

# Project #2 Specification

2018. 4. 27

# Project #2 Introduction



Grab the Coin!

This is the real screenshot of the game! (not a generated image)

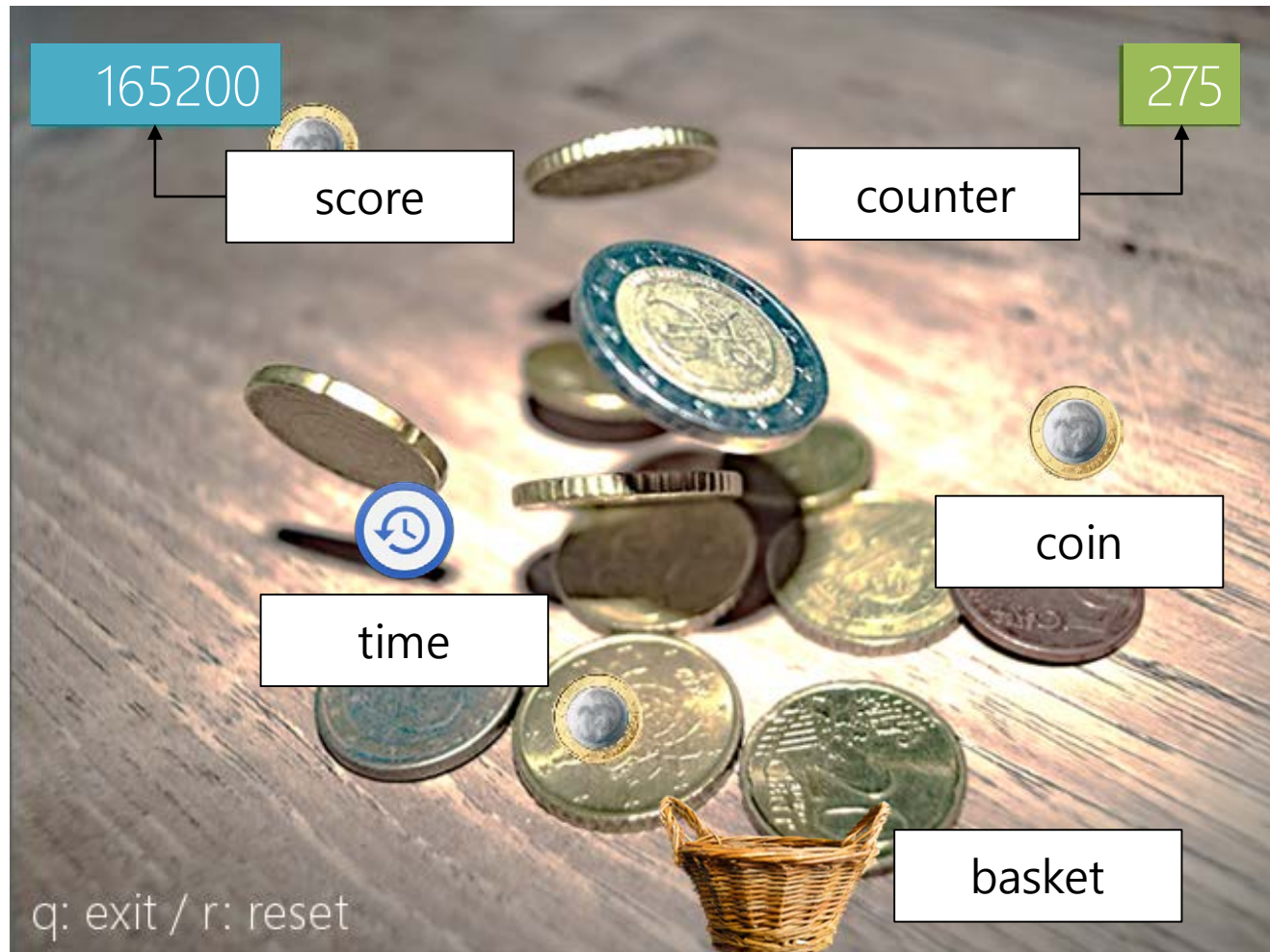
# Project #2 Introduction

- In this project, you have to implement the learning agent for playing the given game.
  - objective: get a high score
- This is individual project.
- You can apply the reinforcement learning (Q-Learning) to the real case by this project.
  - The skeleton code for Q-Learning will be given
    - original code: <https://github.com/dennybritz/reinforcement-learning/blob/master/TD/Q-Learning%20Solution.ipynb>
  - You have to (1) **define the state and reward** for the game, and (2) **modify the code about Q value** for the given state.



# Files, Environment Setting

- Language: Python 3.6
  - I think Python 3.4~3.6 would be okay
- All resources are available in the project site
  - <http://an.yonsei.ac.kr/coindrop/>
- You must install numpy and pygame to play the game
  - `pip install numpy pygame`
    - You may run the command prompt with administrator mode
  - Windows: <https://www.lfd.uci.edu/~gohlke/pythonlibs/>
    - `pip install ~~.whl`

# Game Rule (1/2)



# Game Rule (2/2)

- You can move the basket in two directions
  - Left / Right
- You can obtain items dropping from the top
  - Coin()
    - +500 point if you get a coin in the basket
    - -150 point if you failed to get a coin
  - Time()
    - Does not give the point (+0 point)
    - Extend the counter (+100)
    - Only appears:
      - The counter is under 500
      - when your score is under 300,000
- Game ends when counter become 0
  - Counter starts at 200

# How to play the game

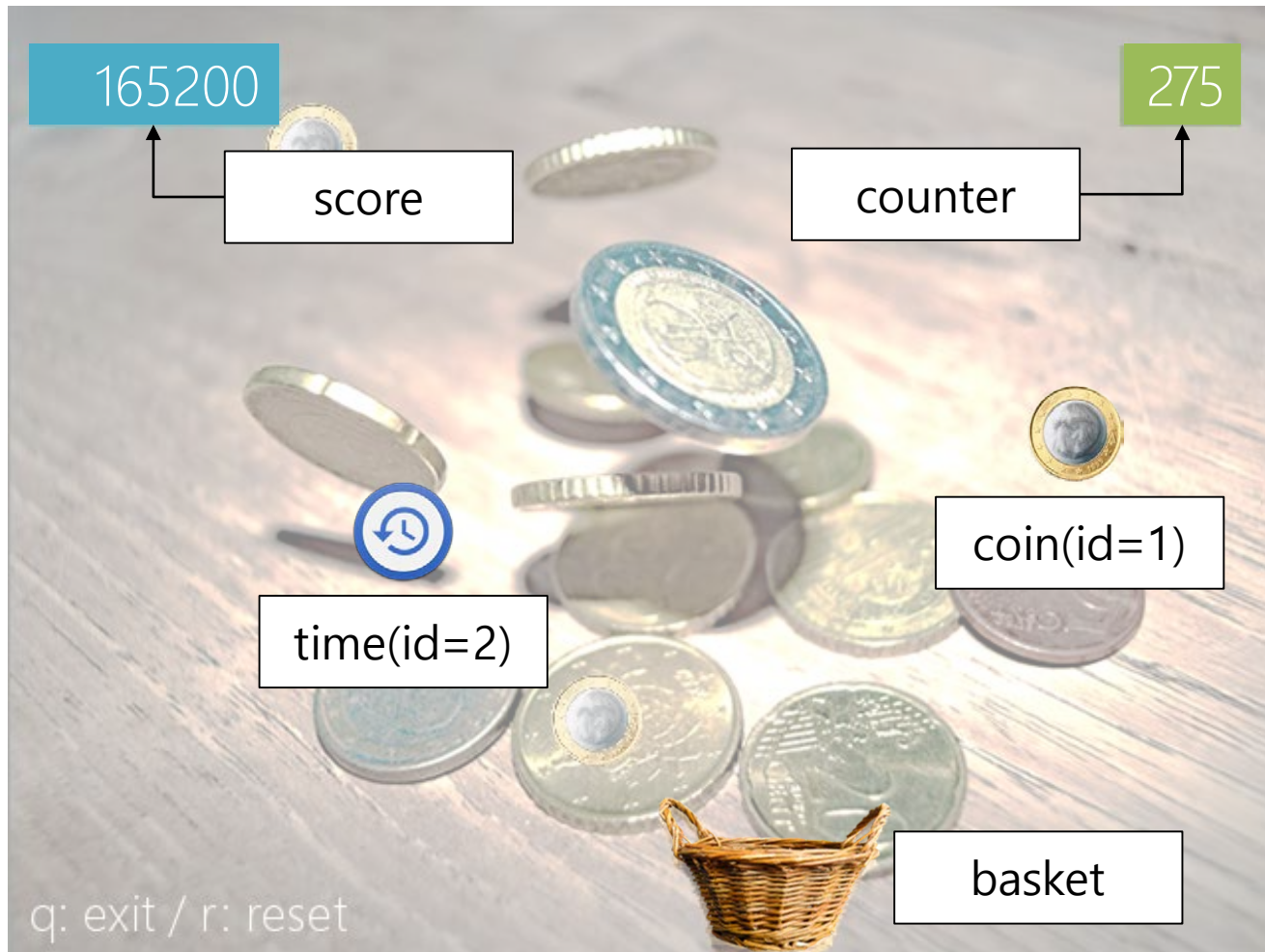
```
game>python game.py -h
usage: game.py [-h] [-s SHOW] [-p PLAYMODE] [-m MODEL]

optional arguments:
  -h, --help            show this help message and exit
  -s SHOW, --show SHOW  1: show game in GUI, 0: only print the state
  -p PLAYMODE, --playmode PLAYMODE
                        1: play the game (--show must be 1), 0: run your agent
  -m MODEL, --model MODEL
                        your model file
```

- If you want to play the game:
  - `python3 game.py --show 1 --playmode 1`
- If you want to see how your agent play the game:
  - `python3 game.py --show 1 --playmode 0`

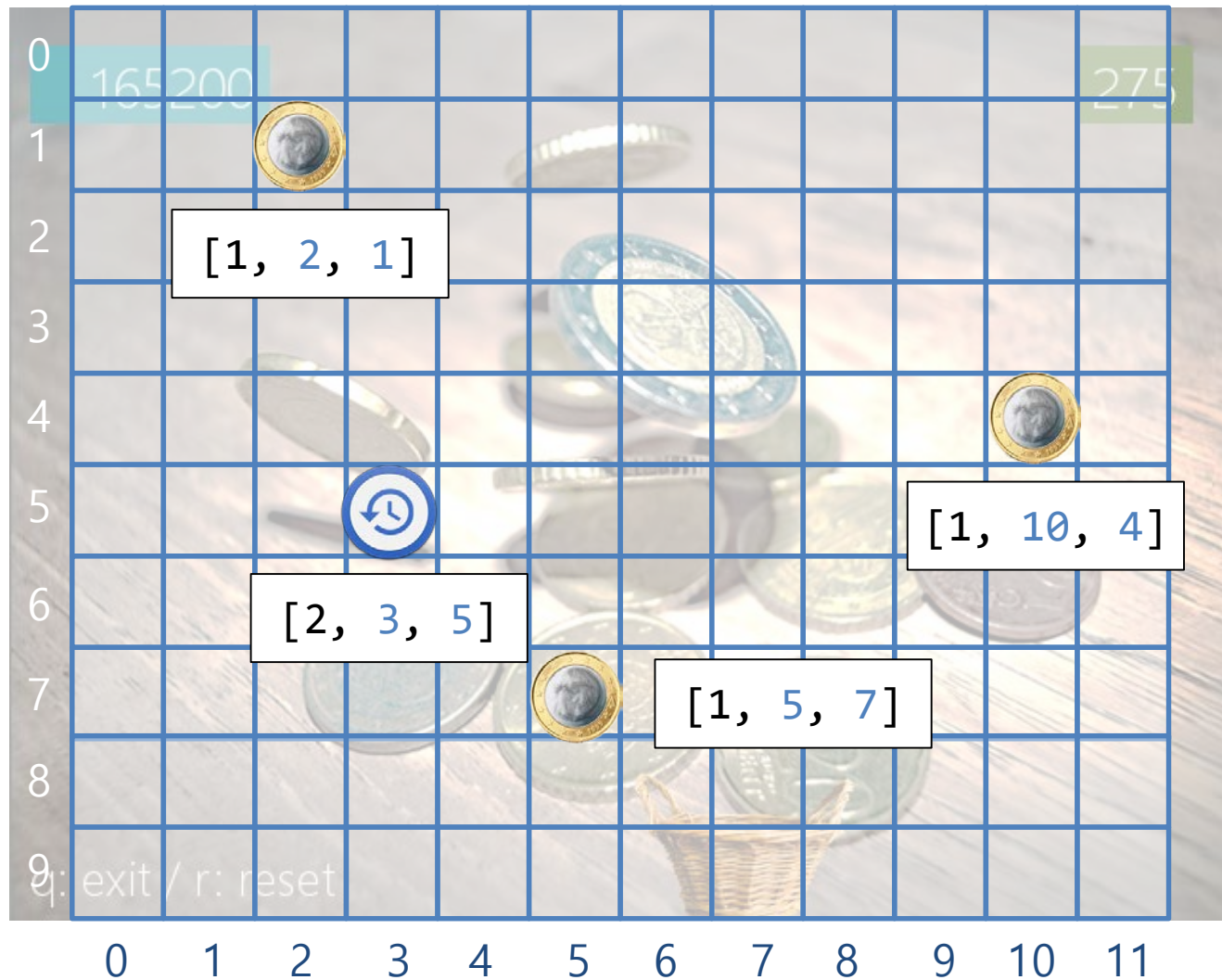


# Given information (1)

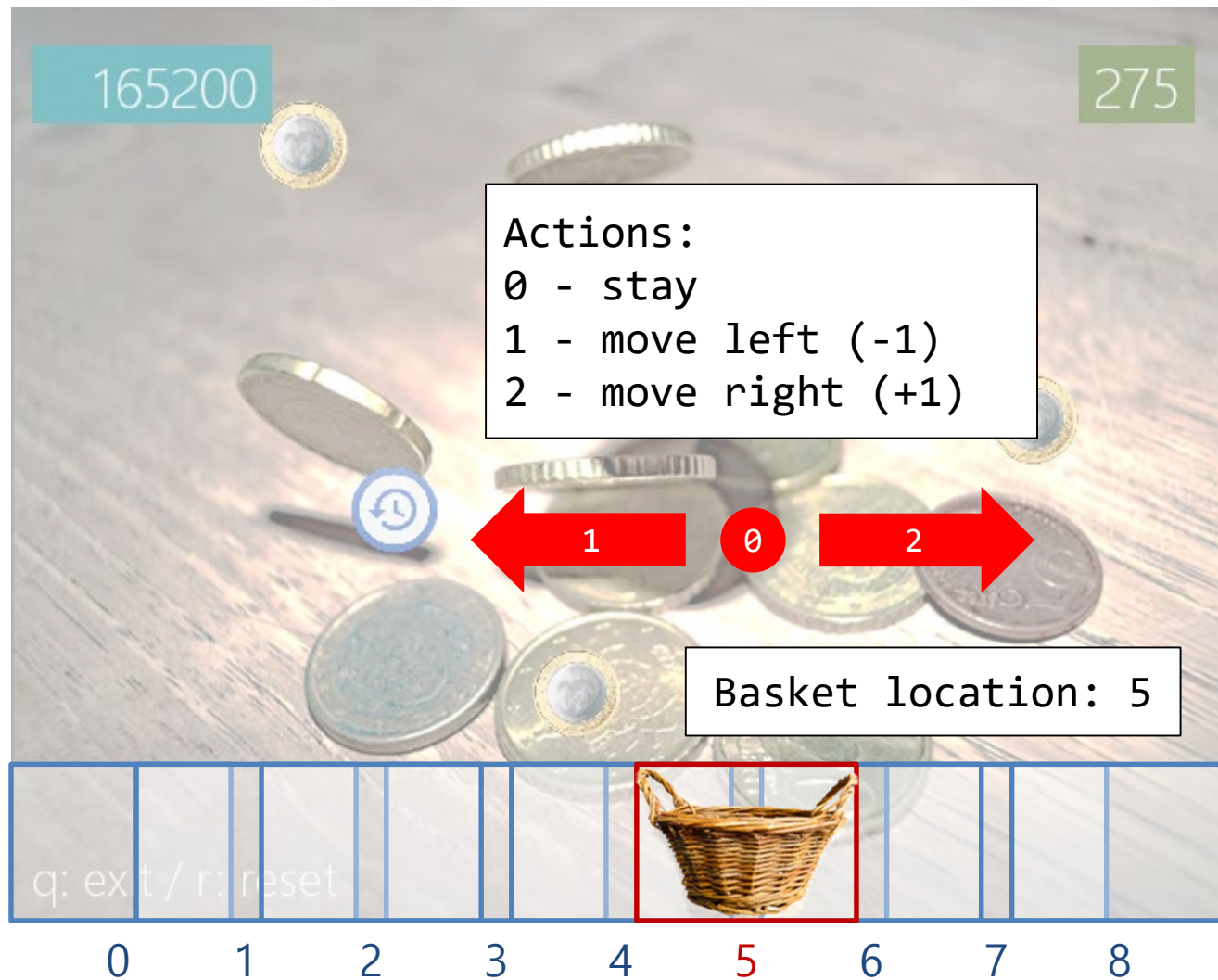




# Given information (2)



# Given information (3)



# Given information (4)

- Input/Output format (for game)
  - `done, counter, score, state = game.step(action)`
    - `done` (True when game finished)
    - `state` (tuple: (basket\_location, [list of item locations]))
    - `action` (0: stay, 1: go left, 2: go right)

```
False 23 -100 (3, [[1, 0, 7], [2, 6, 7], [1, 2, 4], [1, 8, 1]])
False 22 -100 (3, [[1, 0, 8], [2, 6, 8], [1, 2, 5], [1, 8, 2]])
False 21 -100 (2, [[1, 0, 9], [2, 6, 9], [1, 2, 6], [1, 8, 3], [1, 10, 0]])
False 20 -250 (1, [[1, 2, 7], [1, 8, 4], [1, 10, 1]])
```

- Initial state
  - called with `game.reset()`
  - Counter/Score: 200/0
  - Basket location: 4 (center)
  - Item locations: [(1, N, 0)]
  - Action: 0

# Project Requirements (1/3)

- You can only edit `q_learning.py`
- Fill the given code

```
## this function return state from given game information.
```

```
def get_state(counter, score, game_info):  
    basket_location, item_location = game_info
```

```
    """  
    FILL HERE!
```

```
    you can (and must) change the return value.  
    """
```

```
    return basket_location
```

```
## this function return reward from given previous and current score and counter.
```

```
def get_reward(prev_score, current_score, prev_counter, current_counter):  
    """
```

```
    FILL HERE!
```

```
    you can (and must) change the return value.  
    """
```

```
    return (current_score - prev_score) + (current_counter - prev_counter)
```

# Project Requirements (2/3)

- Modify the code related to Q-Learning
  - Modification does not include changing function / variable / constant in existing function
  - It is okay to left them without modification...

```

## this function return policy function to choose the action based on Q value.
def make_policy(Q, epsilon, nA):
    """
    This is the epsilon-greedy policy, which select random actions for some chance (epsilon).
    (Check dennybritz's repository for detail)

    You may change the policy function for the given task.
    """
    def policy_fn(observation):
        A = np.ones(nA, dtype=float) * epsilon / nA
        best_action = np.argmax(Q[observation])
        A[best_action] += (1.0 - epsilon)
        return A
    return policy_fn

    """
    this code performs TD Update. (Update Q value)
    You may change this part for the given task.
    """
    best_next_action = np.argmax(Q[next_state])
    td_target = reward + discount_factor * Q[next_state][best_next_action]
    td_delta = td_target - Q[state][action]
    Q[state][action] += alpha * td_delta

```

# Project Requirements (3/3)

- You can modify the other part of the `q_learning.py`
  - Like adding parameters
  - Please specify your contribution and how to run your code (arguments info, etc) in the report
- Train your agent!
  - `python3 q_learning.py -n 10000`
  - `-e` (epsilon), `-lr` (learning rate)
    - Recommend to adjust these factors to make performance better
- Trained Q value table is saved in `model_q.pkl`
  - You must submit this file that I can reproduce your result.

# Allowed

- You are allowed to:
  - Using well-known libraries
    - Numpy/Scipy for Python
    - If you are not sure, feel free to ask TA.
  - Asking about the specification detail
    - Use YSCEC Q&A Board, I recommend to make it public
  - Discuss your solutions with your colleagues
    - If you don't share the code
  - It is OK to check the related code here:
    - <https://github.com/dennybritz/reinforcement-learning/blob/master/TD/>
    - But not allowed to copy and paste the code (this includes only changing variable name)



# Disallowed

- You are NOT allowed to:
  - Using existing code/library for the given task
  - Copy & Paste the source code from Internet/Book
    - This includes the change of variable name/order of the code!
  - Sharing the source code of your friends
    - You will get **0 points for ALL PROJECTS** if plagiarism detected
  - Fix given `game.py` file
    - You will get 0 point for this project if your code is not running
  - Attack or spoil the web page server
    - I'm making log for all access log

# Submission (1/3)

- Due date: 2018. 5. 20 (Sun) 23:59:59
  - 10% deduction per day
- Tip: Recommend to check the code and implement your solution as soon as possible!
  - Don't put off until tomorrow what you can do now.
  - The learning may take a lot of time,  
so it is not a good idea to do this project near the deadline.



# Submission (2/3)

- Project Report
  - There is no limit for the format and the # of pages
    - But please write it as report-style
  - You **MUST** include:
    - How did you define the state and reward for this game?
    - How did you modify the policy function / Q-score?
      - If you didn't modify them, it is okay to write how does it work instead
    - Learning parameter (learning rate, epsilon, etc)
    - Result and analysis for your agent
      - Quantitative results (ex: the avg scores for many games)
      - Analysis for successful case / failure case

# Submission (3/3)

- Compress your report, code and data with zip and
  - Filename: your student ID (ex: 2016147500.zip)
  - Code and data: `q_learning.py`, `model_q.pkl`
  - You need not to make hard-copy for your report
  - You need not to write comment for your code, but it would be help to understand how you implement it
    - I'll give a little score when the submitted code is not running, but comment for the code exists.
- Submit your zip file to the YSCEC Homework Board

# Grading Policy (tentative)

- Agent Score 20%
  - Public Scoreboard Submission 10%
    - You will get whole 10% when sharing your avg score for 10~20 games in the public ranking board
    - If you submit the fake result, you will get 0 point for this project
      - Ex: submit result from modified game.py
  - Score from your code 10%
    - You can't get the agent score(20%) if your code is not working
    - 5% if your agent can obtain more than 15000 in average
    - +5% if you are in top 10, +4% if you are above 80%, +2% else
- Project Report 80%
  - Containing all requirements?
  - Does the result and analysis is well described?

# Questions?

- TA: Lee, Gyeongbok
  - E-mail: [alias\\_n@yonsei.ac.kr](mailto:alias_n@yonsei.ac.kr)
  - Engineering Hall 4, D718
  - Office Hours: Thu 16:00~17:30
    - Please send email to me before your visit outside of the OH
- Have problem in coding with python?
  - Utilize Q&A Board!
    - Also, student who gives informative answer to the colleagues' question (about programming) will get some extra credit