- 과제 목표 (도출해야 할 결과)
  ➔ 소켓 프로그래밍을 통한 서버 / 클라이언트 구현

- 코드 설명과 과제 해결 방법

1. 서버
   - 소켓을 만들고, bind를 통해 ip 주소와 포트 번호를 바인딩한다.

```
TCP_IP = '192.168.0.24'
TCP_PORT = 5001

# socket -> bind -> listen -> accept
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind((TCP_IP, TCP_PORT))
```

   - 소켓으로 오는 connect 요청을 3개까지 listen 하고, accept 한다.
   - accept의 결과로 connection이 나오는데, 이를 이용하여 한 소켓에 여러 연결을 할 수 있다. 이 후로는 send / recv를 이용하여 통신할 수 있다.
   - 각 연결들을 리스트에 추가한 뒤, 전체에 메시지를 보낸다.

```
# Socket 으로 오는 connect 요청 listen, 3개까지 accept
# accept 과정에서 conn 이 나오는데, 이를 이용하면 한 소켓에 여러 연결 가능
# 한 개의 conn 은 각 인스턴스의 연결
connections = []
while True:
    sock.listen()
    conn, addr = sock.accept()
    connections.append(conn)
    print('Connection address : ' + str(addr))
    if len(connections) == 3:
        break

# 개별 connection 들에 메시지 보냄
for conn in connections:
    conn.send("Okay... All players have gathered. Start the game.\n"\
              "Please select 1 number from 1 to 10.".encode())
```

   - Client가 순번으로 선택할 난수와, Client 별로 할당될 난수를 생성한다.

```
# Client 가 선택할 난수생성
ten_random_numbers = []
for i in range(10):
    ten_random_numbers.append(randint(1, 101))

# Client 별로 할당할 난수 생성
four_random_numbers = []
for i in range(4):
    four_random_numbers.append(randint(-1, 5))
```

- 3개의 Connection 목록을 순회하며 순서대로 숫자 입력을 받아오고, 그에 따른 결과를 돌려준다. 클라이언트가 보낼 때에는 서버는 수신모드여야 하고, 그 반대의 경우도 마찬가지이므로 때에 맞게 모드를 send / receive 를 설정해준다. 무슨 난수가 골라졌는지 서버 입장에서는 알 수 있도록 하기 위해, 출력문을 추가하였다..

```
# 3개의 클라이언트로부터 숫자 입력 받아옴
choose_result_list = []
for conn in connections:

    # input number 받음
    data = conn.recv(1024)
    if not data: break
    received_number = data.decode()

    # SERVER LOG
    print('Received Data : ' + received_number)

    # input number 결과 보냄
    choose_result = ten_random_numbers[int(received_number)]
    choose_result_list.append(choose_result)
    choose_result_prompt = "You chose the number " + str(choose_result) + ". Please wait."
    conn.send(choose_result_prompt.encode())
```

- 이 또한 마찬가지로, Connection 목록을 순회하며 순서대로 연산 입력을 받아온 후, 그에 따른 결과를 계산한다.

```
# 3개의 클라이언트로부터 arith 입력 받아옴, 계산
final_result_list = []
conn_idx = 0
for conn in connections:

    # arith 선택 안내 전송
    temp = "Do you want multiply or add...?"
    conn.send(temp.encode())

    # input arith 받음
    data = conn.recv(1024)
    if not data: break
    received_arith = data.decode()

    # SERVER LOG
    print('Received arith : ' + received_arith)
```

```
# 고른 순번의 난수와, 사용자 별 할당된 난수를 계산하여 보내기
final_result = -10000
if received_arith == "add":
    final_result = choose_result_list[conn_idx] + four_random_numbers[conn_idx]

elif received_arith == "multiply":
    final_result = choose_result_list[conn_idx] * four_random_numbers[conn_idx]

else:
    for conn in connections:
        conn.close()
    print("error. plz retry")

final_result_list.append(final_result)

# SERVER LOG
print('Received result : ' + str(final_result))

conn.send("Okay... please wait.".encode())

conn_idx += 1
```

- 점수 목록을 순회하며, 최고점인 경우 승리 메시지를 보내고 아닐 경우 패배 메시지를 보낸다. 이후 connection 들을 모두 닫는다.

```
# 승자 확인
win_number = max(final_result_list)
for conn_idx in range(len(connections)):
    if final_result_list[conn_idx] == win_number:
        connections[conn_idx].send("Congratulations. You won!".encode())
    else:
        connections[conn_idx].send("Unfortunately, you have been defeated.".encode())


# Connection 들 전부 닫기
for conn in connections:
    conn.close()
```

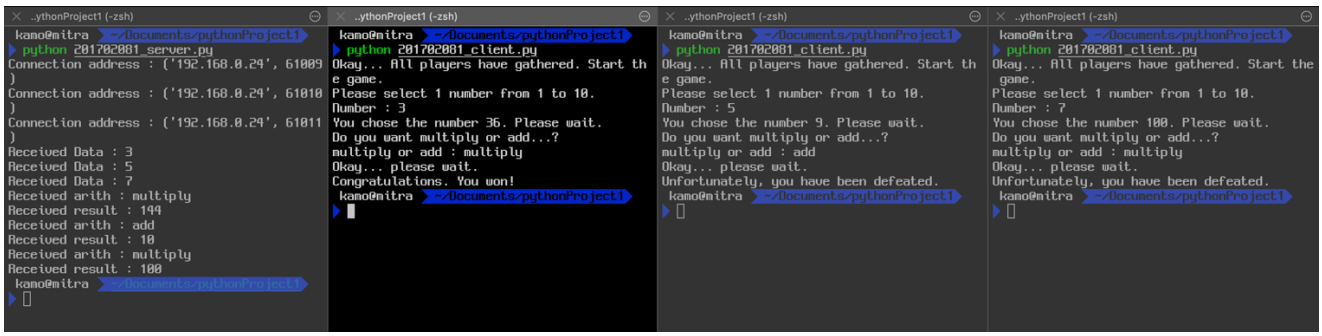
2. 클라이언트
   - 소켓을 만들고, 서버에게 connect 요청을 보낸다. 서버가 listen 중이라면 둘이 연결되어 send / recv 로 통신할 수 있다.

```
# socket -> connect -> send / recv -> close
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((TCP_IP, TCP_PORT))
```

- 서버가 보낼 때에는 recv 하고, 서버에게 보내야 할 때에는 send를 한다.
- 한쪽이 recv 하고 있을 때에는 다른 쪽은 send를 하고 있어야 한다.
- 마지막에는 소켓을 닫아준다.

```
# 3명 다 모였을때의 확인 & 숫자입력 메시지 기다림
print(sock.recv(1024).decode())

# 숫자 입력 보냄
sock.send(input("Number : ").encode())

# 숫자 결과 받음
print(sock.recv(1024).decode())


# 3명에게 숫자 입력 다 받았을 때의 확인 메시지 기다림
print(sock.recv(1024).decode())

# 연산 입력 보냄
sock.send(input("multiply or add : ").encode())

# 결과 기다리라는 알림 받음
print(sock.recv(1024).decode())

# 최종 결과 받음
print(sock.recv(1024).decode())

sock.close()
```

3. 실행 결과
   - 서버와 클라이언트 간 상호작용이 잘 일어남을 확인할 수 있었다.
   - https://youtu.be/J-RD6WOJGCc



- 느낀 점 및 하고싶은 말

코드 설명 스크린샷도 찍고 영상도 찍으려니 너무 힘들다