**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**3-Mavzu:** Pythonda tarmoq dasturlashga kirish.

**6-mashg‘ulot.** Pythonda GUI paketidan foydalanib chat dasturini tuzishni yakunlash

**O‘quv savollari:**

1. TCP client-server dasturini GUI ko’rinishda ishlab chiqish.

**1. TCP client-server dasturini GUI ko’rinishda ishlab chiqish.**

import argparse

from socket import AF\_INET, socket, SOCK\_STREAM

from threading import Thread

def accept\_incoming\_connections():

"""Sets up handling for incoming clients."""

while True:

client, client\_address = SERVER.accept()

print("%s:%s has connected." % client\_address)

addresses[client] = client\_address

Thread(target=handle\_client, args=(client,)).start()

def handle\_client(client): # Takes client socket as argument.

"""Handles a single client connection."""

name = ""

prefix = ""

while True:

msg = client.recv(BUFSIZ)

if not msg is None:

msg = msg.decode("utf-8")

if msg == "":

msg = "{QUIT}"

# Avoid messages before registering

if msg.startswith("{ALL}") and name:

new\_msg = msg.replace("{ALL}", "{MSG}"+prefix)

send\_message(new\_msg, broadcast=True)

continue

if msg.startswith("{REGISTER}"):

name = msg.split("}")[1]

welcome = '{MSG}Welcome %s!' % name

send\_message(welcome, destination=client)

msg = "{MSG}%s has joined the chat!" % name

send\_message(msg, broadcast=True)

clients[client] = name

prefix = name + ": "

send\_clients()

continue

if msg == "{QUIT}":

client.close()

try:

del clients[client]

except KeyError:

pass

if name:

send\_message("{MSG}%s has left the chat." % name, broadcast=True)

send\_clients()

break

# Avoid messages before registering

if not name:

continue

# We got until this point, it is either an unknown message or for an

# specific client...

try:

msg\_params = msg.split("}")

dest\_name = msg\_params[0][1:] # Remove the {

dest\_sock = find\_client\_socket(dest\_name)

if dest\_sock:

send\_message(msg\_params[1], prefix=prefix, destination=dest\_sock)

else:

print("Invalid Destination. %s" % dest\_name)

except:

print("Error parsing the message: %s" % msg)

def send\_clients():

send\_message("{CLIENTS}" + get\_clients\_names(), broadcast=True)

def get\_clients\_names(separator="|"):

names = []

for \_, name in clients.items():

names.append(name)

return separator.join(names)

def find\_client\_socket(name):

for cli\_sock, cli\_name in clients.items():

if cli\_name == name:

return cli\_sock

return None

def send\_message(msg, prefix="", destination=None, broadcast=False):

send\_msg = bytes(prefix + msg, "utf-8")

if broadcast:

"""Broadcasts a message to all the clients."""

for sock in clients:

sock.send(send\_msg)

else:

if destination is not None:

destination.send(send\_msg)

clients = {}

addresses = {}

parser = argparse.ArgumentParser(description="Chat Server")

parser.add\_argument(

'--host',

help='Host IP',

default="127.0.0.1"

)

parser.add\_argument(

'--port',

help='Port Number',

default=33002

)

server\_args = parser.parse\_args()

HOST = server\_args.host

PORT = int(server\_args.port)

BUFSIZ = 2048

ADDR = (HOST, PORT)

stop\_server = False

SERVER = socket(AF\_INET, SOCK\_STREAM)

SERVER.bind(ADDR)

if \_\_name\_\_ == "\_\_main\_\_":

try:

SERVER.listen(5)

print("Server Started at {}:{}".format(HOST, PORT))

print("Waiting for connection...")

ACCEPT\_THREAD = Thread(target=accept\_incoming\_connections)

ACCEPT\_THREAD.start()

ACCEPT\_THREAD.join()

SERVER.close()

except KeyboardInterrupt:

print("Closing...")

ACCEPT\_THREAD.interrupt()