ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет «Инфокоммуникационных технологий»
Направление подготовки «09.03.03 Прикладная информатика»
Бакалаврская программа «Мобильные и сетевые технологии»

Лабораторная работа №1

по дисциплине «Веб программирование»

«Работа с сокетами»

Выполнил

_____ / Сулейманов Р. И., К33402
(подпись) (Фамилия И.О., группа)

Проверил

_____ / Говоров А. И.

Дата _____

Санкт-Петербург
2021

Цель работы

Овладеть практическими навыками использования сокетов.

Выполнение работы

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.

Файл task1_server.py

```
import socket

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

while True:
    data = conn.recv(16384)
    if not data:
        break
    udata = data.decode("utf-8")
    print(udata)
    conn.send(b'Hello, client \n')
```

Файл task1_client.py

```
import socket

sock = socket.socket()
sock.connect(('localhost', 9090))
sock.send(b"Hello, server")

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

udata = data.decode("utf-8")
print(udata)
```

Задание 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: Поиск площади трапеции.

Файл task2_server.py

```
import socket
import pickle

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

while True:
    data = conn.recv(4096)
    data_variable = pickle.loads(data)
    if not data:
        break
    print('a = ', data_variable['a'])
    print('b = ', data_variable['b'])
    print('h = ', data_variable['h'])
    summ =
str(int(data_variable['h'])*(int(data_variable['a'])+int(data_variable['b']))/2)

    conn.send(summ.encode())

conn.close()
```

Файл task2_client.py

```
import socket
import pickle

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

print('Поиск площади трапеции.')
obj = {
    'a': input('Введите первое оснвоание a: '),
    'b': input('Введите второе оснвоание b: '),
    'h': input('Введите расстояние между основаниями - высоту h: ')
}
data = pickle.dumps(obj)
if data:
    sock.send(data)

data = sock.recv(4096)
sock.close()
udata = data.decode("utf-8")
print(udata)
```

Задание 3.

Необходимо написать простой web-сервер для обработки GET и POST http запросов средствами Python и библиотеки socket.

Задание: 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.

Файл task3_server.py

```
import socket
import sys
import time


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

    # 1. Запуск сервера на сокете, обработка входящих соединений
    def serve_forever(self):
        # создание сокета
        serv_sock = socket.socket(
            socket.AF_INET,  # задамем семейство протоколов 'Интернет' (INET)
            socket.SOCK_STREAM,  # задаем тип передачи данных 'потоковый' (TCP)
            proto=0)  # выбираем протокол 'по умолчанию' для TCP, т.е. IP
        try:
            serv_sock.bind((self._host, self._port)) # привязываем созданный сокет
            serv_sock.listen() # переводим сокет в состояние ожидания подключения
            # чтения и записи данных в клиентский сокет:
            while True:
                # Бесконечно обрабатываем входящие подключения
```

```
                client, addr = serv_sock.accept()
                self.serve_client(client)
        finally:
            # закрываем сокет
            serv_sock.close()

    # 2. Обработка клиентского подключения
    def serve_client(self, client):
        try:
            # парсинг заголовка
            method, url, version, parametrs, headers = self.parse_request(client)
            # получив запрос обработаем и отправим все клиенту
            self.handle_request(method, url, headers, client, parametrs)
        except ConnectionResetError:
            client = None
        if client:
            client.close()

    # 3. функция для обработки заголовка http+запроса.Первую строку нужно разбить на 3 элемента  (метод + url + версия протокола). URL необходимо разбить на адрес и параметры
    def parse_request(self, client):
        #получим заголовки запроса
        rfile = client.makefile('rb')
        method, url, version, parametrs= None,None,None,None
        for line in rfile: #сделал способом, который понимаю
            words = line.decode('utf-8').split()
            if len(words) != 3:  # и ожидаем ровно 3 части
                raise Exception('Malformed request line')
            method, url, version = words
            # проверим, есть ли параметры
            if ("?" in url) :
                url, parametrs = url.split('?')
            break
        #теперь прочитаем остальные строки в другой функции
        headers = self.parse_headers(rfile)
        return method, url, version, parametrs, headers


    # 4. Функция для обработки headers. Необходимо прочитать все заголовки после первой строки до появления пустой строки и сохранить их в массив.
    def parse_headers(self, rfile):
        headers = []
        for line in rfile:
            if line in (b'\r\n', b'\n', b''):
                # завершаем чтение заголовков
                break
            headers.append(line)
        return headers

    # 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.
    def handle_request(self, method, url, headers, client, parametrs):
        if url == "/":
            if method == "GET":
                pass
            if method == "POST" and parametrs != None:
                data = parametrs.split('&')
                Discipline.append(data[0].split('=')[1])
                Mark.append(data[1].split('=')[1])
        self.send_response(client)
        return

    # 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.
    def send_response(self, client):
        resp = "HTTP/1.1 200 OK\n\n"
        with open('index.html', 'r') as f:
            # запишем в ответ весь респонс
            for line in f:
                if('<div id="in_serv">\n'== line):
                    # если имеется значени в массиве, выводим их
                    for i in range(len(Discipline)):
                        resp +='<p id="in_serv"> Discipline:' + Discipline[i] +', Mark:
'+ Mark[i] + '</p>'
                resp += line
        client.send(resp.encode('UTF-8'))


if __name__ == '__main__':
    host = 'localhost'
    port = 9090
    name = 'example'
    serv = MyHTTPServer(host, port, name)
    Discipline = []
    Mark = []
    try:
        serv.serve_forever()
    except KeyboardInterrupt:
        pass
```

Файл task3_index/html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<p>Hello world</p>
<form name="mark_input" id="form" method="post" accept-charset="UTF-8">
    <div>
        <input id="Discipline" name="Discipline" required type="text"
placeholder="Discipline">
    </div>
    <p></p>
    <div>
        <input id="Mark" name="Mark" required type="number" placeholder="Mark"
value="value" >
    </div>
    <p></p>
    <div id="cont-button"></div>
</form>
<div id="in_serv">
</div>
<script>
   function sub(e) {
        let Discipline = document.getElementById('Discipline');
        let Mark = document.getElementById('Mark');
        fetch('/?Discipline=' + Discipline.value+ '&Mark=' + Mark.value, {
            method: 'POST'
        })
    }

    let button = document.createElement('button');
    button.onclick=(sub)
```

```
    button.innerHTML='Submit'

    let cont = document.getElementById('cont-button');
    document.getElementById('cont-button').append(button);
</script>
</body>
</html>
```

Задание 4.

Реализовать двухпользовательский или многопользовательский чат.

Файл task4_server.py

```
import socket
from threading import Thread

server = socket.socket(
    socket.AF_INET,
    socket.SOCK_STREAM
)
server.bind(('127.0.0.1', 9090))
server.listen(10)

users = []

# отправляем новое сообщение всем юзерам
def send_all(mes):
    for user in users:
        user.send(mes)

# случшаем юзеров
def user_listen(user):
    print(users)
    print('user listen')
    while True:
        data = user.recv(2048)  # не раскодируем байты, так как их будем дальше
отправлять
        if not data:
            # Клиент отключился
            break
        send_all(data)


# запусск сервера и добавление юзеров
def start_server():
    while True:
        user_sock, addr = server.accept()  # блокирующий поток
        print(f'User<{addr[0]}> con')

        users.append(user_sock)
        user_thread = Thread(target=user_listen,
                             args=[user_sock])  # запятая - показывает питону, что
список неизменяемый
        user_thread.start()  # запуск потока



if __name__ == '__main__':
    start_server()
```

Файл task4_client.py

```
import socket
from threading import Thread

client = socket.socket(
    socket.AF_INET,
    socket.SOCK_STREAM
)
client.connect(('127.0.0.1', 9090))

def listen():
    while True:
        data = client.recv(2048)
```

```
        print(data.decode('utf-8'))

def send():
    lis_thread = Thread(target=listen) #слушаем в потоке
    lis_thread.start()
    while True:
        client.send((nick + ': ' + input()).encode('utf-8'))

if __name__== '__main__':
    nick = input('Введите ник: ')
    send()
```

Вывод

В результате выполненной работы были изучены основы клиент-серверного взаимодействия, работа с сокетами, протокол HTTP, потоки в Python.