Report

> 20221494 박찬솔

Requirements

The program uses `redis` and `memcache` library and server, so extra installation process required.

```
pip3 install python3-memcached redis
```

```
apt install memcached redis-server
```

Run server

```
memcached -d -m 1024 -u root -l 127.0.0.1 -p 11211
```

Screenshot

Send rule information to client from server & select game mode

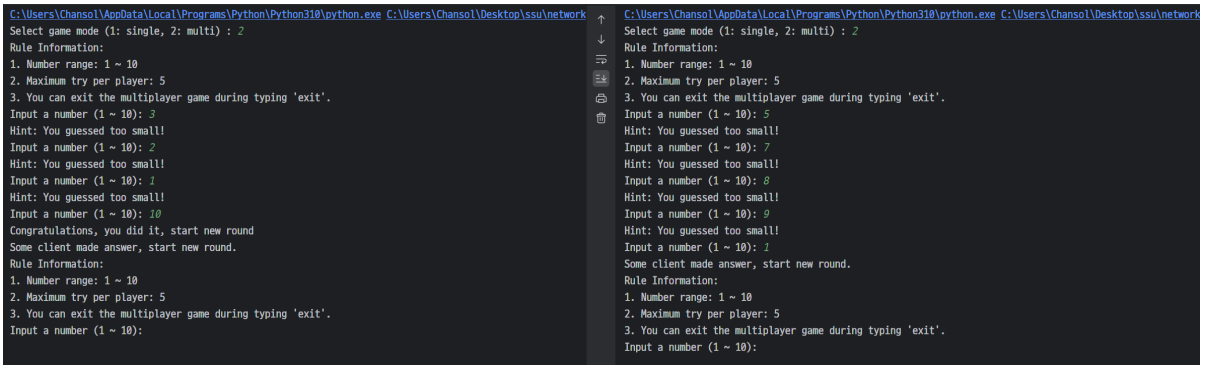
```
Select game mode (1: single, 2: multi) : 1
Rule Information:
1. Number range: 1 ~ 10
2. Maximum try per player: 5
3. You can exit the multiplayer game during typing 'exit'.
Input a number (1 ~ 10): exit
Select game mode (1: single, 2: multi) : 2
Rule Information:
1. Number range: 1 ~ 10
2. Maximum try per player: 5
3. You can exit the multiplayer game during typing 'exit'.
Input a number (1 ~ 10): exit
```

[Single Player] Game Process

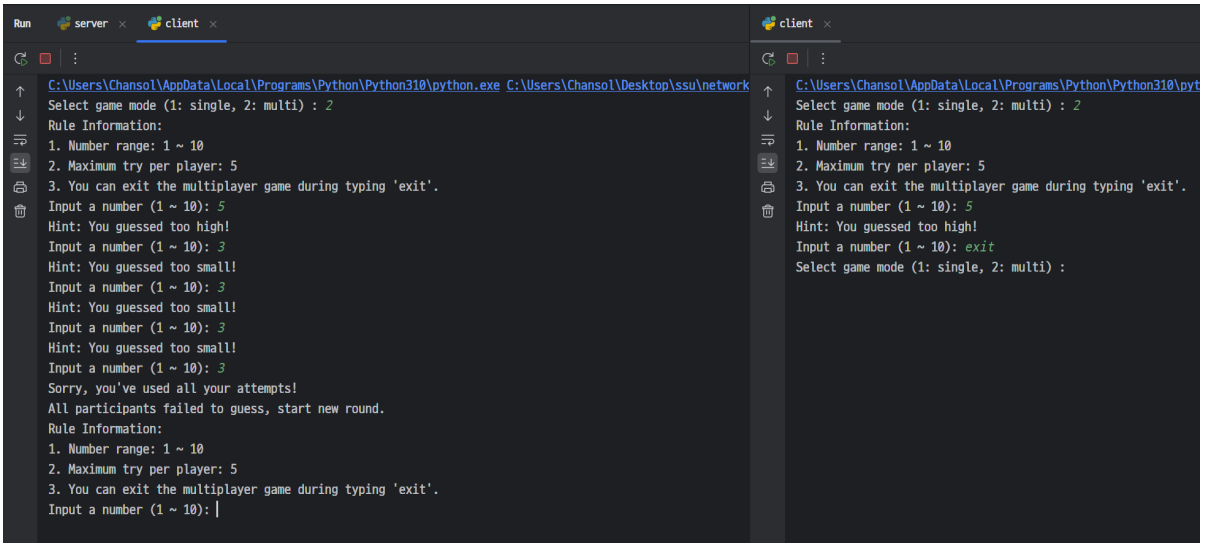
```
C:\Users\Chansol\AppData\Local\Programs\Python\Python310\pyt
Select game mode (1: single, 2: multi) : 1
Rule Information:
1. Number range: 1 ~ 10
2. Maximum try per player: 5
3. You can exit the multiplayer game during typing 'exit'.
Input a number (1 ~ 10): 5
Hint: You guessed too high!
Input a number (1 ~ 10): 3
Congratulations, you did it, start new round
Rule Information:
1. Number range: 1 ~ 10
2. Maximum try per player: 5
3. You can exit the multiplayer game during typing 'exit'.
Input a number (1 ~ 10): 1
Hint: You guessed too small!
Input a number (1 ~ 10): 1
Hint: You guessed too small!
Input a number (1 ~ 10): 1
Hint: You guessed too small!
Input a number (1 ~ 10): 1
Hint: You guessed too small!
Input a number (1 ~ 10): 1
Sorry, you've used all your attempts!
Rule Information:
1. Number range: 1 ~ 10
2. Maximum try per player: 5
3. You can exit the multiplayer game during typing 'exit'.
Input a number (1 ~ 10): 5
```

- When starts each round, game rule will inform to client.
- Round will end in guessing number successfully or using all given attempts.
- Rounds will start automatically after round ends.

[Multi Player] Game Process



- Note that client can join a already started game.
- If a one of participating players make answer, current game ends immediately.
- If all participating players uses given all attempts, current game ends immediately.
- After the game ends, next game will be started immediately.



- It is also allowed to leave game during playing the game.
- Server will process well.

[Multi Player] Message Queue

- Message Queue uses `redis`.
- Message Queue only used in broadcasting message.
  - Send all clients to `All participants failed to guess, start new round.` or,
  - `Some client made answer, start new round.`
- Also it used to inform all client to finish current game.

Memcache

- The server manages the count that client make answer.
- Code snippet: `memcache_client.incr(self.key(), 1)`
- The session will be distinguish by "`{REMOTE_ADDR}:{REMOTE_PORT}`".

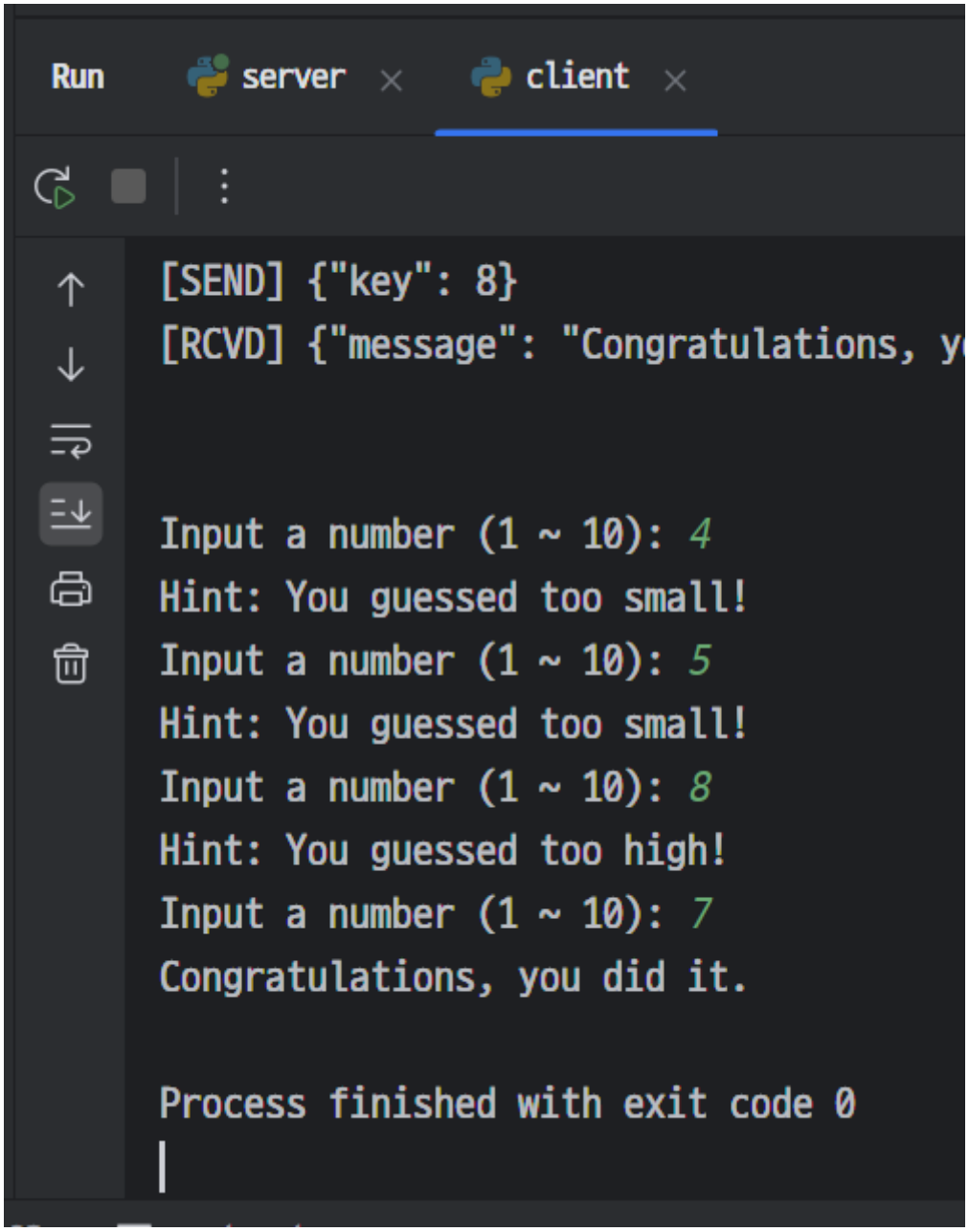
Another function

- Another functionalities such as TLS, Logging, Error Handling are same as previous homework, HW02.
- I will attach a report of HW02 after this report.

Report

> 20221494 박찬솔

Server-Client Communication



Data handling (JSON) & Log

server.py | server_log.txt

```
[CONNECTION] service started.
[CONNECTION] socket accepted.
[RECV] b'start.'
[GAME] Set initial number to 9
[RECV] {"key": 1}
[SEND] Hint: You guessed too small!
[RECV] {"key": 5}
[SEND] Hint: You guessed too small!
[RECV] {"key": 7}
[SEND] Hint: You guessed too small!
[RECV] {"key": 9}
[SEND] Congratulations, you did it.
[CONNECTION] service started.
[CONNECTION] service started.
[CONNECTION] socket accepted.
[RECV] b'start.'
[GAME] Set initial number to 4
[RECV] {"key": 5}
[SEND] {"message": "Hint: You guessed too high!", "finish": false}
[RECV] {"key": 3}
[SEND] {"message": "Hint: You guessed too small!", "finish": false}
[RECV] {"key": 4}
[SEND] {"message": "Congratulations, you did it.", "finish": true}
[CONNECTION] close socket
```

client_log.txt

```
[CONNECTION] connected.
[SEND] start.
[SEND] {"key": 5}
[RCVD] {"message": "Hint: You guessed too small!", "finish": false}
[SEND] {"key": 7}
[RCVD] {"message": "Hint: You guessed too small!", "finish": false}
[SEND] {"key": 9}
[RCVD] {"message": "Hint: You guessed too high!", "finish": false}
[SEND] {"key": 8}
[RCVD] {"message": "Congratulations, you did it.", "finish": true}
[CONNECTION] disconnected.
[CONNECTION] connected.
[SEND] start.
[CONNECTION] disconnected.
[ERROR] invalid input
[CONNECTION] connected.
[SEND] start.
[SEND] {"key": 4}
[RCVD] {"message": "Hint: You guessed too small!", "finish": false}
[SEND] {"key": 5}
[RCVD] {"message": "Hint: You guessed too small!", "finish": false}
[SEND] {"key": 8}
[RCVD] {"message": "Hint: You guessed too high!", "finish": false}
[SEND] {"key": 7}
[RCVD] {"message": "Congratulations, you did it.", "finish": true}
[CONNECTION] disconnected.
[CONNECTION] connected.
[SEND] start.
[SEND] {"key": 5}
[RCVD] {"message": "Hint: You guessed too high!", "finish": false}
[SEND] {"key": 3}
[RCVD] {"message": "Hint: You guessed too small!", "finish": false}
[SEND] {"key": 4}
[RCVD] {"message": "Congratulations, you did it.", "finish": true}
[CONNECTION] disconnected.
```

- It records all exchanged message, and connection/error/game information.

Game History (And replay)

server.py | server_games.txt


```
["789c6b60995ac7c80006b153347a04a2835c9dc3621592d48b4b128b4af4d4a7f4084285aa95b2532b95ac144c6ba7f4384707bbfab980c472538b8b13d35381e24a1e997925560a9
```


- Save play data as json with hex.

```
encoded = pickle.dumps(curr_game)
compressed = zlib.compress(encoded)
game_data.append(compressed.hex())
```

- It is compressed and saved as pickle & zlib.
- Replay game record when client or server start.

```
C:\Users\Chansol\AppData\Local\Programs\Python\Python310\python.exe C:\Users\Chansol\Desktop\ssu\network-2024\hw02\server.py
Showing previous game data log...
0th game data:
[RECV] b'start.'
[RECV] {"key": 5}
[SEND] {"message": "Hint: You guessed too small!", "finish": false}
[RECV] {"key": 7}
[SEND] {"message": "Hint: You guessed too small!", "finish": false}
[RECV] {"key": 9}
[SEND] {"message": "Hint: You guessed too high!", "finish": false}
[RECV] {"key": 8}
[SEND] {"message": "Congratulations, you did it.", "finish": true}

1th game data:
[RECV] b'start.'
[RECV] {"key": 5}
[SEND] {"message": "Hint: You guessed too small!", "finish": false}
[RECV] {"key": 7}
[SEND] {"message": "Hint: You guessed too high!", "finish": false}
[RECV] {"key": 6}
[SEND] {"message": "Congratulations, you did it.", "finish": true}

2th game data:
[RECV] b'start.'
[RECV]

3th game data:
[RECV] b'start.'
[RECV] {"key": 5}
[SEND] {"message": "Hint: You guessed too small!", "finish": false}
[RECV] {"key": 7}
[SEND] {"message": "Hint: You guessed too small!", "finish": false}
[RECV] {"key": 9}
[SEND] {"message": "Hint: You guessed too small!", "finish": false}
[RECV]

4th game data:
```

Error Handling

- It handles all errors of socket, and record them to log.

Security

Communication between client and server is protected by TLS with self-signed private key using locally generated CA.

```python
context = ssl.SSLContext(protocol=ssl.PROTOCOL_TLS_SERVER)
context.load_cert_chain( certfile: "cert.crt",  keyfile: "key.pem")
context.check_hostname = False

print("Server listening on 127.0.0.1:12345")
logs.write("[CONNECTION] %s" % "service started.\n")

while True:
    raw_sock, _ = sock.accept()
    sc = context.wrap_socket(raw_sock, server_side=True)
```