ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет «Инфокоммуникационных технологий»
Направление подготовки «45.03.04 Интеллектуальные системы в гуманитарной сфере»

О Т Ч Е Т

Лабораторная работа №1

Тема задания: Работа с сокетами

Выполнил:
Студент Самощенков А.А. K33421
(Фамилия И.О.) номер группы

Проверил:
Преподаватель Говоров А.И.
(Фамилия И.О)

Санкт-Петербург
2020

Ход работы.

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.

Клиент:

```python
import socket


class Client:
    def __init__(self, port):
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.sock.connect(('localhost', port))

    def send(self, message):
        print(f'{message}')
        self.sock.sendall(bytes(f'{message}','utf-8'))
        data = self.sock.recv(1024)
        data.decode('utf-8')
        print(data)
        self.sock.close()


if __name__ == '__main__':
    cl = Client(12345)
    cl.send('Hello, server')
```

Сервер:

```python
import socket


class Server:
    def __init__(self, port, clients):
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.sock.bind(('localhost', port))
        self.sock.listen(clients)

    def run(self, message):
        while True:
            client_sock, client_addr = self.sock.accept()
            print(f'{client_addr} has just connected')
            while True:
                data = client_sock.recv(1024)
                if not data:
                    print(f'{client_addr} has just lost connection')
                    break
                client_sock.sendall(bytes(f'Server says "{message}" to {client_addr}', 'utf-8'))


if __name__ == '__main__':
    serv = Server(12345, 1)
    serv.run('Hello, Client')
```

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.

Клиент:

```python
import socket


class Client:
    def __init__(self, port):
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.sock.connect(('localhost', port))

    def send(self, a, b, c):
        self.sock.sendall(bytes(f'{a} {b} {c}','utf-8'))
        data = self.sock.recv(1024)
        data.decode('utf-8')
        print(data)


if __name__ == '__main__':
    cl = Client(12345)
    cl.send(a=1, b=-8, c=12)
    cl.send(a=5, b=3, c=7)
    cl.send(a=1, b=-6, c=9)
```

Сервер:

```python
import socket
import math


class Server:
    def __init__(self, port, clients):
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.sock.bind(('localhost', port))
        self.sock.listen(clients)

    def run(self):
        while True:
            client_sock, client_addr = self.sock.accept()
            print(f'{client_addr} has just connected')
            while True:
                data = client_sock.recv(1024)
                if not data:
                    print(f'{client_addr} has just lost connection')
                    break
                data.decode('utf-8')
                x1, x2 = Server.solve(data)
                client_sock.sendall(bytes(f'x1 = {x1}"; x2 = {x2}', 'utf-8'))

    @staticmethod
    def solve(data):
        coef = data.split()
        a = int(coef[0])
        b = int(coef[1])
        c = int(coef[2])
        D = b**2-4*a*c
        if D>0:
            x1 = (-b+math.sqrt(D))/(2*a)
            x2 = (-b-math.sqrt(D))/(2*a)
            return x1, x2
        elif D==0:
            x1 = -b/(2*a)
            return x1, None
        else:
            return None, None

if __name__ == '__main__':
    serv = Server(12345, 1)
    serv.run()
```

Реализовать серверную часть приложения. Клиент подключается к серверу. В ответ клиент получает http-сообщение, содержащее html-страницу, которую сервер подгружает из файла index.html.

Клиент:

```python
import socket


class Client:
    def __init__(self, port):
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.sock.connect(('localhost', port))

    def send(self):
        self.sock.sendall(bytes(f'.','utf-8'))
        data = self.sock.recv(1024)
        data.decode('utf-8')
        print(data)


if __name__ == '__main__':
    cl = Client(12345)
    cl.send()
```

Индекс:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1 align="center">Hello, World!</h1>
</body>
</html>
```

Сервер:

```python
import socket


class Server:
    def __init__(self, port, clients):
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.sock.bind(('localhost', port))
        self.sock.listen(clients)

    def run(self):
        while True:
            client_sock, client_addr = self.sock.accept()
            while True:
                data = client_sock.recv(4096)
                if not data:
                    print(f'{client_addr} has just lost connection')
                    break
                with open('index.html', 'r') as file:
                    html = file.read()
                    client_sock.sendall(bytes(f'HTTP/1.1 200 OK\nContent-Type: text/html\n\n{html}', 'utf-8'))

if __name__ == '__main__':
    serv = Server(12345, 1)
    serv.run()
```

Реализовать двухпользовательский или многопользовательский чат. Реализация многопользовательского часа позволяет получить максимальное количество баллов.

Три одинаковых клиента:

```python
import socket
from threading import Thread

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(('localhost', 12367))

nick = input('Who are you?\n')
print("Welcome to the chat!")


def send_message():
    while True:
        text = input("")
        message = (f'{nick}: {text}')
        sock.send(message.encode('utf-8'))


def receive_message():
    while True:
        data = sock.recv(4096)
        print(data.decode('utf-8'))


send_thread = Thread(target=send_message)
get_thread = Thread(target=receive_message)
send_thread.start()
get_thread.start()
```

Сервер:

```python
import threading
import socket

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('localhost', 12367))
server.listen()

clients = []
addresses = []

def chat(client):
    while True:
        message = client.recv(4096)
        print(message)
        for client in clients:
            client.send(message)


def new_connection():
    while True:
        client, address = server.accept()
        print(f'{address} has connected')

        client.send('You are connected!'.encode('utf-8'))

        clients.append(client)
        addresses.append(address)

        thread = threading.Thread(target=chat, args=(client,))
        thread.start()

new_connection()
```

Работу скриптов можно посмотреть по следующей ссылке:
https://www.youtube.com/watch?v=3U6tsIclzxY

Заключение:
Научился работать с модулем socket в Python.