presentation

MariFlow 재현 프로젝트

project_intro

00.

MariFlow 프로젝트 소개

목표 : 순환신경망 (RNN)을 이용해 Mario Kart를 자율 주행하도록 학습시키는 AI 시스템 구현

주요 기술 : Python, TensorFlow/PyTorch, BizHawk, Emulator, RNN(LSTM)

원작자 : SethBling

project_flow

전체 프로젝트 흐름

1단계

환경 설정
(Python + 에뮬레이터)

2단계

플레이 데이터 수집

3단계

RNN 모델 설계 및 훈련

4단계

실시간 주행 테스트

5단계

시연 및 발표

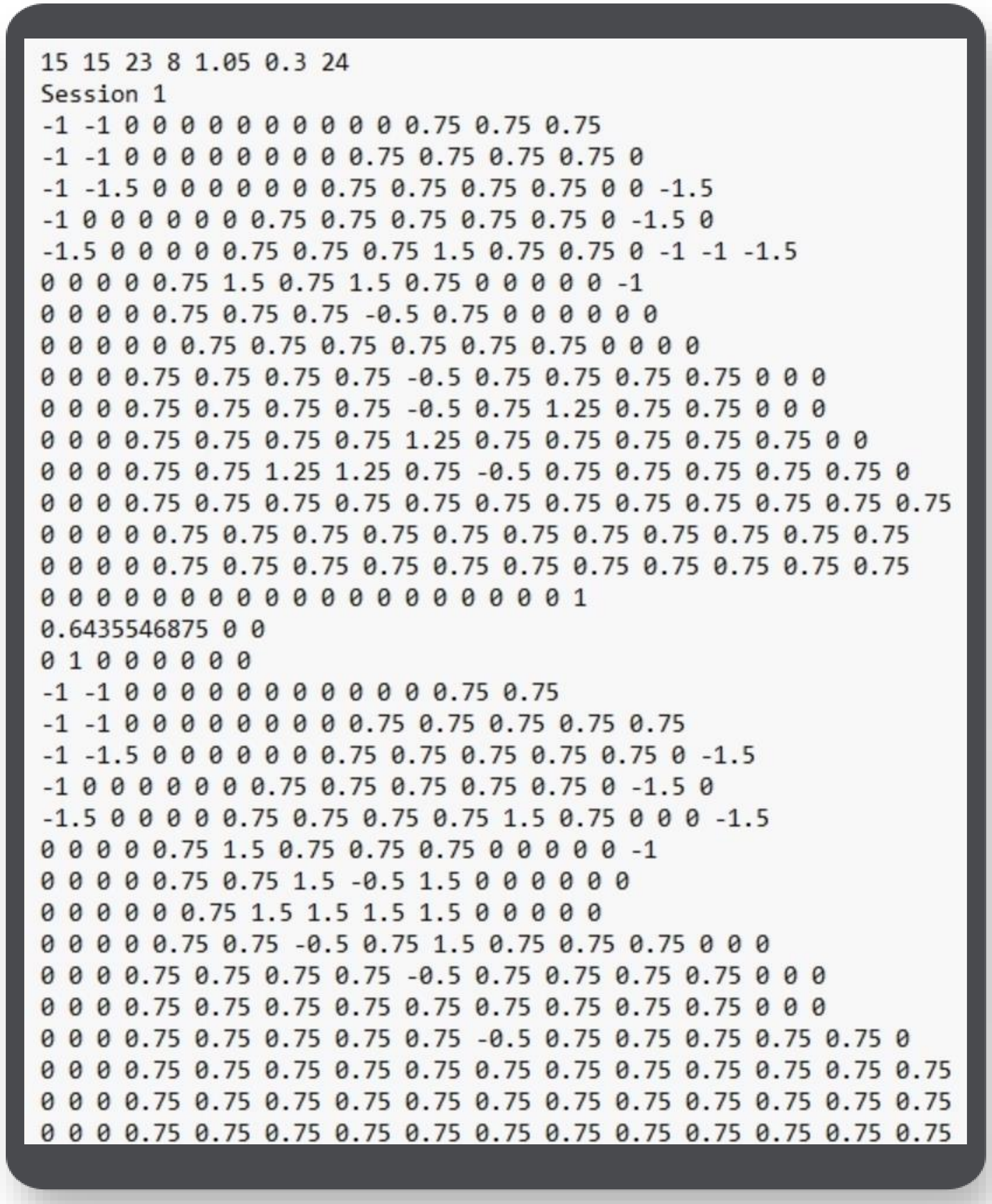
project_setup

1단계 : 환경 설정

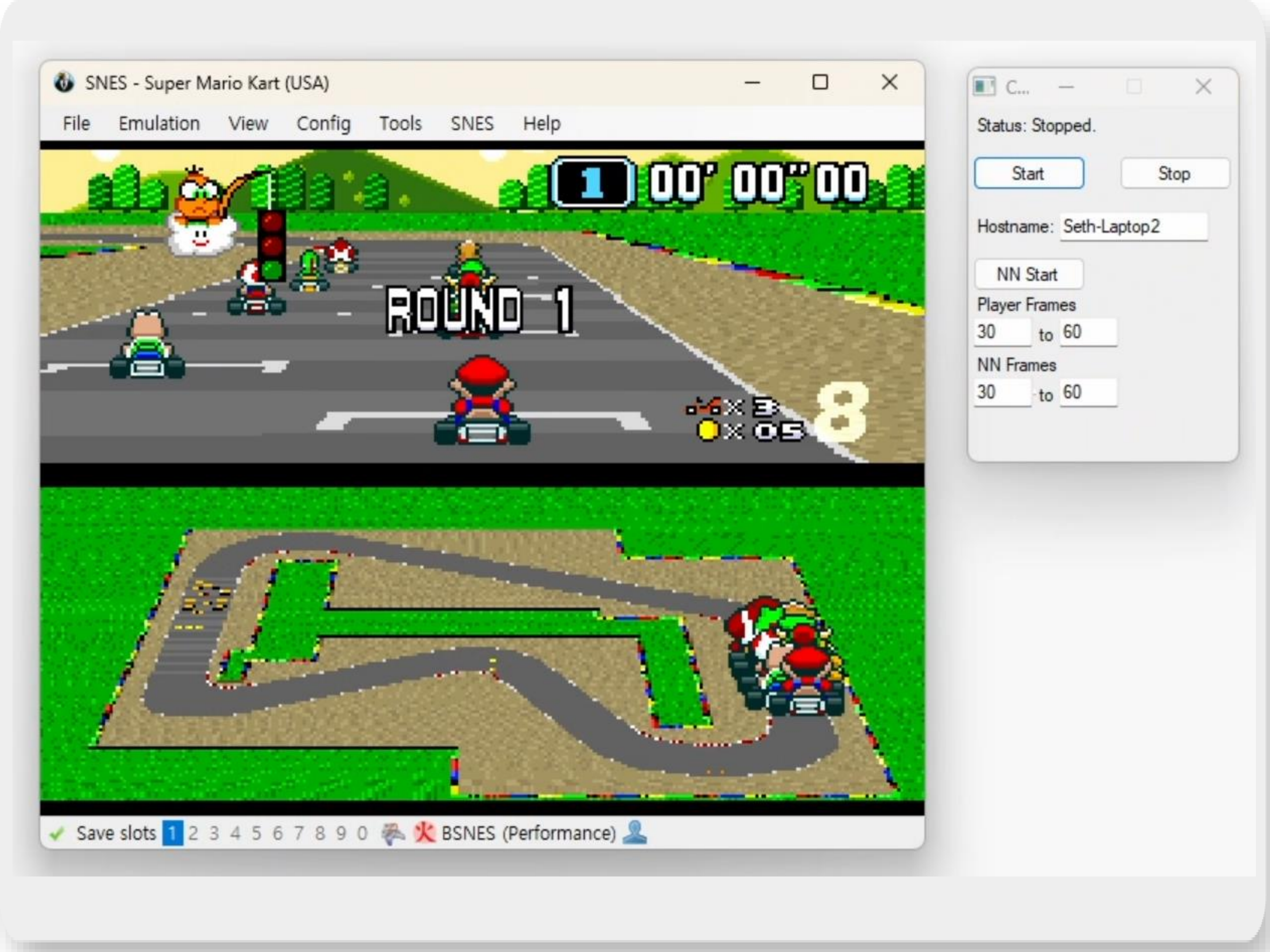
- Python 3.5 → 3.8.5로 변경
- TensorFlow 1.3 → 2.xx로 변경
- BizHawk 1.12.2 + Lua 연동
- Super Mario Kart ROM
- Python과 에뮬레이터 연동 (스크립트로 조작)

→ 결과물 : Mario를 최소 10초 동안 움직이는 Python 스크립트

data_collection_process



```
15 15 23 8 1.05 0.3 24
Session 1
-1 -1 0 0 0 0 0 0 0 0 0 0 0.75 0.75 0.75
-1 -1 0 0 0 0 0 0 0 0 0.75 0.75 0.75 0.75 0
-1 -1.5 0 0 0 0 0 0 0.75 0.75 0.75 0.75 0 0 -1.5
-1 0 0 0 0 0 0 0.75 0.75 0.75 0.75 0.75 0 -1.5 0
-1.5 0 0 0 0 0.75 0.75 0.75 1.5 0.75 0.75 0 -1 -1 -1.5
0 0 0 0 0.75 1.5 0.75 1.5 0.75 0 0 0 0 0 -1
0 0 0 0 0.75 0.75 0.75 -0.5 0.75 0 0 0 0 0 0
0 0 0 0 0 0.75 0.75 0.75 0.75 0.75 0.75 0 0 0 0
0 0 0 0.75 0.75 0.75 0.75 -0.5 0.75 0.75 0.75 0.75 0 0 0
0 0 0 0.75 0.75 0.75 0.75 -0.5 0.75 1.25 0.75 0.75 0 0 0
0 0 0 0.75 0.75 0.75 0.75 1.25 0.75 0.75 0.75 0.75 0.75 0 0
0 0 0 0.75 0.75 1.25 1.25 0.75 -0.5 0.75 0.75 0.75 0.75 0.75 0
0 0 0 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75
0 0 0 0 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75
0 0 0 0 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0.6435546875 0 0
0 1 0 0 0 0 0 0
-1 -1 0 0 0 0 0 0 0 0 0 0 0 0.75 0.75
-1 -1 0 0 0 0 0 0 0 0 0.75 0.75 0.75 0.75 0.75
-1 -1.5 0 0 0 0 0 0 0.75 0.75 0.75 0.75 0.75 0 -1.5
-1 0 0 0 0 0 0 0.75 0.75 0.75 0.75 0.75 0 -1.5 0
-1.5 0 0 0 0 0.75 0.75 0.75 0.75 1.5 0.75 0 0 0 -1.5
0 0 0 0 0.75 1.5 0.75 0.75 0.75 0 0 0 0 0 -1
0 0 0 0 0.75 0.75 1.5 -0.5 1.5 0 0 0 0 0 0
0 0 0 0 0 0.75 1.5 1.5 1.5 1.5 0 0 0 0 0
0 0 0 0 0.75 0.75 -0.5 0.75 1.5 0.75 0.75 0.75 0 0 0
0 0 0 0.75 0.75 0.75 0.75 -0.5 0.75 0.75 0.75 0.75 0 0 0
0 0 0 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0 0 0
0 0 0 0.75 0.75 0.75 0.75 0.75 -0.5 0.75 0.75 0.75 0.75 0.75 0
0 0 0 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75
0 0 0 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75
0 0 0 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75
```

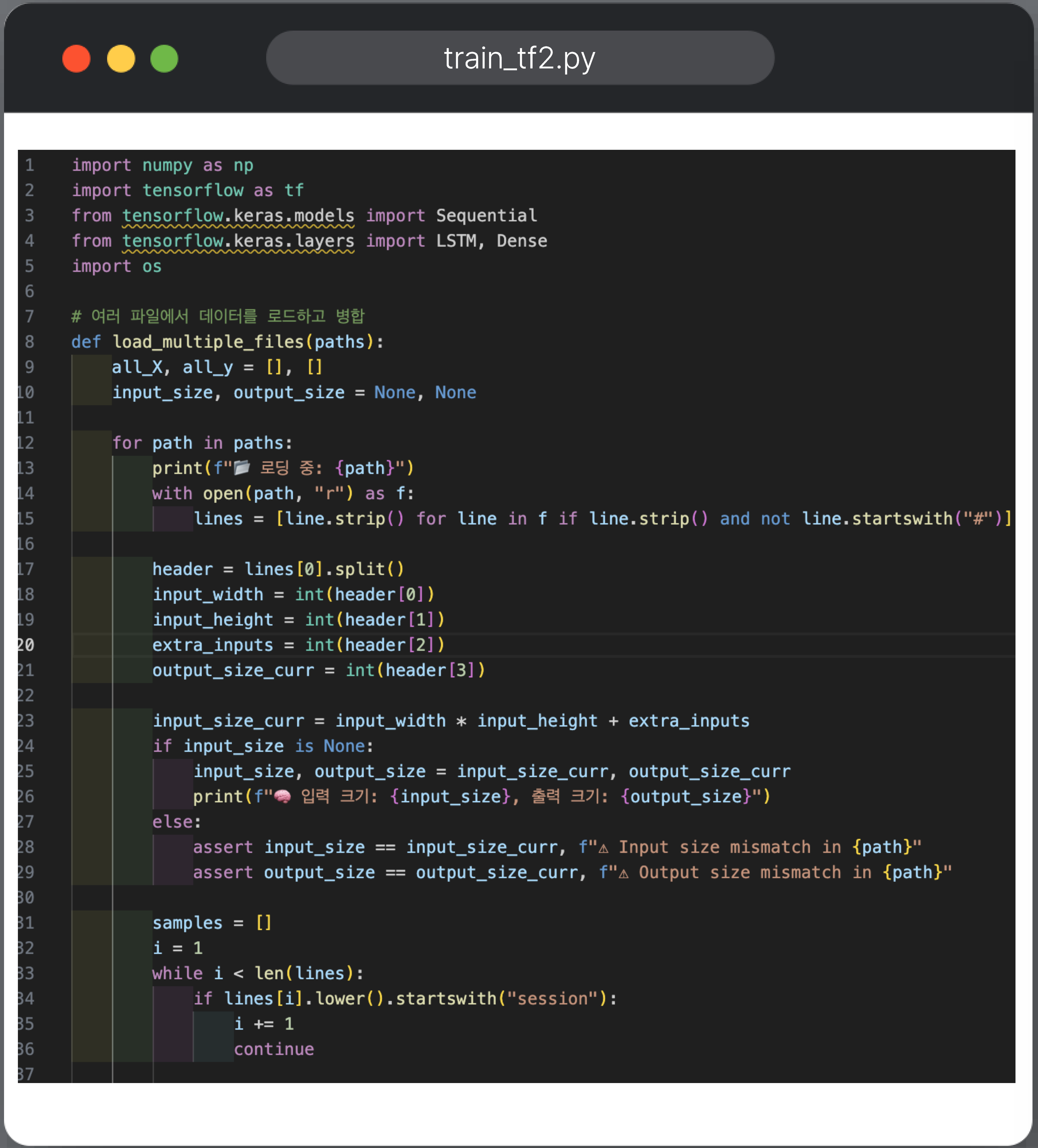


Status: Stopped.

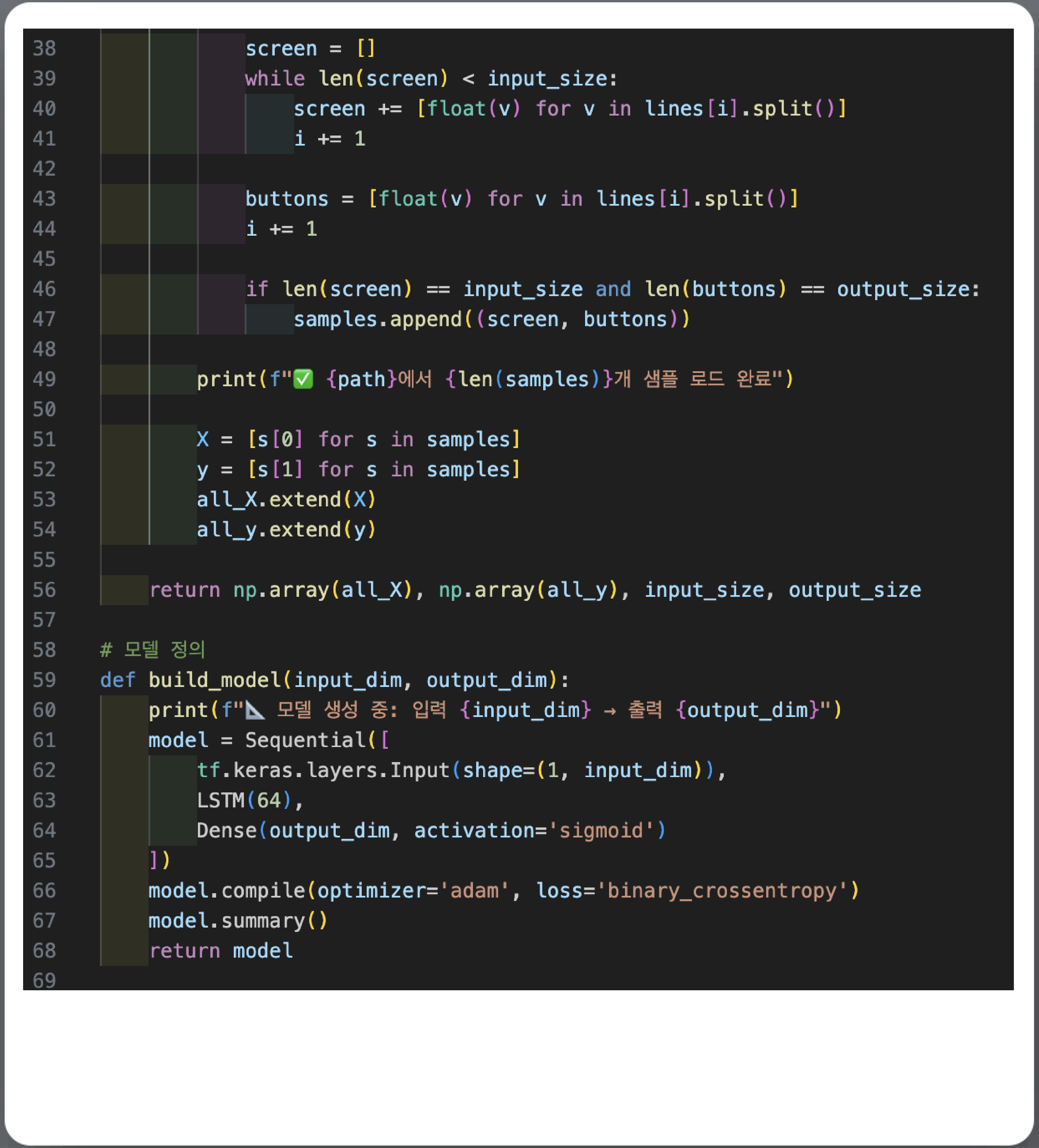
Hostname: Seth-Laptop2

Player Frames 30 to 60

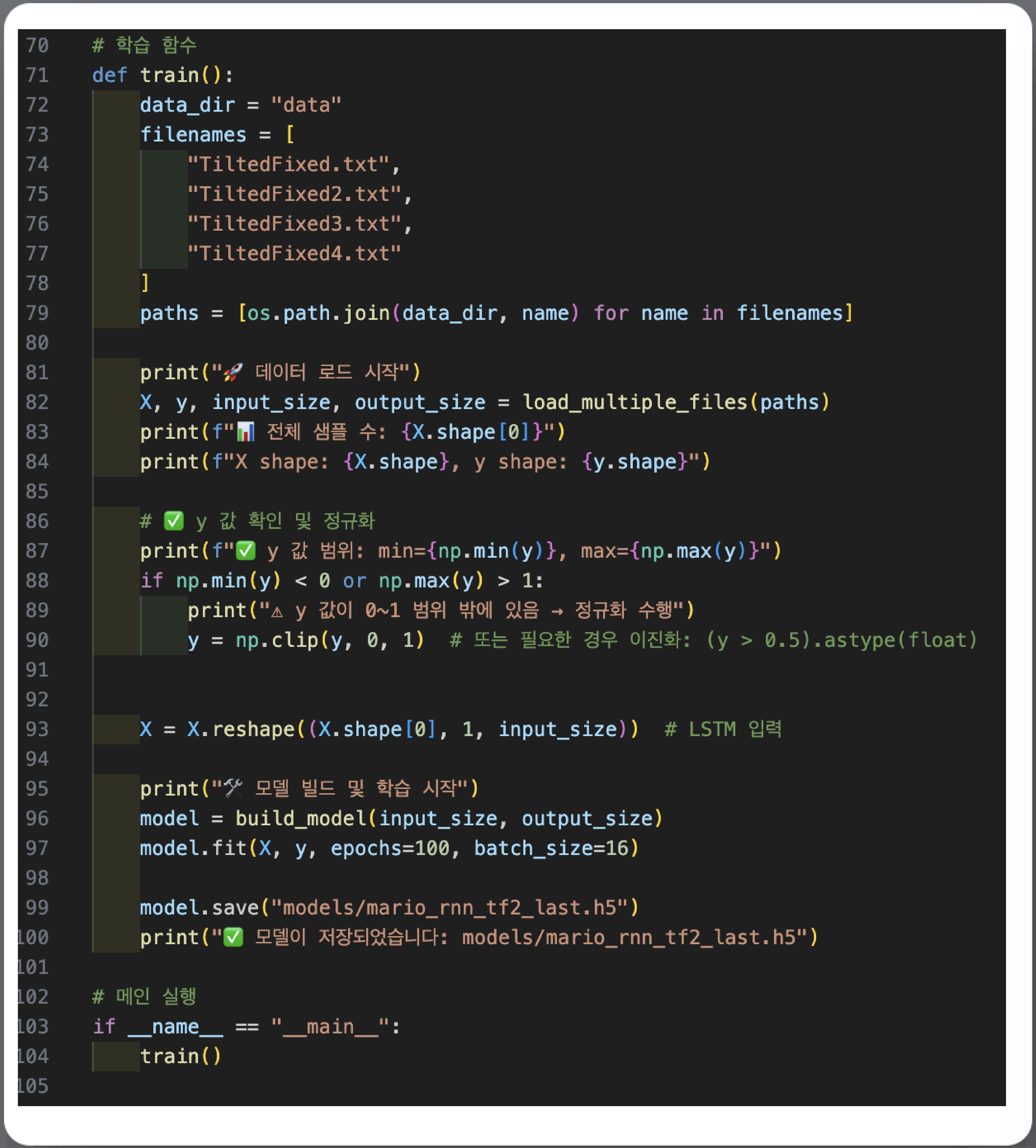
NN Frames 30 to 60



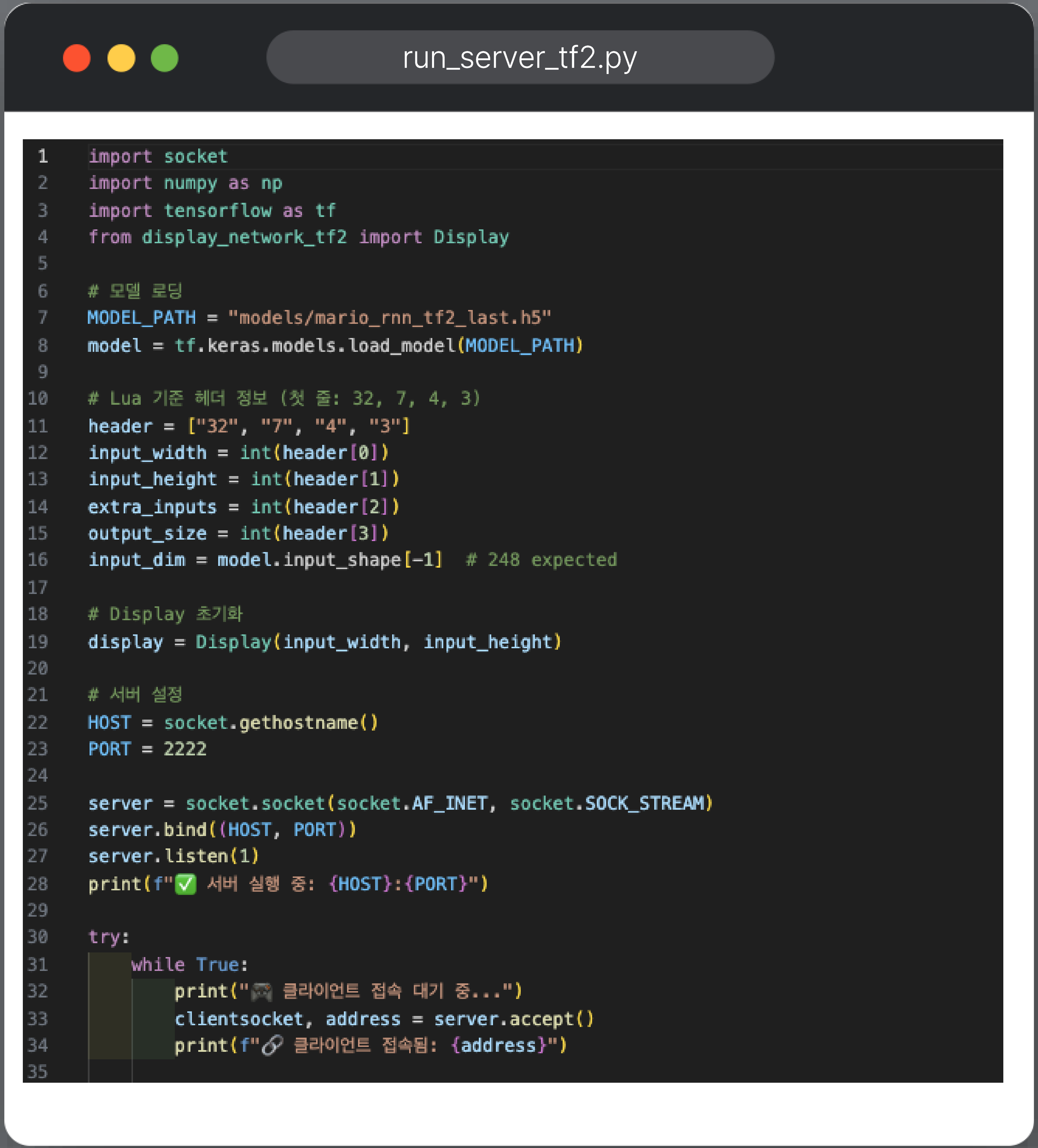
```python
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import os

# 여러 파일에서 데이터를 로드하고 병합
def load_multiple_files(paths):
    all_X, all_y = [], []
    input_size, output_size = None, None

    for path in paths:
        print(f"📂 로딩 중: {path}")
        with open(path, "r") as f:
            lines = [line.strip() for line in f if line.strip() and not line.startswith("#")]

        header = lines[0].split()
        input_width = int(header[0])
        input_height = int(header[1])
        extra_inputs = int(header[2])
        output_size_curr = int(header[3])

        input_size_curr = input_width * input_height + extra_inputs
        if input_size is None:
            input_size, output_size = input_size_curr, output_size_curr
            print(f"🧠 입력 크기: {input_size}, 출력 크기: {output_size}")
        else:
            assert input_size == input_size_curr, f"⚠ Input size mismatch in {path}"
            assert output_size == output_size_curr, f"⚠ Output size mismatch in {path}"

        samples = []
        i = 1
        while i < len(lines):
            if lines[i].lower().startswith("session"):
                i += 1
                continue

```



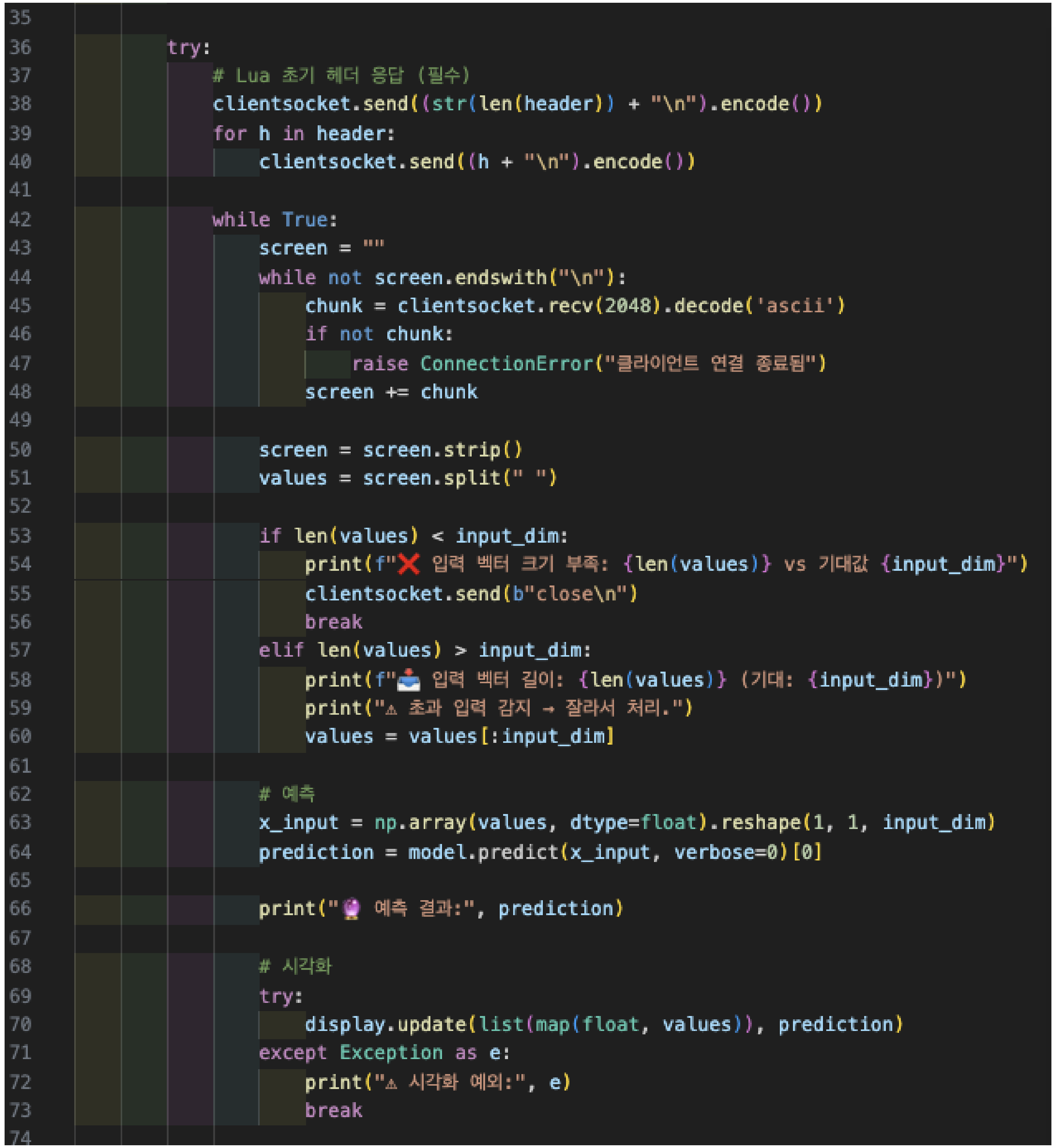
```python
            screen = []
            while len(screen) < input_size:
                screen += [float(v) for v in lines[i].split()]
                i += 1

            buttons = [float(v) for v in lines[i].split()]
            i += 1

            if len(screen) == input_size and len(buttons) == output_size:
                samples.append((screen, buttons))

        print(f"✅ {path}에서 {len(samples)}개 샘플 로드 완료")

        X = [s[0] for s in samples]
        y = [s[1] for s in samples]
        all_X.extend(X)
        all_y.extend(y)

    return np.array(all_X), np.array(all_y), input_size, output_size

# 모델 정의
def build_model(input_dim, output_dim):
    print(f"📐 모델 생성 중: 입력 {input_dim} → 출력 {output_dim}")
    model = Sequential([
        tf.keras.layers.Input(shape=(1, input_dim)),
        LSTM(64),
        Dense(output_dim, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy')
    model.summary()
    return model

```



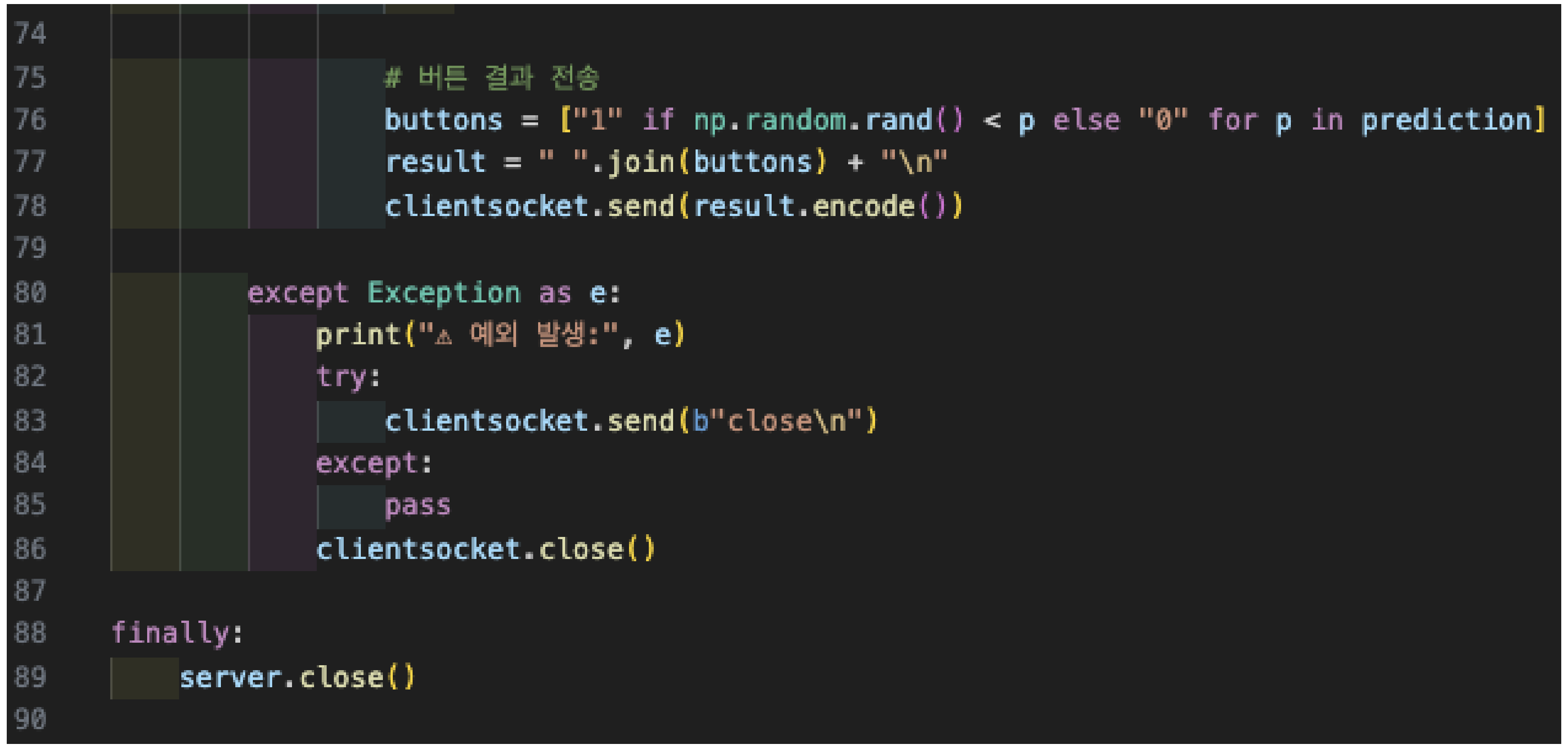
```python
# 학습 함수
def train():
    data_dir = "data"
    filenames = [
        "TiltedFixed.txt",
        "TiltedFixed2.txt",
        "TiltedFixed3.txt",
        "TiltedFixed4.txt"
    ]
    paths = [os.path.join(data_dir, name) for name in filenames]

    print("🚀 데이터 로드 시작")
    X, y, input_size, output_size = load_multiple_files(paths)
    print(f"📊 전체 샘플 수: {X.shape[0]}")
    print(f"X shape: {X.shape}, y shape: {y.shape}")

    # ✅ y 값 확인 및 정규화
    print(f"✅ y 값 범위: min={np.min(y)}, max={np.max(y)}")
    if np.min(y) < 0 or np.max(y) > 1:
        print("⚠ y 값이 0~1 범위 밖에 있음 → 정규화 수행")
        y = np.clip(y, 0, 1)  # 또는 필요한 경우 이진화: (y > 0.5).astype(float)


    X = X.reshape((X.shape[0], 1, input_size))  # LSTM 입력

    print("🛠 모델 빌드 및 학습 시작")
    model = build_model(input_size, output_size)
    model.fit(X, y, epochs=100, batch_size=16)

    model.save("models/mario_rnn_tf2_last.h5")
    print("✅ 모델이 저장되었습니다: models/mario_rnn_tf2_last.h5")

# 메인 실행
if __name__ == "__main__":
    train()

```



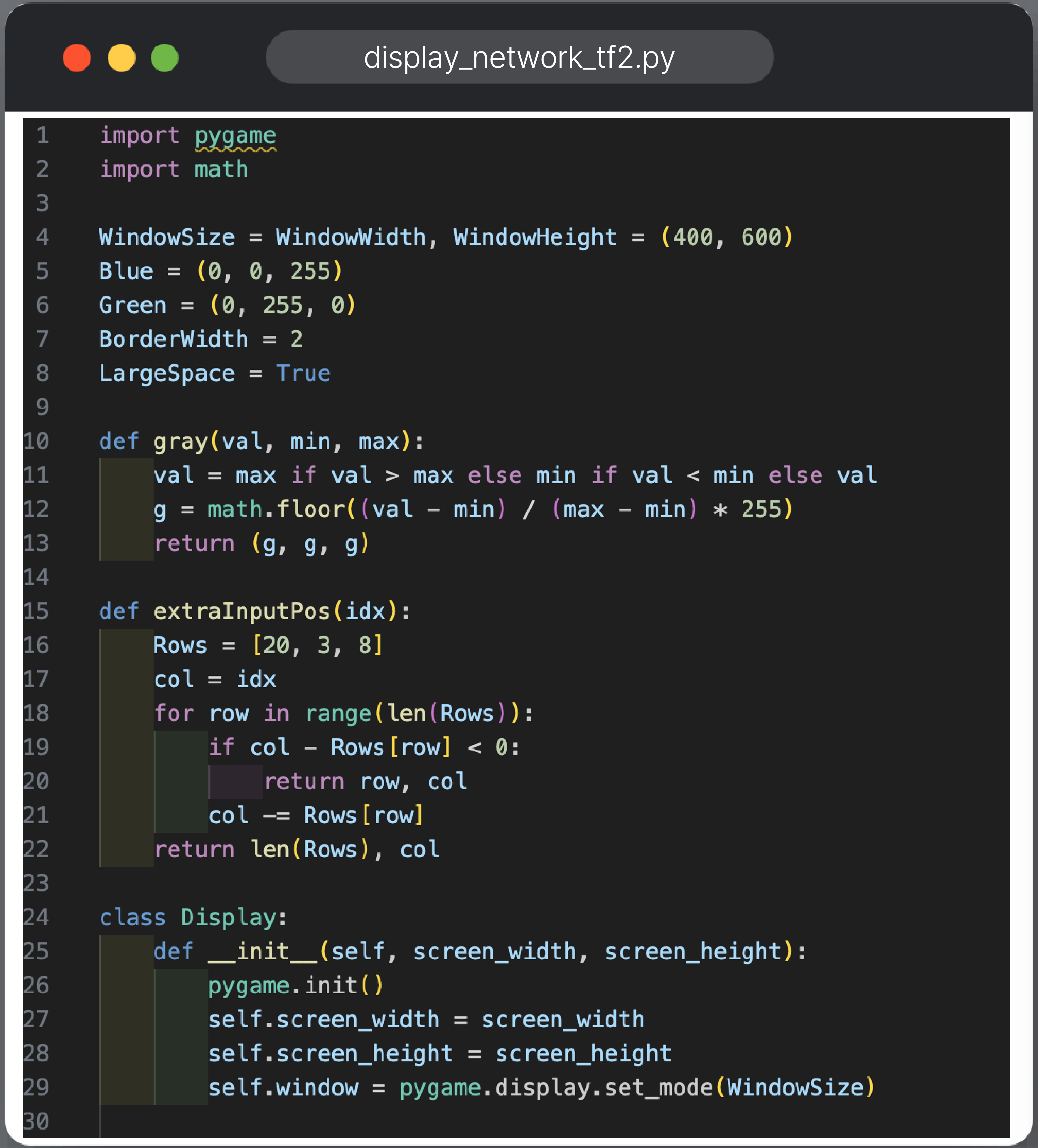
```python
import socket
import numpy as np
import tensorflow as tf
from display_network_tf2 import Display

# 모델 로딩
MODEL_PATH = "models/mario_rnn_tf2_last.h5"
model = tf.keras.models.load_model(MODEL_PATH)

# Lua 기준 헤더 정보 (첫 줄: 32, 7, 4, 3)
header = ["32", "7", "4", "3"]
input_width = int(header[0])
input_height = int(header[1])
extra_inputs = int(header[2])
output_size = int(header[3])
input_dim = model.input_shape[-1]  # 248 expected

# Display 초기화
display = Display(input_width, input_height)

# 서버 설정
HOST = socket.gethostname()
PORT = 2222

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((HOST, PORT))
server.listen(1)
print(f"✅ 서버 실행 중: {HOST}:{PORT}")

try:
    while True:
        print("🎮 클라이언트 접속 대기 중...")
        clientsocket, address = server.accept()
        print(f"🔗 클라이언트 접속됨: {address}")

```



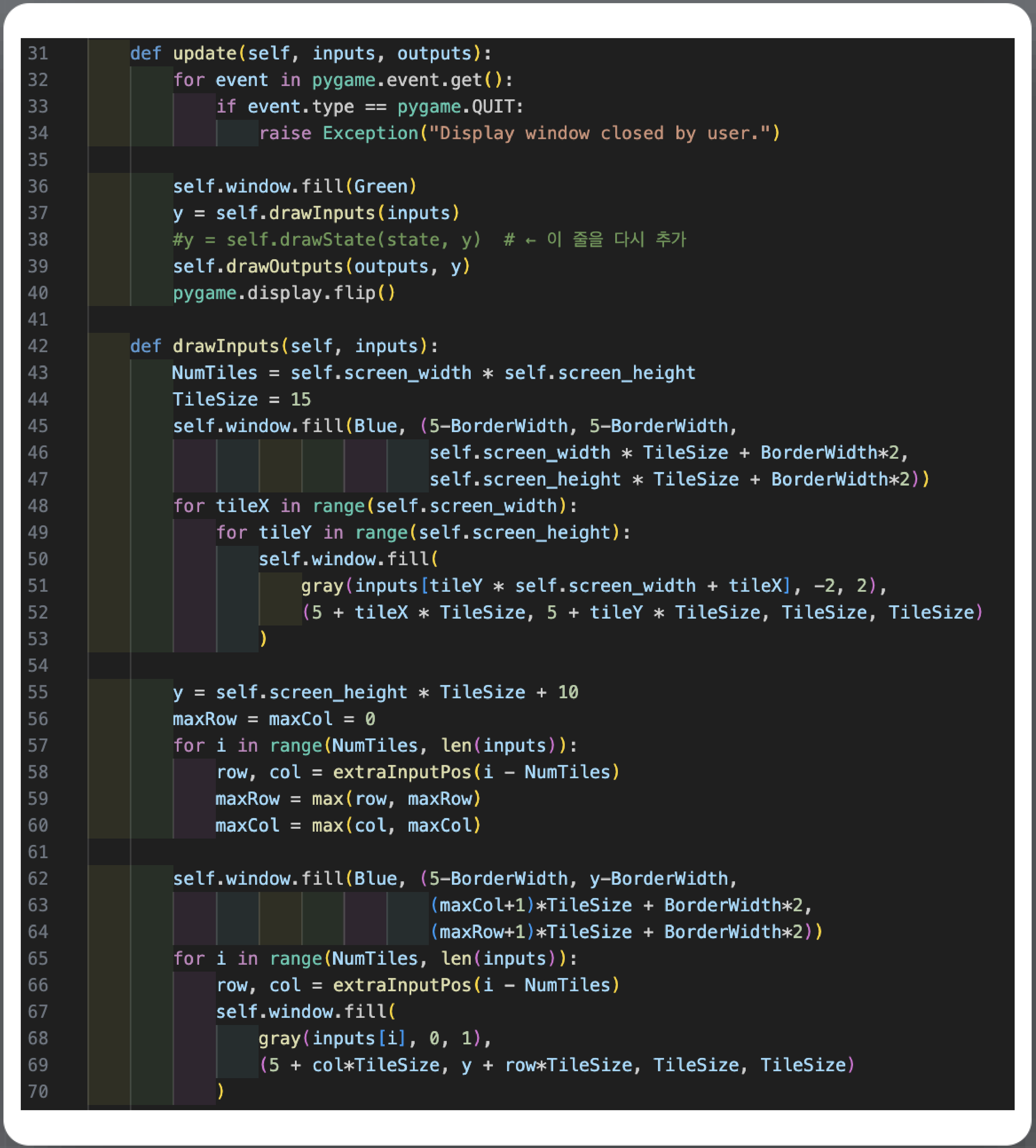
```python

        try:
            # Lua 초기 헤더 응답 (필수)
            clientsocket.send((str(len(header)) + "\n").encode())
            for h in header:
                clientsocket.send((h + "\n").encode())

            while True:
                screen = ""
                while not screen.endswith("\n"):
                    chunk = clientsocket.recv(2048).decode('ascii')
                    if not chunk:
                        raise ConnectionError("클라이언트 연결 종료됨")
                    screen += chunk

                screen = screen.strip()
                values = screen.split(" ")

                if len(values) < input_dim:
                    print(f"❌ 입력 벡터 크기 부족: {len(values)} vs 기대값 {input_dim}")
                    clientsocket.send(b"close\n")
                    break
                elif len(values) > input_dim:
                    print(f"📥 입력 벡터 길이: {len(values)} (기대: {input_dim})")
                    print("⚠ 초과 입력 감지 → 잘라서 처리.")
                    values = values[:input_dim]

                # 예측
                x_input = np.array(values, dtype=float).reshape(1, 1, input_dim)
                prediction = model.predict(x_input, verbose=0)[0]

                print("🔮 예측 결과:", prediction)

                # 시각화
                try:
                    display.update(list(map(float, values)), prediction)
                except Exception as e:
                    print("⚠ 시각화 예외:", e)
                    break

```



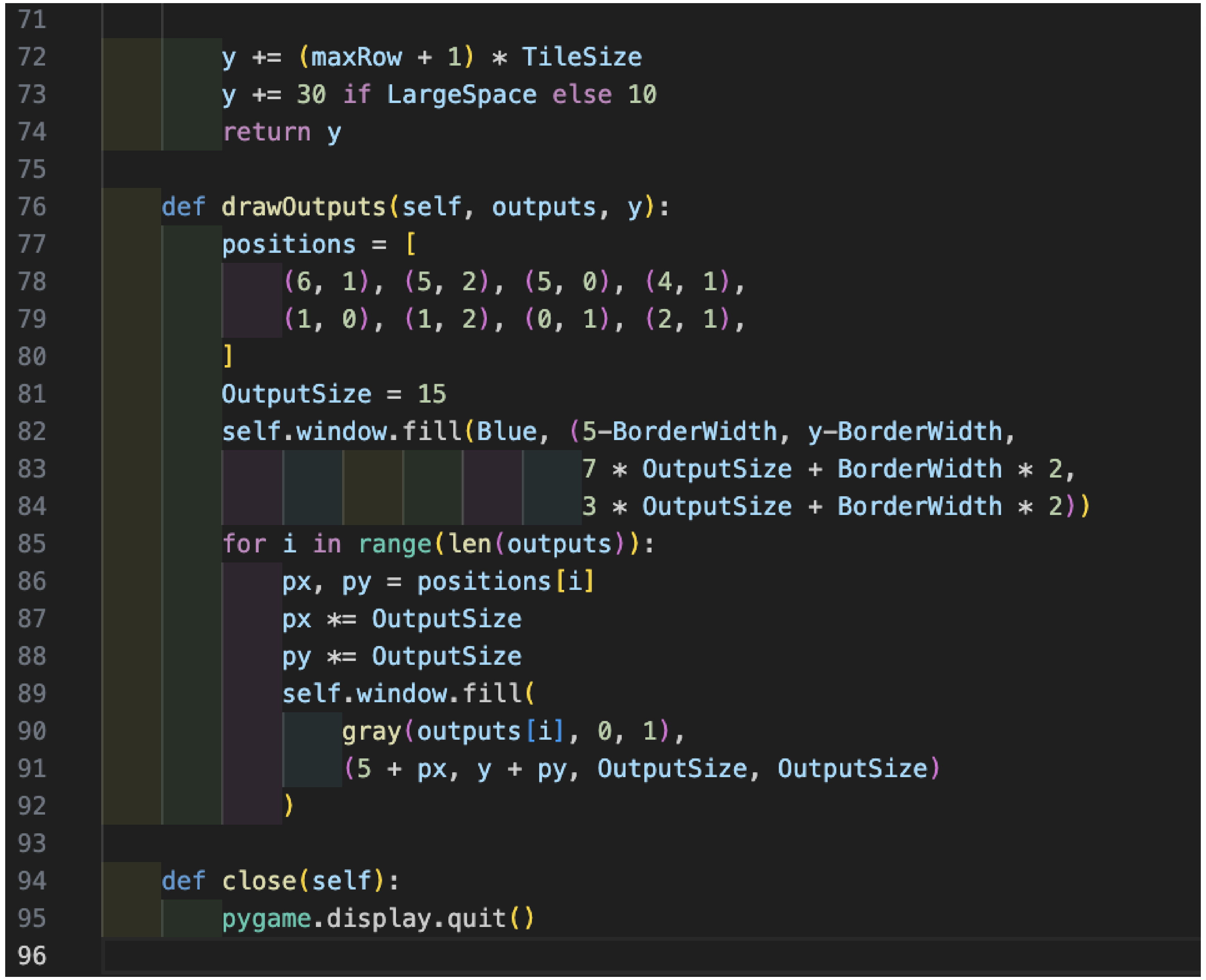
```python

                # 버튼 결과 전송
                buttons = ["1" if np.random.rand() < p else "0" for p in prediction]
                result = " ".join(buttons) + "\n"
                clientsocket.send(result.encode())

        except Exception as e:
            print("⚠ 예외 발생:", e)
            try:
                clientsocket.send(b"close\n")
            except:
                pass
            clientsocket.close()

finally:
    server.close()

```



```python
import pygame
import math

WindowSize = WindowWidth, WindowHeight = (400, 600)
Blue = (0, 0, 255)
Green = (0, 255, 0)
BorderWidth = 2
LargeSpace = True

def gray(val, min, max):
    val = max if val > max else min if val < min else val
    g = math.floor((val - min) / (max - min) * 255)
    return (g, g, g)

def extraInputPos(idx):
    Rows = [20, 3, 8]
    col = idx
    for row in range(len(Rows)):
        if col - Rows[row] < 0:
            return row, col
        col -= Rows[row]
    return len(Rows), col

class Display:
    def __init__(self, screen_width, screen_height):
        pygame.init()
        self.screen_width = screen_width
        self.screen_height = screen_height
        self.window = pygame.display.set_mode(WindowSize)

```

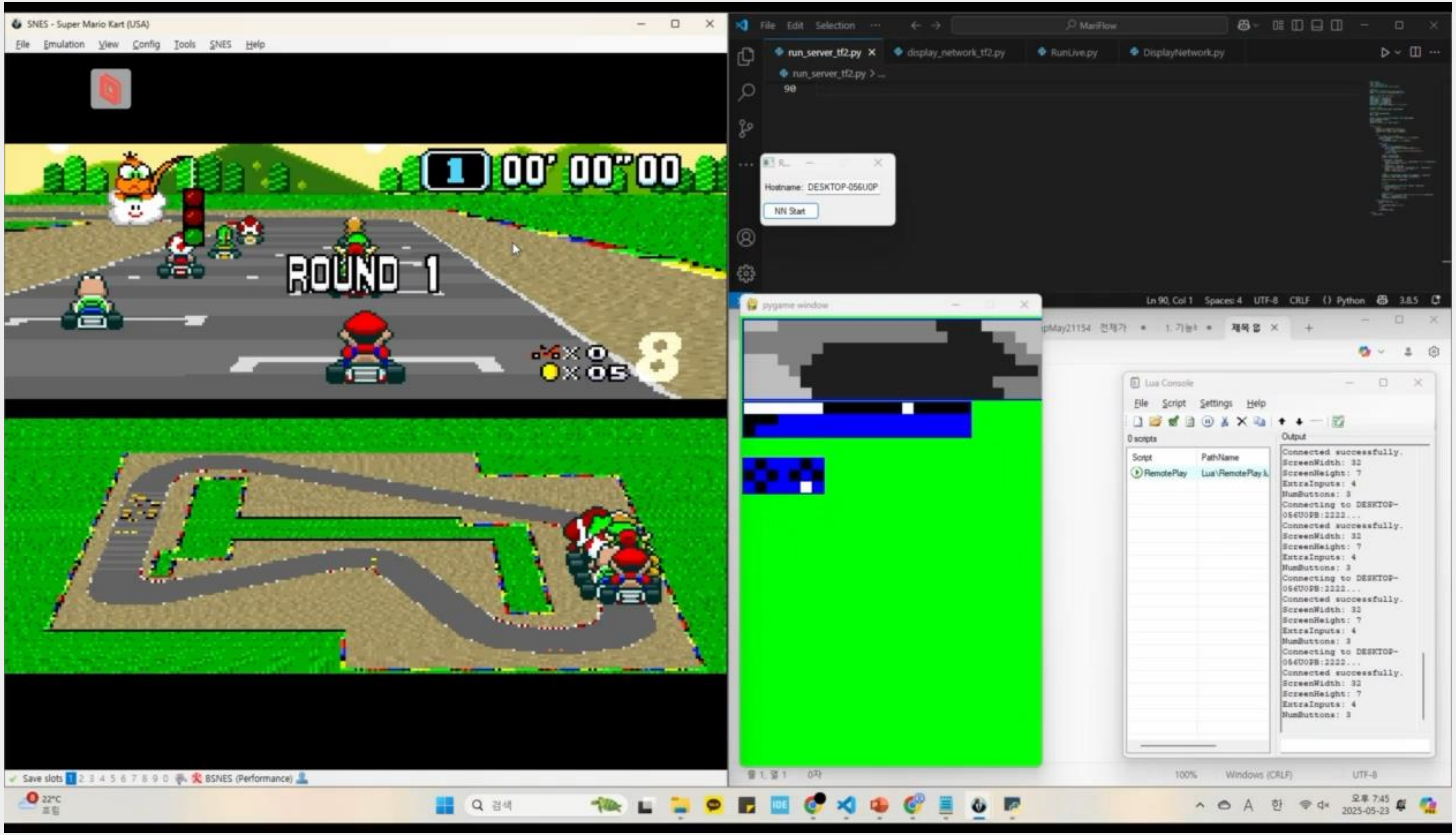


```python
    def update(self, inputs, outputs):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                raise Exception("Display window closed by user.")

        self.window.fill(Green)
        y = self.drawInputs(inputs)
        #y = self.drawState(state, y)  # ← 이 줄을 다시 추가
        self.drawOutputs(outputs, y)
        pygame.display.flip()

    def drawInputs(self, inputs):
        NumTiles = self.screen_width * self.screen_height
        TileSize = 15
        self.window.fill(Blue, (5-BorderWidth, 5-BorderWidth,
                                self.screen_width * TileSize + BorderWidth*2,
                                self.screen_height * TileSize + BorderWidth*2))
        for tileX in range(self.screen_width):
            for tileY in range(self.screen_height):
                self.window.fill(
                    gray(inputs[tileY * self.screen_width + tileX], -2, 2),
                    (5 + tileX * TileSize, 5 + tileY * TileSize, TileSize, TileSize)
                )

        y = self.screen_height * TileSize + 10
        maxRow = maxCol = 0
        for i in range(NumTiles, len(inputs)):
            row, col = extraInputPos(i - NumTiles)
            maxRow = max(row, maxRow)
            maxCol = max(col, maxCol)

        self.window.fill(Blue, (5-BorderWidth, y-BorderWidth,
                                (maxCol+1)*TileSize + BorderWidth*2,
                                (maxRow+1)*TileSize + BorderWidth*2))
        for i in range(NumTiles, len(inputs)):
            row, col = extraInputPos(i - NumTiles)
            self.window.fill(
                gray(inputs[i], 0, 1),
                (5 + col*TileSize, y + row*TileSize, TileSize, TileSize)
            )
```



```python

        y += (maxRow + 1) * TileSize
        y += 30 if LargeSpace else 10
        return y

    def drawOutputs(self, outputs, y):
        positions = [
            (6, 1), (5, 2), (5, 0), (4, 1),
            (1, 0), (1, 2), (0, 1), (2, 1),
        ]
        OutputSize = 15
        self.window.fill(Blue, (5-BorderWidth, y-BorderWidth,
                                7 * OutputSize + BorderWidth * 2,
                                3 * OutputSize + BorderWidth * 2))
        for i in range(len(outputs)):
            px, py = positions[i]
            px *= OutputSize
            py *= OutputSize
            self.window.fill(
                gray(outputs[i], 0, 1),
                (5 + px, y + py, OutputSize, OutputSize)
            )

    def close(self):
        pygame.display.quit()

```

demonstration_video



Hostname: DESKTOP-056U0P

NN Start

```
Connected successfully.
ScreenWidth: 32
ScreenHeight: 7
ExtraInputs: 4
NumButtons: 3
Connecting to DESKTOP-
056U0PB:2222...
Connected successfully.
ScreenWidth: 32
ScreenHeight: 7
ExtraInputs: 4
NumButtons: 3
Connecting to DESKTOP-
056U0PB:2222...
Connected successfully.
ScreenWidth: 32
ScreenHeight: 7
ExtraInputs: 4
NumButtons: 3
Connecting to DESKTOP-
056U0PB:2222...
Connected successfully.
ScreenWidth: 32
ScreenHeight: 7
ExtraInputs: 4
NumButtons: 3
```