

Когда пользователь отправляет команду /start, вызывается функция cmd\_start. Бот отвечает приветственным сообщением, в котором объясняет, как использовать команду /search для поиска книг.

Когда пользователь вводит команду /search <название книги>, вызывается функция cmd\_search.

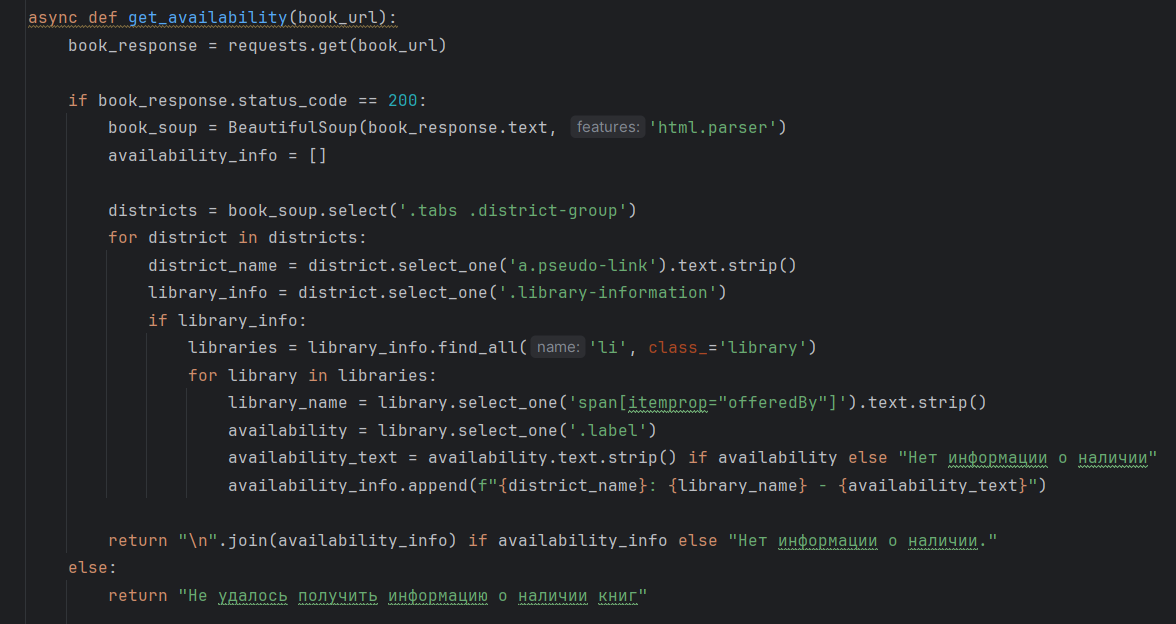
Внутри этой функции происходит следующее:

**Разделение текста**: С помощью message.text.split(maxsplit=1) текст сообщения разбивается на части. Первая часть — это команда (/search), а вторая — название книги, которое пользователь хочет найти.

**Проверка ввода**: Функция проверяет, указал ли пользователь название книги. Если длина списка query меньше 2 (то есть название не указано), бот отправляет сообщение с просьбой указать название книги и завершает выполнение функции с помощью return.

**Поиск книги**: Если название указано, бот отвечает сообщением о том, что он ищет книги и вызывает асинхронную функцию get\_books(query[1]), передавая название книги в качестве аргумента.

**Получение и отправка результатов**: После завершения поиска (в функции get\_books), бот получает информацию о найденных книгах и отправляет ее пользователю.



**Запрос к URL книги**: book\_response = requests.get(book\_url) — здесь выполняется HTTP GET-запрос к указанному URL книги. Эта строка получает HTML-код страницы книги.

**Проверка статуса ответа**: if book\_response.status\_code == 200: — проверяется, успешен ли запрос. Код 200 означает, что запрос выполнен успешно. Если код отличается, функция вернет сообщение об ошибке.

**Парсинг HTML**: book\_soup = BeautifulSoup(book\_response.text, 'html.parser') — HTML-код страницы преобразуется в объект BeautifulSoup, который позволяет легко извлекать данные из HTML-документа.

**Сбор информации о наличии**: availability\_info = [] — создается пустой список для хранения информации о наличии книг в разных библиотеках.

**Поиск районов**: districts = book\_soup.select('.tabs .district-group') — с помощью CSS-селектора выбираются элементы, содержащие информацию о районах (группах библиотек).

**Цикл по районам**: for district in districts: — перебираются все найденные районы.

district\_name = district.select\_one('a.pseudo-link').text.strip() — извлекается название района, находя элемент с классом pseudo-link.

library\_info = district.select\_one('.library-information') — ищется информация о библиотеках в данном районе.

**Проверка наличия информации о библиотеках**:

if library\_info: — если информация о библиотеках найдена, выполняется следующий шаг.

libraries = library\_info.find\_all('li', class\_='library') — собираются все библиотеки в этом районе.

**Цикл по библиотекам**: for library in libraries: — перебираются все библиотеки в районе.

library\_name = library.select\_one('span[itemprop="offeredBy"]').text.strip() — извлекается название библиотеки.

availability = library.select\_one('.label') — ищется информация о наличии книги в библиотеке.

availability\_text = availability.text.strip() if availability else "Нет информации о наличии" — если информация о наличии найдена, она извлекается; в противном случае устанавливается текст "Нет информации о наличии".

**Сбор информации о наличии**: availability\_info.append(f"{district\_name}: {library\_name} - {availability\_text}") — формируется строка с информацией о районе, библиотеке и наличии книги, которая добавляется в список availability\_info.

**Возврат информации**: return "\n".join(availability\_info) if availability\_info else "Нет информации о наличии." — если список availability\_info не пустой, он преобразуется в строку с разделением по новой строке и возвращается. Если список пуст, возвращается сообщение "Нет информации о наличии."

**Обработка ошибок**: Если запрос к URL не удался (т.е. код состояния не равен 200), функция возвращает строку "Не удалось получить информацию о наличии книг". Это позволяет пользователю понять, что возникла проблема с получением данных, возможно, из-за недоступности сайта или неправильного URL.