**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Кафедра “фундаментальная информатика и информационные технологии”**

**отчет**

**по домашнему заданию**

**по дисциплине «Функциональное программирование»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 22Б16 |  | Олизько С.С. |
| Преподаватель |  | Киямов Ж.У. |

**Санкт-Петербург**

**2023 г.**

**Оглавление**

1. Цель работы
2. Задача
3. Теоритическая часть
4. Алгоритм метода
5. Описание программы
6. Рекомендации по использованию
7. Вывод

**Цель:**

Разработка асинхронного чат-сервера, способного обслуживать множество клиентов одновременно и позволяющего им обмениваться сообщениями в режиме реального времени.

**Задача:**

Разработать асинхронный чат-сервер, способного обслуживать множество клиентов одновременно и позволяющего им обмениваться сообщениями в режиме реального времени с использованием библиотеки asyncio.

**Теоретическая часть:**

### Основы сокетов:

Сокеты представляют собой программные интерфейсы для обеспечения обмена данными между процессами, работающими на одном или разных узлах сети. Они позволяют устанавливать соединения, передавать данные и обеспечивать взаимодействие между клиентом и сервером. В вашем случае, вы используете библиотеку asyncio для создания асинхронных сокетов, что обеспечивает эффективное обслуживание множества клиентов без блокировки основного потока.

### Асинхронность в Python и библиотека asyncio:

Асинхронность в Python позволяет создавать асинхронные функции (корутины), которые выполняются независимо от основного потока выполнения. Благодаря этому можно эффективно обрабатывать множество одновременных задач, таких как подключения к клиентам. Библиотека asyncio предоставляет инструменты для работы с асинхронными сокетами и корутинами, обеспечивая эффективное и удобное программирование в асинхронной парадигме.

**Алгоритм метода:**

1. Сервер создает асинхронный сокет и начинает прослушивание на заданном порту.
2. Клиент подключается к серверу и отправляет свое имя.
3. Сервер приветствует клиента и предлагает ввести имя.
4. Клиент и сервер обмениваются сообщениями, используя асинхронные корутины.

**Описание программы:**

**Сервер (server.py):**

|  |  |  |
| --- | --- | --- |
| **Имя функции** | **Тип возвращаемого значения** | **Описание функции** |
| client\_loop | None | Асинхронная корутина, обрабатывающая сообщения от клиента. |
| handle\_client | None | Асинхронная корутина, управляющая подключением нового клиента. |
| join\_room | None | Асинхронная корутина для входа клиента в чат-комнату. |
| create\_room | None | Асинхронная корутина для создания новой чат-комнаты. |
| leave\_room | None | Асинхронная корутина для выхода клиента из текущей чат-комнаты. |
| add\_user\_to\_chat | None | Асинхронная корутина для добавления пользователя в чат. |
| show\_current\_chat | str | Асинхронная корутина, возвращающая текущую чат-комнату клиента. |
| broadcast\_message | None | Асинхронная корутина для рассылки сообщений в чат-комнате. |
| main | None | Асинхронная функция, запускающая сервер и обслуживающая подключения. |
| start\_async\_loop | None | Функция для запуска асинхронного цикла в отдельном потоке. |

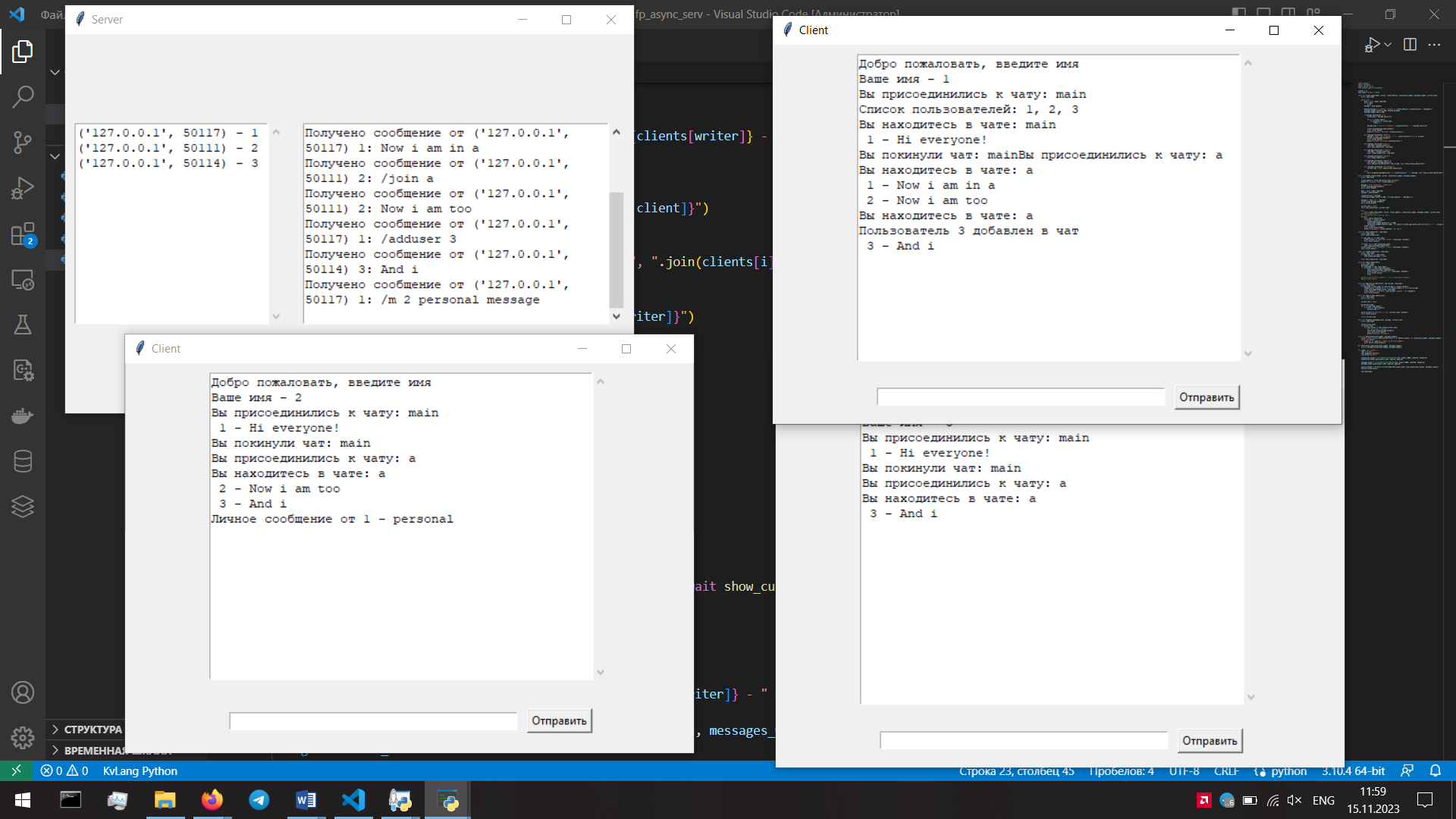
**Клиент (client.py):**

|  |  |  |
| --- | --- | --- |
| **Имя функции** | **Тип возвращаемого значения** | **Описание функции** |
| receive\_messages | None | Асинхронная корутина для приема сообщений от сервера и отображения их в текстовом виджете. |
| send\_messages | None | Асинхронная корутина для отправки сообщений на сервер. |
| get\_input | str | Функция для получения ввода от пользователя в асинхронном стиле. |
| on\_send\_button\_click | None | Функция для обработки нажатия кнопки отправки сообщения. |
| main | None | Асинхронная функция, устанавливающая подключение к серверу и запускающая корутины для приема и отправки сообщений. |
| start\_async\_loop | None | Функция для запуска асинхронного цикла в отдельном потоке. |

**Рекомендации по использованию:**

1. Запустите сервер, указав порт и IP-адрес, если необходимо изменить значения по умолчанию.
2. Запустите клиентское приложение, чтобы подключиться к серверу.
3. Введите имя пользователя.
4. Выберите команды для управления чатом, например, /join, /create, /leave.
5. Отправляйте сообщения и общайтесь в режиме реального времени.

**Контрольный пример:**



**Вывод:**

Предоставленный код реализует асинхронный чат-сервер с базовыми функциями управления подключениями, чат-комнатами и обмена сообщениями. Код также включает простого клиента для взаимодействия с сервером.

**Список литературы:**

* [Документация по библиотеке asyncio](https://docs.python.org/3/library/asyncio.html)
* [Документация по библиотеке tkinter](https://docs.python.org/3/library/tkinter.html)

**GitHub:**

https://github.com/StephanOlizko/fp\_homework