Individual Portfolio Assignment 1

Yasmin Abubaker Hassan

s348816

DATA2410

Introduction

My project includes: server.py, client.py and bot.py. I made a function that takes a string as input and returns the bots response. I made a chat server with four bots. The bots don't take initiative but only respond to the host suggestion.

Bot explanation

The bots take suggestions from the host and then responds to it. The suggestion is a single verb—a suggestion for something. I made four different chatbots: Superman, Spiderman, Batman and Joker. In this application to active them you have to type in their name in the command line. I made the bots as functions so that they take a string or two as input and return one or two strings. I made two arrays one with suggestions and another one with actions

```python
import random


def batman(a, b=None):
    alternatives = ["relax", "play cards", "fight joker.", 'try to catch joker']
    b = random.choice(alternatives)
    res = f"Nahh, {a}ing is an bad option. Or we could {b}."
    return res


def superman(a, b=None):
    if b is None:
        return "im not into {}. can we fly instead... ops forgot batman cant haha".format(a + "ing")
    return "Sure, both {} and {} seems ok to me".format(a, b + "ing")


def joker(a, b=None):
    alternatives = ["play chess", "fight Batman..", 'some killing']
    b = random.choice(alternatives)
    res = f"mmm...{a}ing is an good option. Or maybe we could {b};)."
    return res


def spiderman(a, b=None):
    action = a + "ing"
    bad_things = ["fighting", "sleeping", "killing"]
    good_things = ["singing", "playing", "working", "eating", "sleeping", "playing"]
```

```python
def spiderman(a, b=None):
    action = a + "ing"
    bad_things = ["fighting", "sleeping", "killing"]
    good_things = ["singing", "playing", "working", "eating", "sleeping", "playing"]

    if action in bad_things:
        return "yeaahhh!im down for {}, but can we invite Tony Stark?".format(action)
    elif action in good_things:
        return "What the hell? {} seems lame. Not doing that.".format(action)
    return "that seems lame, what about joining the avengers!!?"


suggestions = ["lets go outside and play", "Let's take a walk to Iron man",
               "Let's sleep a little bit!", "I feel like fighting right now", " We can do some eating"]

actions = ["work", "play", "eat", "cry", "sleep",  # List of known actions
           "fight", "sing", "hug", "bicker", "sleep",
           "complain", "walk"]
```

Client explanation

Each client can connect from its own terminal and run a single bot each. Several terminal windows can run in parallel, connecting different bots to the same server. Starts of taking three command line arguments. Client.py takes an input which is encoded and sent to the server. The server than receives this message, decode it and check it.

```python
import socket
import sys
import os
import threading
from Bots import *  # Import all bot functions from Bot.py.

ip = sys.argv[1]
port = int(sys.argv[2])
bot = sys.argv[3]
# All the bots that can you select from:
bots = {"batman": batman,
        "joker": joker,
        "superman": superman, "spiderman": spiderman}

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect((ip, port))
client_socket.setblocking(False)
# Sends the bot'name you selected to the server
client_socket.send((bot.capitalize()).encode())
print(f"You have selected Bot: {bot.capitalize()}, and are now connected to server at {ip}:{port}...")
print("Waiting for suggestion from server...")


# Function that returns the actions
def receive_message(server=None,
                    client=None):
    msg = client_socket.recv(1024)
    if not len(msg):  # if the message is an empty string, assume server is disconnecting
```

```
# Function that returns the actions
def receive_message(server=None,
                    client=None):
    msg = client_socket.recv(1024)
    if not len(msg):  # if the message is an empty string, assume server is disconnecting
        print('Connection closed by server')
        os._exit(0)
    # This if statment checks the host's message
    if msg.decode().find("Host") != -1:
        print(f"\n--------------CHATROOM-------------")
        print(f"{msg.decode()}")
        server = [action for action in actions if msg.decode().find(action) != -1][0]
        return (server, client)
    else:
        client = [action for action in actions if msg.decode().find(action) != -1][
            0]  # Loop over all actions and check if that action can be found in the message
        return (server, client)


# Send a message to the server
def send_msg(server, client):
    msg = (f"{bot.capitalize()}: " + bots[bot](server, client)).encode()
    print(f"{msg.decode()}\n")
    client_socket.send(msg)
```

Server explanation

The server takes in a command line parameter for the port and not the host.

```
try:
    port = int(sys.argv[2])
    host = ""
    if port == '--help' or port == '-h':
        print("")
        sys.exit()
except (IndexError, ValueError):
    print('--------------------------------')
    print("To start a server you have to connect to specified ip and port:\n")
    sys.exit()
```

I instantiate a socket by specifying (address family, socket type). And I made some list to store all the clients.

```
# Bind the server to ip and port and listen for clients
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind((host, port))
# listen is used below to mark our socket instance as a passive socket
# - A socket that will be used to accept inncoming requess with the accept() method
server_socket.listen(4)
FORMAT = 'utf-8'
socket_list = [server_socket]
clients = {}  # Lists for clients
names = []
msg_count = 0
expected_msg_count = 0
```

```
def receive():  # A function that adds clients to the socket list
    # our server is now listening for connections.
    # When a client attempts to connect, we use the accept()
    # mehod to establish a connection
    conn, address = server_socket.accept()
    socket_list.append(conn)
    name = conn.recv(2042).decode()
    names.append(name)  # Add client to names array to wait until next round of conversations
    print(str.upper('\n'f'{name} has joined the chatroom\n'))
    # sends a message from our server socket to client socket
    conn.send('Connected to the server'.encode(FORMAT))
```

How to run the program and the outcome:

To connect to the server:

Py server.py localhost <port number>

To connect as a client:

Py client.py <Ip address> <port number>

```
C:\Users\Yasmin\PycharmProjects\eks1>py server.py lochalhost 5000
listening for connections on :5000...
```

```
C:\Users\Yasmin\PycharmProjects\eks1>py client.py 192.168.0.162 5000 joker
You have selected Bot: Joker, and are now connected to server at 192.168.0.162:5000...
Waiting for suggestion from server...
```

```
C:\Users\Yasmin\PycharmProjects\eks1>py client.py 192.168.0.162 5000 batman
You have selected Bot: Batman, and are now connected to server at 192.168.0.162:5000...
Waiting for suggestion from server...
```

```
C:\Users\Yasmin\PycharmProjects\eks1>py client.py 192.168.0.162 5000 superman
You have selected Bot: Superman, and are now connected to server at 192.168.0.162:5000...
Waiting for suggestion from server...
```

```
C:\Users\Yasmin\PycharmProjects\eks1>py client.py 192.168.0.162 5000 spiderman
You have selected Bot: Spiderman, and are now connected to server at 192.168.0.162:5000...
Waiting for suggestion from server...
```

Joker was the first bot who connected to server therefore he would show up in the first round.

Then the other bots will join in the second round

```
Listening for connections on :5000...

JOKER HAS JOINED THE CHATROOM

------------------------------

You have thesse options to choose from:

1: lets go outside and play
2: Let's take a walk to Iron man
3: Let's sleep a little bit!,
4: I feel like fighting right now ,
5: We can do some eating

Select a suggestion by index (1-5):
```

```
BATMAN HAS JOINED THE CHATROOM


SPIDERMAN HAS JOINED THE CHATROOM


SUPERMAN HAS JOINED THE CHATROOM
```

This is how it looks like when the host selects a number.

```
Select a suggestion by index (1-5): 1

--------------CHATROOM------------
Batman: Nahh, playing is an bad option. Or we could try to catch joker.
Spiderman: What the hell? playing seems lame. Not doing that.
Joker: mmm...playing is an good option. Or maybe we could some killing;).
Superman: im not into playing. can we fly instead... ops forgot batman cant haha
------------------------------
```

```
Select a suggestion by index (1-5): 2

---------------CHATROOM-------------
Superman: im not into walking. can we fly instead... ops forgot batman cant haha
Batman: Nahh, walking is an bad option. Or we could relax.
Spiderman: that seems lame, what about joining the avengers!!?
Joker: mmm...walking is an good option. Or maybe we could play chess;).
------------------------------
```

```
elect a suggestion by index (1-5): 3

---------------CHATROOM-------------
oker: Nahh, sleeping is an good option. Or maybe we could some killing;).
atman: Nahh, sleeping is an bad option. Or we could relax.
uperman: im not into sleeping. can we fly instead... ops forgot batman cant?
piderman: yeaahhh!im down for sleeping, but can we invite Tony Stark?
------------------------------
```

On the client side the bots are alone with host and sees its own response but the server side can see alle the bots' answers

```


---------------CHATROOM-------------
Host: lets go outside and play
Spiderman: What the hell? playing seems lame. Not doing that.


---------------CHATROOM-------------
Host: I feel like fighting right now
Spiderman: yeaahhh!im down for fighting, but can we invite Tony Stark?
```

If the host type in a number outside the range of 1-6, then the connection will shutdown and the bots will disconnect

```
Connection closed by server
```