План.

0. Источники.

1. Python Библиотека HTTP запросов - Request

2. Что содержит requests

3. Назначение

4. Установка Python Requests

5. Методы HTTP запросов

6. Содержимое обекта Response

7. Передача параметров в URL

8. POST Request

9. Неготивные запросы

10. Получение куков

11. Header for Request

12. Тайм-ауты

13. Исключениями

**0. Источники.**

<https://pythonru.com/biblioteki/kratkoe-rukovodstvo-po-biblioteke-python-requests>

<https://requests.readthedocs.io/en/master/>

<https://requests.readthedocs.io/en/master/user/quickstart/>

=======================================================

**1. Python Библиотека HTTP запросов - Request**

Стандартная библиотека Python имеет ряд готовых модулей по работе с HTTP.

* [httplib](https://docs.python.org/3/library/http.html)
* [urllib](https://docs.python.org/3/library/urllib.html) (это надстройка над "низкоуровневой" библиотекой httplib)
* [socket](https://docs.python.org/3/library/socket.html" \l "module-socket) (если уж совсем хочется хардкора, то можно и сразу юзать) .

Но у всех этих модулей есть один большой недостаток - неудобство работы.

1. Большое обилие классов и функций

2. Код получается вовсе не pythonic

Решение библиотека - Request

Сравнение кода:



Requests ничто иное как обёртка над urllib3, а последняя является надстройкой над стандартными средствами Python. Удобство написания кода в большинстве случаев является приоритетом номер один!

=======================================================

**2. Что содержит requests**

* Множество методов http аутентификации
* Сессии с куками
* Полноценная поддержка SSL
* Различные методы-плюшки вроде .json(), которые вернут данные в нужном формате
* Проксирование
* Грамотная и логичная работа с исключениями

=======================================================

3. Назначение

[Requests](https://docs.python-requests.org/en/master/) - это модуль Python, который вы можете использовать для отправки всех видов HTTP-запросов.

=======================================================

4. Установка Python Requests

pip install requests

5. Методы HTTP запросов

<https://developer.mozilla.org/uk/docs/Web/HTTP/Methods>

[GET](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/GET) - Запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные.

[HEAD](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/HEAD) - Запрашивает ресурс так же, как и метод GET, но без тела ответа.

[POST](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/POST) - Используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.

[PUT](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/PUT) - Заменяет все текущие представления ресурса данными запроса.

[DELETE](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE) - Удаляет указанный ресурс.

[OPTIONS](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/OPTIONS) - Используется для описания параметров соединения с ресурсом.

**6. Содержимое обекта Response**

response.status\_code

response.headers

response.url

response.encoding

response.content

response.text

response.json

* response.status\_code

Полный список: https://en.wikipedia.org/wiki/List\_of\_HTTP\_status\_codes

Краткий обзор значений кодов состояния:

1XX — информация

2XX — успешно

3XX — перенаправление

4XX — ошибка клиента (ошибка на вашей стороне)

5XX — ошибка сервера (ошибка на их стороне)

* response.headers

Заголовки отправляются вместе с запросом и возвращаются с ответом. Заголовки используются для того, чтобы клиент и сервер понимали, как интерпретировать данные, отправляемые и получаемые в запросе и ответе. Обычно требуется заголовок content type, поскольку он показывает формат данных, например HTML, JSON, PDF, обычный текст и т. д.

* response.text

Если мы посмотрим response.text мы увидим текстовый документ (это работает для текстовых данных, таких как просматриваемая нами страница HTML),

* response.content

Вы можете также получить доступ к телу ответа в виде байтов для не текстовых ответов.

* response.json()

Если вы работаете с данными в формате JSON, воспользуйтесь встроенным JSON декодером. Если декодирование в JSON не удалось, response.json() вернет исключение. Например, если ответ с кодом 204 (No Content), или на случай если ответ содержит не валидный JSON, попытка обращения к response.json() будет возвращать ValueError: No JSON object could be decoded. Следует отметить, что успешный вызов response.json() не указывает на успешный ответ сервера. Некоторые серверы могут возвращать объект JSON при неудачном ответе (например, сведения об ошибке HTTP 500). Такой JSON будет декодирован и возвращен. Для того, чтобы проверить успешен ли запрос, используйте response.raise\_for\_status() или проверьте какой response.status\_code.

=======================================================

**7. Передача параметров в URL**

=======================================================

**8. POST Request**

=======================================================

**9. Неготивные запросы**

=======================================================

**10. Получение куков**

=======================================================

**11. Header for Request**

Нужно в случае скрапинга.

=======================================================

**12. Тайм-ауты**

Вы можете сделать так, чтобы Requests прекратил ожидание ответа после определенного количества секунд с помощью параметра timeout:  
Почти весь производственный код должен использовать этот параметр почти во всех запросах. Несоблюдение этого требования может привести к зависанию вашей программы

Timeout это не ограничение по времени полной загрузки ответа. Исключение возникает, если сервер не дал ответ за timeout секунд (точнее, если ни одного байта не было получено от основного сокета за timeout секунд).

**13. Исключениями**

Проблемы, которые могут возникнуть в момент соеденения:

* Хост недоступен. Обычно такого рода ошибка происходит из-за проблем конфигурирования DNS. (DNS lookup failure)
* "Вылет" соединения по таймауту
* Ошибки HTTP. Подробнее о HTTP кодах можно посмотреть [здесь](http://www.restapitutorial.com/httpstatuscodes.html).
* Ошибки SSL соединений (обычно при наличии проблем с SSL сертификатом: просрочен, не является доверенным и т.д.)

Базовым классом-исключением в requests является RequestException. От него наследуются все остальные

* HTTPError
* ConnectionError
* Timeout
* SSLError
* ProxyError