**Санкт-Петербургский политехнический университет Петра Великого**

**Институт кибербезопасности и защиты информации**

**ЛАБОРАТОРНАЯ РАБОТА № 2**

**Извлечение и обработка данных из стороннего**

**web-сервера с помощью Python-приложения**

по дисциплине "Цифровая культура"

Выполнил

студент гр. 4851004/10001 Веремейчук Я. Ю.

<*подпись*>

Помощник преподавателя Данилов В. Д.

<*подпись*>

Санкт-Петербург

2023

**Цель работы**

Получение навыков работы с протоколом HTTP: построение запросов к удаленному веб-серверу для получения содержимого веб-страницы. Получение навыков обработки и извлечения данных из HTMLстраницы при помощи Python-библиотек.

**Задание на работу**

1. Установить необходимые Python-библиотеки: beautifulsoup, requests, matplotlib.

2. Перехватить экземпляр запроса к веб-сайту https://ruz.spbstu.ru. Для этого возможно использование утилиты burpsuite (https://portswigger.net/burp/communitydownload). Изучить заголовки запроса.

3. Включив фильтрацию результатов поиска (по номеру группы/по преподавателю), изучить то, как меняется URL запроса. Сделать выводы о параметрах в URL.

4. С помощью библиотеки requests сформировать GET-запрос к веб-сайту, проанализировать тело полученного ответа.

5. Модифицировать параметры URL и заголовки запроса таким образом, чтобы в теле полученного ответа содержалась отфильтрованная информация.

6. В зависимости от варианта извлечь (например, с помощью beautifulsoap или регулярных выражений) из тела ответа расписание в формате: неделя (четная/нечетная), название предмета, дата, время, аудитория, преподаватель.

7. Реализовать утилиту, которая в качестве входных данных принимает: 1) дату; 2) номер группы (ИКИЗИ). В качестве результирующих данных выводятся: 1) расписание занятий (неделя, название предмета, дата, время, аудитория, преподаватель); 2) расписание в виде графика; по оси абсцисс – день недели, по оси ординат – количество занятий. Фильтрация по номеру группы (только группы института ИКИЗИ).

8. Написать отчет о проделанной работе.

9. Ответить на контрольные вопросы

**Ход работы**

Перед началом работы были установлены Python-библиотеки: beautifulsoup4, requests, matplotlib.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок – Установка необходимых библиотек

Была установлена утилита burpsuite, с помощью которой был перехвачен экземпляр веб-сайта <https://ruz.spbstu.ru>. Были изучены заголовки запроса.

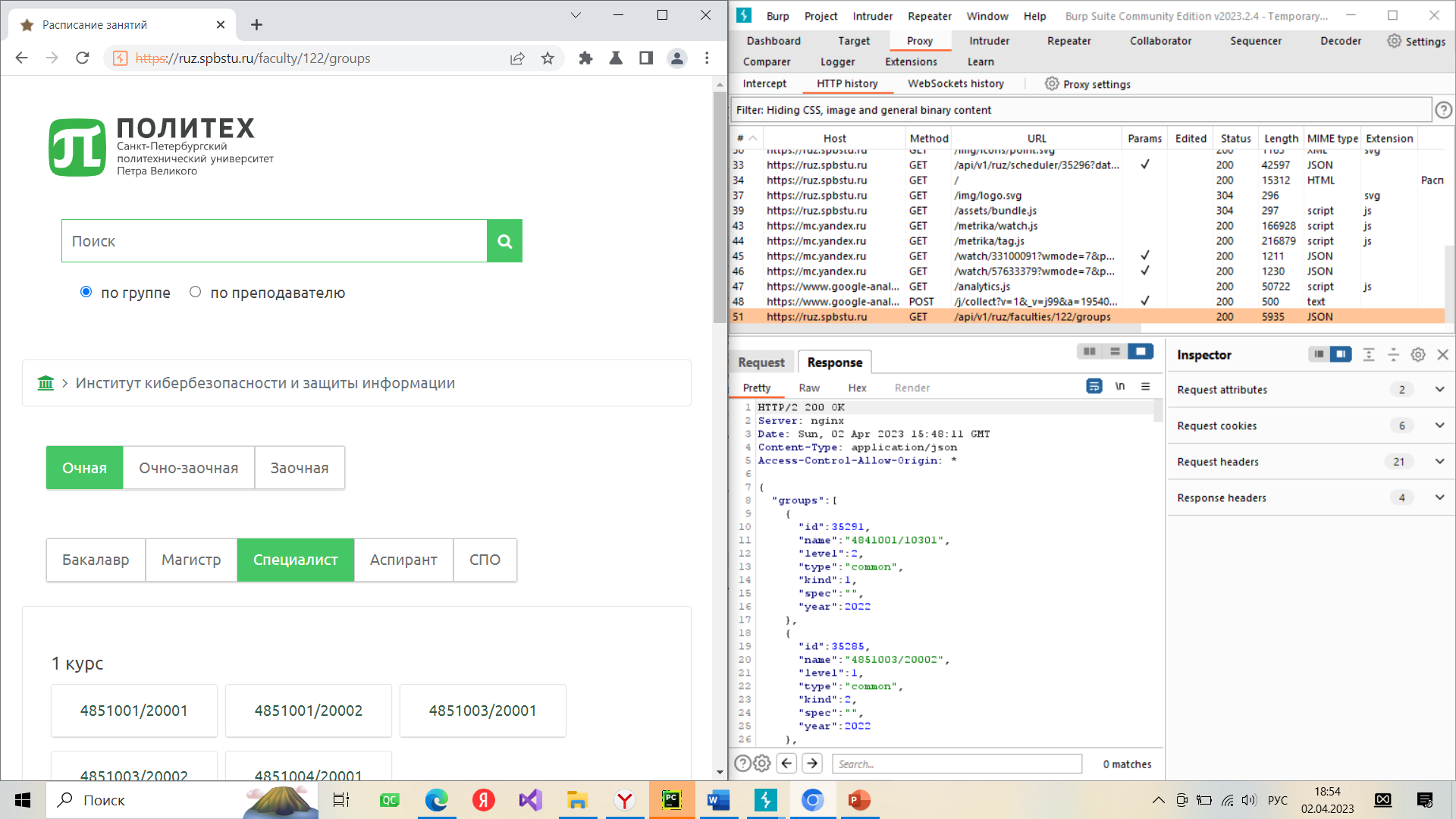


Рисунок – Перехват экземпляра веб-сайта

HTTP/2 200 OK – **строка статуса**

Server: nginx – **HTTP заголовок**

Date: Sun, 02 Apr 2023 15:48:11 GMT – **HTTP заголовок**

Content-Type: application/json – **HTTP заголовок**

Access-Control-Allow-Origin: \* – **HTTP заголовок**

**Тело ответа** –

{

"groups": [

{

"id": 35291,

"name": "4841001/10301",

"level": 2,

"type": "common",

"kind": 1,

"spec": "",

"year": 2022

},

Далее была включена фильтрация по группе 4851004/10001. URL запроса меняется. Прошлый URL со значением «group» очевидно говорит о том, что на данной странице мы можем выбрать группу. В новом URL «schedule», переводящееся, как «расписание», на сайте можно наблюдать расписание выбранной группы на конкретную неделю, которая имеет код 35296.

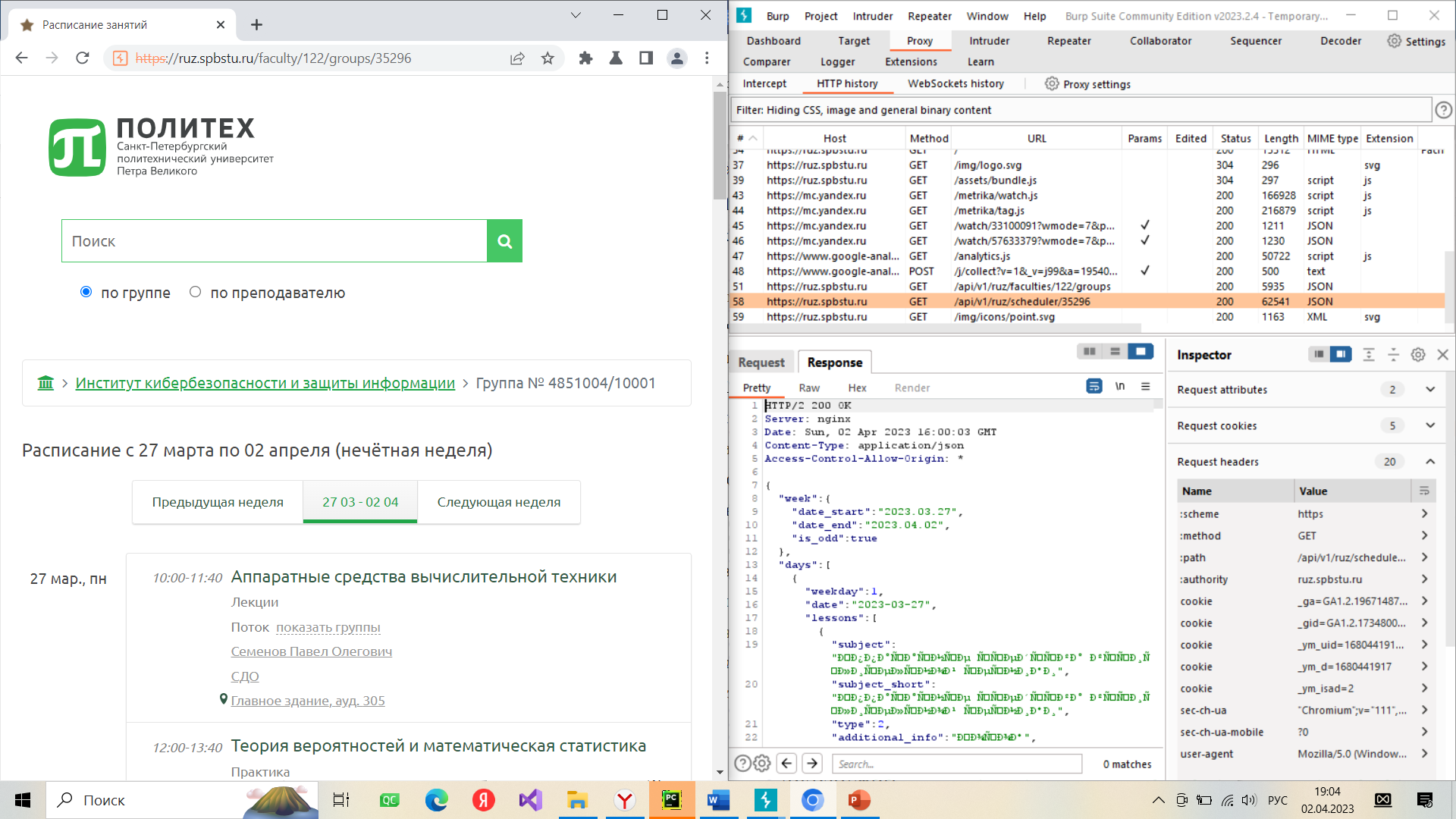


Рисунок – Изменение URL

GET является одним из самых популярных HTTP методов. Метод GET указывает на то, что происходит попытка извлечь данные из определенного ресурса. Для того, чтобы выполнить запрос GET, используется requests.get(). Объект response является мощным средством для просмотра содержимого и обработки результатов запросов.

response = requests.get('https://ruz.spbstu.ru/')

Первым битом информации, которую можно получить от объекта ответа Response, является код состояния запроса к серверу status codes. 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. Существует много других кодов состояния запроса, которые могут дать более детальное представление о том, что произошло с отправленным запросом.

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

response = requests.get('https://ruz.spbstu.ru/')

print(response.status\_code)

Вывод после работы кода выше: 200.

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.

Чтобы получить содержимое ответа в бинарном виде, можно использовать свойство response.content.

response = requests.get('https://ruz.spbstu.ru/')

print(response.content)

Вывод: b'<!DOCTYPE html>\r\n<html lang="ru">\r\n<head>\r\n <meta charset="UTF-8">\r\n <title>\xd0\xa0\xd0\xb0\xd1\x81\xd0\xbf\ …

Response.content предоставляет доступ к “сырым” байтам полезного содержимого ответа. Response.text решает проблему кодировки.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок – Наглядный вывод ответа на запрос GET

Тело полученного ответа состоит из различных скриптов, содержащих информацию, которую можно видеть на сайте, например, названия институтов.

Чтобы получить отфильтрованную информацию, в URL были добавлены параметры. Вывод программы удовлетворяет запросу.

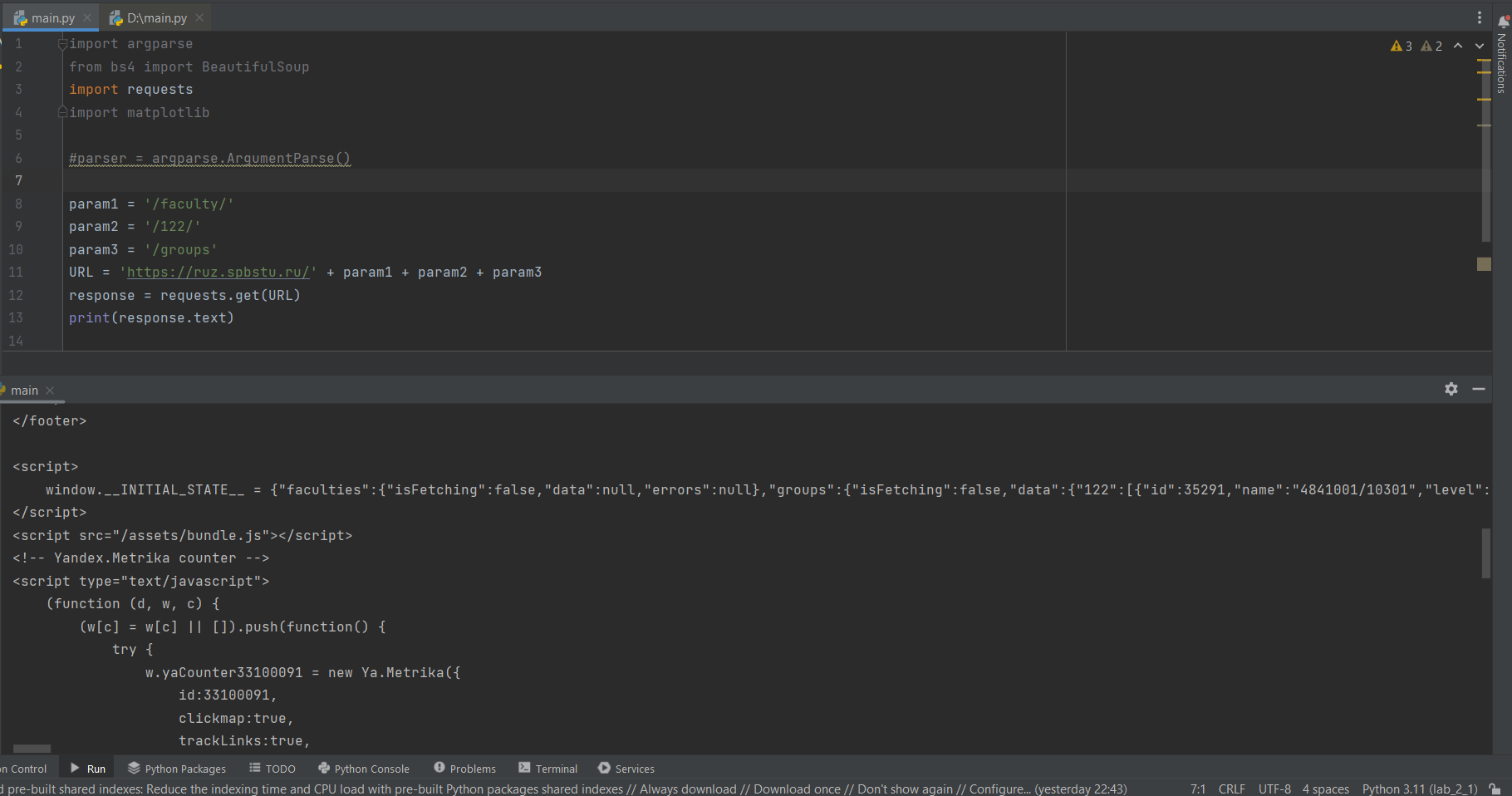


Рисунок – Вывод отфильтрованной информации

Была установлена библиотека lxml для работы с миии. Из тела ответа было извлечено расписание.

Была реализована утилита, которая в качестве входных данных принимает: 1) дату; 2) номер группы (ИКИЗИ). В качестве результирующих данных выводятся: 1) расписание занятий (неделя, название предмета, дата, время, аудитория, преподаватель); 2) расписание в виде графика; по оси абсцисс – день недели, по оси ординат – количество занятий. Фильтрация по номеру группы (только группы института ИКИЗИ).

Были использованы некоторые библиотеки.  
Re – для регулярных выражений. Регулярное выражение — это строка, задающая шаблон поиска подстрок в тексте. Одному шаблону может соответствовать много разных строчек.  
Json. JSON — текстовый формат обмена данными, основанный на JavaScript.  
Datetime. Модуль позволяет управлять датами и временем, представляя их в таком виде, в котором пользователи смогут их понимать.  
Argparse. Модуль упрощает написание удобных интерфейсов командной строки.

Работа утилиты начинается с функции find\_group(): прописывается необходимый URL, создается запрос на веб-сайт, данные преобразуются в более понятный вид с помощью JSON. Из ответа на запрос можно получить параметры: номер группы, код института, его наименование. Если было найдено несколько совпадений по группе, необходимо выбрать одну, чтобы вывести ее расписание.

С помощью функций get\_table() и get\_daily\_table() обрабатывается запрос на расписание выбранной группы или дня: аудитория, преподаватель, время начала пары, время окончания. Если расписание на какой-либо день отсутствует, выводится соответствующая строка, сообщающая об этом. Функции get\_title() и print\_graph() выводят диаграмму, отражающую загруженность студента рассматриваемой группы на неделе.

**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 webресурсу?**

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 (изображения, CSS и т.п). 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 SSLсертификатов. 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.

В примере ниже стартовая строка указывает, что в качестве метода используется GET, обращение будет произведено к ресурсу /index.html, по версии протокола HTTP/1.1:

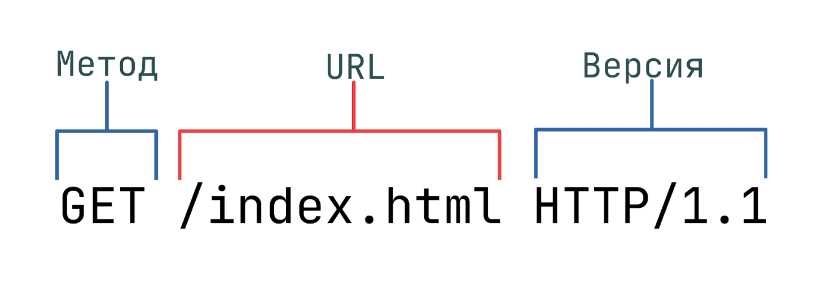


Рисунок – Стартовая строка HTTP-запроса

GET самый известный метод HTTP-запроса, предназначенный для получения информации с сервера.

HEAD Метод похож на облегченный метод GET, так как он также запрашивает информацию с ресурса, но в ответ от сервера не приходит тело ответа, хотя заголовки ответа возвращаются.

POST Используется для отправки информации на ресурс. Запрос приводит к изменениям на сервере, многократное применение запроса POST может привести к дублировании записей в БД, в отличие от запроса PUT, который полностью перезаписывает данные на ресурсе, не создавая новую сущность.

PUT Предназначен для изменения данных на ресурсе, отправка запроса с методом PUT полностью обновить запись на ресурсе.

PATCH Благодаря отправке запроса с методом PATCH можно частично изменять указанный ресурс. У запроса обязательно должно быть тело, где передаются параметры для корректировки данных на стороне сервера.

DELETE Метод позволяет отправить запрос на удаление указанной сущности. У запроса может быть тело, но оно необязательно. В нем можно указать параметры для поиска записи в базе данных и ее последующего удаления.

OPTIONS Данный HTTP-метод позволяет получить сведения о деталях соединения с ресурсом. Благодаря данному методу можно узнать какие другие методы допустимы при обращении к ресурсу.

CONNECT Метод дает возможность установить связь с ресурсом, организовывает своеобразный “туннель”. Может использоваться для установления соединения с ресурсами, которые применяют шифрование соединения по SSL.

TRACE Позволяет следить за тем, что происходит с запросом на пути к целевому ресурсу, как изменяется запрос промежуточными серверами, например, прокси-серверами. Метод предназначен прежде всего для целей диагностики.

1. **Что такое API в контексте веб-серверов?**

API (Application Programming Interface, или программный интерфейс приложения) — это набор протоколов, подпрограмм и других инструментов для создания сайтов или приложений. API регламентирует, как клиент и сервер могут взаимодействовать друг с другом.

API помогает отделить одно приложение от другого, выступая «прослойкой» между системами. API накладывают свои правила на запрос клиента или ответ от сервера. Благодаря этому приложения работают согласованно и данные защищены с обеих сторон.

**Вывод**

В ходе лабораторной работы были получены навыки работы с протоколом HTTP: были построены запросы к удаленному веб-серверу для получения содержимого веб-страницы. Были получены навыки обработки и извлечения данных из HTMLстраницы при помощи Python-библиотек.

**Приложение**

import requests

import matplotlib.pyplot

import numpy

from bs4 import BeautifulSoup

import re

import json

import datetime

import argparse

'''tmp = ' '

def get\_data\_today():

current\_date = datetime.datetime.now()

print(current\_date.year)

tmp1 = f'{current\_date.day:02}'

tmp2 = f'{current\_date.month:02}'

tmp3 = f'{current\_date.year}'

tmp = tmp1 + '.' + tmp2 + '.' + tmp3

print(tmp)

get\_data\_today()'''

parser = argparse.ArgumentParser()

parser.add\_argument("-s", "--search", help="enter string to search", default="4851004/10001")

parser.add\_argument("-d", "--date", help="enter date (DD.MM.YYYY)", default="03.04.2023")

parser.add\_argument("--graph", help="enter key to hide graph", action="store\_true")

args = parser.parse\_args()

table = [0, 0, 0, 0, 0, 0, 0]

def get\_json(response):

start\_json = re.search(r"window\.\_\_INITIAL\_STATE\_\_ = (.\*?);", response).group(1)

json\_with\_groups = json.loads(start\_json)

return json\_with\_groups

def find\_group(name):

number\_of\_groups\_found = 0

find\_group\_url = 'https://ruz.spbstu.ru/search/groups?q=' + name.replace(' ', '%20')

response = requests.get(find\_group\_url).text

json\_with\_groups = get\_json(response)

found\_groups = []

if "searchGroup" in json\_with\_groups:

raw\_found\_groups = json\_with\_groups["searchGroup"]["data"]

number\_of\_groups\_found = len(raw\_found\_groups)

for group in range(number\_of\_groups\_found):

group\_name = raw\_found\_groups[group]["name"]

group\_id = raw\_found\_groups[group]["id"]

group\_faculty = raw\_found\_groups[group]["faculty"]["id"]

found\_groups.append({"name": group\_name, "group\_id": group\_id, "faculty": group\_faculty})

if number\_of\_groups\_found:

number\_group = 0

print('Найдены группы:')

for group in range(number\_of\_groups\_found):

print(str(group + 1) + ' - ' + found\_groups[group]["name"])

if number\_of\_groups\_found != 1:

print("Выберите группу (1 - " + str(number\_of\_groups\_found) + "): ", end='')

number\_group = int(input()) - 1

if number\_group < 0 or number\_group > number\_of\_groups\_found - 1:

print('Некорректное значение группы')

return {}

else:

print('Группы не найдены')

return {}

print()

return found\_groups[number\_group]

def get\_daily\_table(study\_week, input\_date):

number\_of\_pairs = 0

output = ""

for i in range(len(study\_week)):

date = datetime.datetime.strptime(study\_week[i]["date"], "%Y-%m-%d").strftime("%d.%m.%Y")

if date == input\_date:

number\_of\_pairs += 1

if study\_week[i]["weekday"] == 1:

week\_day = 'Понедельник'

elif study\_week[i]["weekday"] == 2:

week\_day = 'Вторник'

elif study\_week[i]["weekday"] == 3:

week\_day = 'Среда'

elif study\_week[i]["weekday"] == 4:

week\_day = 'Четверг'

elif study\_week[i]["weekday"] == 5:

week\_day = 'Пятница'

elif study\_week[i]["weekday"] == 6:

week\_day = 'Суббота'

elif study\_week[i]["weekday"] == 7:

week\_day = 'Воскресенье'

lessons = study\_week[i]["lessons"]

output += week\_day + ' (' + date + ')\n'

for j in range(len(lessons)):

audience = ''

teacher = ''

subject\_name = lessons[j]["subject"] + '\n '

type\_subject = lessons[j]["typeObj"]["name"] + '\n '

time\_start = lessons[j]["time\_start"] + '-'

time\_end = lessons[j]["time\_end"] + '\n '

if "teachers" in lessons[j] and lessons[j]["teachers"]:

teacher = lessons[j]["teachers"][0]["full\_name"] + '\n '

if "auditories" in lessons[j] and lessons[j]["auditories"]:

audience = lessons[j]["auditories"][0]["building"]["name"]

audience += ', ауд. ' + lessons[j]["auditories"][0]["name"] + '\n'

output += str(j + 1) + '. ' + time\_start + time\_end + subject\_name + type\_subject + teacher + audience

output += '\n'

if not number\_of\_pairs:

output += 'Вам не нужно идти на пары, если их нет'

return output

def get\_title(response, date):

soup = BeautifulSoup(response, "lxml")

item = soup.find("div", {'id': 'rootPageContainer', 'class': 'page\_\_container'})

title = item.find("h3").text + '\n'

title = re.search(r'\(.\*?\)', title).group()

title = 'Расписание на неделю с днём ' + date + ' ' + title + '\n'

return title

def get\_table(group, date):

group\_name = str(group['name'])

group\_id = str(group['group\_id'])

group\_faculty = str(group['faculty'])

get\_table\_url = 'https://ruz.spbstu.ru/faculty/' + group\_faculty + '/groups/' + group\_id + '?date='

get\_table\_url += datetime.datetime.strptime(date, "%d.%m.%Y").strftime("%Y-%m-%d")

response = requests.get(get\_table\_url).text

json\_with\_table = get\_json(response)

title = get\_title(response, date)

output = 'Группа: ' + group\_name + '\n' + title + '\n'

study\_week = json\_with\_table['lessons']['data'][group\_id]

output += get\_daily\_table(study\_week, date)

for i in range(len(study\_week)):

table[study\_week[i]["weekday"] - 1] = len(study\_week[i]["lessons"])

return output

def print\_graph(graph, output):

title = re.search(r"Расписание.\*неделя\)", output).group(0)

x\_list = list(range(0, 7))

x\_indexes = numpy.arange(len(x\_list))

matplotlib.pyplot.setp('r')

matplotlib.pyplot.figure(figsize=(9, 5))

matplotlib.pyplot.title(title)

matplotlib.pyplot.xlabel('День недели')

matplotlib.pyplot.ylabel('Количество пар')

matplotlib.pyplot.bar(x\_indexes, table)

week\_days = ['Понедельник', 'Вторник', 'Среда', 'Четверг', 'Пятница', 'Суббота', 'Воскресенье']

matplotlib.pyplot.xticks(x\_indexes, week\_days)

if not graph:

matplotlib.pyplot.show()

if \_\_name\_\_ == '\_\_main\_\_':

found\_group = find\_group(args.search)

if found\_group:

output = get\_table(found\_group, args.date)

print(output)

print\_graph(args.graph, output)

parser.print\_help()