**Санкт-Петербургский политехнический университет Петра Великого**

**Институт кибербезопасности и защиты информации**

**ЛАБОРАТОРНАЯ РАБОТА № 2**

1. «Извлечение и обработка данных из стороннего web-сервера с помощью Python-приложения»
2. по дисциплине «Цифровая культура»
3. Выполнил

студент гр. 5131001/20003 Черникова В.М.

<*подпись*>

1. Преподаватель Писков А. А.

<*подпись*>

1. Санкт-Петербург
2. 2024

**Цель лабораторной работы**

Извлечение и обработка данных из стороннего web-сервера с помощью Python-приложения

**Задание на работу**

1. Установить необходимые Python-библиотеки: **requests**.
2. Перехватить экземпляр запроса к веб-сайту <https://ruz.spbstu.ru>. Для этого необходимо использовать утилиту **burpsuite**. 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. 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).
5. Описать последовательность и логику запросов к API, которые необходимы для извлечения расписания.
6. 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.
7. В зависимости от варианта, 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.

**Вариант 1** – фильтрацияпо номеру группы.

**Вариант 2** – фильтрация по преподавателю.

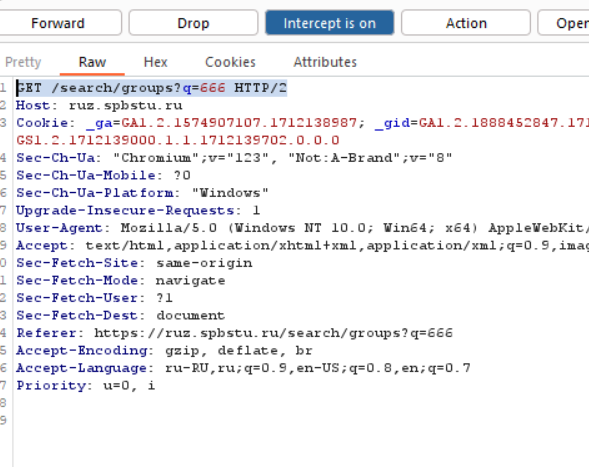
В качестве результата работы программы в консоль выводится расписание занятий в зависимости от входных данных.

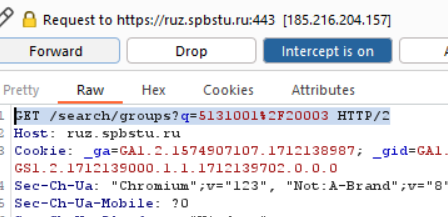
Помимо текстового вывода необходимо представить расписание в виде графика. По оси абсцисс – день недели, по оси ординат – количество занятий. Для построения графика необходимо использовать библиотеку **matplotlib**.

**Ход работы**

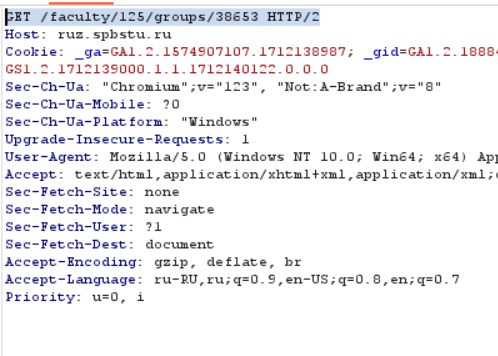
**Анализ заголовков в запросе**

В одном из заголовков в запросе GET можно обратиться к расписанию определленой группы или преподавателя





Следующий http запрос может обратиться к группе по ее уникальному номеру



А этот http запрос может обратиться к расписанию в аудитории по ее уникальному номеру, с указанием здания



**Код**

import argparse

import requests

import pprint

import matplotlib.pyplot as plt

def stud(date\_r, group):

# date\_r = date = "2024-4-5" # - значит, что в этой неделе будет этот день

# group = str(38653) # - уникальный id группы

response = requests.get('https://ruz.spbstu.ru/api/v1/ruz/scheduler/' + group + '?date=' + date\_r)

text = response.json()["days"]

rasp = []

week\_is\_odd = response.json()["week"]["is\_odd"] # True - нечетная

for i in range(len(text)):

rasp.append([])

rasp[i].append({"date": text[i]["date"]})

for j in range(len(text[i]["lessons"])):

rasp[i].append(dict())

rasp[i][j]["subject"] = text[i]["lessons"][j]["subject"]

# print(text[i]["lessons"][j])

rasp[i][j]["time\_start"] = text[i]["lessons"][j]["time\_start"]

rasp[i][j]["time\_end"] = text[i]["lessons"][j]["time\_end"]

rasp[i][j]["type"] = text[i]["lessons"][j]["typeObj"]["name"]

rasp[i][j]["auditories"] = text[i]["lessons"][j]["auditories"][0]["name"]

try:

rasp[i][j]["teachers"] = text[i]["lessons"][j]["teachers"][0]["full\_name"]

except:

rasp[i][j]["teachers"] = "None"

if week\_is\_odd:

print("Неделя нечетная")

else:

print("Неделя четная")

for i in range(len(rasp)):

print("День " + str(i))

pprint.pprint(rasp[i])

graph = []

date = []

for i in range(len(text)):

graph.append(len(text[i]["lessons"])) # для графика

date.append(text[i]["date"]) # для графика

index = [0, 1, 2, 3, 4]

values = [5, 7, 3, 4, 6]

plt.figure(figsize=(10,7))

plt.bar(date, graph)

plt.title(f'ID - Группа {group}')

plt.xlabel('День')

plt.ylabel('Количество занятий')

plt.show()

def aud(date\_r, b, a):

# date\_r = date = "2024-4-5" # - значит, что в этой неделе будет этот день

# b = str(1) # - уникальный id здания

# a = str(546) # - уникальный id аудитории

# /api/v1/ruz/buildings/1/rooms/546/scheduler?date=2024-5-27

response = requests.get('https://ruz.spbstu.ru/api/v1/ruz/buildings/' + b + '/rooms/' + a + '/scheduler?date=' + date\_r)

print(response.json())

text = response.json()["days"]

rasp = []

week\_is\_odd = response.json()["week"]["is\_odd"] # True - нечетная

for i in range(len(text)):

rasp.append([])

rasp[i].append({"date": text[i]["date"]})

for j in range(len(text[i]["lessons"])):

rasp[i].append(dict())

rasp[i][j]["subject"] = text[i]["lessons"][j]["subject"]

# print(text[i]["lessons"][j])

rasp[i][j]["time\_start"] = text[i]["lessons"][j]["time\_start"]

rasp[i][j]["time\_end"] = text[i]["lessons"][j]["time\_end"]

rasp[i][j]["type"] = text[i]["lessons"][j]["typeObj"]["name"]

rasp[i][j]["auditories"] = text[i]["lessons"][j]["auditories"][0]["name"]

try:

rasp[i][j]["teachers"] = text[i]["lessons"][j]["teachers"][0]["full\_name"]

except:

rasp[i][j]["teachers"] = "None"

if week\_is\_odd:

print("Неделя нечетная")

else:

print("Неделя четная")

for i in range(len(rasp)):

print("День " + str(i))

pprint.pprint(rasp[i])

graph = []

date = []

for i in range(len(text)):

graph.append(len(text[i]["lessons"])) # для графика

date.append(text[i]["date"]) # для графика

index = [0, 1, 2, 3, 4]

values = [5, 7, 3, 4, 6]

plt.figure(figsize=(10, 7))

plt.bar(date, graph)

plt.title(f'ID - Здание {b}, аудитория {a}')

plt.xlabel('День')

plt.ylabel('Количество занятий')

plt.show()

def gz\_to\_id(b, a):  
 resp=[]  
 # date\_r = date = "2024-4-5" # - значит, что в этой неделе будет этот день  
 # b = str(1) # - уникальный id здания  
 # a = str(546) # - уникальный id аудитории  
 # /api/v1/ruz/buildings/1/rooms/546/scheduler?date=2024-5-27  
 response = requests.get('https://ruz.spbstu.ru/api/v1/ruz/buildings/')  
 text=response.json()["buildings"]  
 #print(text)  
 for i in range(len(text)):  
 if text[i]['name']==b:  
 resp.append(str(text[i]['id']))  
 response = requests.get('https://ruz.spbstu.ru/api/v1/ruz/buildings/'+str(resp[0])+"/rooms")  
 text = response.json()["rooms"]  
 for j in range(len(text)):  
 if text[j]['name'] == a:  
 resp.append(str(text[j]['id']))  
 return resp  
 return None  
 return None  
  
  
def g\_to\_id(id):  
 req\_get\_group\_id = requests.get('https://ruz.spbstu.ru/search/groups?q='+id)  
 soup = BeautifulSoup(req\_get\_group\_id.text, 'html.parser')  
 groups = []  
 for link in soup.find\_all('a'):  
 if link.get('class'):  
 if link.get('class').count("groups-list\_\_link"):  
 groups.append(  
 [link.get\_text(), link.get('href')[link.get('href').rfind('/') + 1:]])  
 if len(groups) > 1:  
 print("Введите правильный номер группы")  
 return None  
 elif len(groups) < 1:  
 print("Не найдено ни одной группы.")  
 return None  
 else:  
 return groups[0][1]

def main():

p = argparse.ArgumentParser()

# file system args

p.add\_argument("-group", "--g", nargs=2, help="Rasp for your group. Args: data, id")

p.add\_argument("-auditor", "--a", nargs=3, help="Rasp in auditore. Args: data, building, id")

p.add\_argument("-select", "--s", nargs=1, help="Список всех зданий")

args = p.parse\_args() # массив с аргументами

print(args)

args = p.parse\_args() # массив с аргументами

print(args)

if args.s:

stud(args.s[0], args.s[1])

if args.a:

aud(args.a[0], args.a[1], args.a[2])

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

main()

**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?***

Тип HTTP-запроса указывает серверу на то, какое действие мы хотим произвести с ресурсом. Для разграничения действий с ресурсами на уровне HTTP-методов и были придуманы следующие варианты:

GET — получение ресурса

POST — создание ресурса

PUT — обновление ресурса

DELETE — удаление ресурса

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?***

Объектная Модель Документа (DOM) – это программный интерфейс (API) для HTML и XML документов. DOM предоставляет структурированное представление документа и определяет то, как эта структура может быть доступна из программ, которые могут изменять содержимое, стиль и структуру документа. Представление DOM состоит из структурированной группы узлов и объектов, которые имеют свойства и методы. По существу, DOM соединяет веб-страницу с языками описания сценариев либо языками программирования.

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

По умолчанию проверка SSL включена, и запросы выдадут SSLError, если она не сможет проверить сертификат:

>>> **import** **requests**

>>> requests.get('https://requestb.in')

*# requests.exceptions.SSLError: hostname 'requestb.in' doesn't match ...*

Можно передать аргументу verify путь к файлу CA\_BUNDLE или каталогу с доверенными сертификатами CA:

*# указание доверенных сертификатов в запросе*

>>> requests.get('https://example.com', verify='/path/to/certfile')

*# указание доверенных сертификатов для сессии*

>>> sess = requests.Session()

>>> sess.verify = '/path/to/certfile'

Если для параметра verify задан путь к каталогу, то этот каталог должен быть обработан с помощью утилиты c\_rehash, поставляемой с OpenSSL.

Список доверенных CA также можно указать с помощью переменных сред REQUESTS\_CA\_BUNDLE. Если REQUESTS\_CA\_BUNDLE не установлена, то CURL\_CA\_BUNDLE будет использоваться в качестве запасного варианта.

Запросы также могут игнорировать проверку SSL-сертификата, если для параметра verify задано значение False:

>>> requests.get('https://kennethreitz.org', verify=**False**)

*# <Response [200]>*

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

HTTP-запрос состоит из трех основных частей:

Строка запроса (Request line) - содержит метод запроса (GET, POST, PUT, DELETE и т.д.), URI (Uniform Resource Identifier) и версию протокола HTTP.

Заголовки (Headers) - содержат информацию о запросе, такую как тип контента, длина контента, язык, кодировку и другие параметры.

Тело запроса (Body) - содержит данные, которые передаются в запросе.

Существует несколько способов передачи параметров в запрос:

1. В строке запроса (query string) - параметры указываются после символа вопроса (?) в URI.

2. В заголовках запроса (headers) - параметры передаются в виде пар "ключ-значение" в заголовках запроса.

3. В теле запроса (body) - параметры передаются в теле запроса, например, при отправке формы или JSON-данных.

**Вывод**

В ходе лабораторной работы были получены навыки работы с протоколом HTTP: построение запросов к удаленному веб-серверу для получения содержимого веб-страницы, а также его обработки и извлечения данных при помощи Python-библиотек и построение гистограмм.

**Приложение.**

Полный листинг программы

import argparse  
import requests  
import pprint  
import matplotlib.pyplot as plt  
from bs4 import BeautifulSoup  
  
def stud(date\_r, group):  
 # date\_r = date = "2024-4-5" # - значит, что в этой неделе будет этот день  
 # group = str(38653) # - уникальный id группы  
 response = requests.get('https://ruz.spbstu.ru/api/v1/ruz/scheduler/' + group + '?date=' + date\_r)  
 #print(response.json())  
 text = response.json()["days"]  
 rasp = []  
 week\_is\_odd = response.json()["week"]["is\_odd"] # True - нечетная  
 for i in range(len(text)):  
 rasp.append([])  
 rasp[i].append({"date": text[i]["date"]})  
 for j in range(len(text[i]["lessons"])):  
 rasp[i].append(dict())  
 rasp[i][j]["subject"] = text[i]["lessons"][j]["subject"]  
 # print(text[i]["lessons"][j])  
 rasp[i][j]["time\_start"] = text[i]["lessons"][j]["time\_start"]  
 rasp[i][j]["time\_end"] = text[i]["lessons"][j]["time\_end"]  
 rasp[i][j]["type"] = text[i]["lessons"][j]["typeObj"]["name"]  
 rasp[i][j]["auditories"] = text[i]["lessons"][j]["auditories"][0]["name"]  
 try:  
 rasp[i][j]["teachers"] = text[i]["lessons"][j]["teachers"][0]["full\_name"]  
 except:  
 rasp[i][j]["teachers"] = "None"  
 if week\_is\_odd:  
 print("Неделя нечетная")  
 else:  
 print("Неделя четная")  
 for i in range(len(rasp)):  
 print("День " + str(i))  
 pprint.pprint(rasp[i])  
  
 graph = []  
 date = []  
 for i in range(len(text)):  
 graph.append(len(text[i]["lessons"])) # для графика  
 date.append(text[i]["date"]) # для графика  
  
 index = [0, 1, 2, 3, 4]  
 values = [5, 7, 3, 4, 6]  
 plt.figure(figsize=(10,7))  
 plt.bar(date, graph)  
 plt.title(f'ID - Группа {group}')  
 plt.xlabel('День')  
 plt.ylabel('Количество занятий')  
 plt.show()  
  
def aud(date\_r, b, a):  
 # date\_r = date = "2024-4-5" # - значит, что в этой неделе будет этот день  
 # b = str(1) # - уникальный id здания  
 # a = str(546) # - уникальный id аудитории  
 # /api/v1/ruz/buildings/1/rooms/546/scheduler?date=2024-5-27  
 response = requests.get('https://ruz.spbstu.ru/api/v1/ruz/buildings/' + b + '/rooms/' + a + '/scheduler?date=' + date\_r)  
 print(response.json())  
 text = response.json()["days"]  
 rasp = []  
 week\_is\_odd = response.json()["week"]["is\_odd"] # True - нечетная  
 for i in range(len(text)):  
 rasp.append([])  
 rasp[i].append({"date": text[i]["date"]})  
 for j in range(len(text[i]["lessons"])):  
 rasp[i].append(dict())  
 rasp[i][j]["subject"] = text[i]["lessons"][j]["subject"]  
 # print(text[i]["lessons"][j])  
 rasp[i][j]["time\_start"] = text[i]["lessons"][j]["time\_start"]  
 rasp[i][j]["time\_end"] = text[i]["lessons"][j]["time\_end"]  
 rasp[i][j]["type"] = text[i]["lessons"][j]["typeObj"]["name"]  
 rasp[i][j]["auditories"] = text[i]["lessons"][j]["auditories"][0]["name"]  
 try:  
 rasp[i][j]["teachers"] = text[i]["lessons"][j]["teachers"][0]["full\_name"]  
 except:  
 rasp[i][j]["teachers"] = "None"  
 if week\_is\_odd:  
 print("Неделя нечетная")  
 else:  
 print("Неделя четная")  
 for i in range(len(rasp)):  
 print("День " + str(i))  
 pprint.pprint(rasp[i])  
  
 graph = []  
 date = []  
 for i in range(len(text)):  
 graph.append(len(text[i]["lessons"])) # для графика  
 date.append(text[i]["date"]) # для графика  
  
 index = [0, 1, 2, 3, 4]  
 values = [5, 7, 3, 4, 6]  
 plt.figure(figsize=(10, 7))  
 plt.bar(date, graph)  
 plt.title(f'ID - Здание {b}, аудитория {a}')  
 plt.xlabel('День')  
 plt.ylabel('Количество занятий')  
 plt.show()  
  
def gz\_to\_id(b, a):  
 resp=[]  
 # date\_r = date = "2024-4-5" # - значит, что в этой неделе будет этот день  
 # b = str(1) # - уникальный id здания  
 # a = str(546) # - уникальный id аудитории  
 # /api/v1/ruz/buildings/1/rooms/546/scheduler?date=2024-5-27  
 response = requests.get('https://ruz.spbstu.ru/api/v1/ruz/buildings/')  
 text=response.json()["buildings"]  
 #print(text)  
 for i in range(len(text)):  
 if text[i]['name']==b:  
 resp.append(str(text[i]['id']))  
 response = requests.get('https://ruz.spbstu.ru/api/v1/ruz/buildings/'+str(resp[0])+"/rooms")  
 text = response.json()["rooms"]  
 for j in range(len(text)):  
 if text[j]['name'] == a:  
 resp.append(str(text[j]['id']))  
 return resp  
 return None  
 return None  
  
  
def g\_to\_id(id):  
 req\_get\_group\_id = requests.get('https://ruz.spbstu.ru/search/groups?q='+id)  
 soup = BeautifulSoup(req\_get\_group\_id.text, 'html.parser')  
 groups = []  
 for link in soup.find\_all('a'):  
 if link.get('class'):  
 if link.get('class').count("groups-list\_\_link"):  
 groups.append(  
 [link.get\_text(), link.get('href')[link.get('href').rfind('/') + 1:]])  
 if len(groups) > 1:  
 print("Введите правильный номер группы")  
 return None  
 elif len(groups) < 1:  
 print("Не найдено ни одной группы.")  
 return None  
 else:  
 return groups[0][1]  
  
def main():  
 p = argparse.ArgumentParser()  
 # file system args  
 p.add\_argument("-group", "--g", nargs=2, help="Rasp for your group. Args: data, id")  
 p.add\_argument("-auditor", "--a", nargs=3, help="Rasp in auditore. Args: data, building, id")  
 p.add\_argument("-select", "--s", nargs=1, help="Список всех зданий")  
 args = p.parse\_args() # массив с аргументами  
 print(args)  
 if args.g:  
 if g\_to\_id(args.g[1]):  
 stud(args.g[0], g\_to\_id(args.g[1]))  
 if args.a:  
 if gz\_to\_id(args.a[1],args.a[2]):  
 aud(args.a[0], gz\_to\_id(args.a[1],args.a[2])[0], gz\_to\_id(args.a[1],args.a[2])[1])  
 if args.s:  
 print("1-й учебный корпус\n"  
 "11-й учебный корпус\n"  
 "15-й учебный корпус\n"  
 "16-й учебный корпус\n"  
 "17-й учебный корпус\n"  
 "2-й учебный корпус\n"  
 "3-й учебный корпус\n"  
 "4-й учебный корпус\n"  
 "48 учебный корпус\n"  
 "5-й учебный корпус\n"  
 "50 учебный корпус\n"  
 "6-й учебный корпус\n"  
 "9-й учебный корпус\n"  
 "DL\n"  
 "ВИТ\n"  
 "Военная кафедра\n"  
 "Гидробашня\n"  
 "Гидротехнический корпус-1 Северная пристройка\n"  
 "Гидротехнический корпус-1\n"  
 "Гидротехнический корпус-2\n"  
 "Главное здание\n"  
 "Завод «Силовые Машины»\n"  
 "Завод АО «Невский завод»\n"  
 "ИМОП\n"  
 "Институт ядерной энергетики (филиал ФГАОУ ВО СПбПУ) г. Сосновый Бор\n"  
 "Лабораторный корпус\n"  
 "Литейный корпус\n"  
 "Механический корпус\n"  
 "НОЦ СЛК\n"  
 "Научно-исследовательский корпус\n"  
 "Не определено\n"  
 "СЗГМУ им. И.И.Мечникова\n"  
 "Санкт-Петербургский институт машиностроения\n"  
 "Спорткомплекс\n"  
 "ТВН\n"  
 "ТВН-2\n"  
 "Химический корпус\n"  
 "Холдинг «ЛЕНПОЛИГРАФМАШ»\n"  
 "ЦНИИ РТК")  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()