Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Факультет Инфокоммуникационных Технологий

**Web-программирование**

**Лабораторная работа 6**

Выполнил

Носов А.Д.

Проверила

Марченко Е. В.

Санкт-Петербург, 2025

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

**Задачи:**

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.

Номер в списке 17, соответственно вариант a. Теорема Пифагора.

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

**Ход работы**

**Задание 1**

Для выполнения лабораторной работы использовалась стандартная библиотека Python – socket, предназначенная для сетевого взаимодействия.

Файл server.py реализует серверную часть приложения, использующую сокеты для сетевого взаимодействия. При запуске сервер начинает прослушивать указанный порт и ожидает подключения от клиента. После установления соединения он принимает сообщение от клиента, отображает его в консоли и отправляет в ответ строку «Hello, client».

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 1 – Файл server.py

Файл client.py реализует клиентскую часть, которая устанавливает соединение с сервером по указанному адресу и порту. Клиент отправляет серверу сообщение «Hello, server», после чего ожидает ответ и выводит полученную строку от сервера в консоль.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2 – Файл client.py

Для запуска необходимо сначала запустить server.py в одном окне терминала, чтобы сервер начал прослушивать соединения. Затем в другом окне терминала запустить client.py. В результате на стороне сервера отобразится принятое сообщение от клиента, а клиент получит и выведет ответ сервера.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3 – Получение сообщения сервером

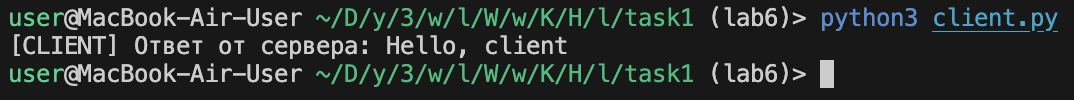


Рисунок 4 – Получение сообщения клиентом

**Задание 2**

Во втором задании было реализовано клиент-серверное, позволяющее выполнять вычисления по теореме Пифагора. Серверная часть (server.py) ожидает подключения клиента, принимает два значения катетов, вычисляет гипотенузу по формуле ​ с помощью модуля math и отправляет результат обратно клиенту.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 5 – Файл server.py

Клиентская часть (client.py) запрашивает у пользователя значения двух катетов через консольный ввод, устанавливает соединение с сервером, передаёт введённые значения, получает результат вычисления и отображает его пользователю.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 6 – Файл client.py

Для запуска необходимо открыть два терминала. В первом терминале запускается server.py, ожидающий подключения. Во втором — client.py, который отправляет значения катетов и получает результат. При успешном взаимодействии клиент выводит, например: «Гипотенуза c = 5.00», если были введены значения 3 и 4.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7 – Получение ответа клиентом от сервера

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 8 – Получение данных сервером от клиента

**Задание 3**

В данном задании был реализован простейший HTTP-сервер на Python, использующий библиотеку socket. Сервер слушает на порту 8080, принимает подключения от клиентов и отправляет им HTML-страницу в ответ. При получении запроса сервер открывает файл index.html, читает его содержимое и формирует HTTP-ответ с этим контентом. Если файл не найден, возвращается сообщение об ошибке в формате HTML.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 9 – Файл server.py

Была написана простейшая HTML-страница в файле index.html.

Рисунок 10 – Файл index.html

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 11 – Ответ от сервера



Рисунок 12 – Отображение HTML-страницы в браузере

**Задание 4**

В данном задании был реализован многопользовательский чат. Чтобы обеспечить многозадачности как на стороне сервера, так и на стороне клиента использовалась библиотека threading. В сервере она необходима для того, чтобы обрабатывать несколько клиентов одновременно. Каждый новый клиент, который подключается к серверу, получает свой собственный поток, что позволяет серверу параллельно обслуживать несколько пользователей без блокировки.

На стороне клиента threading используется для параллельного получения и отправки сообщений. Один поток отвечает за непрерывное получение сообщений от сервера и их вывод на экран, а другой поток позволяет пользователю вводить сообщения и отправлять их на сервер. Это обеспечивает удобное и плавное взаимодействие с чатом, где оба процесса происходят одновременно, не мешая друг другу.

Файл server.py реализует серверную часть чата. При запуске сервера создается сокет, который слушает порт 8080 и ожидает подключения клиентов. Каждый новый клиент добавляется в список clients, после чего для него запускается отдельный поток, обрабатывающий его сообщения. Функция broadcast отвечает за рассылку сообщений всем подключенным клиентам, кроме отправителя. Когда клиент отправляет сообщение, оно получает форматирование с именем пользователя, и это сообщение отправляется всем остальным участникам чата. В случае отключения клиента его сокет удаляется из списка активных соединений.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 13 – Файл server.py

Файл client.py представляет собой клиентскую часть чата. Он устанавливает соединение с сервером и позволяет пользователю отправлять и получать сообщения. Для получения сообщений от сервера используется отдельный поток, который непрерывно слушает сообщения и выводит их на экран. В другом потоке пользователь может вводить текстовые сообщения, которые отправляются на сервер. Функция receive\_messages отвечает за получение сообщений с сервера, а функция send\_message — за отправку сообщений от клиента на сервер. Важно, что оба этих процесса выполняются параллельно, что позволяет пользователю одновременно читать и писать сообщения.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 14 – Файл client.py

Протестируем работу многопользовательского чата.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 15 – Запуск сервера

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, снимок экрана

Автоматически созданное описание

Рисунок 16 – Добавление нового клиента и пример чата

**Вывод**

В результате выполнения лабораторной работы были получены практические навыкы и умения реализации web-серверов и использования сокетов, созданы несколько клиент-серверных приложений, а также реализован многопользовательский чат.