

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное  
учреждение высшего образования  
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»  
Факультет инфокоммуникационных технологий

**ОТЧЕТ**  
**О ЛАБОРАТОРНОЙ РАБОТЕ №1**  
по теме: сокет  
по дисциплине: Web-программирование

Специальность:  
09.03.03 Мобильные и сетевые технологии

Проверил:  
Говоров А.И. \_\_\_\_\_  
Дата: «08» октября 2021г.  
Оценка \_\_\_\_\_

Выполнил:  
студент группы  
К33401  
Фоменко Иван

Санкт-Петербург 2021 г.

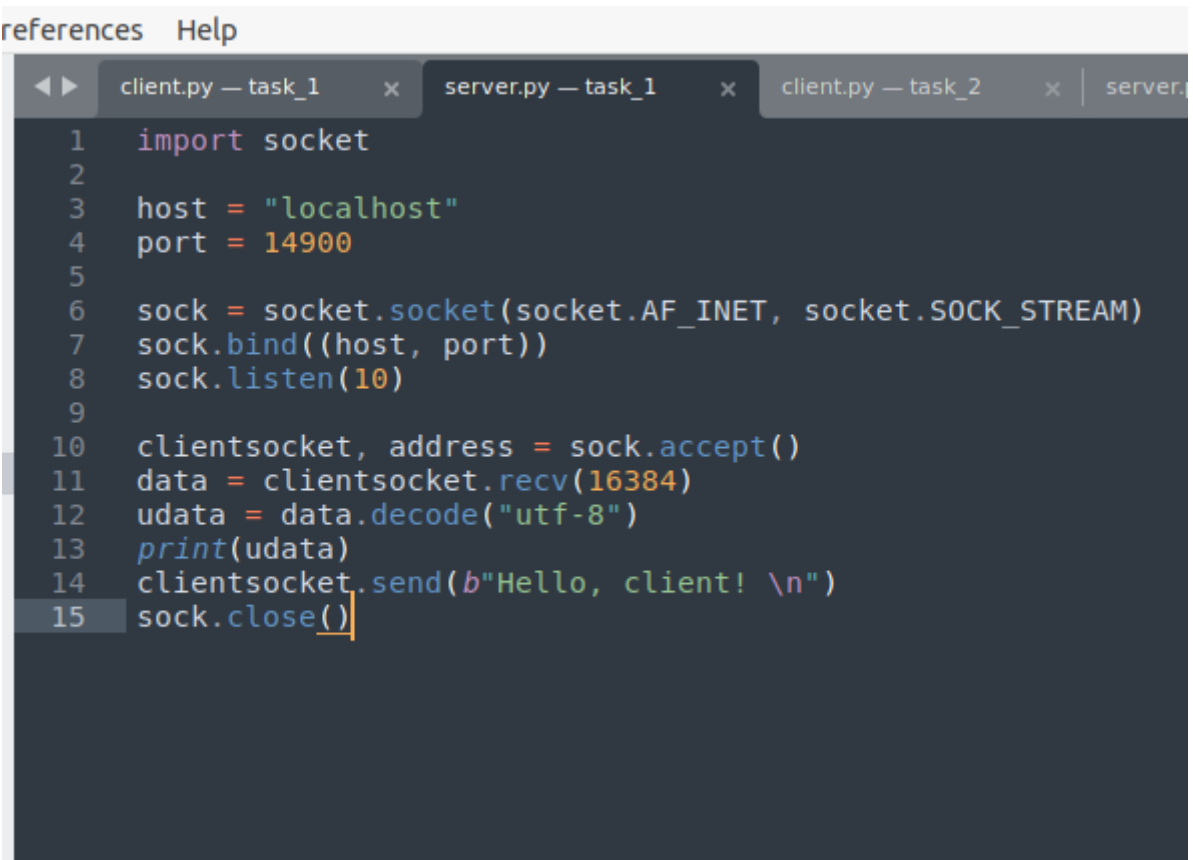
## Цель работы:

Реализация клиентской и серверной части приложений

## Выполнение:

Задание 1.

Сервер

A screenshot of a code editor window with a dark theme. The window has a title bar with 'references' and 'Help' on the left, and several tabs on the right: 'client.py — task\_1', 'server.py — task\_1', 'client.py — task\_2', and 'server.py'. The active tab is 'server.py — task\_1'. The code is written in Python and is as follows:

```
1  import socket
2
3  host = "localhost"
4  port = 14900
5
6  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7  sock.bind((host, port))
8  sock.listen(10)
9
10 clientsocket, address = sock.accept()
11 data = clientsocket.recv(16384)
12 udata = data.decode("utf-8")
13 print(udata)
14 clientsocket.send(b"Hello, client! \n")
15 sock.close()
```

## Клиент

```
client.py — task_1 x server.py — task_1 x client.py — task_2 x server.py — task_2 x
1  import socket
2
3  host = "localhost"
4  port = 14900
5
6  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7  sock.connect((host, port))
8  sock.send(b"Hello, server! \n")
9
10 data = sock.recv(16384)
11 print(data.decode("utf-8"))
12 sock.close()
13
```

## Пример работы

```
Ivan@Ivan-VirtualBox: ~/web/ITMO_ICT_WebDevelopment_2021-2022/
ivan@Ivan-VirtualBox:~$ cd web/ITMO_ICT_WebDevelopment_2021-2022/
ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/$ cd students/K33401/Fomenko_Ivan/Lr1/task_1
ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_1$ alias python=python3
ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_1$ python -i client.py
Hello, client!
>>>

Ivan@Ivan-VirtualBox: ~/web/ITMO_ICT_WebDevelopment_2021-2022/
ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/$ cd students/K33401/Fomenko_Ivan/Lr1
ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1$ ls
task_1
ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1$ cd task_1
ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_1$ alias python=python3
ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_1$ python -i server.py
Hello, server!
>>>
```

## Задание 2.

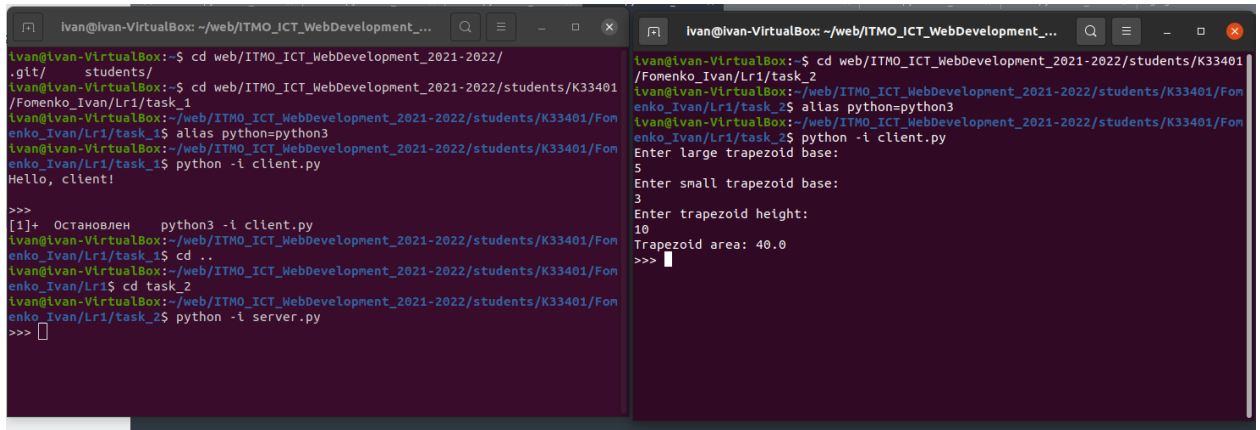
### Сервер

```
client.py — task_1  x  server.py — task_1  x  client.py — task_2  x  server.py — task_2  x
1  import socket
2
3  host = "localhost"
4  port = 14900
5
6  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7  sock.bind((host, port))
8  sock.listen(10)
9
10 #Version with trapezoid
11
12 clientsocket, address = sock.accept()
13 clientsocket.send(b"Enter large trapezoid base:")
14 data = clientsocket.recv(16384)
15 large_base = int(data.decode())
16 clientsocket.send(b"Enter small trapezoid base:")
17 data = clientsocket.recv(16384)
18 small_base = int(data.decode())
19 clientsocket.send(b"Enter trapezoid height:")
20 data = clientsocket.recv(16384)
21 height = int(data.decode())
22 area = height * ((large_base + small_base) / 2)
23 clientsocket.send(f"Trapezoid area: {area}".encode())
24 sock.close()
```

### Клиент

```
client.py — task_1  x  server.py — task_1  x  client.py — task_2  x  server.py — task_2  x
1  import socket
2
3  host = "localhost"
4  port = 14900
5
6  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7  sock.connect((host, port))
8
9  #Version with trapezoid
10
11 for i in range(3):
12     data = sock.recv(16384)
13     text = data.decode()
14     print(text)
15     proportions = input()
16     sock.send(proportions.encode())
17
18 data = sock.recv(16384)
19 trapezoid_area = data.decode()
20 print(trapezoid_area)
21 sock.close()
```

## Пример работы



The image shows two terminal windows from a VirtualBox environment. The left window shows a user navigating through directories and running a Python script to start a web server. The right window shows the same user running a client script that interacts with the server, providing input for a trapezoid and receiving the area calculation.

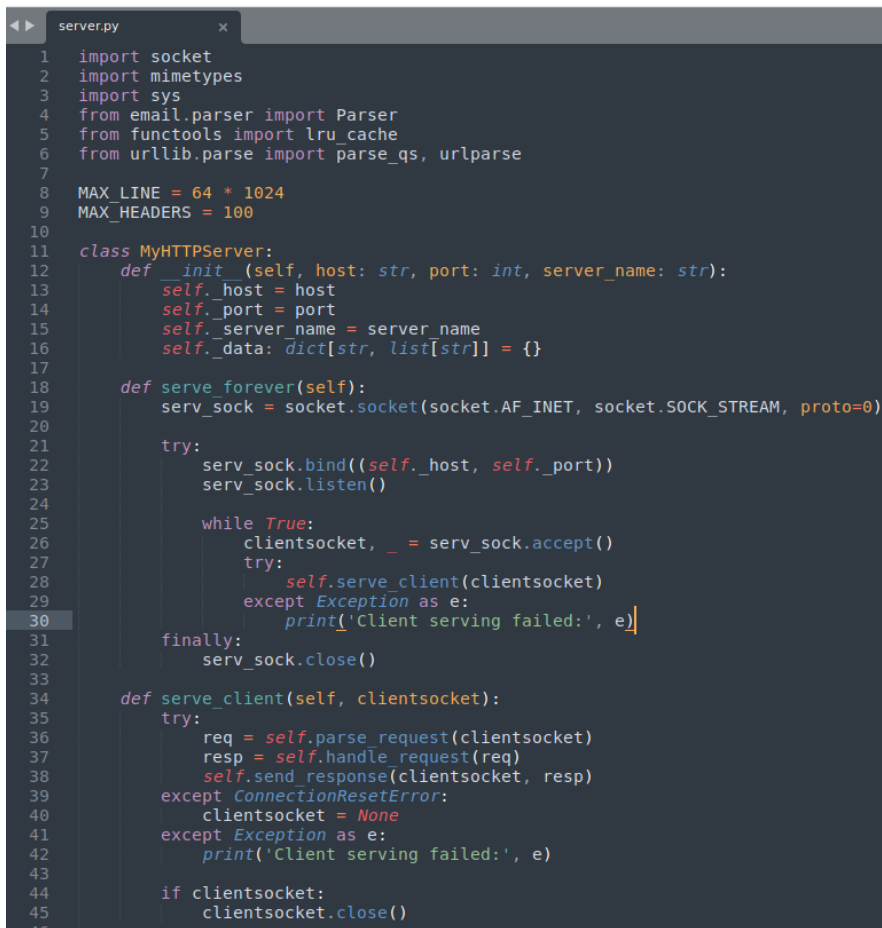
```
Ivan@Ivan-VirtualBox: ~/web/ITMO_ICT_WebDevelopment_2021-2022/
Ivan@Ivan-VirtualBox:~$ cd web/ITMO_ICT_WebDevelopment_2021-2022/
Ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/$ cd web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/
Ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_1$ cd ..
Ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_1$ alias python=python3
Ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_1$ python -i client.py
Hello, client!

>>>
[1]+  Остановлен python3 -i client.py
Ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_1$ cd ..
Ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1$ cd task_2
Ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_2$ python -i server.py
>>>
```

```
Ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_2$ cd ..
Ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_2$ alias python=python3
Ivan@Ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_2$ python -i client.py
Enter large trapezoid base:
5
Enter small trapezoid base:
3
Enter trapezoid height:
10
Trapezoid area: 40.0
>>>
```

## Задание 3

Server:



The image shows a Python script named server.py. It imports necessary modules like socket, mimetypes, sys, email.parser, functools, and urllib.parse. It defines constants for MAX\_LINE and MAX\_HEADERS. A class MyHTTPServer is defined with methods \_\_init\_\_, serve\_forever, and serve\_client. The script uses a while loop in serve\_forever to accept and serve clients.

```
server.py
1 import socket
2 import mimetypes
3 import sys
4 from email.parser import Parser
5 from functools import lru_cache
6 from urllib.parse import parse_qs, urlparse
7
8 MAX_LINE = 64 * 1024
9 MAX_HEADERS = 100
10
11 class MyHTTPServer:
12     def __init__(self, host: str, port: int, server_name: str):
13         self.host = host
14         self.port = port
15         self.server_name = server_name
16         self._data: Dict[str, list[str]] = {}
17
18     def serve_forever(self):
19         serv_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM, proto=0)
20
21         try:
22             serv_sock.bind((self.host, self.port))
23             serv_sock.listen()
24
25             while True:
26                 clientsocket, _ = serv_sock.accept()
27                 try:
28                     self.serve_client(clientsocket)
29                 except Exception as e:
30                     print('Client serving failed:', e)
31             finally:
32                 serv_sock.close()
33
34     def serve_client(self, clientsocket):
35         try:
36             req = self.parse_request(clientsocket)
37             resp = self.handle_request(req)
38             self.send_response(clientsocket, resp)
39         except ConnectionResetError:
40             clientsocket = None
41         except Exception as e:
42             print('Client serving failed:', e)
43
44         if clientsocket:
45             clientsocket.close()
46
```

```

def parse_request(self, clientsocket):
    rfile = clientsocket.makefile('rb')
    raw = rfile.readline(MAX_LINE + 1)

    if len(raw) > MAX_LINE:
        raise Exception('Request line is too long')

    req_line = str(raw, 'iso-8859-1')
    req_line = req_line.rstrip('\r\n')
    words = req_line.split()
    if len(words) != 3:
        raise Exception('Malformed request line')

    method, target, ver = words

    if ver != 'HTTP/1.1':
        raise Exception('Unexpected HTTP version')

    headers = self.parse_headers(rfile)
    host = headers.get('Host')
    if not host:
        raise Exception('Bad request')
    if host not in (self._server_name, f'{self._server_name}:{self._port}'):
        raise Exception('Not found')

    return Request(method, target, ver, headers, rfile)

def parse_headers(self, rfile):
    headers = []
    while True:
        line = rfile.readline(MAX_LINE + 1)
        if len(line) > MAX_LINE:
            raise Exception('Header line is too long')

        if line in (b'\r\n', b'\n', b''):
            break

        headers.append(line)
        if len(headers) > MAX_LINE:
            raise Exception('Too many headers')
    sheaders = b''.join(headers).decode('iso-8859-1')

    return Parser().parsestr(sheaders)

```

```

def handle_request(self, req):
    if req.path == '/rating' and req.method == 'POST':
        return self.handle_post_grades(req)

    if req.path.startswith('/rating/') and req.method == 'GET':
        subject_name = req.path[len('/rating/'): ]
        return self.handle_get_rating(req, subject_name)

    raise Exception('Request not found')

def handle_post_grades(self, req):
    subject_name = req.query['subject_name'][0]
    grade = req.query['grade'][0]
    if subject_name in self._data.keys():
        self._data[subject_name].append(grade)
    else:
        new_subject = []
        new_subject.append(grade)
        self._data[subject_name] = new_subject

    return Response(204, 'Created')

def handle_get_rating(self, req, subject_name):
    if subject_name not in self._data.keys():
        raise Exception('Request not found')

    content_type = 'text/html; charset=utf-8'
    body = '<html><head></head><body>'
    ans = f'{subject_name}: '

    for grade in self._data[subject_name]:
        ans += grade + ' '

    body += f'<div>{ans}</div>'
    body += '</body></html>'
    body = body.encode('utf-8')
    headers = [('Content-Type', content_type), ('Content-Length', len(body))]
    return Response(200, 'OK', headers, body)

```

```

131
132     def send_response(self, clientsocket, response):
133         wfile = clientsocket.makefile('wb')
134         status_line = f'HTTP/1.1 {response.status} {response.reason}\r\n'
135         wfile.write(status_line.encode('iso-8859-1'))
136
137         if response.headers:
138             for (key, value) in response.headers:
139                 header_line = f'{key}: {value}\r\n'
140                 wfile.write(header_line.encode('iso-8859-1'))
141
142         wfile.write(b'\r\n')
143
144         if response.body:
145             wfile.write(response.body)
146
147         wfile.flush()
148         wfile.close()
149
150     class Request:
151         def __init__(self, method, target, version, headers, rfile):
152             self.method = method
153             self.target = target
154             self.version = version
155             self.headers = headers
156             self.rfile = rfile
157
158         @property
159         def path(self):
160             return self.url.path
161
162         @property
163         @lru_cache(maxsize=None)
164         def query(self):
165             return parse_qs(self.url.query)
166
167         @property
168         @lru_cache(maxsize=None)
169         def url(self):
170             return urlparse(self.target)
171
172     class Response:
173         def __init__(self, status, reason, headers=None, body=None):
174             self.status = status
175             self.reason = reason
176             self.headers = headers
177             self.body = body
178
179

```

```

179
180     if __name__ == '__main__':
181         host = sys.argv[1]
182         port = int(sys.argv[2])
183         name = sys.argv[3]
184         server = MyHTTPServer(host, port, name)
185
186         try:
187             server.serve_forever()
188         except KeyboardInterrupt:
189             pass

```

Пример работы:

```

omenko_Ivan/Lr1/task_3$ python server.py 127.0.0.1 14900 example.local
^Civan@ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/F
omenko_Ivan/Lr1/task_3$ python server.py 127.0.0.1 14900 example.local
^Civan@ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/F
omenko_Ivan/Lr1/task_3$

```

```

ivan@ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fom
enko_Ivan/Lr1/task_3$ nc localhost 14900
POST /rating?subject_name=Math&grade=5 HTTP/1.1
Host: example.local

HTTP/1.1 204 Created

^C
ivan@ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fom
enko_Ivan/Lr1/task_3$ nc localhost 14900
POST /rating?subject_name=Russian&grade=3 HTTP/1.1
Host: example.local

HTTP/1.1 204 Created

^C
ivan@ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fom
enko_Ivan/Lr1/task_3$ nc localhost 14900
POST /rating?subject_name=Math&grade=4 HTTP/1.1
Host: example.local

HTTP/1.1 204 Created

^C

```

```

ivan@ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fom
enko_Ivan/Lr1/task_3$ nc localhost 14900
GET /rating/Math HTTP/1.1
Host: example.local

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 60

<html><head></head><body><div>Math: 5 4 </div></body></html>
^C
ivan@ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fom
enko_Ivan/Lr1/task_3$ nc localhost 14900
GET /rating/Russian HTTP/1.1
Host: example.local

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 61

<html><head></head><body><div>Russian: 3 </div></body></html>
^C
ivan@ivan-VirtualBox:~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fom
enko_Ivan/Lr1/task_3$

```

## Задание 4



# Server

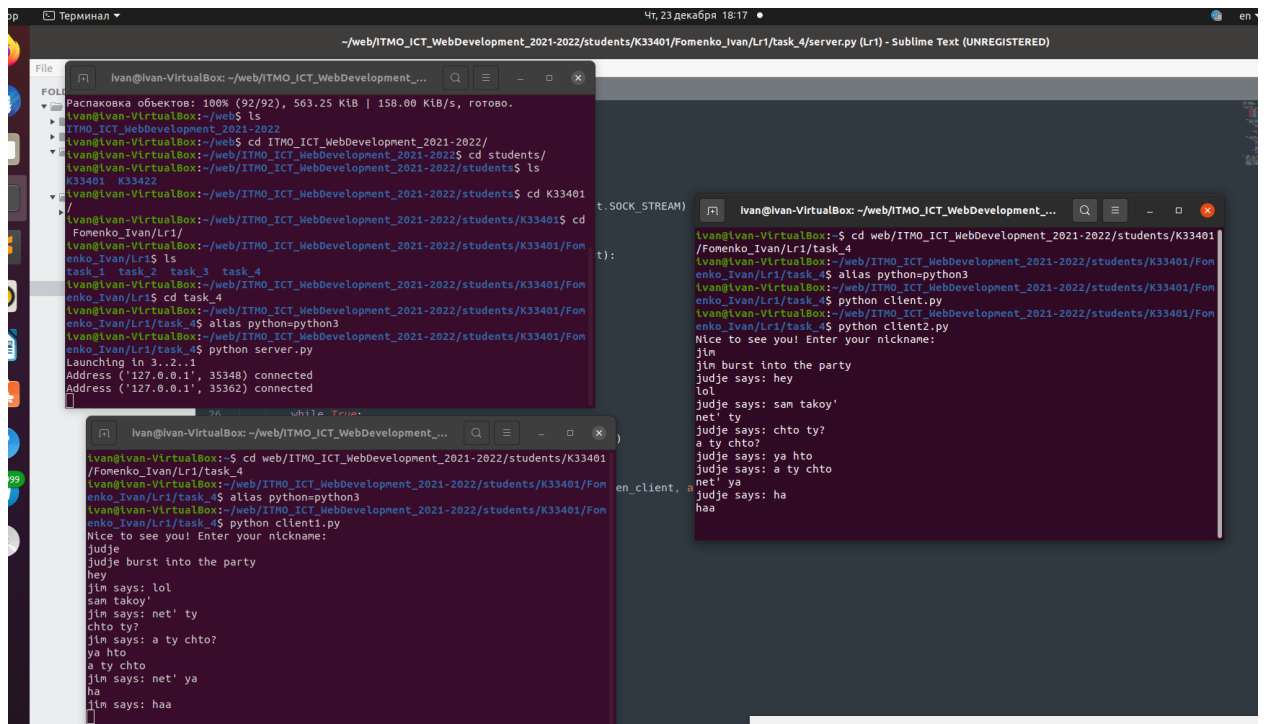
```
server.py — task_3 x client.py x client1.py x server.py — task_4 x
1 import socket
2 import threading
3
4 class Server:
5     def __init__(self, host: str, port: int):
6         self.host = host
7         self.port = port
8         self.users = []
9         self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10        self.sock.bind((self.host, self.port))
11        self.sock.listen(10)
12
13    def listen_client(self, client_socket: socket.socket):
14        client_socket.settimeout(60)
15        while True:
16            try:
17                data = client_socket.recv(1024)
18                for cliento in self.users.copy():
19                    if cliento != client_socket:
20                        cliento.send(data)
21            except KeyboardInterrupt:
22                self.sock.close()
23
24    def run(self):
25        while True:
26            try:
27                client_socket, address = self.sock.accept()
28                print(f"Address {address} connected")
29                if client_socket not in self.users:
30                    self.users.append(client_socket)
31                thread = threading.Thread(target=self.listen_client, args=(client_socket,))
32                thread.start()
33            except KeyboardInterrupt:
34                self.sock.close()
35                break
36
37if __name__ == '__main__':
38    host = "localhost"
39    port = 14900
40    print("Launching in 3..2..1")
41    server = Server(host, port)
42    thread1 = threading.Thread(target=server.run)
43    thread1.start()
```

# Client

```
1 import socket
2 import threading
3
4 def welcome():
5     print("Nice to see you! Enter your nickname: ")
6     nickname = input()
7     print(f"{nickname} burst into the party")
8     return nickname
9
10 class Client:
11     def __init__(self, host: str, port: int, username: str):
12         self.host = host
13         self.port = port
14         self.username = username
15         self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
16
17     def get_message(self):
18         while True:
19             try:
20                 data = self.sock.recv(1024)
21                 udata = data.decode("utf-8")
22                 print(udata)
23             except KeyboardInterrupt:
24                 self.sock.close()
25
26     def send_message(self):
27         while True:
28             try:
29                 data = input()
30                 self.sock.send(f"{self.username} says: {data}".encode())
31             except KeyboardInterrupt:
32                 self.sock.close()
33
34     def run(self):
35         self.sock.connect((self.host, self.port))
36         thread2, thread1 = threading.Thread(target=self.get_message), threading.Thread(target=self.send_message)
37         thread1.start()
38         thread2.start()
```

```
1 from client import Client, welcome
2
3 if __name__ == '__main__':
4     host = "localhost"
5     port = 14900
6     username = welcome()
7     client = Client(host, port, username)
8     client.run()
```

## Пример работы:



```
Ivan@ivan-VirtualBox: ~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_4/server.py (Lr1) - Sublime Text (UNREGISTERED)

Ivan@ivan-VirtualBox: ~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_4$ ls
task_1 task_2 task_3 task_4

Ivan@ivan-VirtualBox: ~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_4$ cd task_4
Ivan@ivan-VirtualBox: ~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_4$ alias python=python3
Ivan@ivan-VirtualBox: ~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_4$ python client1.py
Nice to see you! Enter your nickname:
jln
jln burst into the party
judge says: hey
lol
judge says: san takoy'
net' ty
judge says: chto ty?
a ty chto?
judge says: ya hto
judge says: a ty chto
net' ya
judge says: ha
haa

Ivan@ivan-VirtualBox: ~/web/ITMO_ICT_WebDevelopment_2021-2022/students/K33401/Fomenko_Ivan/Lr1/task_4$ python server.py
Launching in 3.2.2.1
Address ('127.0.0.1', 35348) connected
Address ('127.0.0.1', 35362) connected
```

## Выводы:

Поддерживать чат довольно интересно, а в HTTP запросах уже давно за нас сделали парсеры