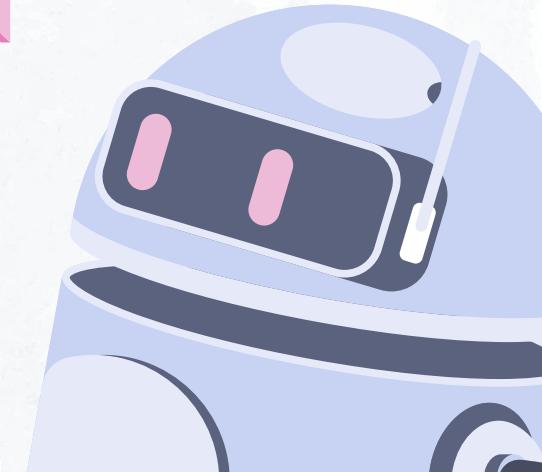


FAI Final Project

Texas Hold'em Casino



FAI 2025 Spring Semester
Instructors: Shang-Tse Chen & Yun-Nung Chen
Due date: 2025/06/11 23:59



(AI)

Update Logs

- 05/28: 補上 LISENCE.md
- 05/29: update raise value restriction

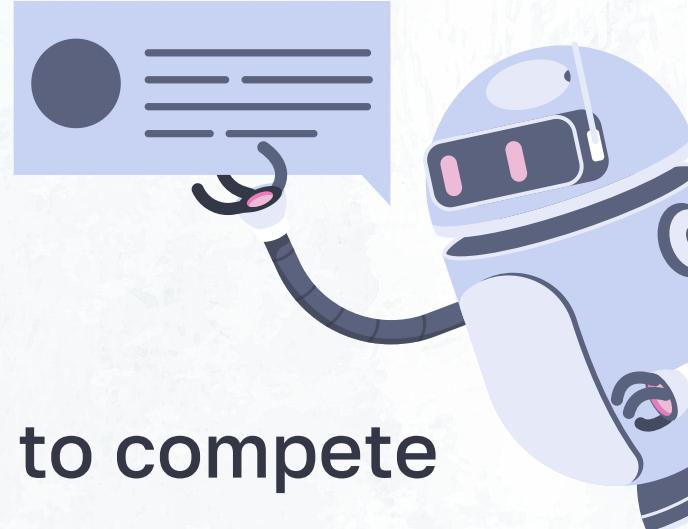
Outline

- 01** → Project Introduction and Rules
- 02** → Game Parameters & Environment in Final Project
- 03** → Sample Code Explanation
- 04** → Competition Rules
- 05** → Submission
- 06** → Grading Policy

01 →

Introduction

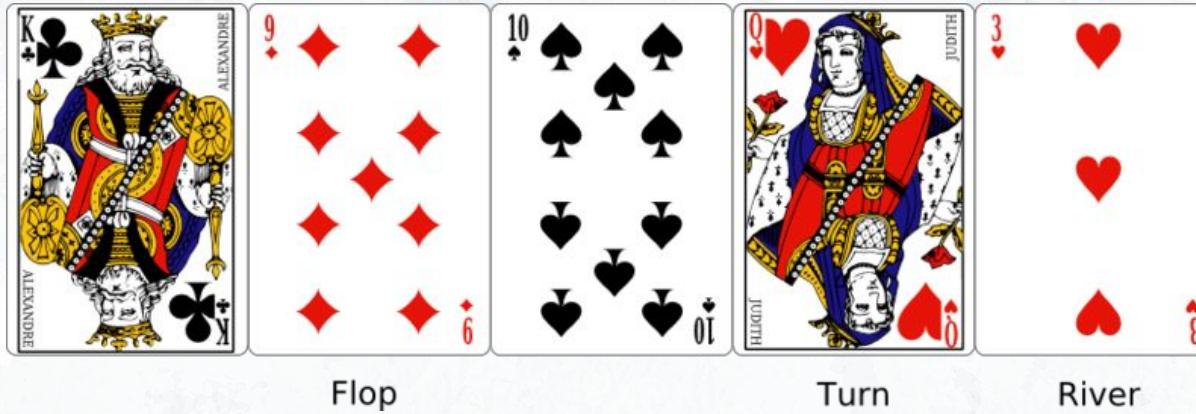
Implement Texas Hold'em AI to compete
in the CSIE casino



(AI)

Texas Hold'em Rules – Heads Up

- Each player has two face-down cards, called the “**hole cards**”
- The face-up cards are called the “**community cards**”
- Goal: use the community cards combined with their hole cards to build a five-card poker hand (Build best 5 combination out of 7 cards)



Small Blind & Big Blind

- A game features **several betting rounds**
- Players get two private and up to five community cards
- All blinds **must** bet
 - 1st player: “small blind”
 - 2nd player: “big blind”
 - Big blind is two times larger than small blind

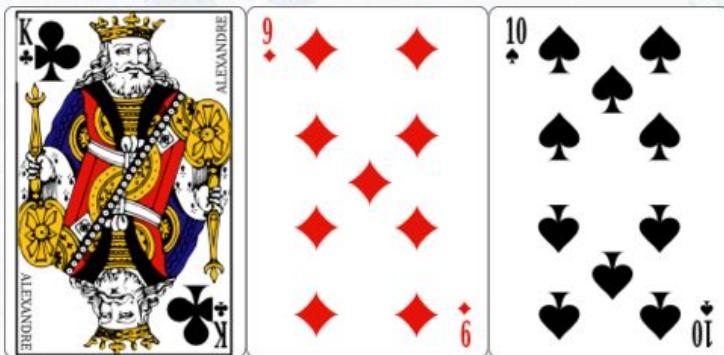


First Betting Round: Preflop

- The player has 3 options
 - **Call:** match the amount of the big blind
 - **Raise:** increase the bet within the limits of the game
 - **Fold:** throw the hand away. If the player chooses to fold, he or she is out of the game and no longer eligible to win the current hand.

Second Betting Round: Flop

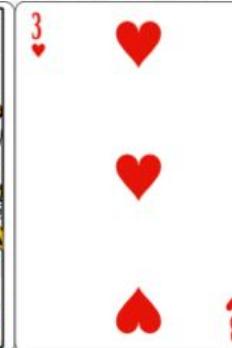
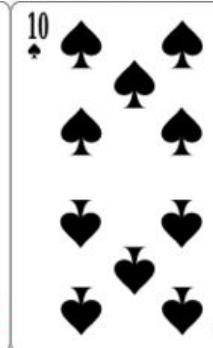
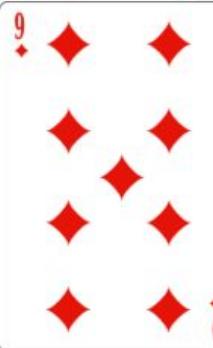
- The first three community cards are dealt
- Players can only choose action “call”, “raise” or “fold”



Flop

Third & Fourth Betting Round: Turn & River

- The fourth community card, called the **turn**, is dealt face-up
- Players can only choose action “call”, “raise” or “fold”
- The fifth community card, called the **river**, is dealt face-up
- Players can only choose action “call”, “raise” or “fold”



Flop

Turn

River

Showdown

- All players expose their holdings to determine a winner.



Hands Rank

- **Royal Flush** — five cards of the same suit, ranked ace through ten; e.g., A♥ K♥ Q♥
J♥ 10♥
- **Straight Flush** — five cards of the same suit and consecutively ranked; e.g., 9♣ 8♣
7♣ 6♣ 5♣
- **Four of a Kind** — four cards of the same rank; e.g., Q♣ Q♥ Q♦ Q♠ 4♦
- **Full House** — three cards of the same rank and two more cards of the same rank; e.g.,
J♣ J♥ J♠ 8♦ 8♥

Hands Rank

- **Flush** — any five cards of the same suit; e.g., A♠ J♠ 8♠ 5♠ 2♠
- **Straight** — any five cards consecutively ranked; e.g., Q♣ J♦ 10♥ 9♠ 8♦
- **Three of a Kind** — three cards of the same rank; e.g., 8♣ 8♠ 8♦ K♣ 4♥
- **Two Pair** — two cards of the same rank and two more cards of the same rank; e.g., A♠ A♣ J♦ J♣ 7♠
- **One Pair** — two cards of the same rank; e.g., 10♥ 10♣ 9♥ 4♦ 2♦
- **High Card** — five unmatched cards; e.g., A♠ J♦ 10♠ 5♣ 2♥ would be called "ace-high"

02 →

Game Parameters & Environment

(AI)

Game Parameters in Final Project

- The winner is the one has more money after all games
 - Max round of game: 20
 - Initial stack for each player: 1000
 - Small blind: 5
- No upper bound for “raise”, you can **all in** at once.
- You must take action within **5 seconds** in every turn, or treated as “**fold**”
- You must take a valid action in every turn, or treated as “**fold**”

Environment

- Install miniconda on `ws{1~7}.csie.org` workstation, following this [guide](#)
 - We recommend to install conda at `/tmp2`
- Create env with name `{env_name}` and python version 3.8.13
 - `conda create -n {env_name} python=3.8.13`
- Activate the environment
 - `conda activate {env_name}`
- Install the packages
 - `pip install -r requirement.txt`
- Run the **`start_game.py`** to see how the game works
 - `python start_game.py`

Environment

- python 3.8.13
- numpy==1.23.5
- torch==2.3.0
- scikit-learn==1.3.2
- tensorflow==2.12.0
- keras==2.12.0
- pytorch_lightning==2.2.4
- tqdm==4.66.4
- If you want to use other packages, please email the TA first.

03 →

Sample Code Explanation

(AI)

Sample Code Explanation

- final_project/
 - |--- start_game.py *<the code to help you test locally>*
 - |--- requirement.txt *<python package needed in this project>*
 - |--- baseline0.cpython-38-x86_64-linux-gnu.so
<the binary file that can be import in csie workstation>
 - |--- game/
<it contains all needed game objects, you should not modify any file in this directory>
 - |--- agents/ *<it contains sample agents for you to play with>*
 - |--- call_player.py *<the agent always to “call” action>*
 - |--- random_player.py *<the agent choose action randomly>*
 - |--- console_player.py *<you can play the game in interaction mode with this player>*

Sample Code Explanation

- Your agent should make parent class as “**BasePokerPlayer**”
- You should override 7 functions
 - declare_action
 - receive_game_start_message
 - receive_round_start_message
 - receive_street_start_message
 - receive_game_update_message
 - Receive_round_result_message
- You should include a function named “**setup_ai**” that return your agent class

```
class CallPlayer< BasePokerPlayer >:
    # Do not forget to make parent class as "BasePokerPlayer"

    # we define the logic to make an action through this method.
    # (so this method would be the core of your AI)
    def declare_action(self, valid_actions, hole_card, round_state):
        # valid_actions format => [raise_action_info, call_action_info, fold_action_info]
        call_action_info = valid_actions[1]
        action, amount = call_action_info["action"], call_action_info["amount"]
        return action, amount # action returned here is sent to the poker engine

    def receive_game_start_message(self, game_info):
        pass

    def receive_round_start_message(self, round_count, hole_card, seats):
        pass

    def receive_street_start_message(self, street, round_state):
        pass

    def receive_game_update_message(self, action, round_state):
        pass

    def receive_round_result_message(self, winners, hand_info, round_state):
        pass

def setup_ai():
    return CallPlayer()
```

Sample Code Explanation

- To test your agent locally, you can use “*python3 start_game.py*”
 1. Import setup_ai function for every agent
 2. Setup game configuration with predefined rules
 3. Register users with the agent
 4. Play the game and get the result

```
1 from game.game import setup_config, start_poker
2 from agents.call_player import setup_ai as call_ai
3 from agents.random_player import setup_ai as random_ai
4 from agents.console_player import setup_ai as console_ai
5
6 config = setup_config(max_round=20, initial_stack=1000, small_blind_amount=5)
7 config.register_player(name="p1", algorithm=call_ai())
8 config.register_player(name="p2", algorithm=random_ai())
9 config.register_player(name="me", algorithm=console_ai())
10 game_result = start_poker(config, verbose=1)
```

Sample Code Explanation

- To play the game interactively, you can use "console_player.py".
 - It allows you to play the game step by step.
 - Baselines cannot be played in an interactive mode; you can only test it locally.

```
1 from game.game import setup_config, start_poker
2 from agents.call_player import setup_ai as call_ai
3 from agents.random_player import setup_ai as random_ai
4 from agents.console_player import setup_ai as console_ai
5
6 config = setup_config(max_round=20, initial_stack=1000, small_blind_amount=5)
7 config.register_player(name="p1", algorithm=call_ai())
8 config.register_player(name="p2", algorithm=random_ai())
9 config.register_player(name="me", algorithm=console_ai())
10 game_result = start_poker(config, verbose=1)
```

Sample Code Explanation

- Example result after submit to baseline server.
- Game Rules
- Game result after playing 20 runs
- p1 won in this example

```
{  
    "rule": {  
        "initial_stack": 1000,  
        "max_round": 20,  
        "small_blind_amount": 5,  
        "ante": 0,  
        "blind_structure": {}  
    },  
    "players": [  
        {  
            "name": "p1",  
            "uuid": "jwdaammhainrbmufgfgodc",  
            "stack": 1996,  
            "state": "participating"  
        },  
        {  
            "name": "p2",  
            "uuid": "ikscmdgatlxjeozsrhhbsd",  
            "stack": 0,  
            "state": "folded"  
        }  
    ]  
}
```

Sample Code Explanation

- We will be given **7 black-box baselines** (different levels of difficulty)
- Baselines are in **binary executable format** compatible in csie workstation
 - <your student id>@ws{1~7}.csie.org
 - If you don't have a csie workstation account, please refer to the [link](#)

```
6 from game.game import setup_config, start_poker
7 from agents.random_player import setup_ai as random_ai
8 from baseline0 import setup_ai as baseline0_ai
9
10 config = setup_config(max_round=20, initial_stack=1000, small_blind_amount=5)
11 config.register_player(name="p1", algorithm=baseline0_ai())
```

04 →

Competition Rules

(AI)

Competition Rules

- Each match (two players compete with each other) consists of 5 games (**B05**)
- Each game consists of 20 rounds starting from 1,000 stacks for both
- The player “**wins**” the game if they have **more money left** after 20 rounds

Competition Rules

- **Baseline Competition**

- You can earn **total 5 points** in a match if you **win 3 or more games**.
- Even if you lose the BO5 match, you can earn at most **3 points** depending on the following:
 - 取前兩場 final stack 最高的:
 - 如果贏了, 得 1.5 分
 - 如果輸了, 但是 $\text{final_stack} \geq \text{initial_stack} * 0.5$, 得到對應比例四捨五入取小數點第一位的分數。例如: $\text{final_stack} = 0.75 * \text{initial_stack} \rightarrow$ 得到 0.8 分
- This scoring system is designed to reduce the impact of randomness in poker by rewarding consistent performance in each game, even if the overall match is lost.

Competition Rules

- **Round-Robin Tournament**
 - Each player is randomly assigned into **a group of 6 people**
 - 5 opponents in total
 - #1: 10pts, #2: 8pts, #3: 6pts, #4: 4pts, #5: 2pts, #6: 0pts
 - Rankings within the group are based on the number of matches each player has won.
 - If there's a tie in match wins, the winner is determined based on the result of the match between the tied participants.

Competition Rules

- **Single Elimination**
 - Top 32 players with highest scores (left money summed over 5 matches) are selected to participate in the single elimination tournament
 - Top 4 can get bonus for the final grades (4pt~1pt)
 - If there's a tie, the winner is determined based on the total stack left across all games in the match.

05 →

Submission

(AI)

Submission

- Due date: 2025/06/11 23:59
- Upload to NTU COOL
- **No late submission is allowed**

Report

- Your report should include but not limited to
 - The methods you have tried
 - Your configurations (e.g. hyperparameters)
 - Comparison of your methods
 - Discussion and conclusion
 - **What method you choose to submit finally.**
- You should write your report in maximum **4 A4 pages in pdf format**
- The grading of the report will base on the **number of methods you tried, completeness, novelty, and clarity of writing**

Submission

- TAs will evaluate both **code functionality** and the **report quality**
- Please compress your agent and the related files in a single .zip named with your student id in **lowercase**.
- **The size of the submitted .zip file should be less than 500MB.**
- Example:
 - bxx902xxx.zip
 - bxx902xxx/
 - |-- report.pdf
 - |-- src/
 - |-- agent.py
 - |-- other related file needed

06 →

Grading Policy

(AI)

Grading (110 pts in total)

- Report (55 pts)
- Baseline Competition (35 pts)
 - baseline1~baseline7 (5 pts for each)
- **Unseen** Strong Agent Competition (10 pts)
 - strong1~strong5 (2 pts for each)
- Round-Robin Tournament (10 pts)
 - Total 5 matches (Points awarded according to your rank.)
- Single Elimination Tournament (bonus 4 pts)

If you have any questions

1. Post your inquiries on NTU COOL course forum
2. Email to: fai-ta@csie.ntu.edu.tw.

Ensure the email title starts with: [FAI25 Final Project]

3. TA Hours: Mon. 14:00~15:00 (Online) and Wed. 14:00~15:00
@德田館 R524

