

בניית אפליקצית צ'ט בפייתון

תוכנת השרת מהווה כמו תחנת הפיקוח על הלקוחות. היא מקבלת את ההודעות מכל לקוח ומציגה אותן בסיגנון של שיחה. למשל, אם ישנם 2 לקוחות אחד בשם client1 והשני בשם client2, השיחה יכולה להיראות כך:

```
client1: Hello client2!  
client2: What's up client1?  
client1: Everything is cool, buddy. What have you been up to, dude?  
client1: Any new games you got?
```

לשם כך נשתמש בתכנות סוקטים ונבנה את הממשק הגרפי (GUI) ב-tkinter. נשתמש גם במודול threading על מנת שלא לתקוע את השיחה ולאפשר לכולם להשתתף בצ'ט.

כדי שהשרת ידע מי הם המשווחים, נשמור את המידע על כל לקוח חדש במילון עם מפתח שיהיה אובייקט הסוקט שלו והערך יהיה שמו, ומילון נוסף שיכיל כערך את הכתובת (ip, port) והמפתח שלו גם יהיה הסוקט.

בשרת תהיינה 2 פונקציות מרכזיות, אחת אחראית על קבלת לקוחות חדשים ולברך אותם על ההצטרפות לשיחה (accept_client_connections), והשנייה תהיה אחראית על השיחה של לקוח מסוים (handle_client)

שתי הפונקציות תנהלנה את פעולתן בלולאה אינסופית ולכן הן תופעלנה כ-thread, כלומר הליך ניפרד בפני עצמו, כך שבמקביל התוכנית שלנו תוכל גם לקבל לקוחות חדשים וגם לנהל שיחה עם הקיימים.

לאחר הבניה הראשונית נוסיף בדיקה אם לקוח לא דיבר יותר מדקה או שתיים, נשלח לו אזהרה שהוא עומד להיות מנותק מהצ'ט, אבל זה בשלב הבא.

ייבוא מודולים והגדרות גלובליות:

```
from socket import AF_INET, socket, SOCK_STREAM  
from threading import Thread  
import datetime  
import time
```

```
clients = {}  
addresses = {}  
time_last_heard = {}  
warnings = {}
```

```
HOST = "  
PORT = 33010
```

```
BUFSIZ = 1024
ADDR = (HOST, PORT)
SERVER = socket(AF_INET, SOCK_STREAM)
SERVER.bind(ADDR)
```

כך תיראה התוכנית הראשית שלנו:

```
main():
    SERVER.listen(5)
    print("Waiting for connection...")
    ACCEPT_THREAD = Thread(target=accept_incoming_connections)
    ACCEPT_THREAD.start()
    ACCEPT_THREAD.join()
    SERVER.close()
```

הפעלנו כהליך (או תהליכון) את קבלת לקוחות חדשים.

כך תיראה הפונקציה של קבלת לקוחות חדשים:

```
def accept_incoming_connections():
    """Sets up handling for incoming clients."""
    while True:
        client, client_address = SERVER.accept()
        print("%s:%s has connected." % client_address)
        client.send(bytes("Greetings from hell! Now type your name and
                           press enter!", "utf8"))
        addresses[client] = client_address
        Thread(target=handle_client, args=(client,)).start()
        #Thread(target=checktimeout).start()
```

למה כדאי לשים לב:

בשורה האחרונה עדיין איננו מפעילים תהליכון של בדיקה אם הלקוח עבר את הזמן המותר ללא דיבור.

בפייתון 3 המשלוח של הודעות ברשת הוא byte code לא מחרוזת רגילה ולכן לפני המשלוח הוא מקודד עם utf8 (שמאפשר לנו גם עברית).

המשתנה socket הוא האובייקט של הסוקט, ו-client_address זהו הצמד (ip,port)

הפעלנו את הטיפול בלקוח שגם הוא לולאה אינסופית (הלקוח תמיד צודק, אז צריך להיות קשוב אליו) כתהליכון ניפרד עם פרמטר שהוא ה-socket שלו.

כך תיראה הפונקציה של טיפול בלקוח:

```

def handle_client(client): # Takes client socket as argument.
    """Handles a single client connection."""

    name = client.recv(BUFSIZ).decode("utf8")
    welcome = 'Welcome %s! If you ever want to quit, type {quit} to exit.' %
        name
    client.send(bytes(welcome, "utf8"))
    msg = "%s has joined the chat!" % name
    broadcast(bytes(msg, "utf8"))
    clients[client] = name
    #time_last_heard[client] = get_seconds()

    while True:
        msg = client.recv(BUFSIZ)
        if msg != bytes("{quit}", "utf8"):
            broadcast(msg, name + ": ")
            #time_last_heard[client] = get_seconds()
        else:
            client.send(bytes("{quit}", "utf8"))
            client.close()
            del clients[client]
            broadcast(bytes("%s has left the chat." % name, "utf8"))
            break

```

מה חשוב לשים לב?
שמנו בהערה את מה שקשור לבדיקת הזמן כרגע.

בתקווה שהלקוח מסר את שמו (בהודעה הראשונה אותה קיבלנו לפני הלולאה),
הוא מוכנס למילון clients.

מתחילים לולאה אינסופית של קבלת הודעות מהלקוח. הקוד עבור הלקוח לצאת
מהשיחה: {quit} כל עוד אין את הקוד בהודעה, השרת משדר אותה לכולם בעזרת
הפונקציה: broadcast

אם הלקוח עוזב, נסגור את הסוקט ונורידו מהמילון של הלקוחות, נודיע לכולם
שיצא.

כך נראית הפונקציה ששולחת הודעות לכולם - broadcast

```
def broadcast(msg, prefix=""): # prefix is for name identification.
    """Broadcasts a message to all the clients."""
    for sock in clients:
        sock.send(bytes(prefix, "utf8") + msg)
```

clients הוא כאמור מילון גלובלי עם מפתחות שהם הסוקטים של הלקוחות השונים וכל מה שעושה פונקציה זו היא לולאה של משלוח הודעה לכל אחד מהלקוחות שעדיין מחוברים.

תוכנת הלקוח

הדבר העיקרי הוא תיכנון הממשק הגראפי, שהוא כאן פשוט למדי ויכול להשתפר בהרבה.

לאחר הממשק הגראפי, מבקשים את כתובת האי פי והפורט (אותו פורט עליו רץ השרת). על אותו מחשב שרת/לקוח, נשתמש בכתובת 127.0.0.1 וניתן את ברירת המחדל לפורט. הנה הקוד של הבקשה והפעלת הקבלה (receive) על ידי תהליכון:

```
HOST = input('Enter host: ')
PORT = input('Enter port: ')
if not PORT:
    PORT = 33010
else:
    PORT = int(PORT)
```

```
BUFSIZ = 1024
ADDR = (HOST, PORT)
```

```
client_socket = socket(AF_INET, SOCK_STREAM)
client_socket.connect(ADDR)
```

```
receive_thread = Thread(target=receive, \
                        args=[client_socket, msg_list, BUFSIZ])
receive_thread.start()
top.mainloop() # Starts GUI execution.
```

יש לשים לב לשורה האחרונה שזו ההפעלה של ה-GUI של tkinter.

כך נראית הפונקציה שמקבלת הודעות עבור הלקוח מהשרת:

```
def receive(client_socket, msg_list, BUFSIZ):
    """Handles receiving of messages."""
    while True:
        try:
            msg = client_socket.recv(BUFSIZ).decode("utf8")
            msg_list.insert( END, msg)
        except OSError: # Possibly client has left the chat.
            break
```

לולאה אינסופית שמחכה להודעות. כשמגיעה הודעה, מכניסים אותה לאיזור של השיחות (msg_list). בסוף הדיון נראה גם את הממשק הגראפי וזה יהיה ברור יותר לאן הולכות ההודעות מהשרת.

כך נראית פונקציה משלוח ההודעות:

```
def send(client_socket, my_msg, top, event=None):
    """Handles sending of messages."""
    msg = my_msg.get()
    my_msg.set("") # Clears input field.
    client_socket.send(bytes(msg, "utf8"))
    if msg == "{quit}":
        client_socket.close()
        top.quit()
```

לאחר המשלוח מנוקה השדה שבו נכתבת ההודעה (my_msg). אם ההודעה הייתה של ניתוק, אנו סוגרים את הסוקט, וסוגרים את הממשק הגראפי זהו פרוטוקול מאד פשוט.

הנה הממשק הגראפי הפשוט עבור השיחות:

```
def main():
    top = Tk()
    top.title("Chatter")
    messages_frame = Frame(top)
    my_msg = StringVar() # For the messages to be sent.
    my_msg.set("Type your messages here.")
    scrollbar = Scrollbar(messages_frame) # To navigate past messages.
    # Following will contain the messages.
    msg_list =Listbox(messages_frame,height=15,width=50,
                      yscrollcommand=scrollbar.set)
    scrollbar.pack(side= RIGHT, fill= Y)
    msg_list.pack(side= LEFT, fill= BOTH)
    msg_list.pack()
```

```

messages_frame.pack()

entry_field = Entry(top, textvariable=my_msg)
entry_field.bind("<Return>", lambda event: send(client_socket, \
        my_msg, top))
entry_field.pack()
send_button = Button(top, text="Send", command= lambda: \
        send(client_socket, my_msg, top))
send_button.pack()
top.protocol("WM_DELETE_WINDOW", \
        lambda : on_closing(client_socket, my_msg, top))

```

השורה האחרונה מיועדת לביצוע כאשר מתכוונים לסגור את הצ'ט על ידי לחיצת ה-X של חלון השיחה. זוהי בעצם אינטראקציה של tkinter עם מה שנקרא: Windows manager. והמתודה protocol יכולה להיות מופעלת רק מהרמה הגבוה ביותר של הממשק. מה שזה אומר, אם נילחץ ה-X (כלומר קרה האירוע: WM_DELETE_WINDOW), בצע את הקוד של ההנדלר שנקרא: on_closing

והנה הקוד של on_closing

```

def on_closing(client_socket, my_msg, top, event=None):
    """This function is to be called when the window is closed with X"""
    my_msg.set("{quit}")
    send(client_socket, my_msg, top)

```

זהו בעצם חיקוי למה שהיה קורה אם המשתמש היה מקיש {quit} ושולח לשרת.

תרגיל

להוסיף את האזהרה של הזמן לשרת. ניכתוב 2 פונקציות, אחת ניקראת checktimeout והשניה ניקראת get_seconds שמצויות בקוד הנ"ל בהערה. נשתמש במילונים (שמוגדרים גלובלית ולכן יש גישה אליהם ולכן checktimeout לא צריכה פרמטרים):

```

time_last_heard = {}
warnings = {}

```

כדי לתת אזהרה לכל לקוח שלוקח יותר מדקה לכתוב הודעה נוספת. שימו לב שההפעלה של checktimeout נעשית על ידי תהליכון ניפרד, כלומר היא רצה ברקע כל הזמן, כך שהיא יכולה לפקח על כל הלקוחות שמשתתפים בשיחה. היא מופעלת כל פעם שנכנס לקוח חדש (בלולאה של accept)

שימו לב שהפונקציה: get_seconds מופעלת גם לפני הלולאה של handle_client וגם בתוך הלולאה. מדוע זה כך לדעתכם?