מטלת פייטון קורס רשתות מחשבים ואינטרנט 1

מגישות: מאיה נאור 315176362, נטשה עדינה הסן 336165139

Operating System: Windows 10

Code language: python

Version: 3

רקע תיאורטי:

שכבת הTRANSPORT ושכבת ושכבת הAPPLICATION (שכבות 4 ו-5) של המודל (OPEN SYSTEMS INTERCONNECTION-OSI) אחראית על התקשורת בין התקנים ברשת. שכבה 4 אחראית לתקשורת מקצה לקצה ומבטיחה שהנתונים מועברים בצורה אמינה ויעילה ממכשיר אחד למשנהו. שכבת האפליקציה אחראית לספק את הממשק בין האפליקציות במכשיר לשכבה 4.

בהקשר של צ'אטים קבוצתיים וירטואליים, ארכיטקטורת שרת – לקוח היא דפוס עיצוב נפוץ שבו מכשירי הלקוח (המשתמשים המעוניינים בצ'אט קבוצתי) מתקשרים עם שרת מרכזי שמנהל את ה'אטים הקבוצתיים. השרת מספק שירות ללקוחות, כמו מתן גישה לצ'אט קבוצתי והדפסת הודעות חדשות.

על מנת ליישם צ'אטים קבוצתיים וירטואליים, השרת חייב להיות מסוגל לטפל במספר לקוחות בו זמנית ולמהל את התקשורת ביניהם. ניתן להשיג זאת באמצעות שילוב של טכנולוגיות שונות כגון sockets ו multithreading המאפשרים לשרת להקשיב ולעבד בקשות ממספר לקוחות במקביל.

השרת חייב להיות מסוגל לאחסן ולאחזר מידע על הצ'אטים הקבוצתיים והודעות שנשלחו. ניתן לעשות זאת באמצעות database או persistent storage system אחר.

המטלה:

במטלה התבקשנו לכתוב קוד של שרת – לקוח.

במקרה שלנו הלקוח הוא מישהו שמעוניין להתכתב בGROUP CHAT. והשרת הוא מחשב האחראי לנהל את כל הקבוצות צ'אט. השירותים שלו מכילים: גישה לקבות צ'אט, כתיבת הודעה לקבוצה וכו.

בעת חיבור של לקוח הוא מקבל הודעה ראשונית המכילה 3 אופציות:

1. חיבור לקבוצת צ'אט-
במקרה זה השרת מבקש מהלקוח שם, מספר מזהה של הקבוצה וסיסמא. אם מספר מזהה של הקבוצה קיים והסיסמא תואמת השרת מחבר את הלקוח לקבוצה הרלוונטית.

2. יצירת קבוצה חדשה-
במקרה זה השרת מקבל מהלקוח שם וסיסמא. לאחר מכן השרת יוצר מספר מזהה לקבוצה ומחבר את הלקוח לקבוצה שנוצר. הלקוח יקבל הודעה כאשר מספר מזהה לקבוצה נוצר.

3. יציאה מהשרת-
במקרה זה השרת מנתק את הלקוח מיידית. השרת ממשיך להאזין גם לאחר ניתק הלקוח.

הקוד:
בהתחלה של קובץ השרת וקובץ הלקוח מופיעות 3 שורות –
Import socket
מאפשר לנו ליצור ולהשתמש בsockets. Sockets הם ממשק תכנון רשת ברמה נמוכה המאפשר לתוכניות לשלוח ולקבל נתונים ברשת. הם משמשים לבניית יישומים ברשת כגון שרתי אינטרנט, לקוחות צ'אט ומשחקים מקוונים.

Import threading
מאפשר לנו עבודה עם threading. Threading הוא אפשרות לזרימה של ביצוע בתוך תכנית. המודל מאפשר ליצור ולנהל threads מרובים בתוכנית אחת.

Import time
מאפשר שימוש בפונקציות שונות לעבודה עם משימות קשורות לזמן.

קטע הקוד של השרת שקולט מהלקוח באיזה אופציה בוחר-

```
while True:
    option = conn.recv(1024).decode(FORMAT)  # receive option number.
    if option == '1' or option == '2' or option == '3':  # for valid option
        print(f"[OPTION SELECTED] client from {address} selected option {option}.\n")
        break
    else:  # for invalid option
        print(f"[ERROR] client from {address} selected an invalid option {option}.\n")
        conn.send("please try again!".encode(FORMAT))
        continue
```

אם בחר אופציה 1:

קטע קוד של הלקוח-

```
if option == '1':  # option 1 - connect an existing chat
    if "no chats available" in message:  # if there is no chat group
        return

    name = input()
    client_socket.send(name.encode(FORMAT))  # send client's name

    message = client_socket.recv(1024).decode(FORMAT)
    print(message)  # print request for group ID

    while True:  # repeat until group ID is valid
        group_id = input()
        client_socket.send(group_id.encode(FORMAT))  # send group ID
        message = client_socket.recv(1024).decode(FORMAT)
        print(message)
        if 'Wrong' not in message:  # if ID is valid
            break

    flag = False
    while not flag:  # repeat until receive  a correct password
        password = input()
        client_socket.send(password.encode(FORMAT))  # send a password
        message = client_socket.recv(1024).decode(FORMAT)
        if "welcome" in message:  # correct password
            print(message)
            flag = True
        else:
            print(message)  # wrong password
```

קטע קוד של השרת-

```
if option == '1':  # option 1 - connect an existing chat
    if not groups:  # if there is no chat group
        conn.send("You have chosen to connect to a chat, but there are no chats available to connect "
                  "to. You are disconnected from the server, you can try to connect again later, "
                  "bye!".encode(FORMAT))
        print("[ERROR] there are no chats available to connect.\n")
        return

    conn.send("Enter your name: ".encode(FORMAT))  # ask for client's name
    name = conn.recv(1024).decode(FORMAT)

    conn.send("Enter group ID: ".encode(FORMAT))  # ask for group ID
    while True:  # repeat until group ID is valid
        group_id = conn.recv(1024).decode(FORMAT)  # receive group ID
        if group_id not in groups.keys():  # group ID is invalid
            conn.send("Wrong ID, try again! ".encode(FORMAT))
        else:
            conn.send("Enter password: ".encode(FORMAT))
            print(f'[MEMBER REQUEST] Client {name} from {address} want to join group {group_id}.')
            break

    while True:  # repeat until receive a correct password
        password = conn.recv(1024).decode(FORMAT)
        if groups[group_id]['password'] == password:  # if received the correct password
            groups[group_id]['connections'].append(conn)  # connect client to the group
            conn.send(f"Hi {name}! welcome to group {group_id}".encode(FORMAT))
            print(f"[NEW MEMBER] {name} has joined to group {group_id}. ")

            broadcast(conn, ['notify message', f'{name}'], group_id)  # send message to group about new member
            break
        else:  # wrong password
            conn.send("Wrong password. Please try again! ".encode(FORMAT))
```

אם בחר אופציה 2:

קטע קוד של הלקוח-

```
if option == '2':  # option 2 - create a new chat group

    name = input()
    client_socket.send(name.encode(FORMAT))  # send client's name

    print(client_socket.recv(1024).decode(FORMAT))  # print password request
    password = input()
    client_socket.send(password.encode(FORMAT))  # send password
    print(client_socket.recv(1024).decode(FORMAT))  # approve message
```

קטע קוד של השרת-

```
if option == '2':  # option 2 - create a new chat group

    conn.send("Enter your name: ".encode(FORMAT))  # ask for client's name
    name = conn.recv(1024).decode(FORMAT)

    conn.send("Enter a new password for your group:".encode(FORMAT))  # ask for new password
    password = conn.recv(1024).decode(FORMAT)  # receive password

    groups[str(id_counter)] = {'connections': [conn], 'password': password, 'threads': []}  # create new group

    group_id = str(id_counter)       # save group ID
    id_counter += 1                  # update counter

    print(f'[NEW CHAT] client {name} has created chat. ID: {group_id}.')
    conn.send(f"Chat {group_id} has been created.".encode(FORMAT))
```

אם בחר אופציה 3:
קטע קוד של הלקוח –

```
if option == '3':  # option 3 - disconnect from server
    return
```

קטע קוד של השרת-

```
if option == '3':   # option 3 - di  sconnect from server
    print(f"[ENDING] client on {address} is disconnected. ")
    conn.send("You have disconnected from the server.".encode(FORMAT))
    return
```

הסבר על הפונקציות שבהן השתמשנו:

עבור השרת-

start_server()

פונקציה שמפעילה את השרת ומאזינה לחיבורים נכנסים. כאשר יש קריאה לפונקציה יש תחילה קישור של ה server socket עם IP portו מסוים באמצעות פונקציית server_socket.bind(). הIP והport מצוינים בADDRESS tuple המוגדר כקבוע בתחילת הקוד. לאחר מכן הפונקציה מדפיסה הודעה שמציינת שהשרת מאזין לחיבורים בכתובת שצוינה.
אחר כך השרת מתחיל להאזין לחיבורים נכנסים באמצעות הפונקציה server_socket.listen().
הפונקציה נכנסת ללולאה אינסופית שמחכה לחיבורים חדשים באמצעות הפונקציה server_socket.accept(). כאשר חיבור חדש מתקבל הפונקציה מאחזרת את החיבור ואת כתובת הלקוח ומדפיסה המציינת שנוצר חיבור חדש.
לבסוף הפונקציה יוצרת thread חדש לטיפול בחיבור לקוח באמצעות הפונקציה threading.Thread, פונקציית handle_client מוגדרת להיות היעד של הthread, משתני הכתובת והחיבור מועברים כארגומנטים. לאחר מכן הפונקציה מתחילה את הthread על מנת לטפל בחיבור לקוחות.
סך הכול נקבל כי הפונקציה start_server() אחראית הפעלת השרת, האזנה לחיבורים נכנסים ויצירת threads חדשים על מנת לטפל בחיבורים חדשים שנכנסים.

handle_client(conn, address)

פונקציה אשר מטפלת בחיבור לקוחות. הפונציה מקבלת את הsocket ואת הכתובת כפרמטרים.
הפונקציה מתחילה בזה שהיא שולחת הודעה ללקוח ששואלת אותו איזה אופציה הוא בוחר מתוך התפריט הראשית. לאחר שהלקוח עונה תשובה השרת מפענחת את התשובה באמצעות פונקציית client_socket.recv(). אם התשובה של הלקוח היא תקינה הפונקציה מסתיימת והשרת ממשיך לתקשר עם הלקוח לפי האופציה שבחר.
אם בחר אופציה 1 – הפונקציה בודקת אם יש קבוצות זמינות, אם אין מוציא הודעת שגיאה.
אם יש הפונקציה מבקשת מהלקוח שם, שם הקבוצה וסיסמה. לאחר מכן הפונקציה בודקת אם הקבוצה הזאת קיימת ואם הסיסמא תואמת, אם אחת הבדיקות נכשלו הפונקציה שולחת הודעת שגיאה ללקוח ומבקשת שוב את המידע הדרוש.
אם שם הקבוצה- המזה והסיסמא תקפים הפונקציה מוסיפה את הלקוח לקבוצה ומתחילה להחליף אתו הודעות.
אם בחר אופציה 2 – השרת מבקש מהלקוח שם ושם הקבוצה - מזהה שהוא רוצה ליצור.
הפונקציה בודקת אם המזהה הרצוי כבר קיים, אם כן הפונקציה שולחת הודעת שגיאה ללקוח ומבקשת מזהה חדש. אם המזהה אינו בשימוש הפונקציה יוצרת קבוצה חדשה עם המזהה שסופק ומוסיפה את הלקוח כמשתתף יחיד. לאחר מכן הפונקציה מתחילה להחליף הודעות עם הלקוח.
אם בחר אופציה 3 – השרת מדפיס הודעה למסך ומבצע return מה שסוגר את החיבור.

broadcast(sender, name, group_to_send)

הפונקציה שולחת הודעה מsender לכל שאר המשתתפים בקבוצה. הפונקציה מקבלת שלושה פרמטרים: חיבור של השולח, שם משתמש של השולח ואת הקבוצה שהוא רוצה לשלוח את ההודעה.
בהתחלה הפונקציה בודקת אם השם של השולח הוא 'notify message' זאת אומרת שזוהי הודעה מיוחדת אשר מייעדת על כך כי משתמש חדש התחבר לקבוצה, במקרה זה הפונקציה שולחת הודעה לכל המשתתפים בקבוצה כי לקוח חדש התחבר לקבוצה.
אם השם שונה מ'notify message' הפונקציה נכנסת ללולאה אינסופית אשר מקבלת הודעות מהלקוח ושולחת אותן לכל משתתפי הקבוצה. הפונקציה מקבלת הודעות באמצעות הפונקציה sender.recv() ומפענחת אותן באמצעות ENCODING_METHOD.
הפונקציה עוברת על כל הלקוחות בקבוצה ושולחת את ההודעה לכולם באמצעות הפונקציה member.send().
הפונקציה ממוקמת בתוך בלוק של try-except על מנת לתפוס שגיאות שמתרחשות בזמן השליחה, אם מתרחש שגיאה הודעת שגיאה מודפסת למסך.

עבור הלקוח-

start_client()

הפונקציה מטפלת בכל אינטרקציה של השרת עם הלקוח הכולל חיבור של הלקוח עם השרת, שליחה וקבלה של הודעות וסגירת החיבור.
הפונקציה מתחלה בחיבור השרת לsocket בשיטת client_socket.connect(). לאחר מכן הוא מקבל ומדפיס את הודעת הפתיחה מהשרת, מבקשים מהלקוח לבחור אפשרות מהתפריט הראשי. הפונקציה נכנסת ללולאה המבקשת מהמשתמש

את אחת האפשרויות ושולחת את התשובה לשרת. אם האפשרות שבחר אינה חוקית הלולאה חוזרת על עצמה. אם המשתמש בחר באפשרות 3, הפונקציה מקבלת ומדפיסה הודעת ניתוק מהשרת וחוזרת. אם המשתמש בחר באפשרות 1 או 2, הפונקציה מקבלת ומדפיסה הודעות מהשרת וממשיכה בחיבור.
אם הלקוח בחר באופציה 1- הפונקציה מבקשת מהמשתמש את שמו ושולחת אותו לשרת. לאחר מכן הוא נכנס ללולאה שמבקש מהמשתמש את מזהה הקבוצה ושולח אותו לשרת עד שהלקוח מספק מזהה קבוצתי חוקי. ברגע שמתקבל מזהה קבוצה חוקי, הפונקציה מבקשת מהמשתמש להזין את הסיסמא עבור הקבוצה ושולח אותו לשרת. אם הסיסמא נכונה, הפונקציה מקבלת ומדפיסה הודעת אישור מהשרת. אם הסיסמא אינה נכונה, הפונקציה מקבלת ומדפיסה הודעת שגיאה מהשרת וממשיכה בלולאה עד למתן סיסמא נכונה.
אם הלקוח בחר באופציה 2- הפונקציה מבקשת מהמשתמש להזין את שמו ואת המזהה והסיסמא עבור הקבוצה שהוא רוצה ליצור. לאחר מכן הוא שולח מידע זה לשרת ומקבל ומדפיס הודעת אישור מהשרת.
לאחר שהמשתמש התחבר בהצלחה לקבוצה הפונקציה נכנסת ללולאה אינסופית המבקשת מהמשתמש לשלוח הודעות באופן רציף ושולחת אותן לשרת.

receive(client_socket)
הפונקציה מקשיבה להודעות מהשרת ומדפיסה אותן.
הפונקציה נמצאת בלולאה אינסופית שבה היא מאזינה להודעות תוך שימוש בשיטת recv של פרמטר client_socket כדי לקבל הודעות באורך 124 בתים ולפענח אותן באמצעות ENCODING_METHOD.

send(client_socket, name)
פונקציה המשמשת לשליחת הודעות מהלקוח לשרת. הפונקציה מקשיבה לקלט מהלקוח, משרשר בין שם הלקוח לבין הודעה ממנו ושולח את ההודעה המתקבלת לשרת. הפונקציה רצה בלולאה אינסופית על מנת שהוא יוכל להמשיך להקשיב להודעות מהלקוח עד שהוא סוגר את החיבור.
אין התייחסות לפרמטרים שמועברים לפונקציה בפונקציה.
הפונקציה מתחילה עם להיכנס ללולאה אינסופית אשר מסתיימת כאשר הלקוח סוגר את החיבור. בתוך הלולאה יש האזנה לקלט מהלקוח, לאחר מכן יש שרשור של ההודעה עם שם הלחות באמצעות ':'. הפיענוח נעשה באמצעות ENCODING_METHOD ונשלח לשרת.

מילון:
groups{}
המילון groups נוצר בקוד של השרת והוא נועד לאכסן את הקבוצות והמשתתפים.
כל קבוצה מיוצגת על ידי מפתח במילון, הערך של כל קבוצה נמצאת במילון אחר אשר מכיל את הפרטים הבאים- מספר הקבוצה, שם, סיסמא ורשימת הלקוחות השייכים לקבוצה.
בנוסף שמרנו רשימה chat המכילה את כל הלקוחות המחוברים לשרת, משתנה id_client שסופר את הלקוחות על מנת לתת להם מספר מזהה חדש, ומשתנה message ששומר את ההודעה האחרונה שהתקבלה מהלקוח.

<u>פרוטוקול handshake</u>
handshake הוא כינוי לתהליך שבו מרכיבים שונים של מערכת תקשורת מבצעים תיאום לגבי שיטת התקשורת שתתקיים ביניהם. השיטה שהשתמשנו בה היא TCP/IP – פרוטוקול שמעביר נתונים באמצעות IP, מוודא את נכונותם ומאשר שהנתונים אכן התקבלו במלואם.
הפקודות שהשתמשנו בהם הן פקודות האזנה וקבלה של חיבורים.
השרת מאזין באמצעות server_socket.listen() וכך הוא בעצם ממתין לחיבור חדש- ללקוח חדש שיתחבר אליו. ברגע שלקוח חדש מתחבר, השרת מקבל אותו עם הפונקציה connection, address = server_socket.accept() שנמצאת בתוך לולאה אינסופית וכך השרת יכול לקבל תמיד לקוחות חדשים. הפונקציה מקבלת לקוח חדש ושומרת thread חדש בהתאם כדי לטפל בחיבור באמצעות הפונקציה handle_client

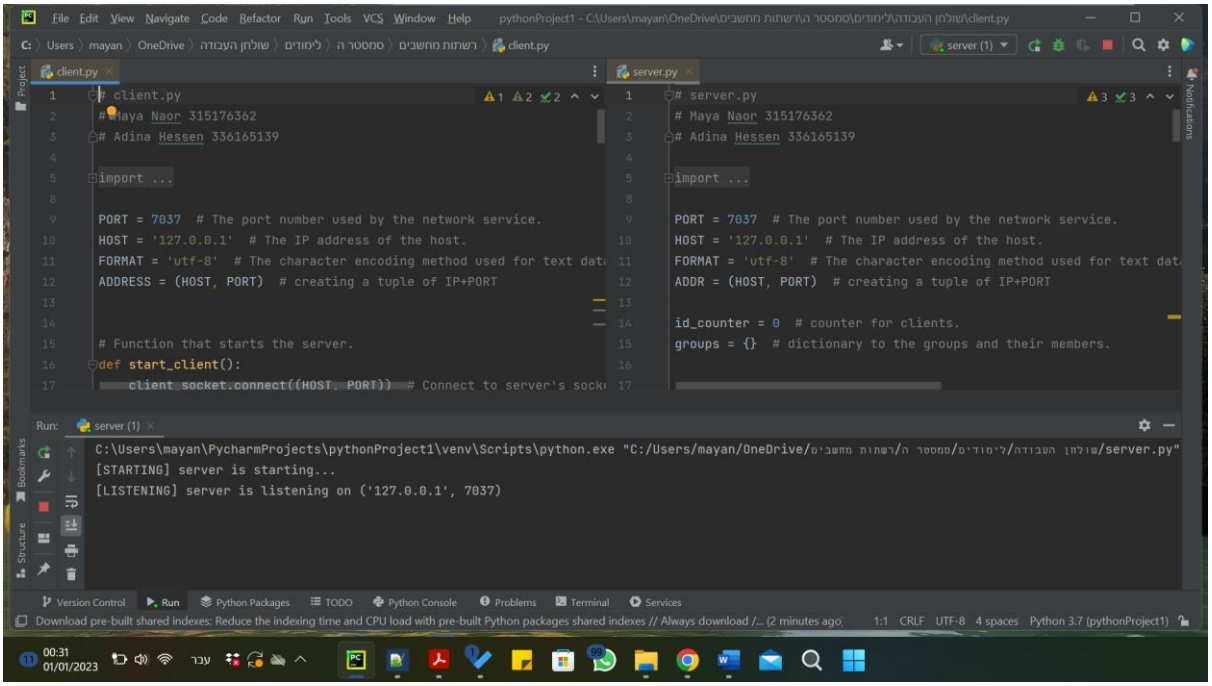
פקודות שקשורות לפרוטוקול handshake שהשתמשנו בהן במהלך התוכנית:

- server_socket.bind(ADDR)- פקודה שמקצה כתובת IP ומספר פורט של השרת לsocket.
- connection, address = server_socket.accept() – פקודה הממתינה לחיבור חדש ומחזירה את הכתובת של הלקוח ואת החיבור שלו.
- client_socket.connect((HOST, PORT))- פקודה היוצרת חיבור לsocket של השרת בכתובת הIP הנתונה.
- client_socket.send(option.encode(FORMAT))- פקודה השולחת נתונים לשרת דרך הsocket.
- message = client_socket.recv(1024).decode(FORMAT) – פקודה המקבלת נתונים מהשרת דרך הsocket.

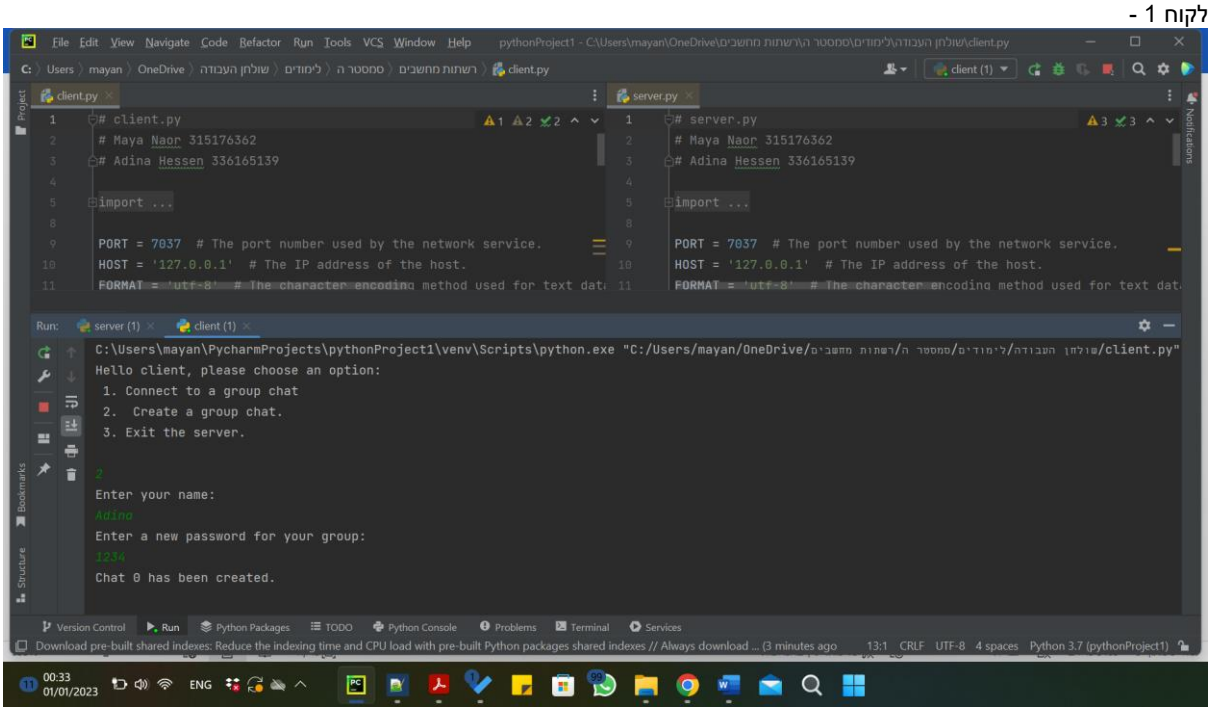
חמשת הפקודות הנ"ל הן פקודות חסימה- פקודות שגורמות לתוכנית להמתין לפעולה מסוימת שתתרחש.

- client_socket.close() – פקודה הסוגרת את חיבור הsocket (פקודה זו אינה פקודת חסימה).

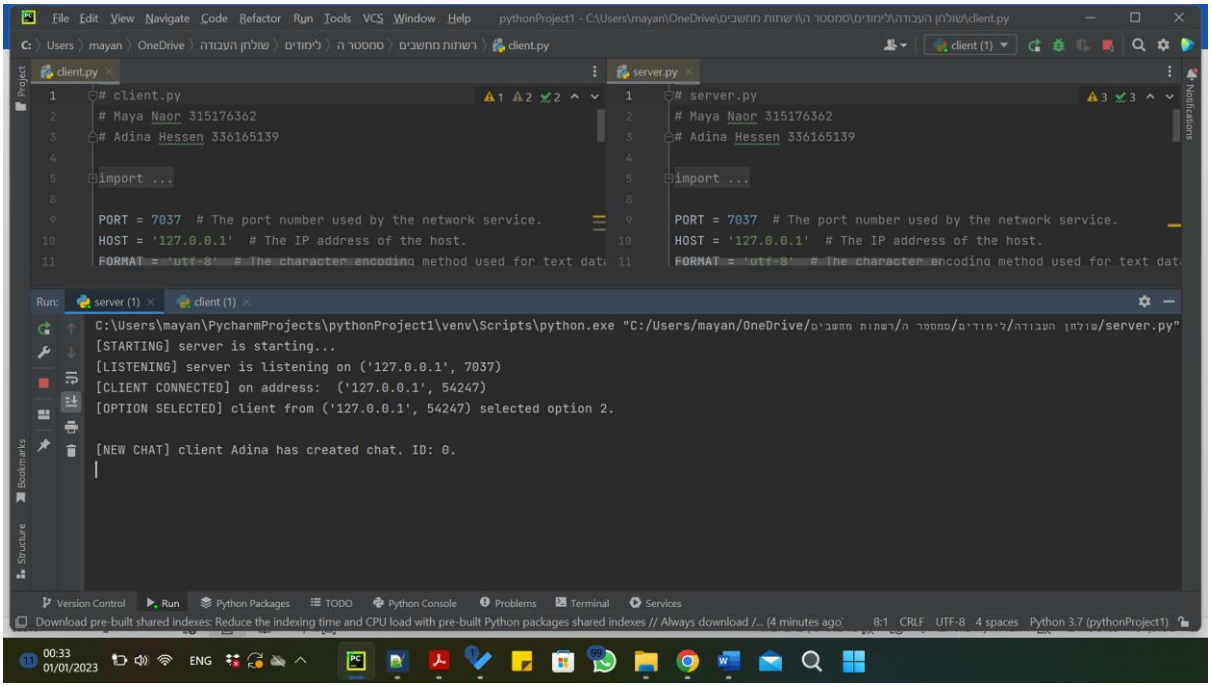
צילומי מסך מתוך הרצה של התוכנית:
פתיחה של השרת-



חיבור של לקוח חדש שבוחר לפתוח קבוצה חדשה (אופציה 2):
לקוח 1 -

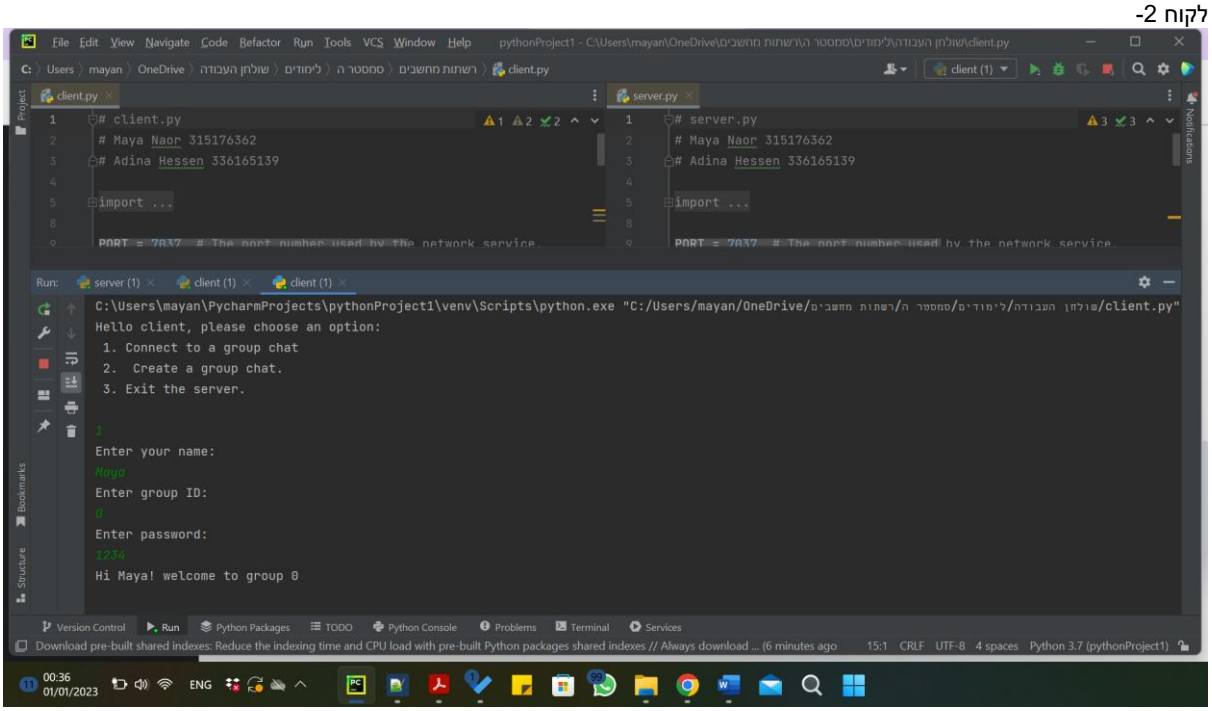


שרת-

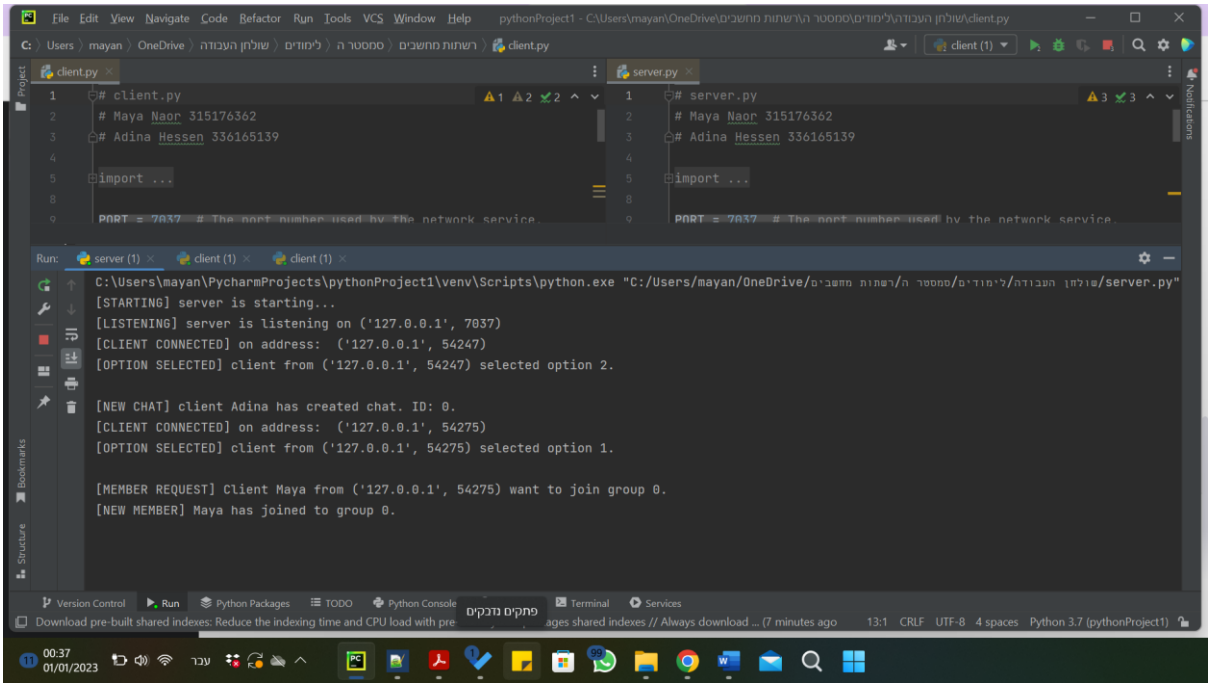


חיבור של לקוח חדש הבוחר להיכנס לקבוצה קיימת (אופציה 1):

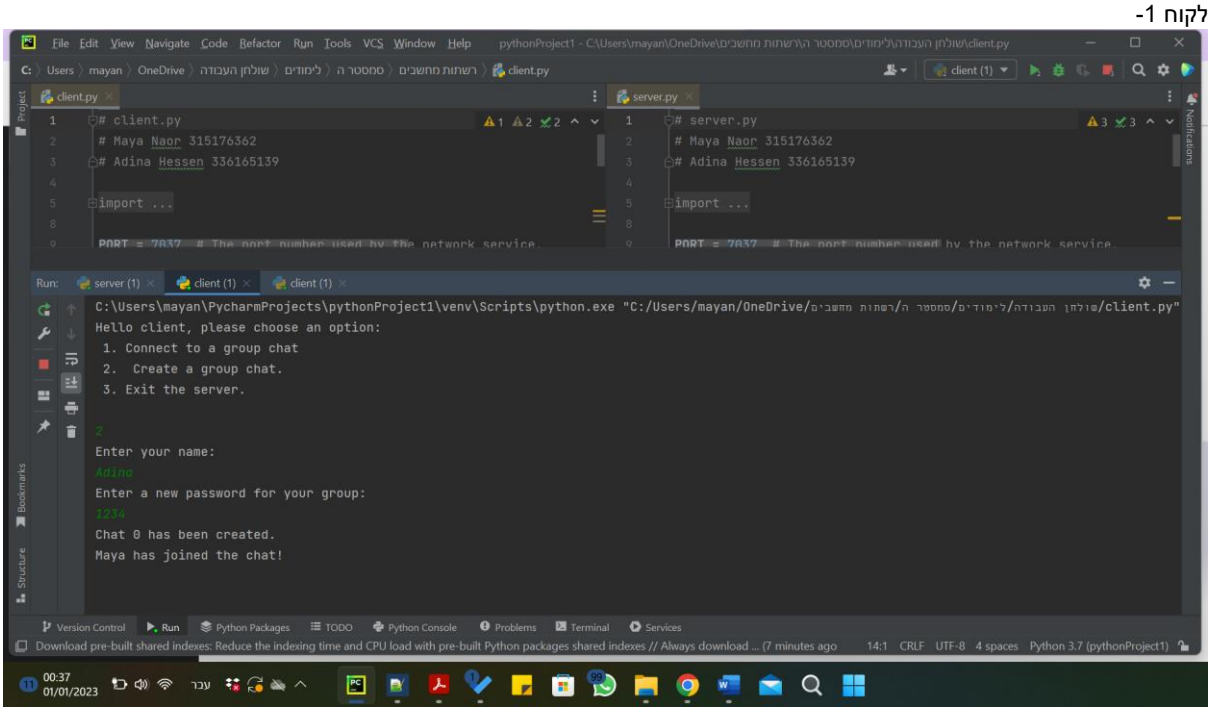
לקוח 2-



שרת-

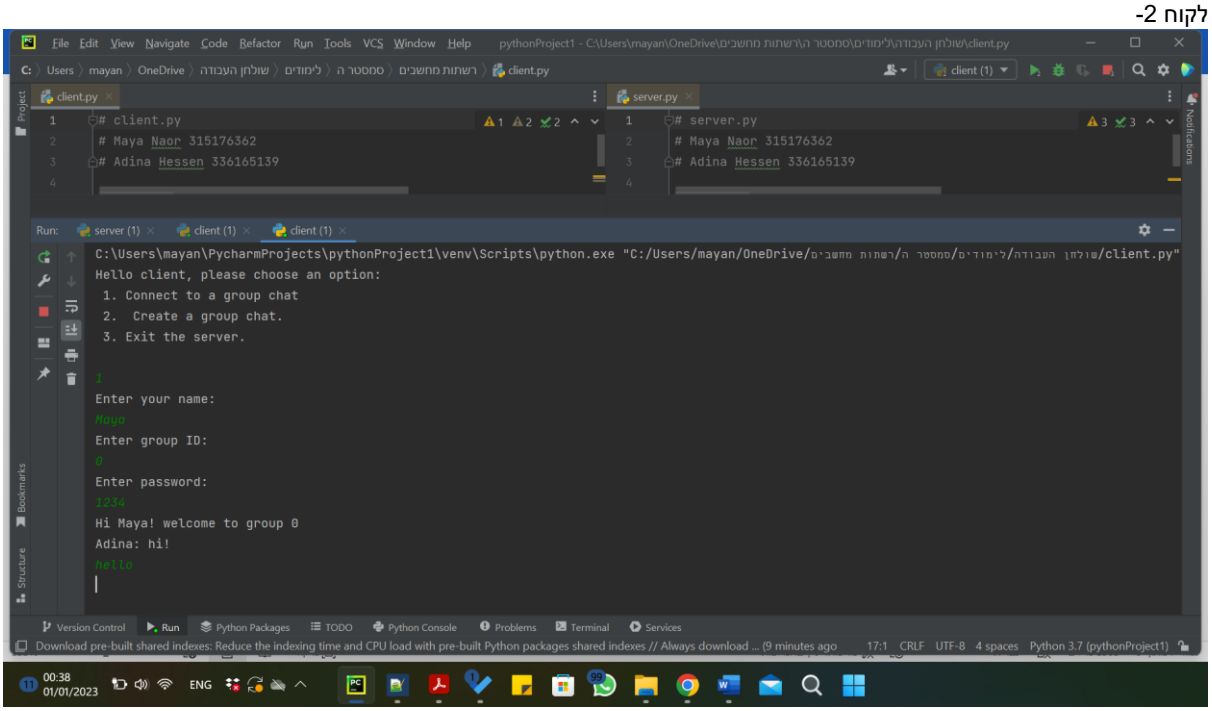

client.py
client.py
Maya Naor 315176362
Adina Hessen 336165139
import ...
server.py
server.py
Maya Naor 315176362
Adina Hessen 336165139
import ...
Run: server (1) client (1) client (1)
C:\Users\mayan\PycharmProjects\pythonProject1\venv\Scripts\python.exe "C:/Users/mayan/OneDrive/שולחן העבודה/לימודים/סמסטר ה/רשתות מחשבים/server.py"
[STARTING] server is starting...
[LISTENING] server is listening on ('127.0.0.1', 7037)
[CLIENT CONNECTED] on address: ('127.0.0.1', 54247)
[OPTION SELECTED] client from ('127.0.0.1', 54247) selected option 2.
[NEW CHAT] client Adina has created chat. ID: 0.
[CLIENT CONNECTED] on address: ('127.0.0.1', 54275)
[OPTION SELECTED] client from ('127.0.0.1', 54275) selected option 1.
[MEMBER REQUEST] Client Maya from ('127.0.0.1', 54275) want to join group 0.
[NEW MEMBER] Maya has joined to group 0.

לקוח 1-

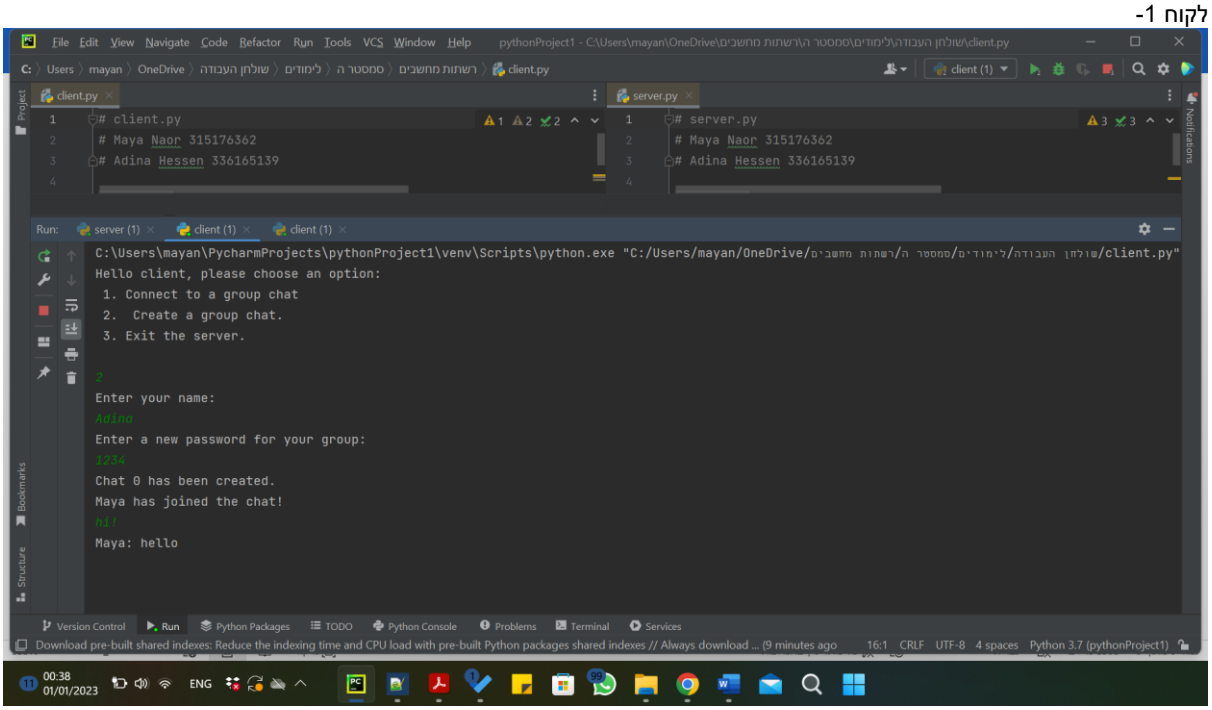

client.py
client.py
Maya Naor 315176362
Adina Hessen 336165139
import ...
server.py
server.py
Maya Naor 315176362
Adina Hessen 336165139
import ...
Run: server (1) client (1) client (1)
C:\Users\mayan\PycharmProjects\pythonProject1\venv\Scripts\python.exe "C:/Users/mayan/OneDrive/שולחן העבודה/לימודים/סמסטר ה/רשתות מחשבים/client.py"
Hello client, please choose an option:
1. Connect to a group chat
2. Create a group chat.
3. Exit the server.
2
Enter your name:
Adina
Enter a new password for your group:
1234
Chat 0 has been created.
Maya has joined the chat!

שליחת הודעות בין שני חברים בקבוצה:

לקוח 2-

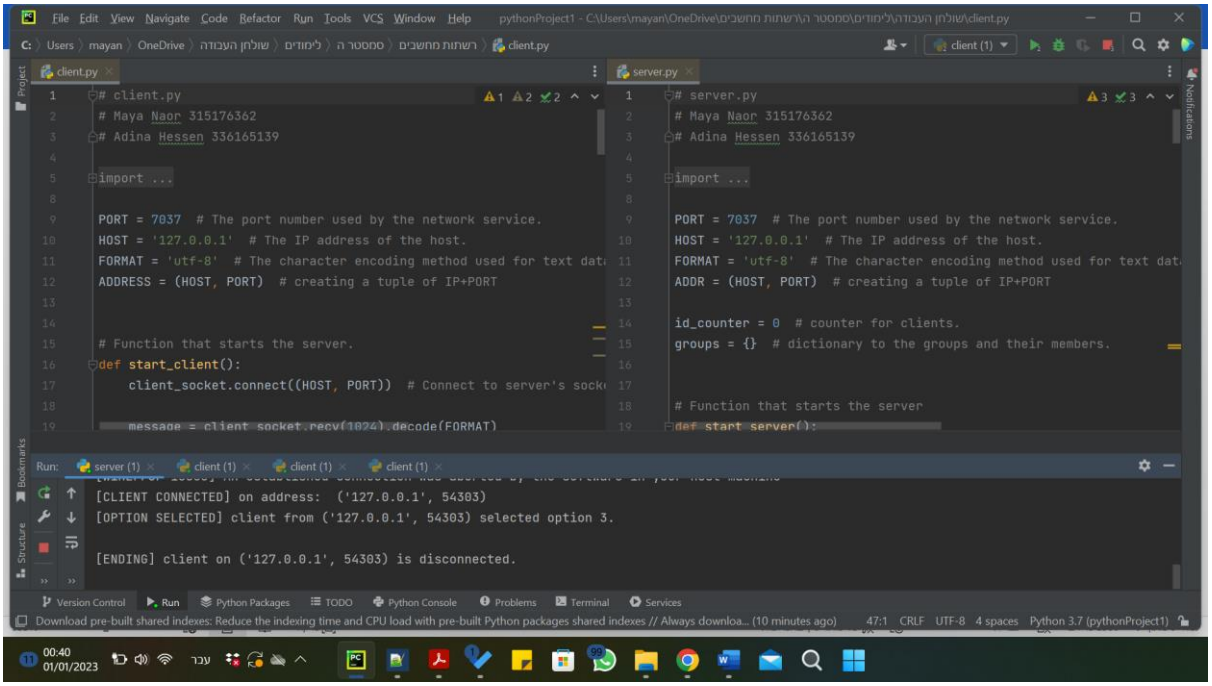


לקוח 1-



לקוח שבוחר להתנתק (אופציה 3):

שרת-



לקוח 3-