**Основы работы с командами протокола HTTP на прикладном уровне.**

**Для работы с HTTP использовать сервер https://httpbin.org/**

**Теоретическая часть:**

Модуль [requests](http://docs.python-requests.org/en/master/) в настоящее время уже давно является стандартом де-факто для реализации **отправки HTTP-запросов** в Python.  Он удобнее и проще встроенного Urllib настолько, что даже в документации Python [рекомендовано использовать](https://docs.python.org/3/library/urllib.request.html#module-urllib.request) requests.

Внимание! Модуль не входит в стандартную библиотеку Python. Его надо установить отдельно, любым стандартным способом. Например, выполнив в терминале

pip install requests *# Для Windows*

**Основные возможности библиотеки Requests**

Модуль разработан с учетом потребностей современных web-разработчиков и актуальных технологий. Многие операции автоматизированы, а ручные настройки сведены к минимуму.  
  
Для понимания инструментария библиотеки перечислим ее основные возможности:  
– поддержка постоянного HTTP-соединения и его повторное использование;  
– применение международных и национальных доменов;  
– использование Cookie: передача и получение значений в формате ключ: значение;  
– автоматическое декодирование контента;  
– SSLверификация;  
– аутентификация пользователей на большинстве ресурсов с сохранением;  
– поддержка proxy при необходимости;  
– загрузка и выгрузка файлов;  
– стриминговые загрузки и фрагментированные запросы;  
– задержки соединений;  
– передача требуемых заголовков на web-ресурсы и др.

**Пример использования модуля:**

Давайте пошлем запрос GET , и сохраним принятое значение с объектом ответа в переменной, и затем поближе познакомиться с его атрибутами (свойствами), а также поведением:

import requests

response = requests.get('https://api.github.com')

### **Код состояния запроса (status codes)**

Первым битом информации, которую вы можете получить от объекта ответа Response, является код состояния запроса к серверу **status codes**. Код состояния информирует вас о статусе нашего запроса.

В большинстве случаев эту информацию мы будем использовать для реализации в своем коде различной логики: управлять дальнейшей работой нашего приложения:

if response.status\_code == 200:

print('Success!')

elif response.status\_code == 404:

print('Not Found.')

## Содержимое ответа (Content)

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

Чтобы получить содержимое ответа в бинарном виде, мы можем использовать свойство Response.content

Несмотря на то, что свойство Response.content предоставляет нам доступ к “сырым” байтам полезного содержимого ответа и в большинстве случаев мы будем преобразовывать их в строку с заданной кодировкой символов, например, UTF-8. Объект Response легко сделает это для нас, предоставляя доступ к свойству Response.text

## Заголовки ответа (Headers)

Заголовки ответа сервера могут дать много полезной информации, такой, например, как тип полезного содержимого ответа, ограничение по времени, в течение которого ответ будет кэшироваться и т.д. Чтобы просмотреть содержимое заголовков, необходимо обратиться к свойству объекта Response.headers

Небольшое руководство на русском языке по библиотеке можно найти например здесь:

<https://www.awesomeandrew.ru/2020/04/23/requests-python-библиотека-для-отправки-http-запросов/>

**Задание к работе:**

Написать программу которая реализует HTTP методы: OPTION, GET, POST с полным выводом на экран (либо текстовый файл) кода ответа сервера, заголовков ответа и тела ответа. Также выводить текст запроса при выполнении методов GET и POST. При необходимости допускается применение JSON библиотек.

Отчет по работе должен содержать:

1. Файл с исходным кодом вашей программы
2. Если результаты выполнения запросов выводились в файл, то файл с результатами, если выводились в консоль, то сделать скриншоты всех результатов и собрать их в вордовском файле, который приложить к отчету.
3. Завернуть все в формат zip