

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

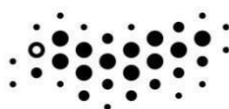
Отчёт
по лабораторной работе №1
по дисциплине “Web-Программирование”

Автор: Панкова Кристина Сергеевна

Факультет: Инфокоммуникационные технологии

Группа: К33421

Преподаватель: Говоров Антон Игоревич



УНИВЕРСИТЕТ ИТМО

Санкт-Петербург 2023

Ход работы:

Практическое задание 1.

Реализовать клиентскую и серверную часть приложения. Клиент отправляет серверу сообщение «Hello, server». Сообщение должно отразиться на стороне сервера. Сервер в ответ отправляет клиенту сообщение «Hello, client». Сообщение должно отобразиться у клиента.

Код:

server.py

```
import socket

HOST = '127.0.0.1' # Локальный адрес
PORT = 65432      # Выбранный порт

# Создаем сокет
with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
    s.bind((HOST, PORT)) # Привязываем сокет к адресу и порту
    print('Server started...')
    while True:
        data, addr = s.recvfrom(1024) # Получаем данные от клиента
        print('Received from client:', data.decode())
        s.sendto(b'Hello, client', addr) # Отправляем ответ клиенту
```

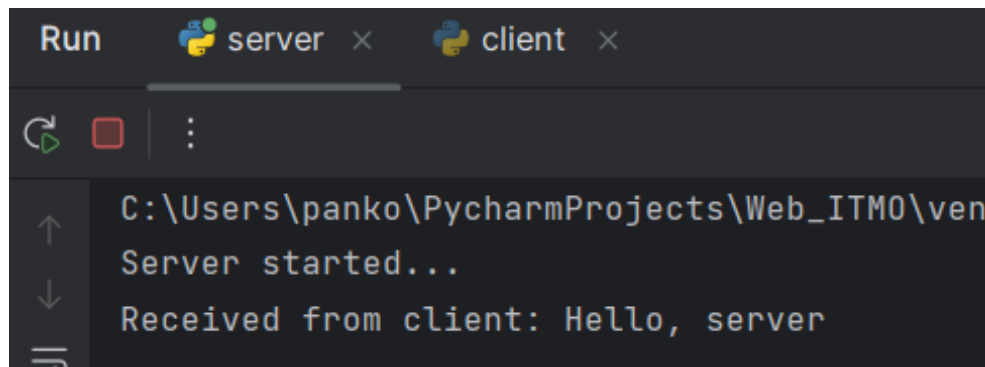
client.py

```
import socket

HOST = '127.0.0.1' # Локальный адрес сервера
PORT = 65432      # Выбранный порт

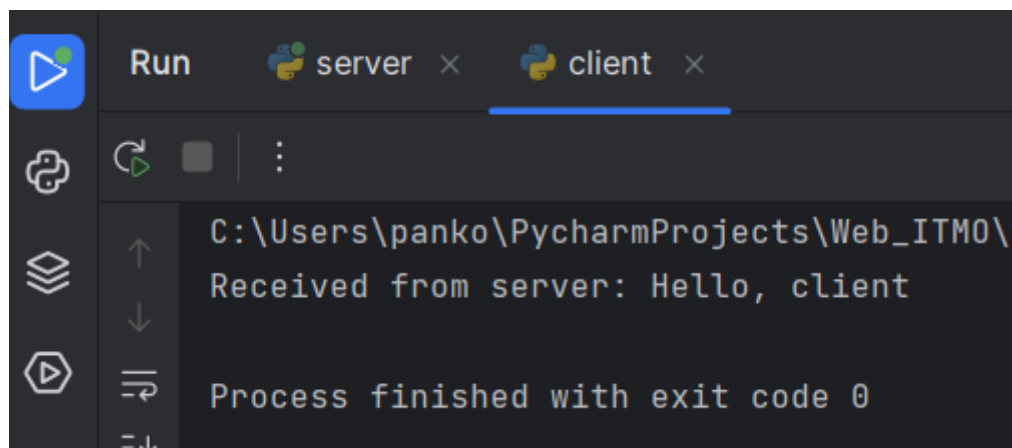
# Создаем сокет и подключаемся к серверу
with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
    s.sendto(b'Hello, server', (HOST, PORT)) # Отправляем сообщение серверу
    data, addr = s.recvfrom(1024)           # Получаем ответ от сервера
    print('Received from server:', data.decode())
```

Результат работы:

The screenshot shows the PyCharm Run console with two tabs: 'server' and 'client'. The 'server' tab is active. The console output shows the path 'C:\Users\panko\PycharmProjects\Web_ITM0\ven', the message 'Server started...', and 'Received from client: Hello, server'.

```
Run server x client x
C:\Users\panko\PycharmProjects\Web_ITM0\ven
Server started...
Received from client: Hello, server
```

Рисунок 1 - Результат работы со стороны сервера

The screenshot shows the PyCharm Run console with two tabs: 'server' and 'client'. The 'client' tab is active. The console output shows the path 'C:\Users\panko\PycharmProjects\Web_ITM0\ven', the message 'Received from server: Hello, client', and 'Process finished with exit code 0'.

```
Run server x client x
C:\Users\panko\PycharmProjects\Web_ITM0\ven
Received from server: Hello, client
Process finished with exit code 0
```

Рисунок 2 - Результат работы со стороны клиента

Практическое 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: Теорема Пифагора

Код:

server.py

```
import socket

HOST = '127.0.0.1'
PORT = 5000

def get_input():
```

```

    a = input('Enter the first cathetus (or leave blank): ')
    b = input('Enter the second cathetus (or leave blank): ')
    c = input('Enter the hypotenuse (or leave blank): ')
    return f'{a},{b},{c}'

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    while True:
        data = get_input()
        s.sendall(data.encode())
        result = s.recv(1024).decode()
        print(f'Result: {result}')

```

client.py

```

import socket

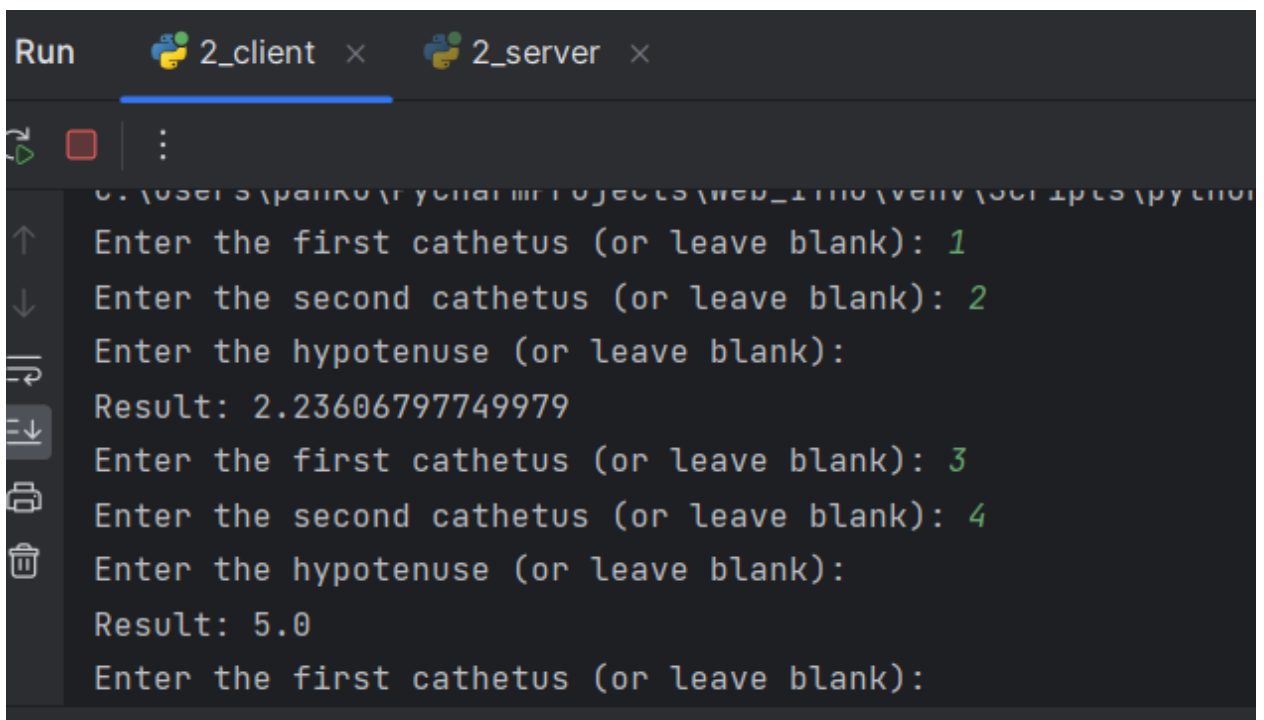
HOST = '127.0.0.1'
PORT = 5000

def get_input():
    a = input('Enter the first cathetus (or leave blank): ')
    b = input('Enter the second cathetus (or leave blank): ')
    c = input('Enter the hypotenuse (or leave blank): ')
    return f'{a},{b},{c}'

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    while True:
        data = get_input()
        s.sendall(data.encode())
        result = s.recv(1024).decode()
        print(f'Result: {result}')

```

Результаты:



```
Run 2_client x 2_server x
C:\Users\franko\PycharmProjects\web_1\venv\Scripts\python
Enter the first cathetus (or leave blank): 1
Enter the second cathetus (or leave blank): 2
Enter the hypotenuse (or leave blank):
Result: 2.23606797749979
Enter the first cathetus (or leave blank): 3
Enter the second cathetus (or leave blank): 4
Enter the hypotenuse (or leave blank):
Result: 5.0
Enter the first cathetus (or leave blank):
```

Рисунок 3 - Результат работы со стороны клиента

Практическое задание 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.

Код:

server.py



```
import socket

TCP_IP = '127.0.0.1'
TCP_PORT = 5005
BUFFER_SIZE = 1024

def load_index_page():
    with open("index.html", "r") as f:
        return f.read()
```

```

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

sock.bind((TCP_IP, TCP_PORT))

sock.listen(1)


while True:

    conn, addr = sock.accept()

    print('Connection address:', addr)


    data = conn.recv(BUFFER_SIZE)

    if not data: break

    response = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n" +
load_index_page()

    conn.send(response.encode())


conn.close()

```

client.py

```

import socket


TCP_IP = '127.0.0.1'

TCP_PORT = 5005

BUFFER_SIZE = 1024


sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

sock.connect((TCP_IP, TCP_PORT))


message = "GET / HTTP/1.1\r\nHost: example.com\r\n\r\n"

sock.send(message.encode())


data = sock.recv(BUFFER_SIZE)

print(data.decode())


sock.close()

```

Результаты работы:

I love web programming

Рисунок 4 - Загруженная html-страница

Практическое задание 4.

Реализовать многопользовательский чат.

Код:

server.py

```
import socket
import threading

TCP_IP = '127.0.0.1'
TCP_PORT = 5005
BUFFER_SIZE = 1024

names = {}
clients = []

def handle_client(client_socket, addr):
    while True:
        try:
            data = client_socket.recv(BUFFER_SIZE)

            if not data:
                del names[addr]

                print(f"Connection closed with {addr}")

                break

            message = data.decode()
```

```

        print(message)

        print('\n')

        if 'Hello! My name is ' in message:
            names[addr] = message.split(' ')[4]

            for client in clients:
                if client != client_socket:
                    client.send(message.encode())

            except (ConnectionResetError, ConnectionAbortedError) as e:
                print(f"Connection closed with {names[addr]}")

                del names[addr]

                break

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind((TCP_IP, TCP_PORT))
sock.listen(10)

print(f"Server started on {TCP_IP}:{TCP_PORT}")

while True:
    client_socket, addr = sock.accept()

    print(f"A new connection! Here's some info about them:")

    clients.append(client_socket)

    client_thread = threading.Thread(target=handle_client, args=(client_socket,
addr))

    client_thread.start()

```

client.py

```

import socket

import threading

TCP_IP = '127.0.0.1'

TCP_PORT = 5005

BUFFER_SIZE = 1024

def receive_messages(sock):

```



```

while True:

    data = sock.recv(BUFFER_SIZE)

    if not data:

        break

    message = data.decode()

    print(message)

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

sock.connect((TCP_IP, TCP_PORT))

name = input("Enter your name: ")

message = "Hello! My name is " + name

sock.send(message.encode())

receive_thread = threading.Thread(target=receive_messages, args=(sock,))

receive_thread.start()

while True:

    inp = input()

    message = name + ": " + inp

    if inp == 'exit':

        break

    sock.send(message.encode())

sock.close()

```

Результат:

```
Server started on 127.0.0.1:5005
A new connection! Here's some info about them:
Hello! My name is Kristina

Kristina: Anyone here?

A new connection! Here's some info about them:
Hello! My name is Alice

Alice: nice chat

A new connection! Here's some info about them:
Hello! My name is Misha

Misha: hey

Misha: actually bye

Connection closed with Misha
Kristina: Thats weird. I'll go too

Connection closed with Kristina
```

Рисунок 5 - Результат работы чата

Практическое задание 5.

Код:

server.py

```
import socket
```

```
class MyHTTPServer:

    def __init__(self, host, port):

        self.host = host

        self.port = port

        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        self.sock.bind((self.host, self.port))

        self.sock.listen(1)

        self.grades = {}

    def serve_forever(self):

        print(f"Server started...")

        while True:

            client, addr = self.sock.accept()

            self.serve_client(client)

    def serve_client(self, client):

        data = client.recv(1024).decode("utf-8")

        self.parse_request(client, data)

    def parse_request(self, client, data):

        lines = data.split("\n")

        if len(lines) < 1:

            raise HTTPError(400, 'Bad request')

        method, url, ver = lines[0].split()

        if ver != 'HTTP/1.1':

            raise HTTPError(505, 'HTTP Version Not Supported')

        params = url[2:].replace('=', ':').split('&')

        rq = {}

        for item in params:

            item = item.split(':')

            rq[item[0]] = item[1]

        self.handle_request(client, method, rq)
```

```

def handle_request(self, client, method, params):
    if method == "GET":
        self.send_response(client, 200, "OK", self.grades_to_html())
    elif method == "POST":
        discipline = params["discipline"]
        grade = params["grade"]
        if discipline in self.grades.keys():
            self.grades[discipline] = f"{self.grades[discipline]}, {grade}"
            self.send_response(client, 200, "OK", "Success!")
        else:
            self.grades[discipline] = grade
            self.send_response(client, 200, "OK", "Success!")
        else:
            raise HTTPError(404, 'Not Found', 'Method must be in ("GET",
"POST") ')

def send_response(self, client, code, reason, body):
    response = f"HTTP/1.1 {code} {reason}\nContent-Type:
text/html\n\n{body}"

    client.send(response.encode("utf-8"))
    client.close()

def grades_to_html(self):
    page = "<html><body><ul>"
    for discipline, grade in self.grades.items():
        page += f"<li>{discipline}: {grade}"
    page += "</ul></body></html>"
    return page

class HTTPError(Exception):
    def __init__(self, status, reason, body=None):
        super()
        self.status = status
        self.reason = reason
        self.body = body

```

```

if __name__ == '__main__':

    host = 'localhost'
    port = 8080
    serv = MyHTTPServer(host, port)

    try:
        serv.serve_forever()
    except KeyboardInterrupt:
        serv.sock.close()

```

Результат работы:

The screenshot displays a web browser interface for a REST client. The top bar shows a POST request to `http://localhost:8080/?discipline=math&grade=1` with a 'Send' button. Below this, the 'Params' tab is active, showing a table of query parameters:

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	discipline	math			
<input checked="" type="checkbox"/>	grade	1			
	Key	Value	Description		

Below the params table, the 'Body' tab is active, showing a response of `200 OK` with a status of `7 ms` and a size of `52 B`. The response body is displayed as `Success!`.

Рисунок 6 - POST-запрос к серверу

POST http://localhost:8080/?discipline=math&grade=10 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	discipline	math			
<input checked="" type="checkbox"/>	grade	10			
	Key	Value	Description		

Body Cookies Headers (1) Test Results 200 OK 6 ms 52 B Save as Example

Pretty Raw Preview Visualize HTML

1 Success!

Рисунок 7 - POST-запрос к серверу

POST http://localhost:8080/?discipline=PE&grade=100 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	discipline	PE			
<input checked="" type="checkbox"/>	grade	100			
	Key	Value	Description		

Body Cookies Headers (1) Test Results 200 OK 6 ms 52 B Save as Example

Pretty Raw Preview Visualize HTML

1 Success!

Рисунок 8 - POST-запрос к серверу

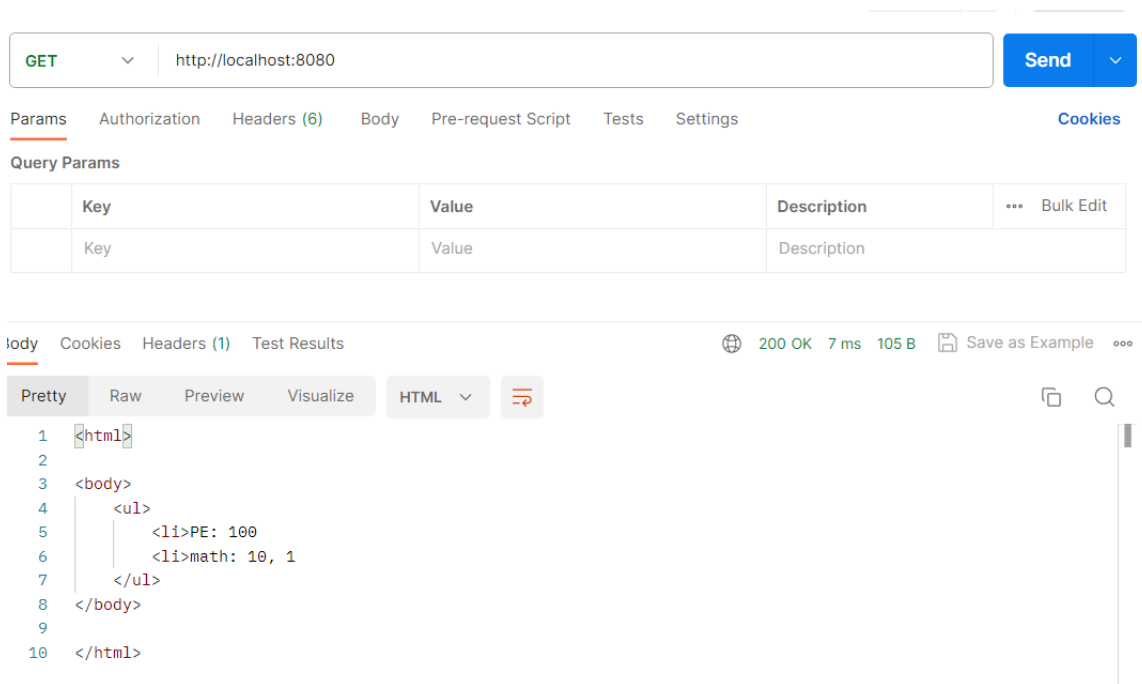


Рисунок 9 - GET-запрос к серверу

Вывод:

В лабораторной работе №1 были изучены способы реализации web-серверов и использования сокетов.