



Socket Bots

Individual Portfolio Assignment 1

Siem Fitwi Fissuh
s351913
DATA2410

Table of Contents

Table of Contents	2
Introduction	3
Bot explanation	3
Client explanation	3
Server explanation	3
Code readability	4
Run the code	4
Communication.py explanation	5
Bots.py explanation	6
Server.py explanation	7
Client.py explanation	9
Conclusion	10
References	11

Introduction

This project is a portfolio assignment in Data network and cloud computing topic at OsloMet university. The goal in this project is to create a communication channel with bots. The programming language used in this project is **python**. Based on what is mentioned in the assignment, I have solved the task by having the clients signed as a bots. Before “opening” the chat room, the server waits for four clients to be connected on the same server. When a client is connected it get identified with random bot. Then, based on the bot’s interest they will respond to suggested action to the chatroom.

GitHub repository: https://github.com/siemff/data2410_potrfolio.git

Bot explanation

We got four insane bots in this project. Madelene, the sporty type. Felix, the nurd type. Marija, the lazy type, and Rateb, the likable type. They are the bots that will identify the clients and respond to the suggested message.

Client explanation

Clients connect to the same server. IPv4 address: 127.10.10.10. Port: 8080. When all four clients are connected the chat room opens. They will receive suggestion from the host. Once all clients/bots have responded to the suggested action the bots leave the chat room and the connection ends.

Server explanation

The server acts as chat room. Accept four connections. Maintain a list of connected clients. Initiate a round of dialogue by suggesting an action. Send suggestion and response to each of the connected clients and disconnect once all clients have responded the suggestion. All those actions can be seen at the server.

Code readability

The code contains four python files. Communication.py, bots.py, server.py and client.py. To solve the task best possible I have used the following important interfaces:

- ✚ **Socket**: to enable two networks/sockets to send and receive data, bi-directionally, at any given moment. It works by connecting two nodes/sockets together and allowing them to communicate in real time.
- ✚ **Serialization/deserializing**: for making/receiving requests/responses - effectively transporting “messages”. Converting data which include arrays into a string so it can be stored as well as transmitted easily. The pickle module is used here for implementing a binary protocol for serializing and de-serializing a python object structure. I have used all four methods the pickle interface provides.

```
Dump(object, file): serialize and store a python object into a file
Dumps(object) -> string: serialize a python object into a byte stream/string
Load(file) -> object: deserializes a python object to its original form.
Loads(string) -> object: deserialize a byte stream into a python object.
```

- ✚ **Multithreading**: to allow threads to communicate and share resources such as message/data, to the same server. So that server and all clients receive all communications between.

Run the code

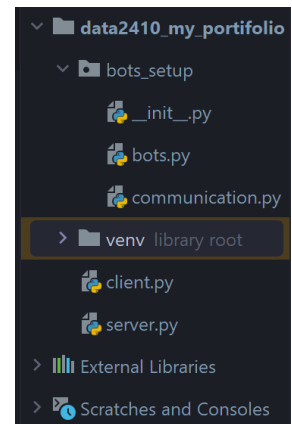
Run from terminal:

1. Open terminal (CMD) on the project file.
2. Run server: `py -3.9 server.py`
3. Run 4 separate clients: `py -3.9 client.py`

Run on the python file.

```
#!/usr/bin/env python
```

At top of the server and client file click the run icon.



Communication.py explanation

```
data2410_my_portfolio > bots_setup > communication.py >
communication.py x bots.py x server.py x client.py x
1 """
2 this Message class contains attributs for communication
3 between server and clients
4 """
5 class Message:
6     def __init__(self, sender: None, response=None, action=None, suggestion_type=None):
7         self.sender = sender # username
8         self.response = response # communicatoin between
9         self.action = action # action from bots
10        self.suggestion_type = suggestion_type # suggestion from bots
11
12    def __str__(self):
13        return self.sender, self.response, self.action, self.suggestion_type
14
15    """
16    Method that takes in name, address and connection
17    so that the client can be assigned into the server.
18    """
19
20    class Client:
21        def __init__(self, name=None, address=None, connection=None):
22            self.name = name
23            self.address = address
24            self.connection = connection
25
26        def __str__(self):
27            return self.name, self.address, self.connection
```

Bots.py explanation

This file contains all response actions from host and bots, as well as actions and suggestion which is important part of the task.

The suggestion message takes in action from the user/host and add a subject to it.

By using `random.choice()` If for example the “host” is interested in sport (action) and would like to play (suggestion) football (subject), by calling Message class it returns sender name and response.

```

data2410_my_portfolio > bots_setup > bots.py >
communication.py x bots.py x server.py x client.py x
1 import random
2 from bots_setup.communication import Message
3
4 actions = {
5     "chill": ["drink", "party", "travel to"],
6     "work": ["code", "write", "read"],
7     "sport": ["play", "run", "drive"],
8     "bad thing": ["steal", "fight"],
9 }
10
11 #This method takes an action from above and adds a subject to it
12 def suggestions(action: str):
13     subjects = {
14         "drink": ['beer', 'coffee', 'wine'],
15         "travel to": ['Paris', 'New York', 'Oslo'],
16         "code": ['chatbot', 'website'],
17         "write": ['book', 'rapport'],
18         "read": ['poem', 'the paper', 'python book'],
19         "play": ['football', 'basketball', 'tennis'],
20         "drive": ['sport car', 'motorcycle'],
21         "fight": ['Floyd Mayweather', 'Conor McGregor', 'our professor'],
22         "steal": ["book", "car", "computer from school"],
23     }
24     subject_in_action = action
25     if action in subjects:
26         return f"{subject_in_action} {random.choice(subjects[action])}"
27     else:
28         return action
  
```

```

30 def my_bot():
31     name = 'Host' # username
32     suggestion_types = ["sport", "chill", "work", "bad thing"] # interests
33     suggestion = random.choice(suggestion_types) # random choice of interests
34     action = random.choice(actions[suggestion]) # random choice action of chosen suggestion
35     print("Host suggested " + suggestion) # print Host suggestion
36     subject = suggestions(action) # action choices
37     responses = [ # message from host
38         f"Do you guys want to {subject}?",
39         f"Let's {subject} guys!!",
40     ]
41     raandom_response = random.choice(responses) #Random choice of messages
42     return Message(sender=name, response=raandom_response, # Return username, message and
43                   action=action, suggestion_type=suggestion) #chosen action and suggestion
44
45 def madelen(response: Message):
46     name = 'Madelen' # username
47     suggestion = "sport" # interests
48     change_suggestion = random.choice(actions[suggestion]) # random choice action of chosen suggestion
49     responses = [ # response from Madelen
50         f"I dont like it, but I will join",
51         f"I have some other stuff to do.",
52         f"That one isn't fair, I would like to {change_suggestion}",
53     ]
54     random_response = random.choice(responses) #Random choice of messages
55     return Message(sender=name, response=random_response) # Return username and response
56
  
```

We can see here the method that returns suggestion of the host and the response from the bots by calling the class Message.

Server.py explanation

In `client_action` method the method takes in connection from the client and checks if the client is still in the chatroom. If client leaves ("bye"), the chatroom it will be informed in the server and the connection between ends. The connection ends automatically when all clients have responded.

```
1  def start_threads():                                #the client handler threads will begin after connection loop
2      for count in threads:
3          threads[count].start()                      # Make the new thread ready.
4          time.sleep(2.5)                             # puts a thread to sleep for 2.0 seconds
5
6  def connection_msg(msg, clients):
7      pickle_msg = pickle.dumps(msg)
8      for client in clients:
9          clients[client].connection.send(pickle_msg)
10
11 #This method checks if the client is still connected
12 #as long as it's connected it prints out responses of the client in server.
13 def client_action(connection):
14     while True:
15         msg = connection.recv(1024)                 #Receive message from server
16         unpickle_msg = pickle.loads(msg)             #deserialize a byte stream into a python object
17         if unpickle_msg.response == "Bye":
18             client = clients.pop(unpickle_msg.sender)
19             print(client.name, "left the chatroom.")
20             time.sleep(5)                            #after 10s the server closes the chatroom
21             client.connection.close()
22             threads.pop(unpickle_msg.sender)
23             break
24         else:
25             connection_msg(unpickle_msg, clients)
26             print(f"{unpickle_msg.sender}: {unpickle_msg.response}")
```

In `get_msg()` function the server connects with client first by accepting the connection. Then I used `pickle` and `unpickle` as I mentioned over for transporting "messages" effectively and to implement binary protocol for pickling/Serializing and unpickling/deserializing an object. The method introduces clients to the chatroom check if all clients are connected and prints all messages at the server once all 4 clients are connected.

```

data2410_my_portfolio > server.py >
communication.py × bots.py × server.py × client.py ×

50 def get_msg():
51     while True:
52         c, addr = server_socket.accept() #wait till a client accepts connection
53         connected = Message(sender="Host", response="USERNAME")
54         c.send(pickle.dumps(connected)) #send connection message to client so we can receive bot name
55         username = pickle.loads(c.recv(1024)).sender #receive message string from server
56         if username not in bot_username: #If connection is done successfully
57             bot_username.append(username) #
58             person = Client(name=username, address=addr, connection=c) # creating object of Person
59             # sending the connection confirmation back to the client
60             clients[username] = person
61             # a print out message just for the server
62             info = f"New connection from {person.address} \"
63                 f\"{person.name} has joined the chatroom\"
64             print(info)
65             connection_ok = Message(sender="Host", response=f"Introducing our awesome bot {person.name}!.\"
66                                     f\"\\nWaiting for all clients to join the chat.\")
67             pickle_msg = pickle.dumps(connection_ok)
68             person.connection.send(pickle_msg) # sending the connection confirmation back to the client
69             thread = threading.Thread(target=client_action, args=(person.connection,)) # creating new thread for handling the client
70             threads[username] = thread # adding the thread to a dictionary of threads
71         else:
72             print(f"Bot {username}, is taken please try again with another bot name.")
73             c.send(pickle.dumps(connection_bad))
74             bot_username.pop(bot_username.index(username))
75             c.close()
76         # if all the bots are connected, then the chat begins with a suggestion message from the server
77         if len(clients) == 4:
78             time.sleep(2)
79             suggested_msg = my_bot() # there is a simple bot function that decides the first message
80             print(f"{suggested_msg.sender}: {suggested_msg.response}")
81             connection_msg(suggested_msg, clients) # sends the message to all the bots
82             break
83         else:
84             print(f"Waiting for {4 - len(clients)} client(s) to join the chat.") # not sure yet
85             continue
86     start_threads() # starting the threads

```


Client.py explanation

Msg_fromServer takes connection socket. Receive message from the server and save messages from server/host in a list get_server_message[] and get_bot_responses[]. get_server_message[] contains suggestion message from host which will be printed out as a suggestion from host in clients side. get_bot_responses[] contains bots response which will be printed out as well. The connection ends after all messages from the list has been sent out.

```

67 def msg_fromServer(connection: socket):
68     while True:
69         try:
70             msg_received = connection.recv(1024) # receive as long as message strings are not empty
71             if msg_received:
72                 msg = Message = pickle.loads(msg_received) # deserialize byte stream to python object
73             else:
74                 continue
75             if msg.sender == 'Host':
76                 if msg.response == 'USERNAME':
77                     random_usr = Message(sender=USERNAME) # serialize Usernames into byte stream
78                     connection.send(pickle.dumps(random_usr))
79                 else:
80                     get_server_messages.append(msg) # appends message from host to be added in the list
81                     print(f"{msg.sender}: {msg.response}") # Message from host, suggesting...
82             else:
83                 get_bot_responses.append(msg) # appends replays from bots to be added in list
84                 time.sleep(2) # suspends execution of the thread for 2s
85                 print(f"{msg.sender}: {msg.response}") # Message from bots
86
87             if len(get_bot_responses) == 4: # When all clients have responded the connection ends
88                 disconnect = Message(sender=USERNAME, response="Bye")
89                 connection.send(pickle.dumps(disconnect))
90                 break
91         except OSError as e:
92             print(e)
93             break
94     connection.close()

```

```

6 # function that imports bots responses by using choose_bot function
7 # which contains the attributes: bot name and response
8 def w_msg(connection: socket):
9     while True:
10         if len(get_server_messages) == 1:
11             msg = choose_bot(USERNAME, get_server_messages[0])
12             pickle_msg = pickle.dumps(msg)
13             connection.send(pickle_msg)
14             break
15         else:
16             continue

```

Conclusion

As the result shows below in the picture. When we run the code, the server waits for four connections. When a client is connected, it prints out confirmation of the connection address and port number as well as the username of the client. On the client's side the host welcome the client and inform to wait for all clients to be connected. When all clients are connected the host suggest action (Do you guys want to travel to Oslo), and all the clients see and respond in the chat. After one round of action and response the clients leave the chatroom, and server ends the connection between.

```

C:\Windows\System32\cmd.exe
C:\Users\siemf\PycharmProjects\data2410_my_portfolio>py -3.9 client.py
Host: Introducing our awesome bot Marija!.
Waiting for all clients to join the chat.
Host: Do you guys want to travel to Oslo?
Marija: I can't join!
Felix: WOW I can't wait for it!
Madelen: I have some other stuff to do.
Rateb: That isn't my thing, you have a great time!
C:\Users\siemf\PycharmProjects\data2410_my_portfolio>

C:\Windows\System32\cmd.exe
C:\Users\siemf\PycharmProjects\data2410_my_portfolio>py -3.9 server.py
SERVER LISTENING on ('127.0.0.1', 8080) ...
New connection from ('127.0.0.1', 61308) Marija has joined the chatroom
Waiting for 3 client(s) to join the chat.
New connection from ('127.0.0.1', 61314) Felix has joined the chatroom
Waiting for 2 client(s) to join the chat.
Bot Felix, is taken please try again with another bot name.
Waiting for 2 client(s) to join the chat.
New connection from ('127.0.0.1', 51423) Madelen has joined the chatroom
Waiting for 1 client(s) to join the chat.
Bot Madelen, is taken please try again with another bot name.
Waiting for 1 client(s) to join the chat.
Bot Marija, is taken please try again with another bot name.
Waiting for 1 client(s) to join the chat.
New connection from ('127.0.0.1', 51426) Rateb has joined the chatroom
Host suggested chill
Host: Do you guys want to travel to Oslo?
Marija: I can't join!
Felix: WOW I can't wait for it!
Madelen: I have some other stuff to do.
Rateb: That isn't my thing, you have a great time!
Rateb left the chatroom.
Felix left the chatroom.
Madelen left the chatroom.
Marija left the chatroom.
C:\Users\siemf\PycharmProjects\data2410_my_portfolio>

C:\Windows\System32\cmd.exe
C:\Users\siemf\PycharmProjects\data2410_my_portfolio>py -3.9 client.py
Host: Introducing our awesome bot Rateb!.
Waiting for all clients to join the chat.
Host: Do you guys want to travel to Oslo?
Marija: I can't join!
Felix: WOW I can't wait for it!
Madelen: I have some other stuff to do.
Rateb: That isn't my thing, you have a great time!
C:\Users\siemf\PycharmProjects\data2410_my_portfolio>

C:\Windows\System32\cmd.exe
C:\Users\siemf\PycharmProjects\data2410_my_portfolio>py -3.9 client.py
Host: Introducing our awesome bot Felix!.
Waiting for all clients to join the chat.
Host: Do you guys want to travel to Oslo?
Marija: I can't join!
Felix: WOW I can't wait for it!
Madelen: I have some other stuff to do.
Rateb: That isn't my thing, you have a great time!
C:\Users\siemf\PycharmProjects\data2410_my_portfolio>

```

References

Geeksforgeeks.org, (01 Jun, 2021), Python object serialization.

<https://www.geeksforgeeks.org/pickle-python-object-serialization/?ref=gcse>

Geeksforgeeks.org, (13 Nov, 2018), Understanding python pickling with example

<https://www.geeksforgeeks.org/understanding-python-pickling-example/?ref=gcse>

Gigi Sayfan, (25 Aug, 2016), Serialization and Deserialization of Python Objects.

<https://code.tutsplus.com/tutorials/serialization-and-deserialization-of-python-objects-part-1--cms-26183>

knowledgehut.com, (04 Juni, 2020), Python Tutorial (various codes)

<https://www.knowledgehut.com/tutorials/python-tutorial>

sentdex, Youtube, (02 Apr, 2019), Socket Chatroom server python

https://www.youtube.com/watch?v=CV7_stUWvBQ&list=RDCMUcfzICWGWYyIQ0aLC5w48gBQ&start_radio=1&t=1299s

sentdex, Youtube, (22 mai 2015), Python Pickle Module for saving objects (serialization)

<https://www.youtube.com/watch?v=2Tw39kZlbhs>

NeuralNine, Youtbe (), Simple TCP Chat Room in Python

<https://www.youtube.com/watch?v=3UOyky9sEQY&t=163s>

+ notes from the lecture, labs, and published files on canvas.