Санкт-Петербургский национальный исследовательский университет ИТМО

Лабораторная работа №1
«Работа с сокетами»
По дисциплине
«Web-программирование»

Выполнила:
Моруга Э.С.
Группа:
К33422
Преподаватель:
Говоров А.И.

Санкт-Петербург
2021 г

Цель: овладеть практическими навыками и умениями реализации web-серверов и использования сокетов.

Практическое задание:

1. Реализовать клиентскую и серверную часть приложения. Клиент отсылает серверу сообщение «Hello, server». Сообщение должно отразиться на стороне сервера. Сервер в ответ отсылает клиенту сообщение «Hello, client». Сообщение должно отобразиться у клиента.
2. Реализовать клиентскую и серверную часть приложения. Клиент запрашивает у сервера выполнение математической операции, параметры, которые вводятся с клавиатуры. Сервер обрабатывает полученные данные и возвращает результат клиенту. Вариант: решение квадратного уравнения.
3. Сделать сервер, который может: принять и записать информацию о дисциплине и оценке по дисциплине; отдать информацию обо всех оценах по дсициплине в виде html-страницы.
4. Реализовать многопользовательский чат.

Задание 1.

Server.py

```
import socket

sock = socket.socket()
sock.bind(('', 9090))
sock.listen(1)
conn, addr = sock.accept()

print('connected:', addr)

while True:
    data = conn.recv(1024)
    if not data:
        break
    print(data)
    conn.send(b"Hello, client!")

conn.close()
```

client.py

```
import socket

sock = socket.socket()
sock.connect(('localhost', 9090))
sock.send('Hello, server!'.encode())

data = sock.recv(1024)
sock.close()
```

```
print(data)
```

Вывод:

```
PS C:\Users\Elya\Desktop\things\веб> python 1_server.py
connected: ('127.0.0.1', 59753)
b'Hello, server!'
PS C:\Users\Elya\Desktop\things\веб>
```

```
PS C:\Users\Elya\desktop\things\веб> python 1_client.py
b'Hello, client!'
PS C:\Users\Elya\desktop\things\веб>
```

Задание 2.

Server.py

```
import socket
import math

sock = socket.socket()
sock.bind(('', 9090))
sock.listen(1)
conn, addr = sock.accept()

print('connected:', addr[0])

data = conn.recv(1024)
data = data.decode()

_, url, ver = data.split(" ")
url, params = url.split("?")

params = params.split("&")
params_dct = {}
for param in params:
    p_lst = param.split("=")
    params_dct[p_lst[0]] = float(p_lst[1])


# решение квадратного уравнения

a = params_dct["a"]
b = params_dct["b"]
c = params_dct["c"]
discr = b ** 2 - 4 * a * c

if discr > 0:
    x1 = (-b + math.sqrt(discr)) / (2 * a)
    x2 = (-b - math.sqrt(discr)) / (2 * a)
    conn.send(("x1 = %.2f \nx2 = %.2f" % (x1, x2)).encode())
elif discr == 0:
    x = -b / (2 * a)
    conn.send(("x = %.2f" % x).encode())
```

```
else:
    conn.send("Корней нет".encode())

conn.close()
```

client.py

```
import socket

sock = socket.socket()
sock.connect(('localhost', 9090))

print("Введите коэффициенты для уравнения")
print("ax^2 + bx + c = 0:")
a = float(input("a = "))
b = float(input("b = "))
c = float(input("c = "))

sock.send(f'GET https://localhost?a={a}&b={b}&c={c} HTTP/1.1'.encode())

print("Ждем ответ от сервера...")
data = sock.recv(1024)

print(data.decode())

sock.close()
```

Вывод:

```
PS C:\Users\Elya\Desktop\things\веб> python 2_server.py
connected: 127.0.0.1
PS C:\Users\Elya\Desktop\things\веб>
```

```
PS C:\Users\Elya\desktop\things\веб> python 2_client.py
Введите коэффициенты для уравнения
ax^2 + bx + c = 0:
a = 3
b = 4
c = 5
Ждем ответ от сервера...
Корней нет
```

Задание 3.

Server.py

```
import socket
import sys

MAX_LINE = 64*1024

class MyHTTPServer:
  # Параметры сервера
    def __init__(self, host, port, name):
        self.host = host
        self.port = port
```

```
        self.name = name

    def serve_forever(self):
    # 1. Запуск сервера на сокете,
    # обработка входящих соединений
        serv_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM, proto=0)

        try:
            serv_sock.bind((self.host, self.port))
            serv_sock.listen(1)

            while True:
                conn, _ = serv_sock.accept()
                try:
                    self.serve_client(conn)
                except Exception as e:
                    print('Client serving failed', e)
        finally:
            serv_sock.close()

    def serve_client(self, conn):
        # 2. Обработка клиентского подключения
        try:
            req = self.parse_request(conn)
            resp = self.handle_request(req)
            self.send_response(conn, resp)
        except ConnectionResetError:
            conn = None
        except Exception as e:
            self.send_error(conn, e)

        if conn:
            conn.close()

    def parse_request(self, conn):
        # 3. функция для обработки заголовка http+запроса.
        # Python, сокет предоставляет возможность создать
        # вокруг него некоторую обертку, которая предоставляет
        # file object интерфейс.
        # Это дайте возможность построчно обработать запрос.
        # Заголовок всегда - первая строка. Первую строку
        # нужно разбить на 3 элемента
        # (метод + url + версия протокола).
        # URL необходимо разбить на адрес и параметры
        # (isu.ifmo.ru/pls/apex/f?p=2143 ,
        # где isu.ifmo.ru/pls/apex/f,
        # а p=2143 - параметр p со значением 2143)
        rfile = conn.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()            # разделяем по пробелу
        if len(words) != 3:                 # и ожидаем ровно 3 части
            raise Exception('Malformed request line')

        method, url, ver = words
        if ver != 'HTTP/1.1':
            raise Exception('Unexpected HTTP version')

        url, params = url.split("?")
        params = params.split("&")
        params_dct = {}
        for i in range(len(params)):
            # p = {}
            p_list = params[i].split("=")
            params_dct[p_list[0]] = p_list[1]
            # params[i] = p

        dct = {"method": method, "url": url,
               "params": params_dct, "ver": ver}
        # print(dct)
        return dct


    def handle_request(self, req):
        # 5. Функция для обработки url в соответствии
        # с нужным методом. В случае данной работы,
        # нужно будет создать набор условий,
        # который обрабатывает GET или POST запрос.
        # GET запрос должен возвращать данные.
        # POST запрос должен записывать данные на основе
        # переданных параметров.
        if req["method"] == "GET":
            with open("index.html", "rb") as f:
                data = f.read()

            # print(data.decode())
            headers = ['Content-Type: text/html\r\n',
                       f'Content-Length: {len(data)}\r\n']

            response = {"code": 200, "reason": "OK", "headers": headers, "body":
data}
            return response

        if req["method"] == "POST":
            # print(req["params"])
            params = req["params"]
            data = ""
            for param in params:
```

```
                st = f'\n\t\t\t<tr>\n\
                \t<td>{param}</td>\n\
                \t<td>{params[param]}</td>\n\
                </tr>'
                data += st

            # читаем текущий index.html
            with open("index.html", "r") as f:
                text = f.read()

            # находим последнюю запись
            if "</tr>" in text:
                ind = text.rindex("</tr>")+len("</tr>")
                new_text = text[:ind]
                new_text += data
                new_text += text[ind:]

            # записываем новые данные
            with open("index.html", "wb") as f:
                f.write(new_text.encode())

            response = {"code": 204, "reason": "Grade added"}

            return response


    def send_response(self, conn, response_data):
        # 6. Функция для отправки ответа.
        # Необходимо записать в соединение
        # status line вида
        # HTTP/1.1 <status_code> <reason>.
        # Затем, построчно записать заголовки
        # и пустую строку,
        # обозначающую конец секции заголовков.
        # print(response_data)
        wfile = conn.makefile('wb')
        status_line = f"HTTP/1.1 {response_data['code']} {response_data['reason']
}\r\n"
        wfile.write(status_line.encode('iso-8859-1'))

        if "headers" in response_data:
            for header in response_data["headers"]:
                wfile.write(header.encode('iso-8859-1'))

        wfile.write(b'\r\n')

        if "body" in response_data:
            wfile.write(response_data["body"])

        wfile.flush()
        wfile.close()
```

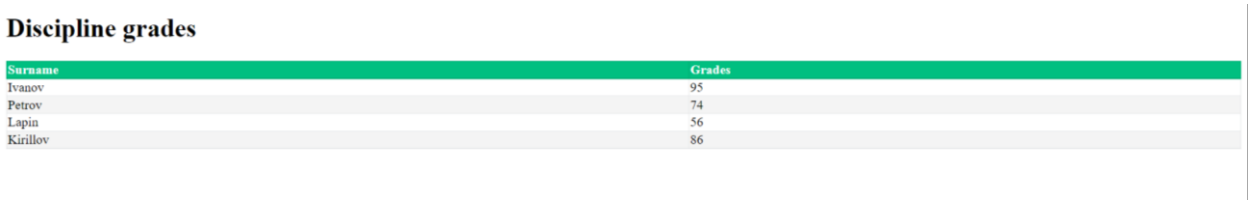
```
    def send_error(self, conn, e):
        print("send_error", e)
        try:
            body = e.encode('utf-8')
        except:
            status = 500
            reason = b'Internal Server Error'
            body = b'Internal Server Error'

            headers = [f'Content-Length: {len(body)}\r\n']
            resp = {"code": status, "reason": reason, "headers": headers, "body":
 body}

        self.send_response(conn, resp)


if __name__ == '__main__':
    host = "localhost"
    port = 9090
    name = "MyServer"

    serv = MyHTTPServer(host, port, name)
    try:
        serv.serve_forever()
    except KeyboardInterrupt:
        pass
```

client.py

```
import socket

sock = socket.socket()
sock.connect(('localhost', 9090))

GET_request = ["GET isu.ifmo.ru/pls/apex/f HTTP/1.1",
        "Host: example.local",
        "Accept: text/html",
        "User-Agent: Mozilla/5.0"]

POST_request = ["POST isu.ifmo.ru/pls/apex/f?Petrov=74 HTTP/1.1",
        "Host: example.local",
        "Accept: text/html",
        "User-Agent: Mozilla/5.0"]

sock.send("\r\n".join(POST_request).encode())

print("Ждем ответ от сервера...")
data = sock.recv(1024)
sock.close()
```

```
print(data.decode())
```

Вывод:



Discipline grades

Surname	Grades
Ivanov	95
Petrov	74
Lapin	56
Kirillov	86

Задание 4.

Server.py

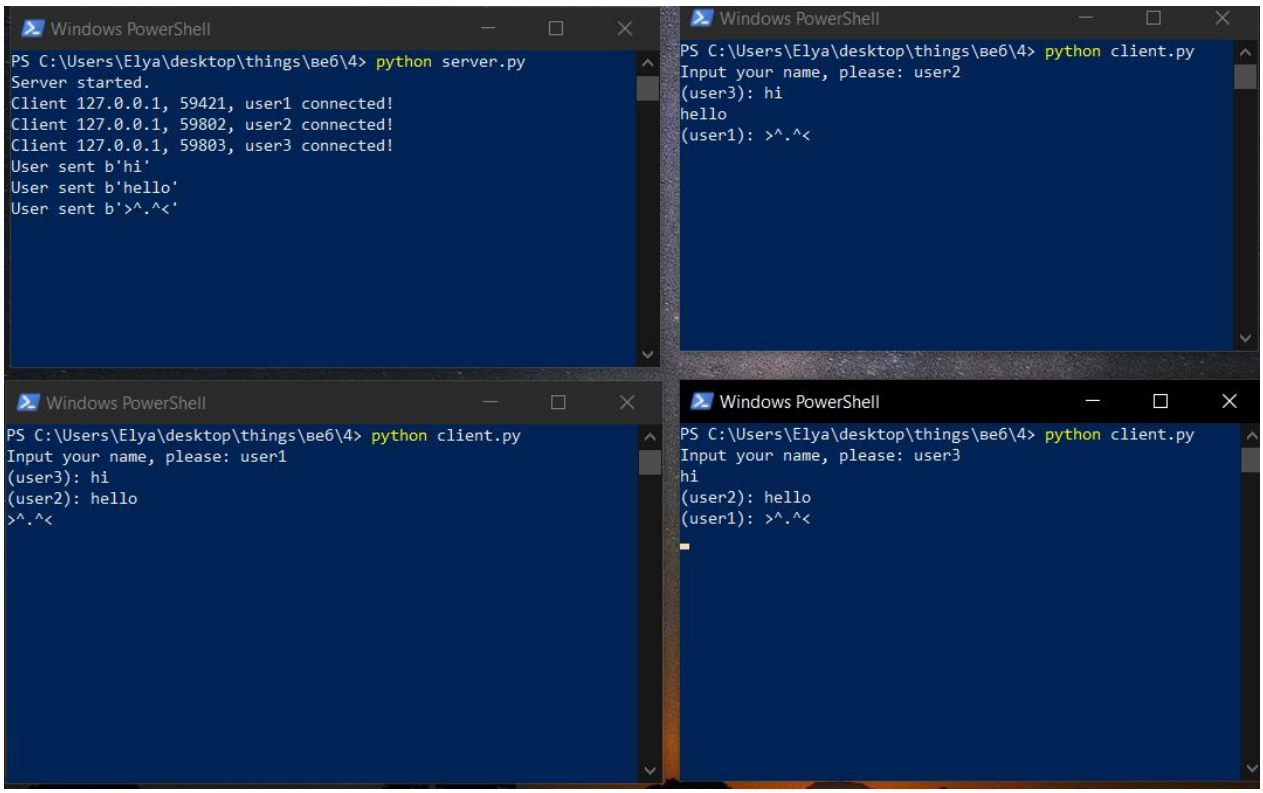
```
import socket
from threading import Thread

server = socket.socket(socket.AF_INET,
                       socket.SOCK_STREAM,
                       proto=0)
host = "localhost"
port = 9090

server.bind((host, port))
server.listen(10)
print("Server started.")

clients = []
names = []

def send_message(client_name, data):
    for i, client in enumerate(clients):
        if client_name != names[i]:
            client.send(f"({client_name}): ".encode() + data)

def listen_client(socket):
    while True:
        data = socket.recv(1024)

        client_name = names[clients.index(socket)]
        print(f"User sent {data}")
        send_message(client_name, data)

def server_accept():
    while True:
        socket, addr = server.accept()

        # получить имя клиента
        data = socket.recv(1024)
        names.append(data.decode())

        print(f"Client {addr[0]}, {addr[1]}, {data.decode()} connected!")
```

```
        clients.append(socket)
        listen_clients = Thread(target=listen_client, args=(socket,))
        listen_clients.start()


server_accept()
```

client.py

```
import socket, threading

client = socket.socket()
client.connect(('localhost', 9090))

name = input("Input your name, please: ")
client.send(name.encode())

def listen_server():
    while True:
        data = client.recv(1024)
        print(data.decode())

def send_to_server():
    listen_thread = threading.Thread(target=listen_server)
    listen_thread.start()

    while True:
        client.send(input().encode())

send_to_server()
```

Вывод:


Windows PowerShell
PS C:\Users\Elya\desktop\things\ве6\4> python server.py
Server started.
Client 127.0.0.1, 59421, user1 connected!
Client 127.0.0.1, 59802, user2 connected!
Client 127.0.0.1, 59803, user3 connected!
User sent b'hi'
User sent b'hello'
User sent b'>^.^<'
Windows PowerShell
PS C:\Users\Elya\desktop\things\ве6\4> python client.py
Input your name, please: user2
(user3): hi
hello
(user1): >^.^<
Windows PowerShell
PS C:\Users\Elya\desktop\things\ве6\4> python client.py
Input your name, please: user1
(user3): hi
(user2): hello
>^.^<
Windows PowerShell
PS C:\Users\Elya\desktop\things\ве6\4> python client.py
Input your name, please: user3
hi
(user2): hello
(user1): >^.^<