Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе № 1

по дисциплине «**Web-программирование**»

Автор: Аль-Мошки Исмаил Абдулвхаб

Факультет: ИКТ

Группа:К33391

Преподаватель: Говоров Антон Игоревич

Санкт-Петербург, 2023

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

Оборудование: компьютерный класс**.**

Программное обеспечение: Python 2.7-3.6, библиотеки Python: sys, socket.

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.
Обязательно использовать библиотеку socket
Реализовать с помощью протокола UDP

Client:

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.sendto(bytes("Hello, server", "utf-8"),(socket.gethostname(),1234))
```

Server:

```
import socket

FORMAT = 'utf-8'
PORT = 8888
HOST = socket.gethostbyname(socket.gethostname())
ADDRESS = (HOST, PORT)

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDRESS)

server.listen(1)

while True:
    client_conn, address = server.accept()
    client_conn.recv(1024)

    response_type = "HTTP/1.0 200 OK\n"
    headers =  "text/html\n\n"

    body = '''
    <html>
        <body>
            <H1> Heloo World! <H1>
        </body>
    </html>
    '''

    response = response_type + headers + body
```

```
    client_conn.send(response.encode(FORMAT))

    client_conn.close()
```

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. Варианты:
a. Теорема Пифагора
b. Решение квадратного уравнения.
c. Поиск площади трапеции.
d. Поиск площади параллелограмма.
Вариант выбирается в соответствии с порядковым номером в журнале. Пятый
студент получает вариант 1 и т.д.
Обязательно использовать библиотеку socket
Реализовать с помощью протокола TCP

Client:

```
import socket

SERVER = socket.gethostbyname(socket.gethostname())
PORT = 1234
FORMAT = "utf-8"
ADDRESS = (SERVER, PORT)
HEADER = 64


def send_massage(msg):
    encoded_msg = msg.encode(FORMAT)
    client.send(encoded_msg)
    print(client.recv(HEADER).decode(FORMAT))


client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(ADDRESS)
send_massage("0 0 6")
```

Server:

```
import math
import socket, threading

SERVER = socket.gethostbyname(socket.gethostname())
PORT = 1234
FORMAT = "utf-8"
ADDRESS = (SERVER, PORT)
HEADER = 64

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDRESS)
```

```
def start_server():
    server.listen()
    print(f"[starting] Server is running on {SERVER}")
    while True:
        client_socket, address = server.accept()
        # print(f"[Connected] client with address {address} connected")
        thread = threading.Thread(target=handle_client, args=(client_socket,
address))
        thread.start()


def handle_client(client_socket, address):
    print(f"[NEW CONNECTION]new connection from address {address}")

    msg = client_socket.recv(HEADER).decode(FORMAT)
    print(f"{address}  {msg}")
    numbers_array = msg.split(" ")

    client_socket.send(getequationroot(numbers_array).encode(FORMAT))


def getequationroot(array):
    # reformat data type [cast to integer]
    for i in range(len(array)):
        array[i] = int(array[i])
    A = array[0]
    B = array[1]
    C = array[2]

    delta = pow(B,2) - 4*A*C

    if A == 0:
        if B == 0:
            if C == 0:
                return "The equation is always valid"
            else:
               return "The equation is always invalid"
        else:
            return f'answer is {-C / B}'



    if delta < 0:
        return f"There is no real root for this equation"

    elif delta == 0:
        answer = (-B / 2 * A)
        return f"the answer is {answer}"

    else:
        first_root = (-B + math.sqrt(delta))/2*A
        second_root = (-B - math.sqrt(delta)) / 2 * A
        return f"Solutions are ({first_root}, {second_root})"



start_server()
```

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

Обязательно использовать библиотеку socket

Client:

```
import socket, webbrowser

FORMAT = 'utf-8'
PORT = 8888
HOST = socket.gethostbyname(socket.gethostname())
ADDRESS = (HOST, PORT)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:



    s.connect(ADDRESS)
    headers = [
        'GET / HTTP/1.1',
        'Host: www.geeksforgeeks.org',
        'Connection: keep-alive',
        'Accept: text/text',
        '\n'
    ]
    content = "\n".join(headers)
    s.send(content.encode(FORMAT))
    result = s.recv(14233)
    print(result.decode(FORMAT))
    ready  = result.decode()
    webbrowser.open_new_tab(ready)
```

Server:

```
import socket

FORMAT = 'utf-8'
PORT = 8888
HOST = socket.gethostbyname(socket.gethostname())
ADDRESS = (HOST, PORT)

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDRESS)

server.listen()

while True:
    client_conn, address = server.accept()
    client_conn.recv(1024)

    response_type = "HTTP/1.0 200 OK\n"
    headers =  "text/html\n\n"

    body = '''
    <html>
        <body>
            <H1> Heloo World! <H1>
        </body>
    </html>
    '''
```

```
    response = response_type + headers + body
    client_conn.send(response.encode(FORMAT))

    client_conn.close()
```

4. Реализовать двухпользовательский или многопользовательский чат. Реализация
многопользовательского часа позволяет получить максимальное количество баллов.
Обязательно использовать библиотеку socket

Client:

```
import socket

PORT = 5050
SERVER = socket.gethostbyname(socket.gethostname())
HEAD = 1024
FORMAT = 'utf-8'
ADDRESS = (SERVER, PORT)

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(ADDRESS)

nickname_request = client.recv(HEAD)
client.send(bytes("Ais ", FORMAT))
massage_request = client.recv(HEAD)
client.send(bytes("Sub Eman", FORMAT))

while True:
    massage = client.recv(HEAD).decode(FORMAT)
    print(massage)
```

Server:

```
import socket
import threading

PORT = 5050
SERVER = socket.gethostbyname(socket.gethostname())
HEAD = 1024
FORMAT = 'utf-8'
ADDRESS = (SERVER, PORT)

clients = []
nicknames = []

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDRESS)


def start_server():
    print(f'[LAUNCHING] server is listening on {SERVER}')
    server.listen()
    while True:
        conn, add = server.accept()
        thread = threading.Thread(target=handle_client, args=(conn, add))
        thread.start()
```

```
def handle_client(conn, add):
    while True:
        conn.send("NICKNAME".encode(FORMAT))
        nickname = conn.recv(HEAD).decode(FORMAT)
        print(f'{nickname} joined the chat')
        if nickname:
            nicknames.append(nickname)
            clients.append(conn)
            conn.send("MASSAGE".encode(FORMAT))
            massage = nickname + " : " + conn.recv(HEAD).decode(FORMAT)
            send_live(massage)


def send_live(massage):
    for client in clients:
        client.send(massage.encode(FORMAT))


start_server()
```

Выводы:

Использование различных протоколов подключения в значительной степени зависит от типа данных и уровня безопасности передачи данных