```python
# ------------------------------------------------
# Client
# ------------------------------------------------


import socket
import threading
import Bot
import sys
import logging
import argparse
import re

# ---Input validation-------------------

# Create the parser
my_parser = argparse.ArgumentParser(description='User/bot chat client for chatychaty server')

# Requierd comand line arguments for  clinet.py
my_parser.add_argument('Ip',
                       metavar='ip',
                       type=str,
                       help='Ip adress of server [0-255].[0-255].[0-255].[0-255] ')

my_parser.add_argument('Port',
                       metavar='port',
                       type=int,
                       help='Port number of server [0 - 65535]')

my_parser.add_argument('Mode',
                       metavar='mode',
                       type=str,
                       help='Two modes: user or bot | [user] or [bot]')

my_parser.add_argument('Name',
                       metavar='name',
                       type=str,
                       help='If in bot mode type bot name,Available bots: Alice, Bob, Dora, Chuck\n If in user mode '
                            'type nickname')

# Execute the parse_args() method

args = my_parser.parse_args()

# Check for valid ip format and set ip

valid_ipaddress_regex = "^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$";
ip_regexp = re.search(valid_ipaddress_regex, args.Ip)

if ip_regexp:
    address = args.Ip  # server address
else:
    logging.error("Not a valid ip format, valid format: [0-255].[0-255].[0-255].[0-255] ")
    sys.exit()

    # Sets port number
port = args.Port

# Check for user mode
if (args.Mode == 'user') or (args.Mode == 'bot'):
    mode = args.Mode  # user or bot mode bot
else:
    logging.error("Not valid mode, valid modes: user, bot ")
```

```
64      sys.exit()
65
66  # Sets checks for vali bot name
67
68  if args.Mode == 'bot':
69
70      bot_check = args.Name
71      bot_check = bot_check.lower()
72      bot_list = ['alice', 'bob', 'dora', 'chuck']
73
74      if bot_check in bot_list:
75          name = args.Name
76      else:
77          logging.error("Invalid bot name, valid bot names: Alice, Bob, Dora, Chuck ")
78          sys.exit()
79
80  # Sets username
81
82  if args.Mode == 'user':
83      name = args.Name
84
85  # ---Net code-------------------
86
87  client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # Define tcp protocol
    for client
88  client.connect((address, port))  # Adress and port of chat server local '127.0.0.1
    ', 55556
89
90
91  # ---Bot code-------------------
92
93
94  def bot_io():  # Funktion for reciving messages form chat server
95      while True:
96          try:
97              message = client.recv(1024).decode('utf8')
98              if message == 'NICK':  # Send nickname of client when server asks for it
99                  client.send(name.encode('utf8'))
100             elif message == 'NICK_INVALID':  # If nickname is used disconnect
101                 print(f'Bot: {name} Nickname already in use')
102             elif message == 'NICK_OK':  # If nickname is ok print connect message
103                 print(f'Bot: {name} connected')
104             elif message == 'KICK':  # Kick message from server disconnects client
105                 client.close()
106                 print(f'Bot: {name} Kicked')
107             elif Bot.name_check(message):  # If message is from a bot ignore
108                 pass
109             else:
110                 keyword = Bot.find_keyword(message)  # Check i chat message has a
    reply keyword
111                 if keyword != "NOMATCH":  # Keword is a match
112                     bot_name = name.lower()
113                     bot_reply = f'{name}: {(Bot.response(bot_name, keyword))}'  #
    Activate bot reply with keyword
114                     client.send(bot_reply.encode('utf8'))
115                     print(f'Bot reply: {bot_reply}')  # Console log info
116
117
118         except:
119             logging.error("Com error!")  # If server is down disconnect ´
120             client.close()
121             break
122
123
124 # ---User code------------------
125
126
```

```
127 def user_receive():  # Funktion for receiving messages form chat server
128     while True:
129 
130         try:
131             message = client.recv(1024).decode('utf8')
132             if message == 'NICK':  # Send nickname of client when server asks for it
133                 client.send(name.encode('utf8'))
134             elif message == 'NICK_INVALID':  # If nickname is used disconnect
135                 client.close()
136                 print(f'User: {name} nickname already in use')  # If nickname is ok
    print connect message
137             elif message == 'NICK_OK':
138                 print(f'{name} connected')
139             elif message == 'KICK':
140                 client.close()
141                 print(f'User: {name} Kicked')
142             else:
143                 print(f'{message}')  # If not nick request print message
144 
145         except:
146             print(f"Disconected from server!")  # If server is down disconnect ´
147             client.close()
148             break
149 
150 
151 def user_send():  # Function for sending messages to chat server
152     while True:
153         try:
154             message = f'{name}: {input("")}'
155             client.send(message.encode('utf8'))
156         except Exception as e:
157             print(f"Com error!{e.__class__}")  # If server is down disconnect ´
158             client.close()
159             break
160 
161 
162 def main():
163     if mode == "user":
164         user_receive_thread = threading.Thread(target=user_receive)  # A thread for
    receiving messages to chat server
165         user_receive_thread.start()
166 
167         user_send_thread = threading.Thread(target=user_send())  # A thread for
    sending messages to chat server
168         user_send_thread.start()
169 
170     if mode == "bot":
171         bot_io_thread = threading.Thread(target=bot_io)  # A thread for receiving
    messages to chat server
172         bot_io_thread.start()
173 
174 
175 if __name__ == "__main__":
176     main()
177 
```