1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт компьютерных наук и кибербезопасности
5. **Кафедра «Высшая школа кибербезопасности»**

**ЛАБОРАТОРНАЯ РАБОТА № 2**

1. **Извлечение и обработка данных из стороннего web-сервера с помощью Python-приложения**
2. по дисциплине «Цифровая культура»
3. Выполнил
4. студент гр. 5131001/30003 Шевчук Н.Е.

<*подпись*>

Руководитель

ассистент ВШК ИКНК Иванов М.С.

<*подпись*>

1. Санкт-Петербург
2. 2025

**Оглавление**

[1 Цель работы 3](#_Toc194240655)

[2 Задачи 3](#_Toc194240656)

[3 Ход работы 4](#_Toc194240657)

[3.1 Анализ сетевого взаимодействия с помощью утилиты 4](#_Toc194240658)

[3.2 Анализ конечных точек API 6](#_Toc194240659)

[3.3 Пример GET-запроса: 6](#_Toc194240660)

[3.4 Реализация программы 7](#_Toc194240661)

[4 Результаты 8](#_Toc194240662)

[5 Ответы на контрольные вопросы 9](#_Toc194240663)

[6 Вывод 11](#_Toc194240664)

[Приложение 1 12](#_Toc194240665)

# Цель работы

Получить навыки работы с протоколом HTTP: построить запросы к удаленному веб-серверу для получения содержимого веб-страницы. Получить навыки обработки и извлечения данных из HTML-страницы при помощи Python-библиотек.

# Задачи

* + - 1. Перехватить экземпляр запроса к веб-сайту <https://ruz.spbstu.ru>. Изучить заголовки запроса и описать их.
      2. Включив фильтрацию результатов поиска (по номеру группы/по преподавателю), изучить то, как меняется URL запроса. Сделать выводы о параметрах в URL и описать их.
      3. Изучить HTTP-трафик, перехваченный при взаимодействии с целевым веб-сервером. Составить список конечных точек API, которые необходимы для извлечения расписания с заданным фильтром (по номеру группы/по преподавателю) и описать их.
      4. Описать последовательность и логику запросов к API, которые необходимы для извлечения расписания.
      5. С помощью библиотеки **requests** выполнить GET-запрос к одной из конечных точек API. Проанализировать принцип передачи параметров в запросе, описать формат данных, содержащихся в ответе.
      6. Используя библиотеку **requests**, извлечь из тела ответа расписание в формате: неделя (четная/нечетная), название предмета, дата, время, аудитория, преподаватель. Последовательность запросов должна воспроизводить изученную на предыдущих шагах логику работы веб-сайта.
      7. Помимо текстового вывода необходимо представить расписание в виде графика. По оси абсцисс – день недели, по оси ординат – количество занятий. Для построения графика необходимо использовать библиотеку **matplotlib**.

# Ход работы

## Анализ сетевого взаимодействия с помощью утилиты

Для анализа http-заголовках запроса к веб-сайту использовалась утилита burpsuite с помощью которой был перехвачен экземпляр запроса к веб-сайту <https://ruz.spbstu.ru>. Были изучены заголовки запроса. Пример перехвата запроса на веб-сайт представлен на рисунке 1. На листинге 1 представлен пример тела ответа.

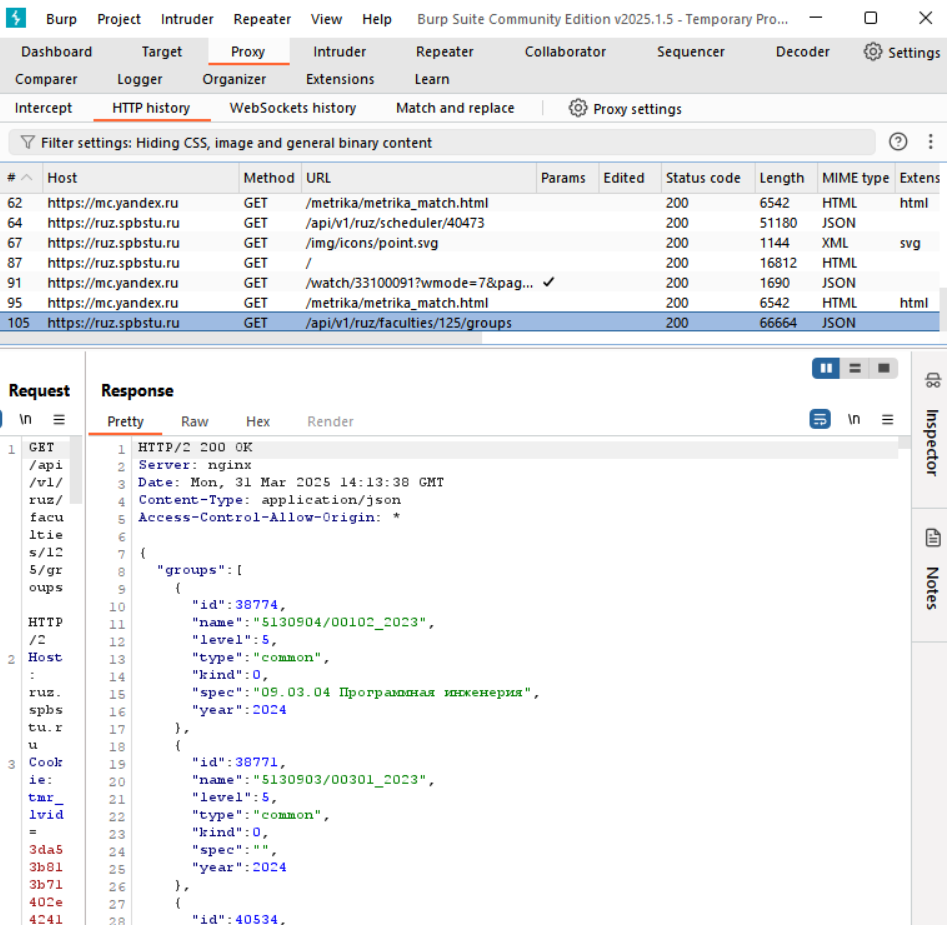


Рисунок 1 – Пример перехвата экземпляра запроса к веб-сайту

Листинг 1 – пример тела Response

|  |
| --- |
| HTTP/2 200 OK #строка статуса  Server: nginx #HTTP-заголовок  Date: Mon, 31 Mar 2025 14:13:38 GMT #HTTP-заголовок  Content-Type: application/json #HTTP-заголовок  Access-Control-Allow-Origin: \* #HTTP-заголовок  #Тело ответа  {  "groups": [  {  "id": 38774,  "name": "5130904/00102\_2023",  "level": 5,  "type": "common",  "kind": 0,  "spec": "09.03.04 Программная инженерия",  "year": 2024  }, |

Далее была включена фильтрация по группе 5131001/30003. URL запроса меняется. Прошлый URL со значением “group” говорит о том, что на данной странице мы можем выбрать группу. В новом URL “schedule” на сайте можно наблюдать расписание выбранной группы на конкретную неделю, которая имеет id 40473. Наглядно изменения URL можно посмотреть на рисунке 2.

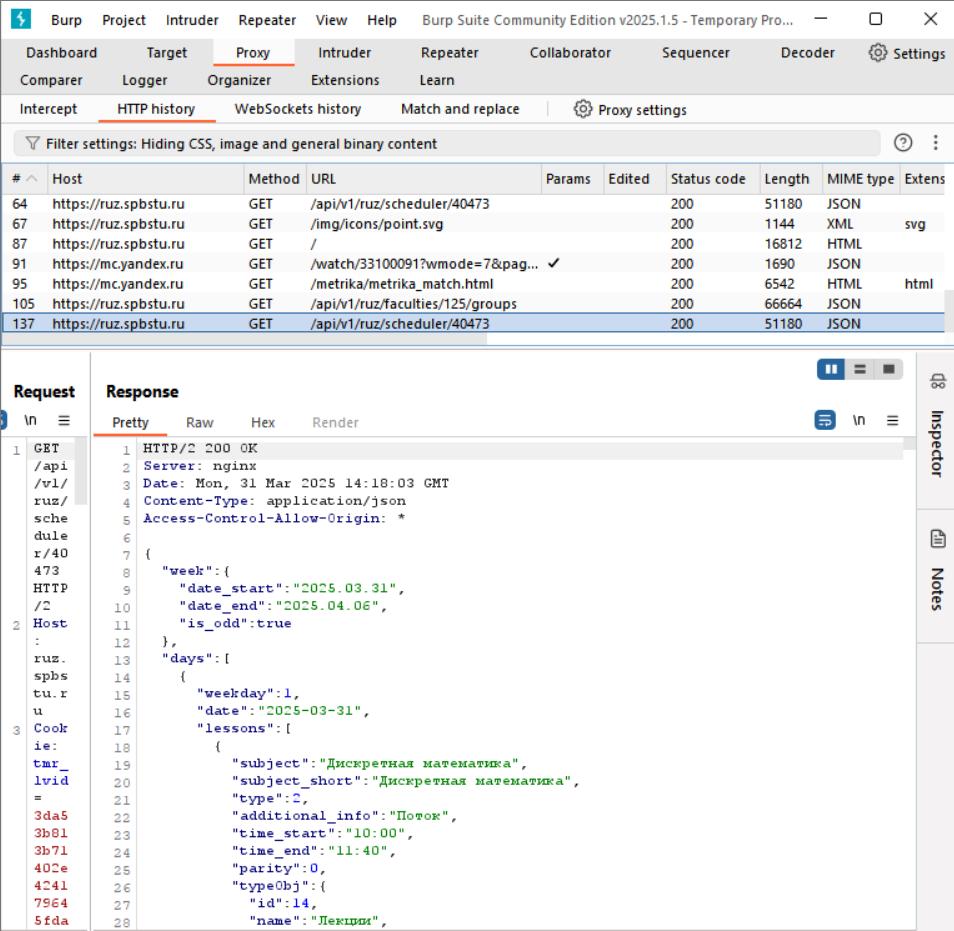


Рисунок 2 – Изменение URL

## Анализ конечных точек API

Конечная точка API — это уникальный URL-адрес, по которому сервер предоставляет доступ к определённым данным или функциям.

Каждая конечная точка:

* Имеет свой путь (например, /api/v1/ruz/scheduler/{group\_id}).
* Принимает параметры (например, ?date=2024-05-20).
* Возвращает данные в определённом формате (обычно JSON).

Для выполнения задания были использованы следующие конечные точки API:

* https://ruz.spbstu.ru/api/v1/ruz/faculties.
* https://ruz.spbstu.ru/api/v1/ruz/faculties/{faculty\_id}/groups.

## Пример GET-запроса

Для того, чтобы выполнить запрос GET, используется requests.get().

Первым битом информации, которую можно получить от объекта ответа Response, является код состояния запроса к серверу. Код состояния информирует о статусе запроса. Например, статус 200 OK означает, что запрос был успешным, а статус 404 NOT FOUND означает, что искомый ресурс не найден.

Используя свойство объекта response.status\_code, можно получить доступ к коду состояния ответа, который вернул удаленный сервер.

Ответ на успешный запрос GET часто содержит в сообщении некоторую ценную информацию, известную так же, как полезное содержимое. Используя атрибуты и методы объекта Response, можно просматривать его содержимое в различных форматах.

В листинге 2 представлен пример GET-запроса к конкретной конечной точки API – расписания для группы 5131001/30003 на конкретную дату.

Листинг 2 – Пример GET-запроса

|  |
| --- |
| import requests  group\_id = 40473 date = "2025-03-20" headers = {  "User-Agent": "Mozilla/5.0",  "Accept": "application/json" }  url = f"https://ruz.spbstu.ru/api/v1/ruz/scheduler/{group\_id}?date={date}" response = requests.get(url, headers=headers)  if response.status\_code == 200:  schedule = response.json()  print("Расписание группы 40473:")  print(schedule) else:  print(f"Ошибка: {response.status\_code}") |

## Реализация программы

Далее была реализована программа для извлечения из тела ответа HTTP-запроса расписания для запрашиваемой пользователем группой. Для этого сначала были подключены следующие библиотеки:

* Requests – для выполнения HTTP-запросов.
* Matplotlib – для построения графиков.
* Datetime – для работы с датами.
* Sys – для ошибочного завершения программы.

Затем, реализована общая логика работы программы с помощью следующих функций:

* get\_faculty\_and\_group\_id – получения id-факультета и id-группы с помощью запроса напрямую к конечной API.
* get\_schedule – извлечение расписания из тела API запроса https://ruz.spbstu.ru/api/v1/ruz/scheduler/{group\_id}?date={date}.
* print\_schedule – вывод расписания на экран.
* plot\_schedule – построение графика.

# Результаты

Листинг реализованной программы представлен в Приложении 1. На рисунках 3 и 4 представлены результаты запуска реализованной программы.

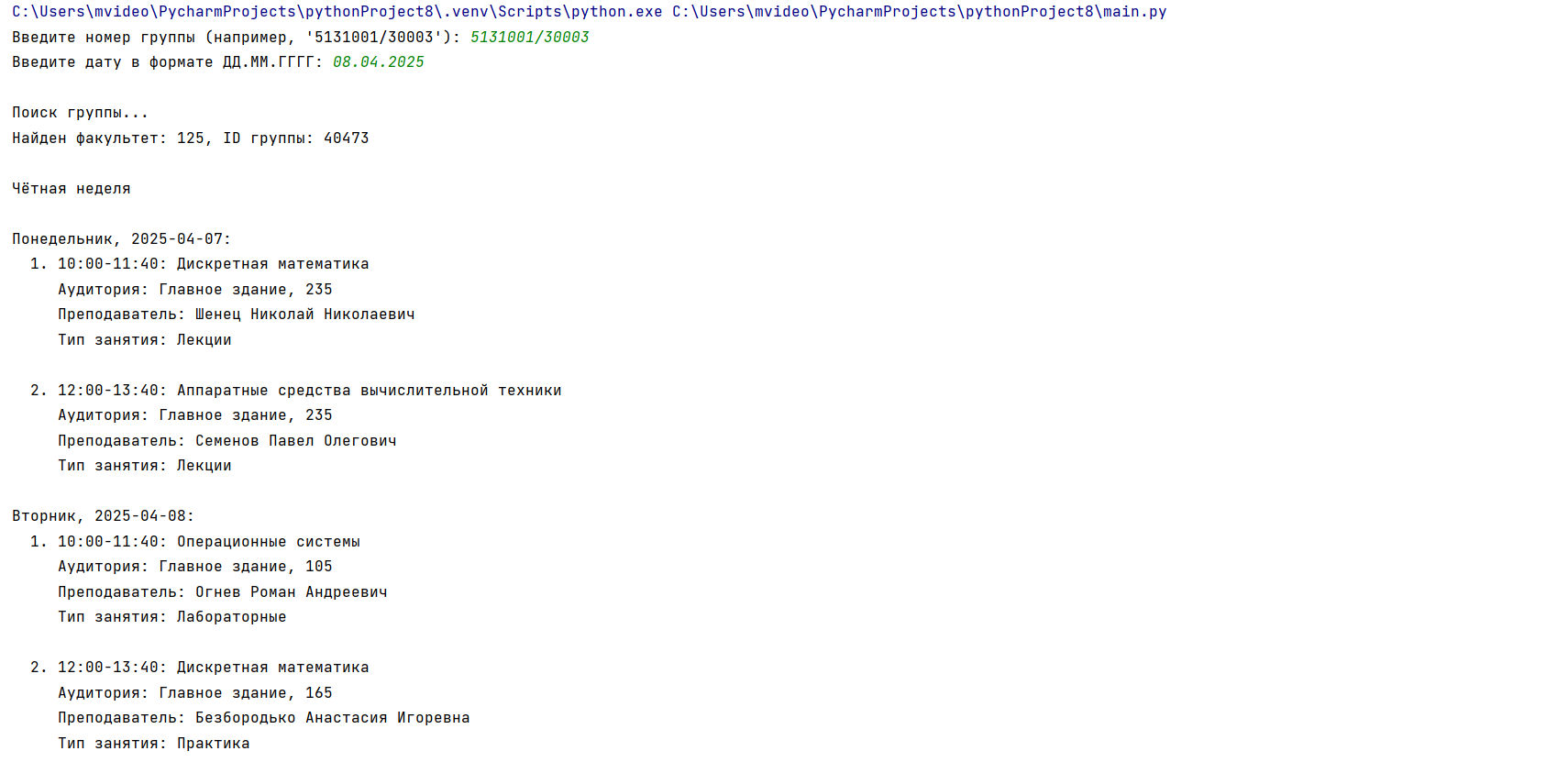


Рисунок 3 – Пример вывода расписания при запуске программы

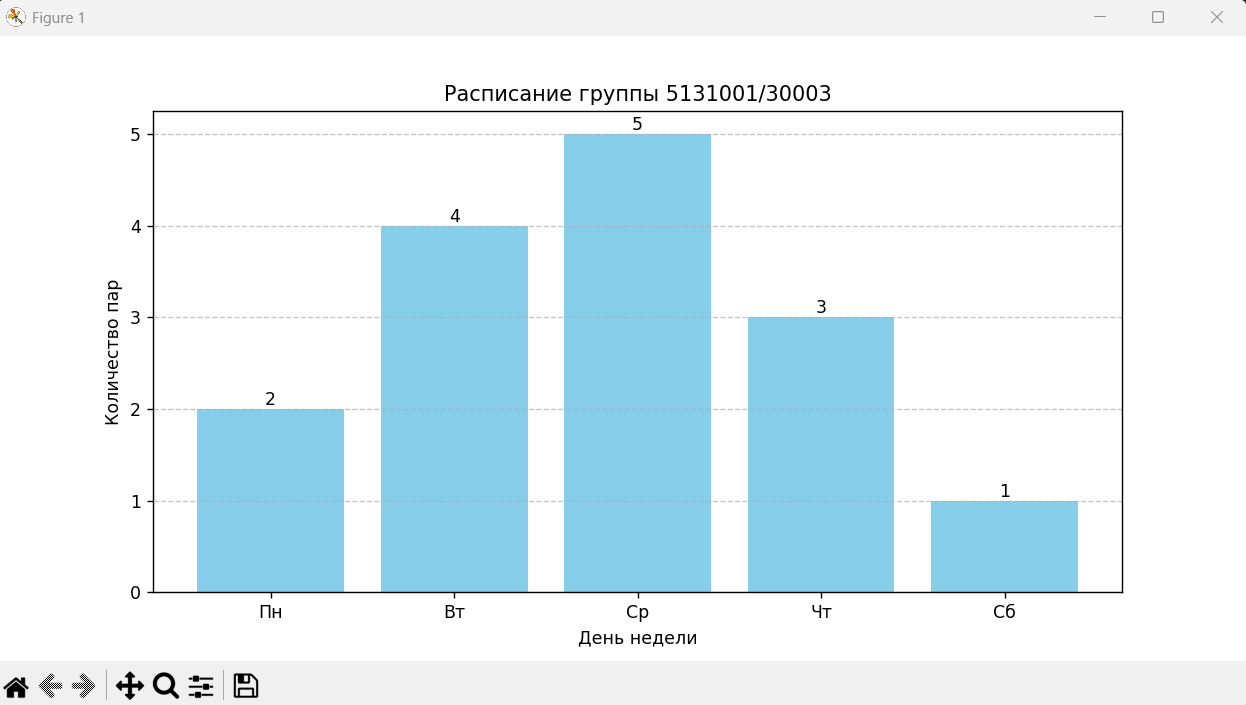


Рисунок 4 – Пример вывода графика при запуске программы

# Ответы на контрольные вопросы

1. *Какие виды HTTP-запросов существуют, для чего нужен каждый из них? Какие данные можно получить при отправке GET-запроса к web-ресурсу?*

Данные между клиентом и сервером в рамках работы протокола передаются с помощью HTTP-сообщений. Они бывают двух видов:

* Запросы (HTTP Requests) — сообщения, которые отправляются клиентом на сервер, чтобы вызвать выполнение некоторых действий. Зачастую для получения доступа к определенному ресурсу. Основой запроса является HTTP-заголовок.
* Ответы (HTTP Responses) — сообщения, которые сервер отправляет в ответ на клиентский запрос.

GET используется для запроса содержимого указанного ресурса. Это с его помощью браузер получает HTML код конкретной страницы и все ее объекты. Тело такого запроса является пустым. Ответ может кэшироваться. GET запрос может передать параметры на сервер для уточнения запрашиваемых данных. Параметры запроса содержатся в адресе запроса, отделяются от URL знаком «?», пары параметр-значение разделяются символом «&».

1. *Что такое DOM-объект, для чего он необходим? Какие библиотеки и функции были использованы при анализе HTML-страницы?*

DOM – это древовидное представление веб-сайта, загружаемого в браузер, в виде серии объектов, вложенных друг в друга. Оно определяет логический каркас документа, способы доступа к нему и управления им. С его помощью мы можем создавать документы, перемещаться по их структуре, а также добавлять, изменять или удалять его элементы и их содержимое.

Корень любого DOM-дерева – сам HTML-документ (document). А уже от корня, в зависимости от структуры HTML-разметки – тянется вверх ствол с ветками. Можно получить доступ ко всем его элементам с помощью набора свойств и методов, предоставляемых DOM API.

1. *По умолчанию в библиотеке requests включена проверка SSL-сертификатов. Что необходимо сделать, чтобы запросы к защищенным страницам работали корректно?*

В request для отключения проверки сертификата используется передача соответствующего параметра verify = False:

requests.get(url, verify=False)

1. *Из каких частей состоит HTTP-запрос? Какие способы передачи параметров в запрос существуют?*

Стартовая строка HTTP-запроса состоит из трех элементов:

1. Метод HTTP-запроса (method, реже используется термин verb). Обычно это короткое слово на английском, которое указывает, что конкретно нужно сделать с запрашиваемым ресурсом. Например, метод GET сообщает серверу, что пользователь хочет получить некоторые данные, а POST — что некоторые данные должны быть помещены на сервер.
2. Цель запроса. Представлена указателем ресурса URL, который состоит из протокола, доменного имени (или IP-адреса), пути к конкретному ресурсу на сервере. Дополнительно может содержать указание порта, несколько параметров HTTP-запроса и еще ряд опциональных элементов.
3. Версия используемого протокола (либо HTTP/1.1, либо HTTP/2), которая определяет структуру следующих за стартовой строкой данных.

Передать параметры в запрос можно с помощью Query String в URL ссылке: она начинается со знака вопроса (?), за которым следует пара «параметр-значение», между которыми расположен символ равно (=). В поле Query String могут быть переданы несколько параметров с помощью символа амперсанд в качестве разделителя.

# Вывод

В ход лабораторной работы были получены навыки работы с протоколом HTTP: построены запросы к удаленному веб-серверу для получения содержимого веб-страницы. Также были получены навыки обработки и извлечения данных из HTML-страницы при помощи Python-библиотеки requests. Была реализована и протестирована программа на языке Python, извлекающая тело ответа из расписания занятий.

# Приложение 1

Листинг программы, извлекающей из тела ответа расписание:

import requests  
import matplotlib.pyplot as plt  
from datetime import datetime  
import sys  
  
*# Получаем ID факультета и группы через API*def get\_faculty\_and\_group\_id(group\_name):  
 *# Запрашиваем список факультетов* url = "https://ruz.spbstu.ru/api/v1/ruz/faculties"  
 headers = {"User-Agent": "Mozilla/5.0"}  
 response = requests.get(url, headers=headers)  
 response.raise\_for\_status() *# Проверяем код состояния ответа на ошибки* faculties\_data = response.json()  
  
 *# Проходим по всем факультетам* for faculty in faculties\_data['faculties']:  
 faculty\_id = faculty['id']  
 groups\_url = f"https://ruz.spbstu.ru/api/v1/ruz/faculties/{faculty\_id}/groups"  
 groups\_response = requests.get(groups\_url, headers=headers)  
 groups\_response.raise\_for\_status()  
 groups\_data = groups\_response.json()  
 for group in groups\_data['groups']:  
 if group['name'] == group\_name:  
 return faculty\_id, group['id']  
 print(f"Группа {group\_name} не найдена!")  
 return None, None  
  
def get\_schedule(group\_id, date):  
 url = f"https://ruz.spbstu.ru/api/v1/ruz/scheduler/{group\_id}?date={date}"  
 headers = {"User-Agent": "Mozilla/5.0"}  
  
 try:  
 response = requests.get(url, headers=headers)  
 response.raise\_for\_status()  
 return response.json()  
 except requests.RequestException as e:  
 print(f"Ошибка при получении расписания: {str(e)}")  
 return None  
  
def print\_schedule(schedule):  
 if not schedule:  
 return  
 week\_type = "Нечётная" if schedule['week']['is\_odd'] else "Чётная"  
 print(f"\n{week\_type} неделя\n")  
 weekdays = { 1: "Понедельник", 2: "Вторник", 3: "Среда", 4: "Четверг", 5: "Пятница", 6: "Суббота", 7: "Воскресенье"}  
 for day in schedule['days']:  
 weekday = weekdays.get(day['weekday'], f"День {day['weekday']}")  
 print(f"{weekday}, {day['date']}:")  
 if not day['lessons']:  
 print(" Нет занятий\n")  
 continue  
 for i, lesson in enumerate(day['lessons'], 1):  
 print(f" {i}. {lesson['time\_start']}-{lesson['time\_end']}: {lesson['subject']}")  
 if lesson.get('auditories'):  
 aud = lesson['auditories'][0]  
 print(f" Аудитория: {aud['building']['name']}, {aud['name']}")  
 if lesson.get('teachers'):  
 print(f" Преподаватель: {lesson['teachers'][0]['full\_name']}")  
 print(f" Тип занятия: {lesson['typeObj']['name']}\n")  
  
def plot\_schedule(schedule):  
 if not schedule:  
 return  
 weekdays = { 1: "Пн", 2: "Вт", 3: "Ср", 4: "Чт", 5: "Пт", 6: "Сб", 7: "Вс"}  
 days = []  
 counts = []  
 for day in schedule['days']:  
 weekday = weekdays.get(day['weekday'], day['weekday'])  
 days.append(weekday)  
 counts.append(len(day['lessons']))  
 plt.figure(figsize=(10, 5))  
 bars = plt.bar(days, counts, color='skyblue')  
 for bar in bars:  
 height = bar.get\_height()  
 plt.text(bar.get\_x() + bar.get\_width() / 2, height, f'{int(height)}', ha='center', va='bottom')  
 plt.title(f"Расписание группы {schedule['group']['name']}")  
 plt.xlabel("День недели")  
 plt.ylabel("Количество пар")  
 plt.grid(axis='y', linestyle='--', alpha=0.7)  
 plt.show()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 group\_name = input("Введите номер группы (например, '5131001/30003'): ").strip()  
 date\_input = input("Введите дату в формате ДД.ММ.ГГГГ: ").strip()  
  
 try:  
 date\_obj = datetime.strptime(date\_input, "%d.%m.%Y")  
 date\_api = date\_obj.strftime("%Y-%m-%d")  
 except ValueError:  
 print("Неверный формат даты! Используйте ДД.ММ.ГГГГ")  
 sys.exit()  
 print("\nПоиск группы...")  
 faculty\_id, group\_id = get\_faculty\_and\_group\_id(group\_name)  
 if not group\_id or not faculty\_id:  
 sys.exit()  
 print(f"Найден факультет: {faculty\_id}, ID группы: {group\_id}")  
 schedule = get\_schedule(group\_id, date\_api)  
 if not schedule:  
 sys.exit()  
 print\_schedule(schedule)  
 plot\_schedule(schedule)