)
  - [DDPG](#) (Deep Deterministic Policy Gradient)
  - [DQN](#) (Deep Q Network)
  - [PPO](#) (Proximal Policy Optimization)
  - [SAC](#) (Soft Actor Critic)
  - [TD3](#) (Twin Delayed DDPG)
- 
- Note that some algorithms don't support every action and observation space type.
  - Make sure that the algorithm you pick fits the environment for this assignment.
  - Read [RL Algorithms](#) and the documentation of each algorithm for more detail.

# Tutorials

- [Stable Baselines 3 Examples](#)



🏠 / Examples

[Edit on GitHub](#)

## Examples

### Note

These examples are only to demonstrate the use of the library and its functions, and the trained agents may not solve the environments. Optimized hyperparameters can be found in the [RL Zoo repository](#).

```
import gymnasium as gym

from stable_baselines3 import DQN
from stable_baselines3.common.evaluation import evaluate_policy

# Create environment
env = gym.make("LunarLander-v2", render_mode="rgb_array")

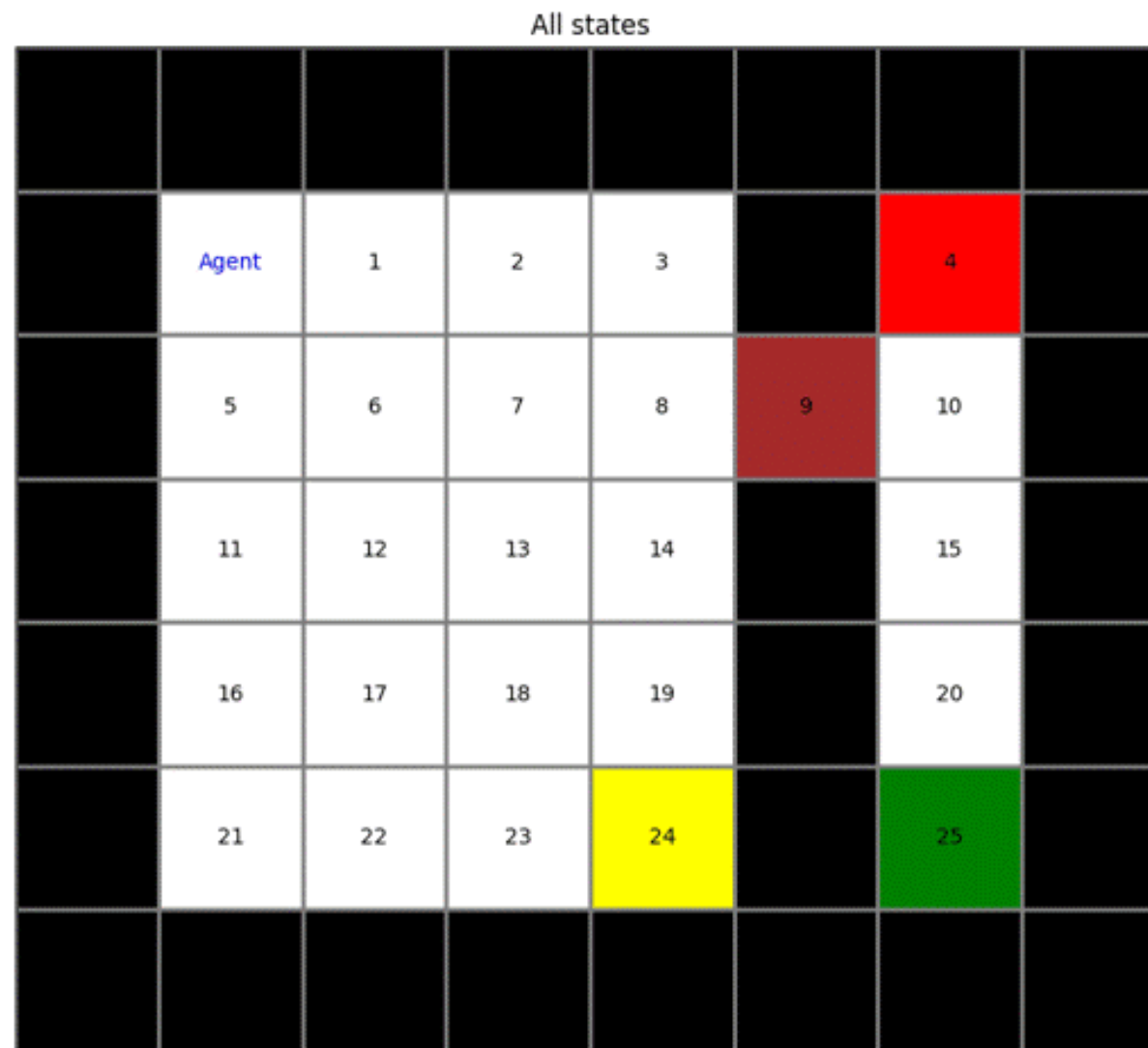
# Instantiate the agent
model = DQN("MlpPolicy", env, verbose=1)
# Train the agent and display a progress bar
model.learn(total_timesteps=int(2e5), progress_bar=True)
# Save the agent
model.save("dqn_lunar")
del model # delete trained model to demonstrate loading

# Load the trained agent
# NOTE: if you have loading issue, you can pass `print_system_info=True`
```

# Tasks

# Embed the GridWorld to SB3 and speed up

- We have learned how to implement algorithms on GridWorld environments. SB3 has some well-known Deep RL algorithms. Make the GridWorld SB3 trainable.
- Add more types of states to the GridWorld and have some interesting findings.
- You can train some algorithms on the GridWorld environment (not for grading)



# Environment

# Terminal states

---

- Original terminal states
  - Trap and Goal: Terminate after taking one step at the state
- New terminal states
  - Exit: Exit the GridWorld with a much less positive reward. The reward is given after taking any step at the Exit step.
  - Lava: Terminate the trajectory when entering the lava.

# Non-terminal state

---

- Original non-terminal states
  - Empty and Wall: The setting of Empty and Wall remains the same as in PA1 and PA2
- New non-terminal states
  - Bait: Enter bait state can get a position bait reward but will have step penalty afterward.
  - Door and Key: The Door will be opened when entering the Key state.
  - Portal: There can be only two portal states in GridWorld. We have to hit the wall at one portal state to teleport to the other portal state.



# All states and its color

---

All states

	Empty		Goal	Trap	Lava	Exit	Key	Door	Bait	Portal	

# Reward

---

- Step reward is given at every transition
- Goal reward is given after reaching goal state
- Trap reward is given after reaching trap state
- Exit reward is given after reaching exit state
- Reach Key can get step reward.
- Going through the opened door can get step reward.
- Reach Lava can get step reward.
- Enter the portal and get a step reward.
- If the agent is baited, the reward is the sum of the step reward and bait penalty.

# Initial state and reset

---

- Initial
  - You can not initialize the state in door, key, bait, lava states.
  - In PA3, the tasks are left-to-right. You should initialize the state at the left of the Lava and door, and the goal state will be at the right of the Lava and door.
  - Door and key and portal-pair are unique.
- Reset
  - Select one valid initial state.
  - Close the door if the door is opened.
  - Place the bait back in the bait state if it is bitten.

# Inner State and Outer State

---

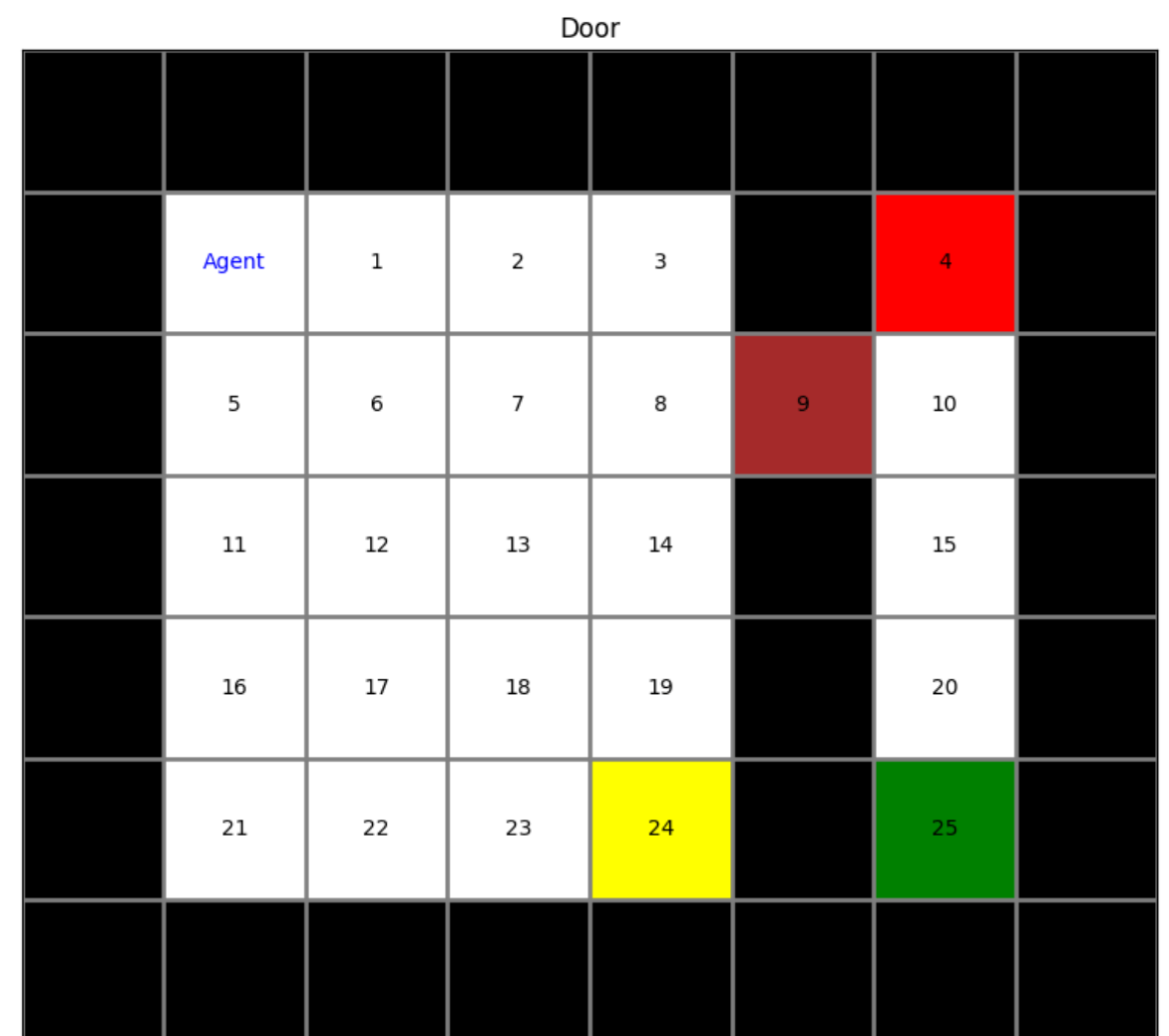
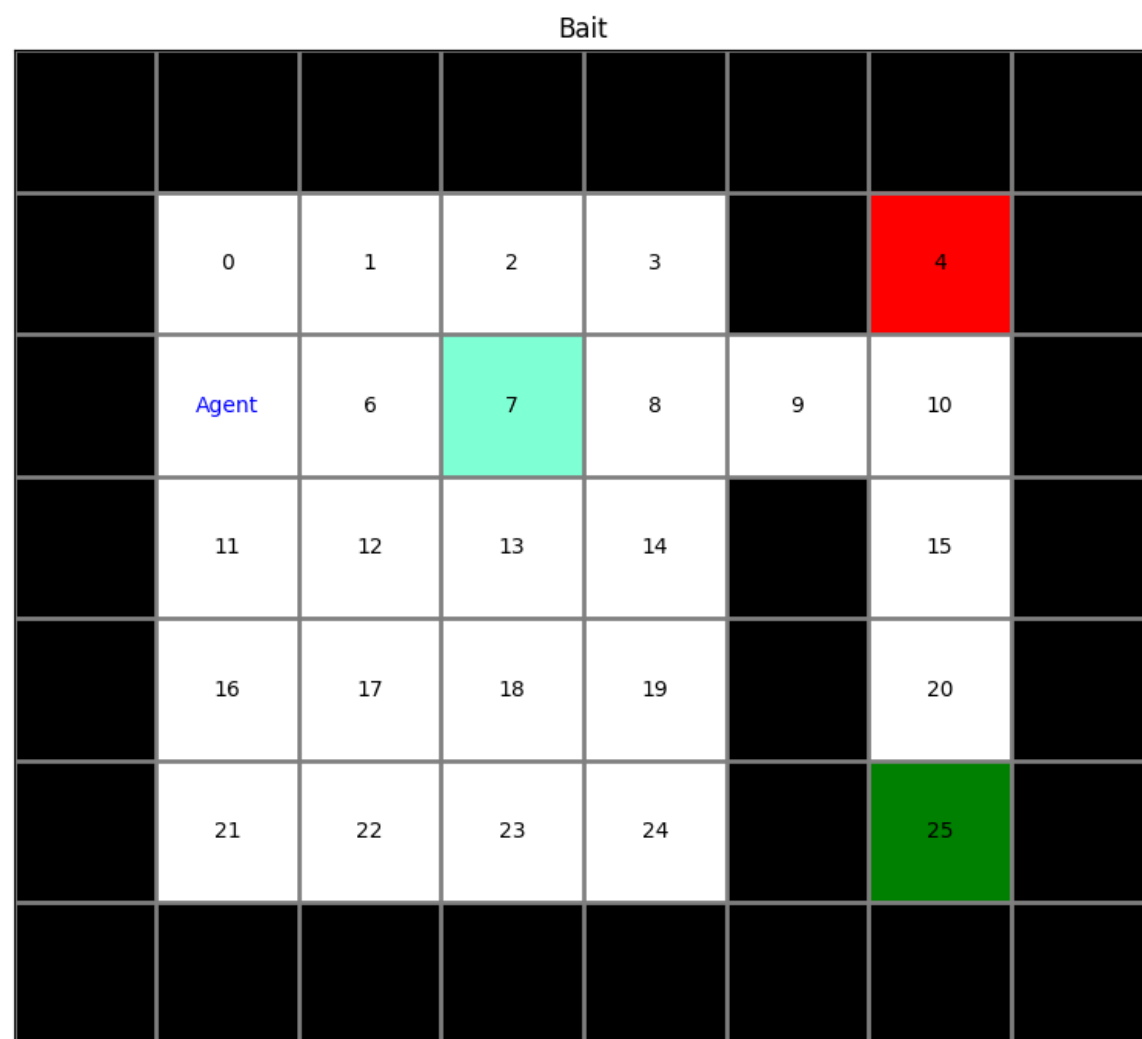
- The outer state is returned by the environment for Deep RL policy.
- The inner state indicates the current agent position in GridWorld.
- The outer state is initially the same as the inner state but can differ when door is **opened**.

If the door is opened, the outer state is  $inner\_state + len(self.\_state\_list)$ .



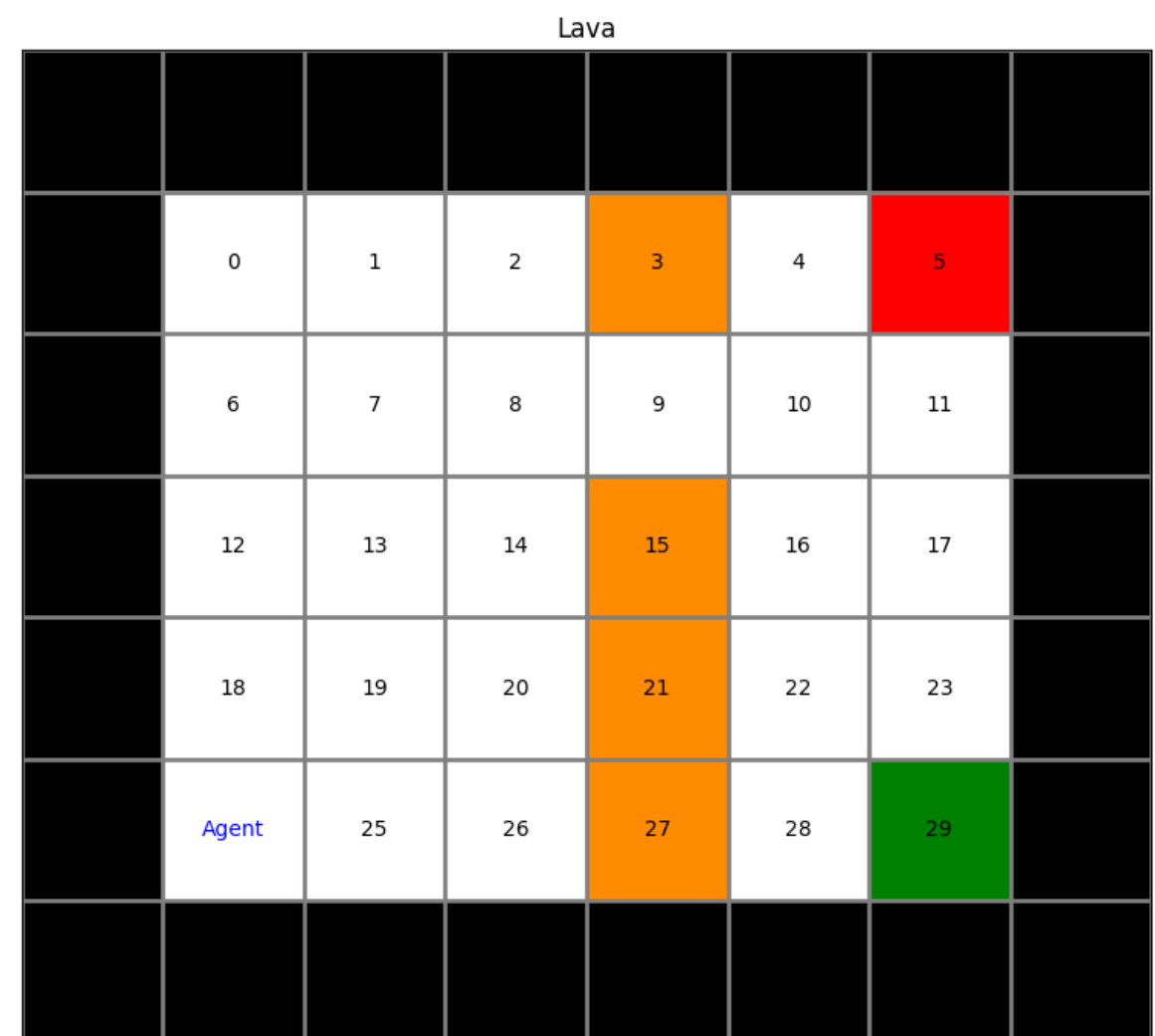
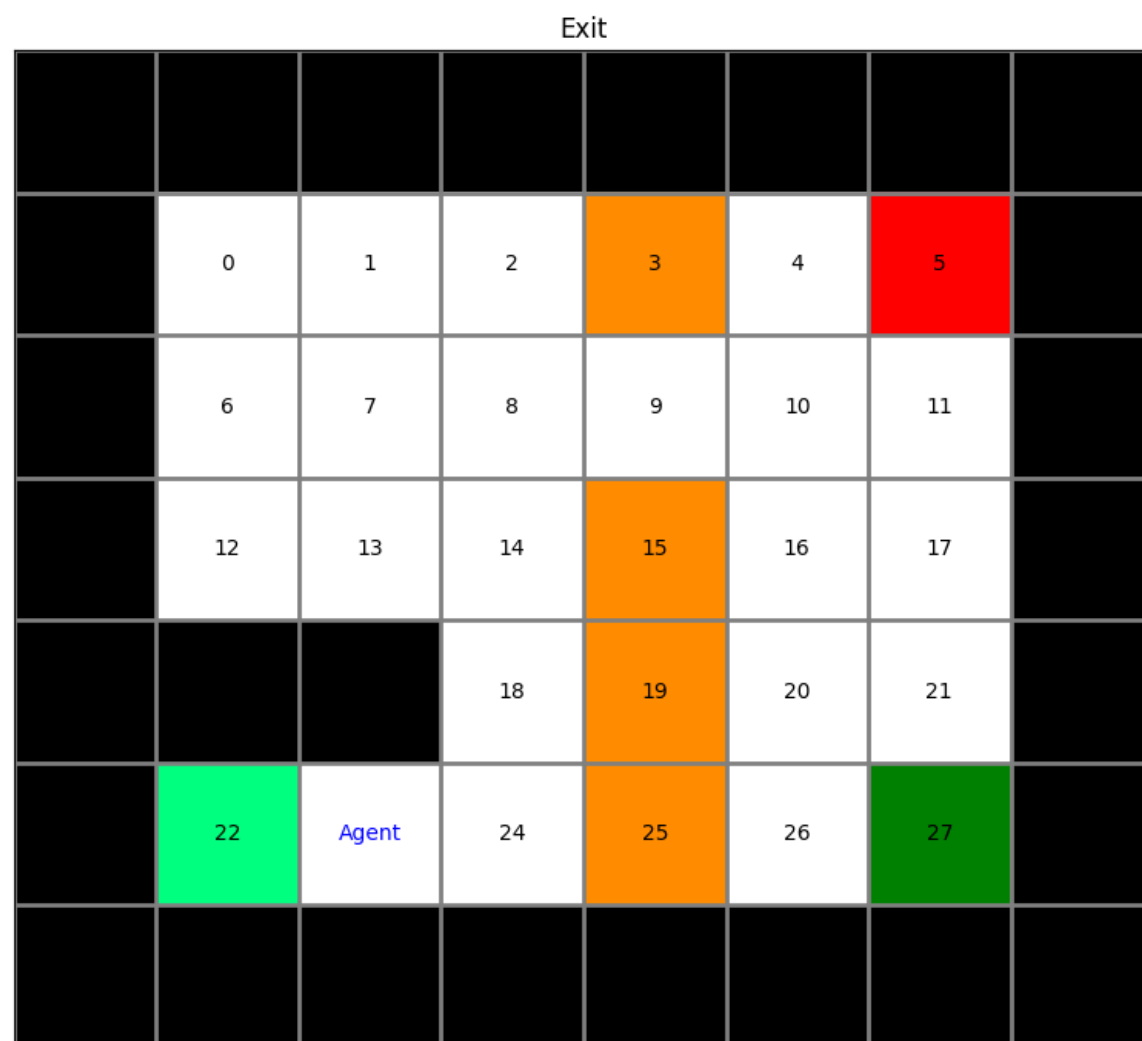
```
next_state = get_next_state(self._current_state)
if door_is_open:
    next_state += len(self._state_list)
```

# Bait and Door&Key



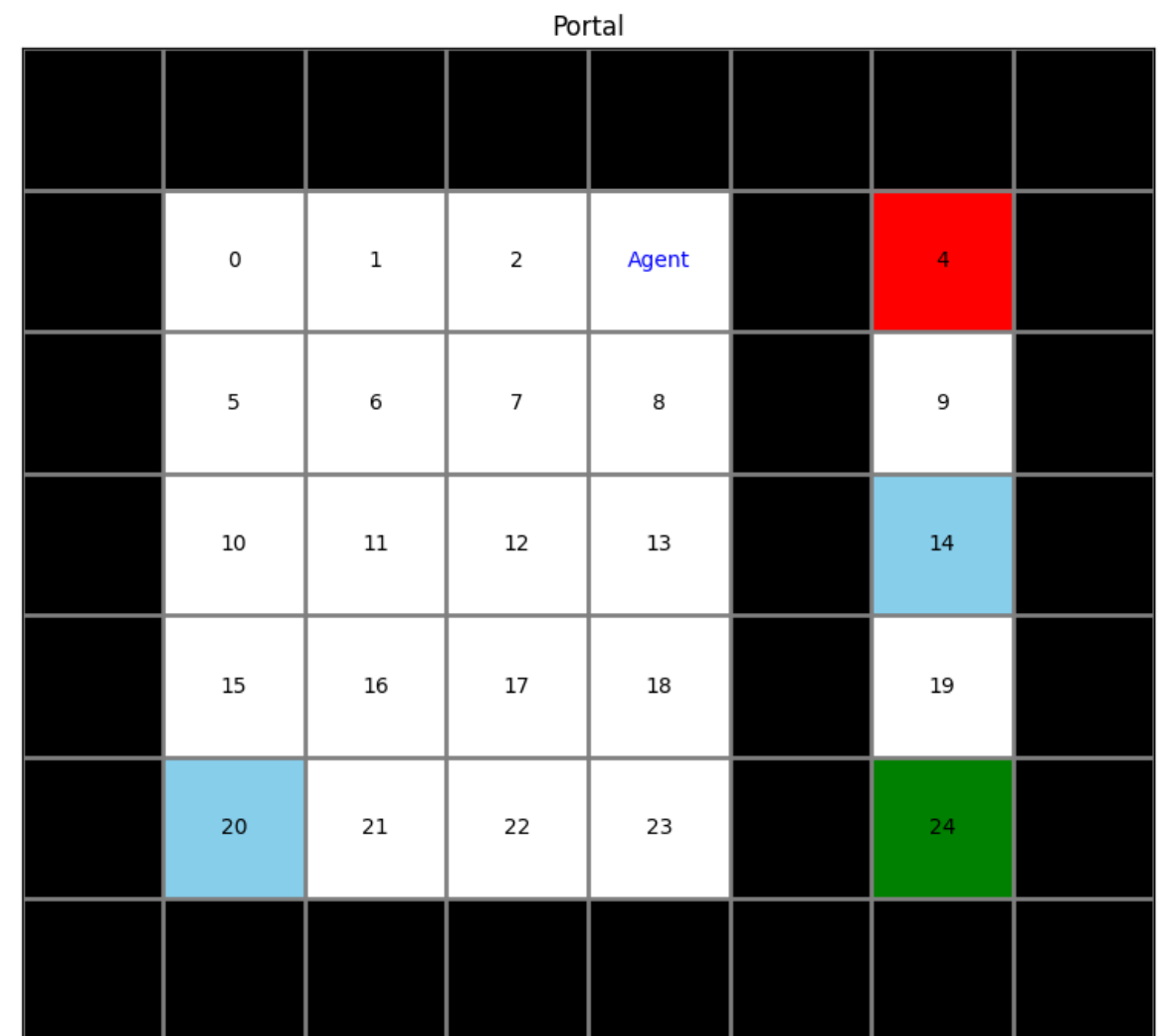
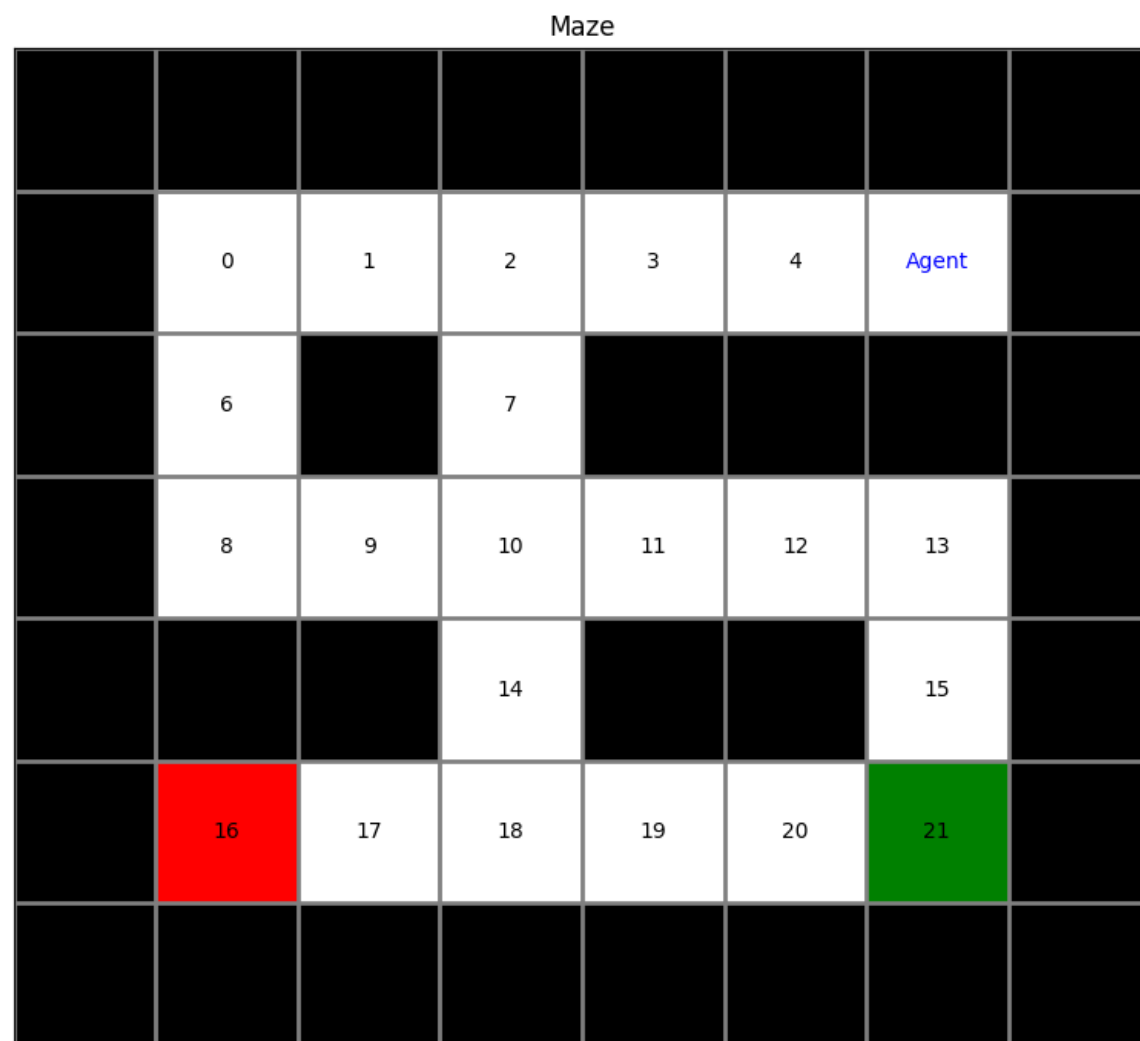
- Any transition in the bait state can get a bait reward but will be penalized for every step afterward.
- Going to the Key state can only get a step reward, but the environment changes.

# Exit and Lava



- You can not stay in lava states at any time.
- You should initial the Agent left to the Lava.

# Maze and Portal



- Hit the wall at portal can go to the other portal with step reward.
- The initial state is not limited to these tasks.

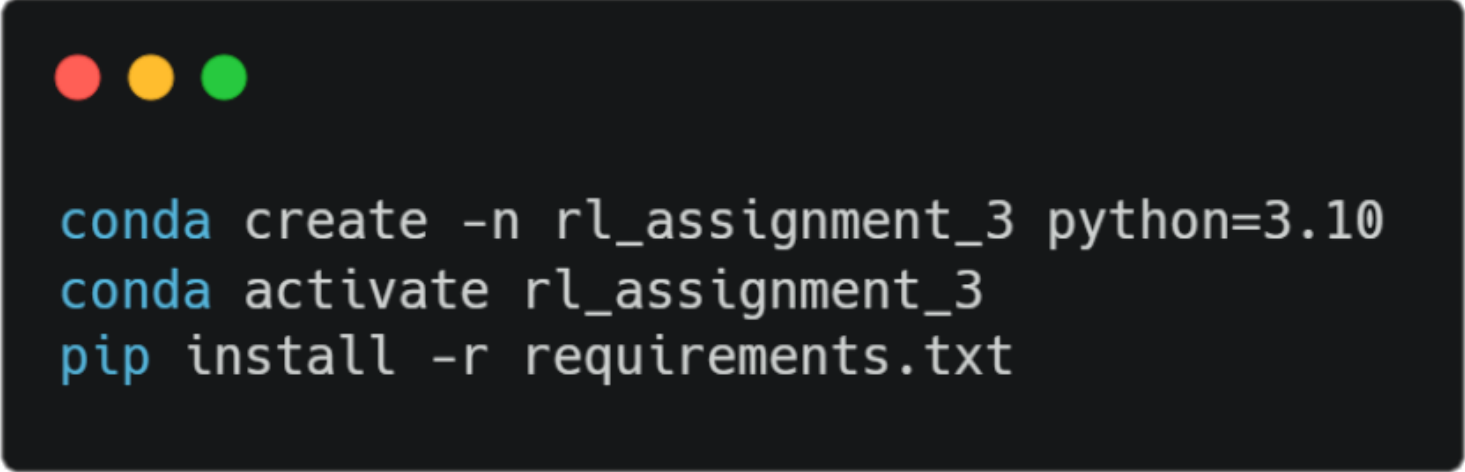
# Code Structure



# requirement.txt

---

- [Conda](#)



```
conda create -n rl_assignment_3 python=3.10
conda activate rl_assignment_3
pip install -r requirements.txt
```

- [venv](#)

- Sorry for virtual environment users. Stable Baseline 3 is hard to install using venv with pip. Do not use [venv](#) in PA3.



# main.py

---

- `print_sa(obs, action, env)`
  - Write state-action pair to console
- `test_task(filename, algorithm)`
  - Test the termination of your environment.
- `test_speed(filename)`
  - Test the speed of your implementation
- `test_correctness(filename)`
  - Test the correctness of your environment
- `write_gif(filename, algorithm)`
  - Write out one trajectory to gif.

# gridworld.py

---

- class GridWorld
  - TODO: step(action).
  - TODO: reset(). Unlike PA1 and PA2, reset will not be called inside the class when terminated. SB3 will call reset from outside.
  - ONLY the return values of step(action) and reset() are used for grading. Feel free to change any other parts except the arguments in `__init__`.
- class GridWorldEnv
  - It is the wrapper function for the gymnasium. Do not change any part of the class.

# Other folders

---

- assets/
  - The assets directory contains trained Deep RL algorithms.
- tasks/
  - There are six sample tasks.
- grading/
  - The grading directory has two types of data. JSON files are the testing files for speed, and CSV files are for correctness.

# Submission

# Report and Submission

---

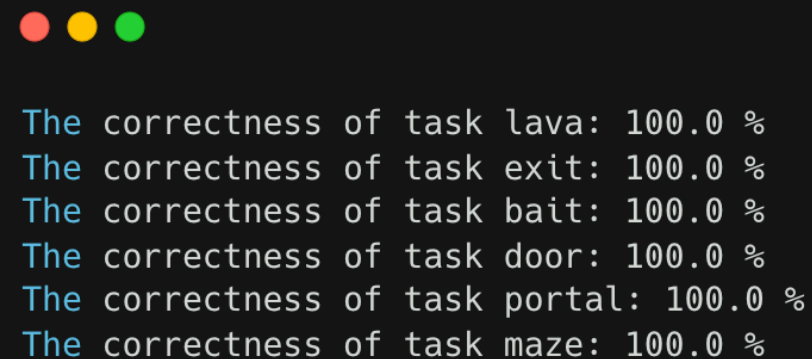
- Deadline 2023/11/09 Thu 09:30am
- No late submission is allowed
- The submission format and report questions will be announced in Assignment #3-2

# Grading

# Grading

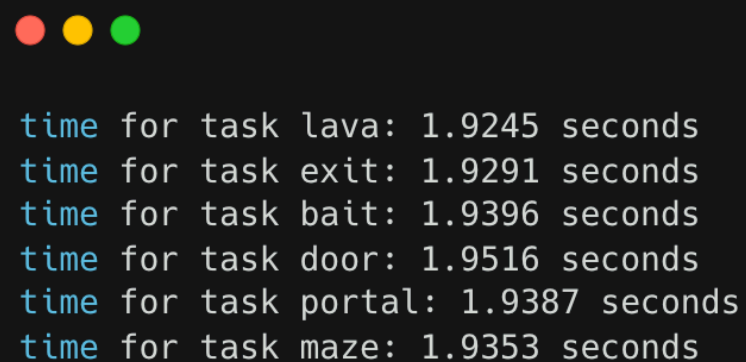
---

- Correctness (15%)
  - Public Test cases (1% x 6 public cases)
  - Private Test cases (1% x 9 private cases)



```
The correctness of task lava: 100.0 %  
The correctness of task exit: 100.0 %  
The correctness of task bait: 100.0 %  
The correctness of task door: 100.0 %  
The correctness of task portal: 100.0 %  
The correctness of task maze: 100.0 %
```

- Speed (10%)
  - Public Test cases (1% x 6 cases)
  - Private Test cases (1% x 4 cases)



```
time for task lava: 1.9245 seconds  
time for task exit: 1.9291 seconds  
time for task bait: 1.9396 seconds  
time for task door: 1.9516 seconds  
time for task portal: 1.9387 seconds  
time for task maze: 1.9353 seconds
```



# Criteria

---

- Test cases
  - Only check the output of step and reward function.
  - **5 minutes** for each test case to avoid infinite loops.
  - You have to be faster than TA's implementation to get points.
  - The private tasks will be more complex and bigger than the sample tasks.
  - The testing sequence of correctness will be pretty long.
  - The truncation will not grade for correctness.
- Sample agents trained on correct dynamics are provided for reference.
  - The trained policy may not be optimal.
  - Some test tasks are sequences of trajectory rolled out by trained agents.

# Policy

# Policy

---

## Package

- You can use any Python standard library (e.g., heap, queue...)
- System level packages are prohibited (e.g., sys, multiprocessing, subprocess...) for security concern

## Collaboration

- Discussions are encouraged
- Write your own codes

## Plagiarism & cheating

- All assignment submissions will be subject to duplication checking (e.g., MOSS)
- Cheater will receive an **F** grade for this course

## Grade appeal

- Assignment grades are considered finalized two weeks after release

# Contact

# Questions?

---

- General questions
  - Use channel **#assignment 3** in slack as first option
  - Reply in thread to avoid spamming other people
- Personal questions
  - DM us on Slack: **TA 李威緒 Wei-Hsu Lee**

**TA 林辰宇 Chen-Yu Lin**



**TA 李威緒 Wei-Hsu Lee**



**TA 林辰宇 Chen-Yu Lin**