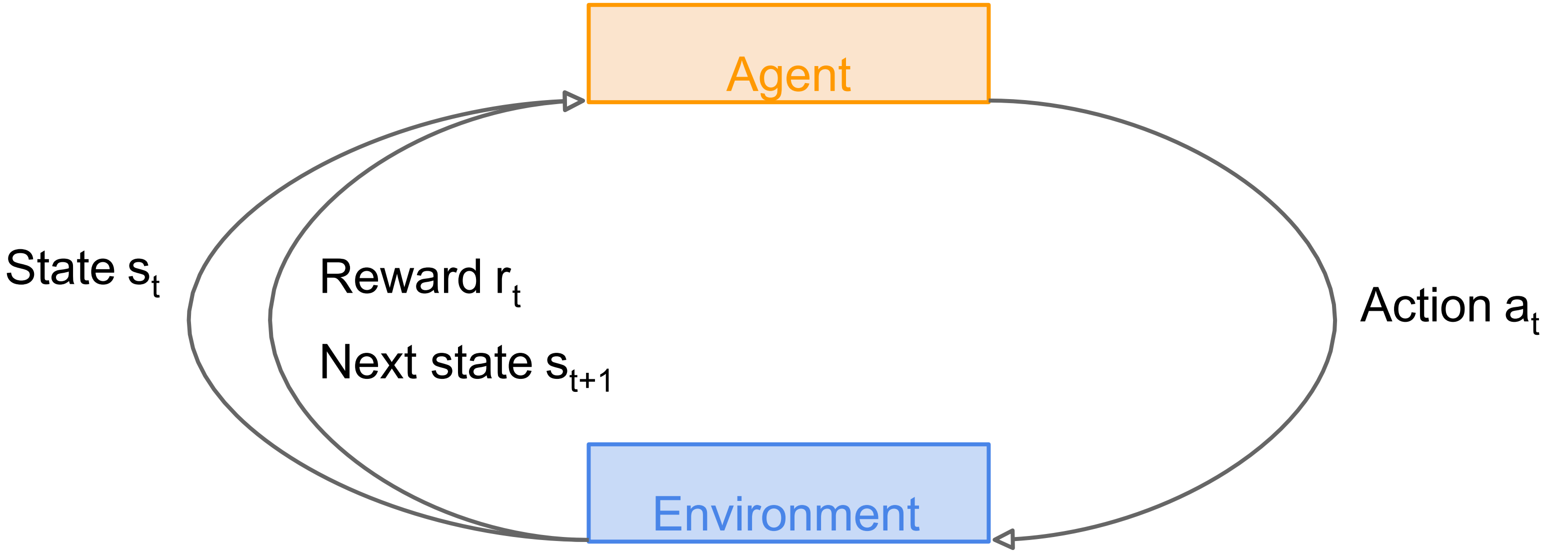
Artificial Intelligence
Assignment 2

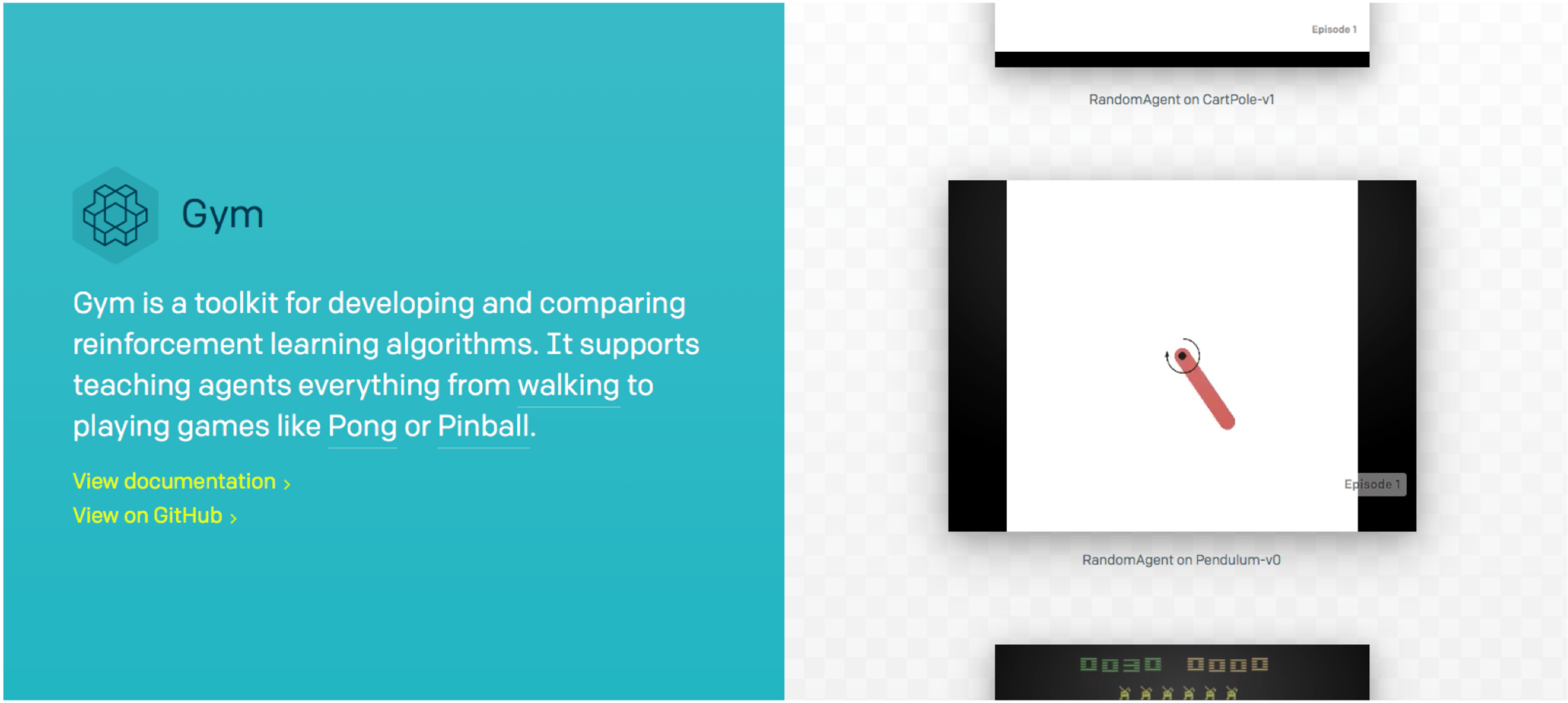
CSI4108-02
Spring, 2018

1. Introduction

- **We will solve the toy text problem that is provided by the gym library using Reinforcement Learning.**



2. OpenAI Gym



Gym

Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Pinball.

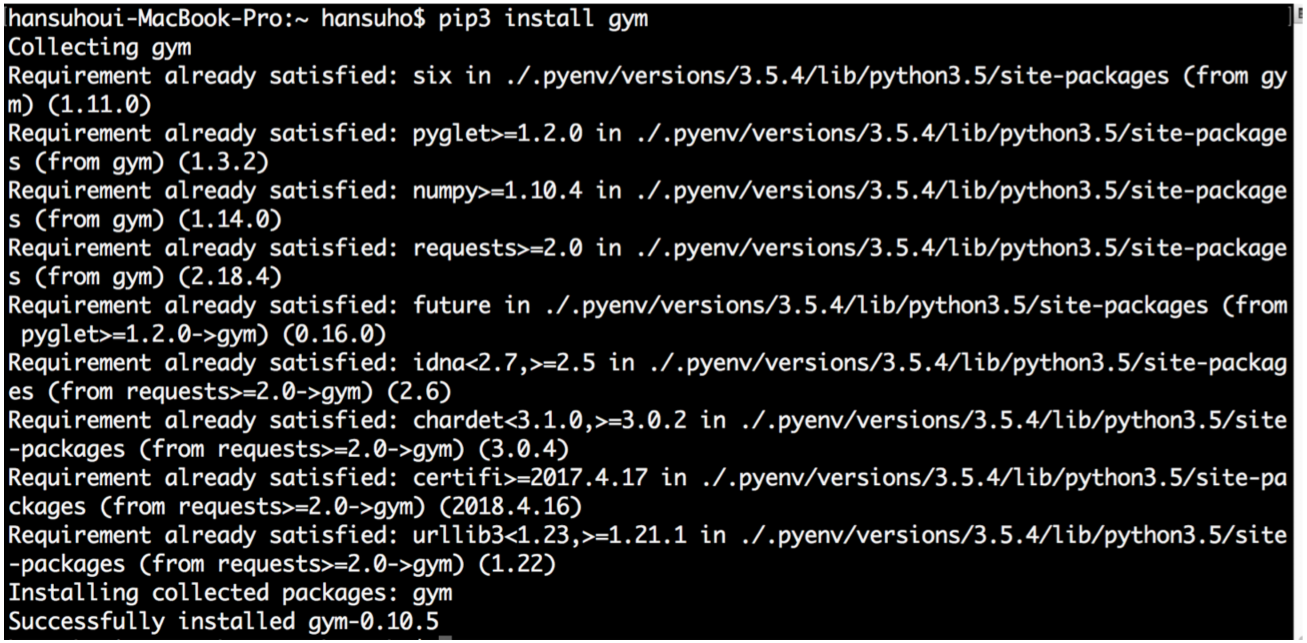
View documentation ›

View on GitHub ›

We provide the environment; you provide the algorithm.
You can write your agent using your existing numerical computation library, such as TensorFlow or Theano.

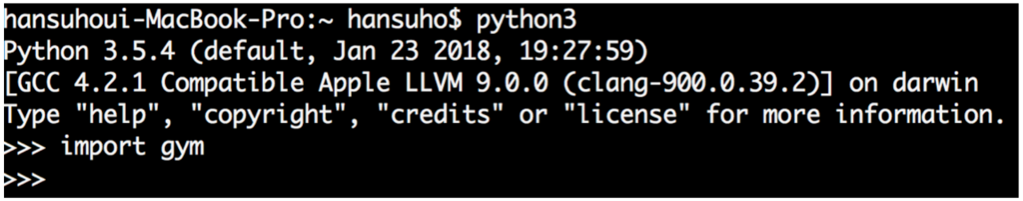
2-1. Basic installation steps

- **Install gym library(pip install gym)**



```
hansuhoui-MacBook-Pro:~ hansuho$ pip3 install gym
Collecting gym
Requirement already satisfied: six in ./.pyenv/versions/3.5.4/lib/python3.5/site-packages (from gy
m) (1.11.0)
Requirement already satisfied: pyglet>=1.2.0 in ./.pyenv/versions/3.5.4/lib/python3.5/site-package
s (from gym) (1.3.2)
Requirement already satisfied: numpy>=1.10.4 in ./.pyenv/versions/3.5.4/lib/python3.5/site-package
s (from gym) (1.14.0)
Requirement already satisfied: requests>=2.0 in ./.pyenv/versions/3.5.4/lib/python3.5/site-package
s (from gym) (2.18.4)
Requirement already satisfied: future in ./.pyenv/versions/3.5.4/lib/python3.5/site-packages (from
 pyglet>=1.2.0->gym) (0.16.0)
Requirement already satisfied: idna<2.7,>=2.5 in ./.pyenv/versions/3.5.4/lib/python3.5/site-packag
es (from requests>=2.0->gym) (2.6)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in ./.pyenv/versions/3.5.4/lib/python3.5/site
-packages (from requests>=2.0->gym) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in ./.pyenv/versions/3.5.4/lib/python3.5/site-pa
ckages (from requests>=2.0->gym) (2018.4.16)
Requirement already satisfied: urllib3<1.23,>=1.21.1 in ./.pyenv/versions/3.5.4/lib/python3.5/site
-packages (from requests>=2.0->gym) (1.22)
Installing collected packages: gym
Successfully installed gym-0.10.5
```

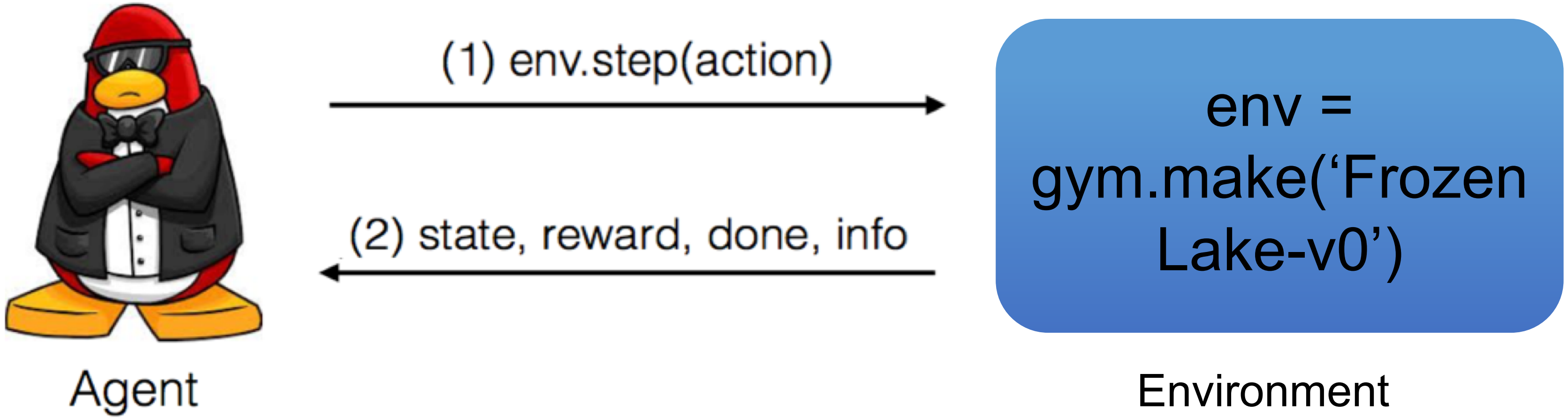
- **Quick checking**



```
hansuhoui-MacBook-Pro:~ hansuho$ python3
Python 3.5.4 (default, Jan 23 2018, 19:27:59)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.39.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import gym
>>>
```

2-2. OpenAI Gym detail

- **Detail information about the gym library can be found below site.**
- **https://gym.openai.com/docs/**



(1) env.step(action)

(2) state, reward, done, info

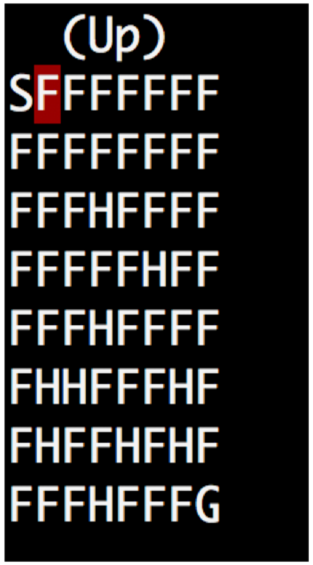
env = gym.make('Frozen Lake-v0')

Agent

Environment

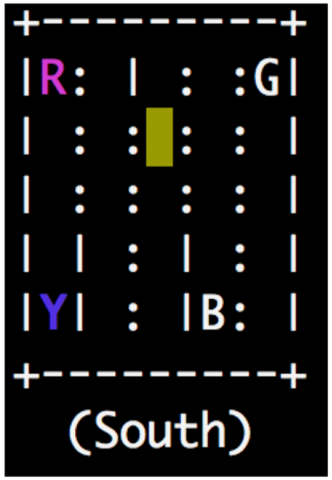
3. Detail

- In Assignment2, there are two problems.
  - FrozenLake8x8
  - Taxi



```
  (Up)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG
```

FrozenLake8x8



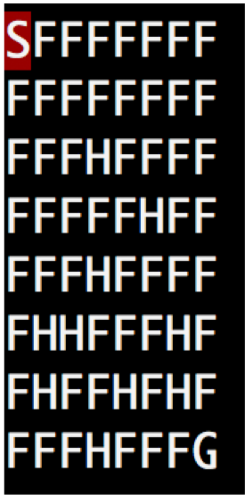
```
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (South)
```

Taxi

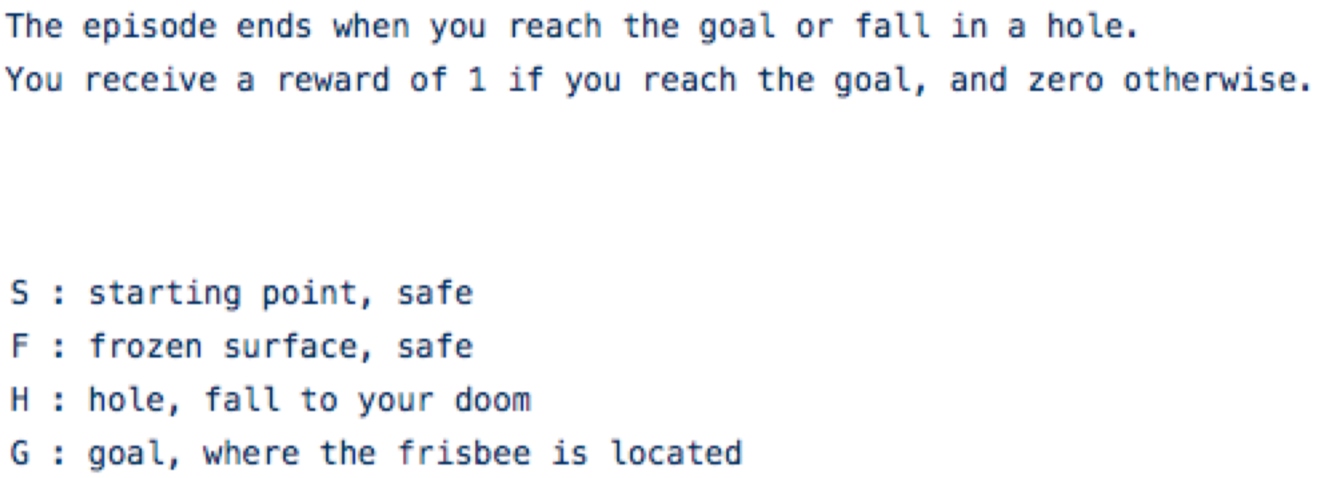
3-1. FrozenLake8x8

- The agent controls the movement of a character in a grid world. Some tiles of the grid are walkable, and others lead to the agent falling into the water. Additionally, the movement direction of the agent is uncertain and only partially depends on the chosen direction. The agent is rewarded for finding a walkable path to a goal tile.

- Use Q-learning to solve this problem
- There are four actions (Left, Down, Right, Up).



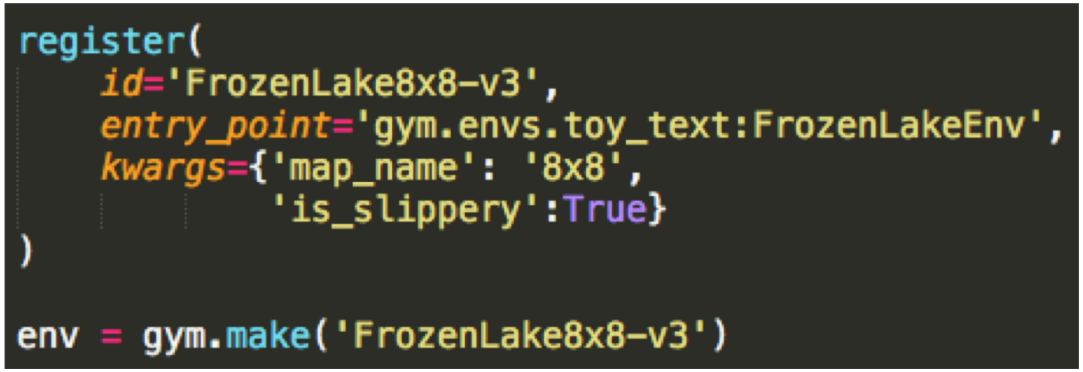
```
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG
```



```
The episode ends when you reach the goal or fall in a hole.
You receive a reward of 1 if you reach the goal, and zero otherwise.



S : starting point, safe
F : frozen surface, safe
H : hole, fall to your doom
G : goal, where the frisbee is located
```

3-1. FrozenLake8x8

- You need to use register to create environment.

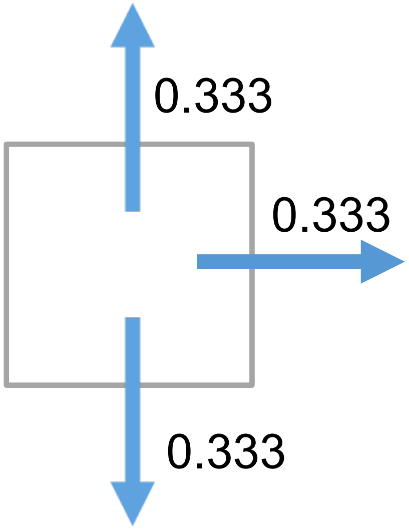


```
register(
    id='FrozenLake8x8-v3',
    entry_point='gym.envs.toy_text:FrozenLakeEnv',
    kwargs={'map_name': '8x8',
            'is_slippery':True}
)

env = gym.make('FrozenLake8x8-v3')
```

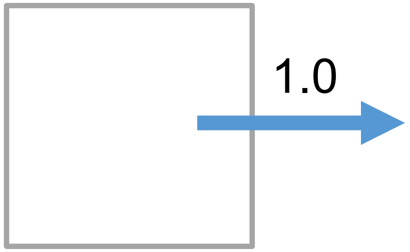
- Your code should work well when the ‘is_slippery’ is True and when it is False

3-1. FrozenLake8x8

- **If the ‘right’ action is selected, the probability of the action depending on the ‘is_slippery’ is as follows.**



Is_slippery : True



Is_slippery : False

3-1. FrozenLake8x8

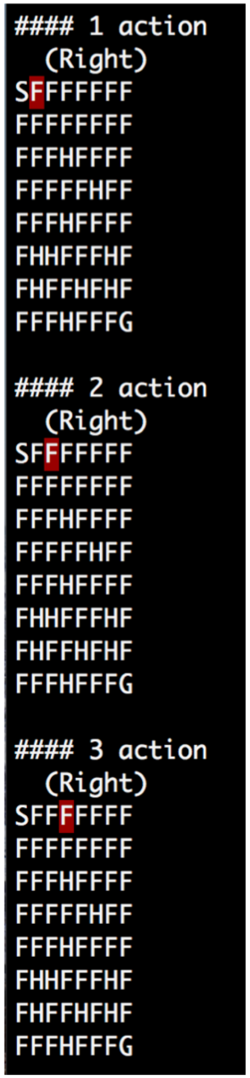
- **Output**

  After learning Q-table, print the path to goal tile according to Q-table value on standard output.

  You just print the path like <example>.

- **Detail information about the FrozenLake8x8 can be found below site.**

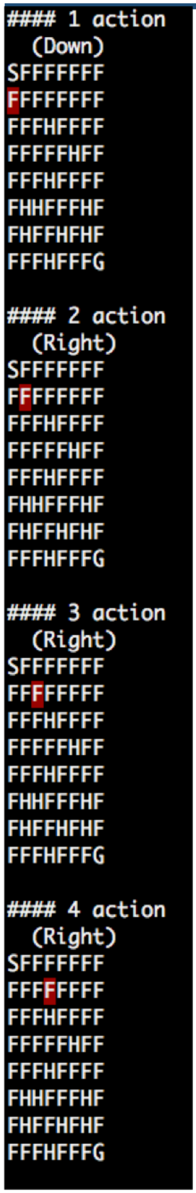
- https://gym.openai.com/envs/FrozenLake8x8-v0/



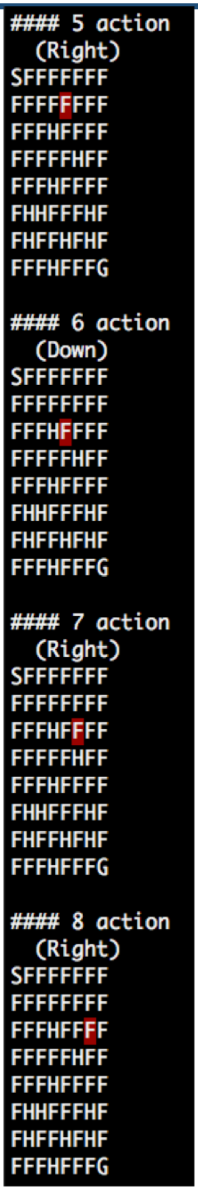
```
#### 1 action
  (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 2 action
  (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 3 action
  (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG
```

<Example>

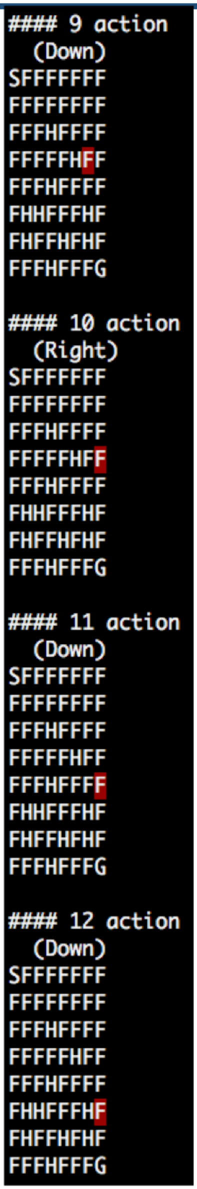
3-1. Output Example



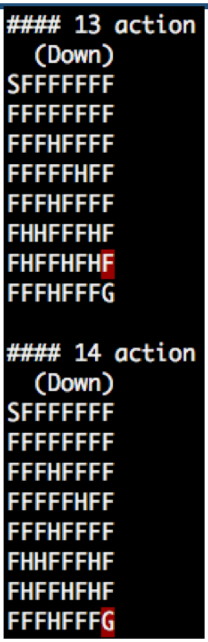
```
#### 1 action
 (Down)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 2 action
 (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 3 action
 (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 4 action
 (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG
```



```
#### 5 action
 (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 6 action
 (Down)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 7 action
 (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 8 action
 (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG
```

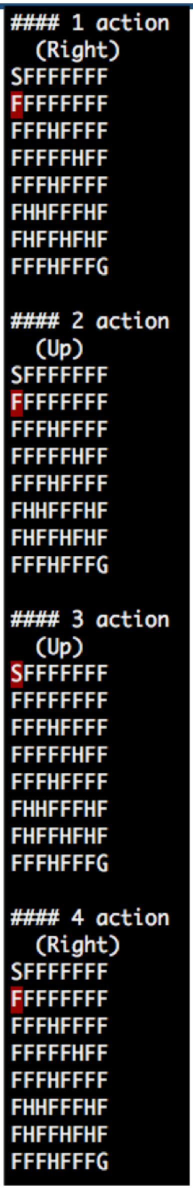


```
#### 9 action
 (Down)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 10 action
 (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 11 action
 (Down)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 12 action
 (Down)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG
```



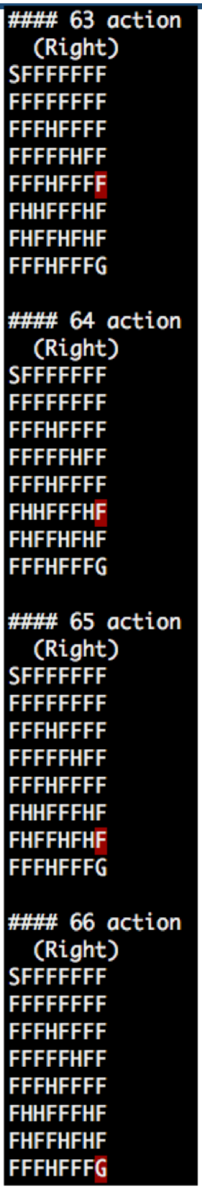
```
#### 13 action
 (Down)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 14 action
 (Down)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG
```

<‘is_slippery’: False>



```
#### 1 action
 (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 2 action
 (Up)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 3 action
 (Up)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 4 action
 (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG
```

......



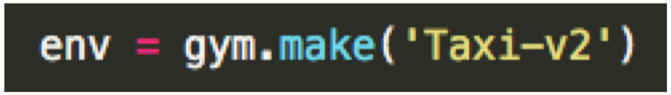
```
#### 63 action
 (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 64 action
 (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 65 action
 (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG

#### 66 action
 (Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFHFFFF
FHHFFFHF
FHFFHFHF
FFFHFFFG
```

<‘is_slippery’: True>

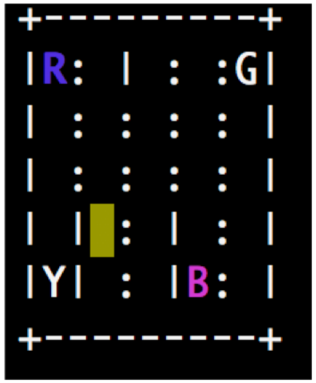
3-2. Taxi

- This task was introduced in [Dietterich2000] to illustrate some issues in hierarchical reinforcement learning. There are 4 locations (labeled by different letters) and your job is to pick up the passenger at one location and drop him off in another. You receive +20 points for a successful dropoff, and lose 1 point for every timestep it takes. There is also a 10 point penalty for illegal pick-up and drop-off actions.

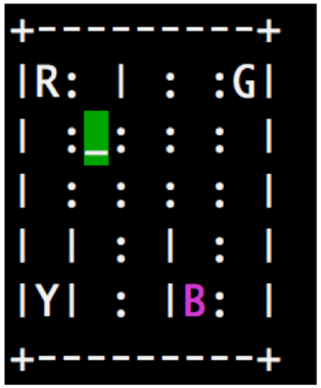
- Use Q-learning to solve this problem
- There are six actions (West, South, East, North, Pickup, Dropoff).
- In this problem, You don't need to use register to create environment.



```
env = gym.make('Taxi-v2')
```



```
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
```



```
+---------+
|R: | : :G|
| :_: : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
```

```
- blue: passenger
- magenta: destination
- yellow: empty taxi
- green: full taxi
- other letters: locations
```

3-2. Taxi

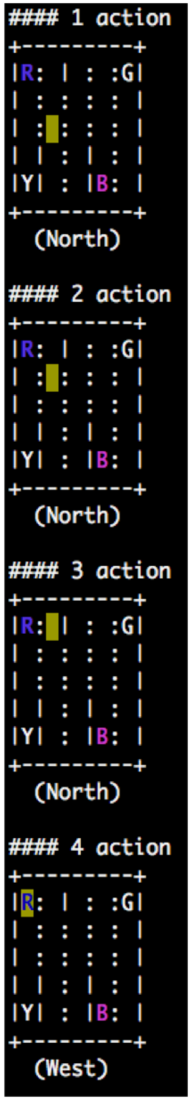
- **Output**

  After learning Q-table, print the states until the end of the episode according to Q-table value on standard output.

  You just print the path like <example>.

- **Detail information about the Taxi can be found below site.**

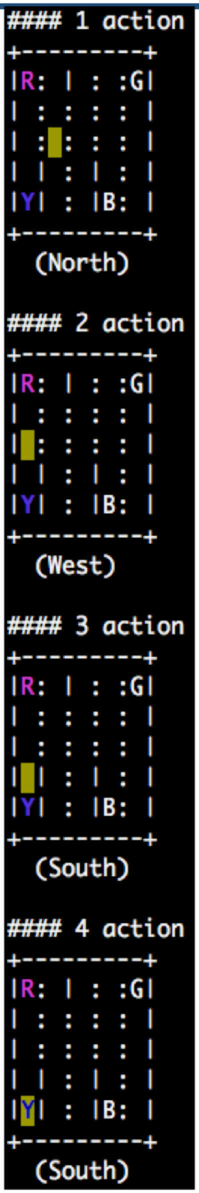
- https://gym.openai.com/envs/Taxi-v2/



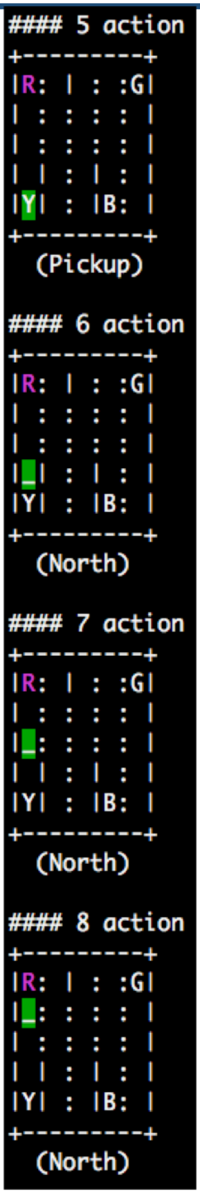
```
#### 1 action
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (North)

#### 2 action
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (North)

#### 3 action
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (North)

#### 4 action
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (West)
```

<Example>

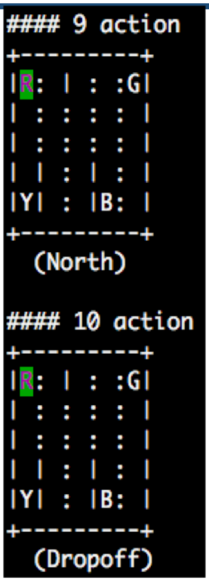
3-2. Output Example



```
#### 1 action
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (North)

#### 2 action
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (West)

#### 3 action
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (South)

#### 4 action
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (South)
```



```
#### 5 action
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (Pickup)

#### 6 action
+---------+
|R: | : :G|
| : : : : |
| : : : : |
|_| : | : |
|Y| : |B: |
+---------+
  (North)

#### 7 action
+---------+
|R: | : :G|
| : : : : |
|_: : : : |
| | : | : |
|Y| : |B: |
+---------+
  (North)

#### 8 action
+---------+
|R: | : :G|
|_: : : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (North)
```



```
#### 9 action
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (North)

#### 10 action
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (Dropoff)
```

3. Detail

- **Summary**

Solve problems through reinforcement learning using the environment and functions given in the gym library.

We give you three Python files, then you have to write the code in that files.

- **FrozenLake_true.py : The FrozenLake problem 'is_slippery' is ture**
- **FrozenLake_false.py: The FrozenLake problem 'is_slippery' is False**
- **Taxi.py: The Taxi problem**

If you need another library(like numpy), you can use it.

3. Detail

- **Tips**

- Use the Q-learning formula

$$sample = R(s, a, s') + \gamma \max_{a'} Q(s', a')$$
$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + (\alpha)\,[sample]$$

- **Set the noise appropriately for fast learning.**
- **Exploit & Explore**

4. Submission

- Deliverables: 2013147xxx_1.zip

- Must include
  - FrozenLake_true.py
  - FrozenLake_false.py
  - Taxi.py
  - Other codes (If you necessary)

  (Your code with detail comments)

5. Grading environment & Directions

- Language: Python
- We grade your score in Linux(Ubuntu 16.04)
- Python3 (>= 3.5.2)

- This is an individual project
- You should follow output format

- Never copy code
- You will get 0 points if you cheat

- If you do not use Reinforcement Learning, you will get 0 points

6. Grading policy

- FrozenLake_true.py
  - print a right path : 25pts

- FrozenLake_false.py
  - print a right path : 25pts

- Taxi.py
  - print right states : 50pts

7. Due Date

- **Due Date: 17/May/2018 23:59:00 KST**
- **Delay Policy: -15pts per day**

Pleae use YSCEC Q&A board to leave your question.