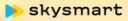



ХОЧУ ПОМОЧЬ
ПРОЕКТУ

HTTP-библиотека requests в Python

 skysmart.ru

РЕКЛАМА



Программирование для детей и подростков в Skysmart

Узнать больше

skillbox.ru

Станьте «DevOps-инженером» всего за 7 месяцев!

Получите поддержку от Центра Карьеры и трудоустроитесь Junior-специалистом.

Налоговый вычет 13%

Спикеры-практики

Трудоустройство

Учитесь сейчас, платите потом

Узнать больше

HTTP-библиотека requests в Python

Что такое HTTP-библиотека для Python

requests — это простая и легкая в использовании HTTP-библиотека для Python, созданная для людей. Модуль позволяет чрезвычайно просто отправлять запросы и получать ответы. Нет необходимости вручную составлять URL-адреса или кодировать данные для PUT и POST (достаточно использовать метод Request.json()).

Установка в виртуальное окружение:

если нет

VirtualEnv

инициализация

```
python3 -m venv venv
source venv/bin/activate
pip install -U requests
```

Свойства:

- Создание HTTP запроса к WEB-странице;
- Код ответа сервера (status code);
- Получение контента в виде текста;
- Получение контента в виде байтов;
- Отслеживание перенаправлений, атрибут Response.history.

Создание HTTP запроса к WEB-странице.

Сделать HTTP-запрос очень просто. Сначала необходимо импортировать модуль requests. Далее попробуем запросить контент веб-страницы. Для примера запросим методом GET общедоступную HTTP страницу с временной шкалой сайта GitHub:

```
>>> import requests
>>> resp = requests.get('https://api.github.com/events')
```

В результате запроса получили объект ответа под названием resp. Далее, из этого объекта можно [получить всю необходимую информацию](#) о этой странице.

Простой API запросов означает, что все формы HTTP-запросов столь же очевидны. Например, вот как вы делаете [запрос HTTP POST](#):

```
>>> import requests
>>> resp = requests.post('https://httpbin.org/post', data = {'key': 'value'})
```

Создание других типов HTTP-запросов, таких как PUT, DELETE, HEAD и OPTIONS все это так же просто и очевидно.

```
>>> import requests
>>> resp_put = requests.put('https://httpbin.org/put', data = {'key': 'value'})
>>> resp_del = requests.delete('https://httpbin.org/delete')
>>> resp_head = requests.head('https://httpbin.org/get')
>>> resp_opt = requests.options('https://httpbin.org/get')
```

При работе с библиотекой requests необходимо уяснить следующее:

Всякий раз, когда вызываются методы requests.get() или requests.post() и т. д., то делаются две важные вещи.

1. [Вверх](#) создается объект Request, который будет отправлен на сервер для запроса ресурса указанного в URL .
2. Создается объект Response, который генерируется после того, как запрос получает [ответ от сервера](#).

https://docs-python.ru/packages/modul-requests-python/

1/4

[Объект Response](#) содержит всю информацию, возвращаемую сервером, а также объект запроса, который вы создали изначально

Код ответа сервера (status code).

Можно проверить код состояния ответа сервера следующим способом:

РЕКЛАМА • 16+



skillbox.ru

Станьте «DevOps-инженером» всего за 7 месяцев!

Получите поддержку от Центра Карьеры и трудоустроитесь Junior-специалистом.

Налоговый вычет 13% >

Спикеры-практики >

Трудоустройство >

Учитесь сейчас, платите потом >

Узнать больше

```
resp.raise_for_status()

# requests.exceptions.HTTPError: 404 Client Error: Not Found (url=httpbin.org/get')
```

Для работы со встроенным объектом поиска кода состояния `requests.codes` для удобства

```
resp.raise_for_status()

# requests.codes.ok
```

В случае получения кода состояния клиента 4XX или ответа на ошибку сервера 5XX), то можно поднять его с помощью объекта `raise_for_status()`:

```
resp.raise_for_status()

# requests.exceptions.HTTPError: 404 Client Error: Not Found (url=httpbin.org/status/404')
```

Вывод ошибки (в данном случае `404`):

```
File ~/requests/requests.py, line 832, in raise_for_status
# raise http_error
# requests.exceptions.HTTPError: 404 Client Error
```

Но если вызвать `Response.raise_for_status()` для ответа сервера со статусом `200`, то в результате получим `None`:

```
>>> resp.status_code == requests.codes.ok
# True
>>> resp.raise_for_status()
# None
```

Получение контента WEB-страницы в виде текста.

Извлекать/читать контент/текста ответа сервера также легко как [делать запросы](#). Еще раз рассмотрим временную шкалу GitHub:

```
>>> import requests
>>> resp = requests.get('https://api.github.com/events')
>>> resp.text
'[{"repository":{"open_issues":0,"url":"https://github.com/...
```

Запросы будут автоматически декодировать содержимое с сервера. Большинство кодировок юникода легко декодируются.

Когда посылается запрос, модуль `requests` делает обоснованные предположения о кодировке ответа на основе HTTP-заголовков. При доступе к атрибуту `resp.text` используется кодировка, прочитанная модулем `requests` во время запроса к серверу. Если сервер не предоставляет кодировку страницы в заголовках ответа или кодировка не распознана, то по умолчанию `requests` использует кодировку `'utf-8'`. Можно узнать, какую кодировку использует конкретный запрос, и изменить ее, используя атрибут `resp.encoding`:

```
# просмотр текущей кодировки страницы
>>> resp.encoding
# 'utf-8'

# изменение кодировки страницы
>>> resp.encoding = 'windows-1251'
```

Если изменить кодировку, то запросы будут использовать новое значение `resp.encoding` всякий раз, когда вызывается `resp.text`. Бывают случаи, когда заголовок ответа сервера выдает неправильную кодировку (отличную от той которая указана в HTML разметке) и в этой ситуации, необходимо применить специальную логику, чтобы определить, какой будет кодировка

контента. Например, языки разметки HTML и XML имеют возможность указывать свою кодировку в своем теле. В подобных ситуациях необходимо использовать `resp.content`, чтобы найти указанную кодировку, а затем установить `resp.encoding`. Это позволит извлекать данные HTML-страницы `resp.text` с правильной кодировкой.

РЕКЛАМА • 16+



S skillbox.ru

Станьте «DevOps-инженером» всего за 7 месяцев!

Получите поддержку от Центра Карьеры и трудоустроитесь Junior-специалистом.

Налоговый вычет 13% >

Спикеры-практики >

Трудоустройство >

Учитесь сейчас, платите потом >

Узнать больше

Виде байтов.

В ответе в [байтах](#) для нетекстовых запросов, например для загрузки изображений:

```
...:0,"url":"https://github.com/...
```

Запросы deflate автоматически декодируются.

Извлечение из двоичных данных, возвращаемых запросом, можно использовать следующий код:

```
...p.content))
```

Извлечений, атрибут `Response.history`.

requests не обрабатывает перенаправление для всех типов запросов, кроме HEAD (по умолчанию). Для перенаправления можно использовать атрибут `.history` объекта `Response`.

Атрибут `history` возвращает список объектов **Response**, созданные для выполнения запроса. Список отсортирован от самого последнего к первому.

Для всех HTTP-запросы на HTTPS:

```
>>> r = requests.get('http://github.com/')
>>> r.url
# 'https://github.com/'
>>> r.status_code
# 200
>>> r.history
# [<Response [301]>]
```

Если используются запросы GET, OPTIONS, POST, PUT, PATCH или DELETE, то можно отключить обработку перенаправления с помощью аргумента `allow_redirects`:

```
>>> import requests
>>> r = requests.get('http://github.com/', allow_redirects=False)
>>> r.status_code
# 301
>>> r.history
# []
```

Если используется HEAD запрос, то также можно включить отслеживание перенаправления:

```
>>> import requests
>>> r = requests.head('http://github.com/', allow_redirects=True)
>>> r.url
# 'https://github.com/'
>>> r.history
# [<Response [301]>]
```

Содержание раздела:

- [КРАТКИЙ ОБЗОР МАТЕРИАЛА.](#)
- [GET и POST запросы с модулем requests](#)
- [Получение/отправка заголовков сервера модулем requests](#)
- [Извлечение и установка cookies с модулем requests](#)
- [Сессии/сеансы Session\(\) модуля requests](#)
- [Вверх](#)
- [Получение и отправка данных в виде JSON с модулем requests](#)

- [Установка timeout для модуля requests](#)
- [Объект PreparedRequest модуля requests](#)
- [Загрузка файлов на сервер модулем requests](#)
- [Загрузка больших данных модулем requests](#)
- [Загрузка больших данных модулем requests](#)
- [Загрузка больших данных модулем requests](#)



[DOCS-Python.ru](#)™, 2023 г.
Станьте «DevOps-инженером» всего за 7 месяцев!

Получите поддержку от Центра Карьеры и трудоустройтесь Junior-специалистом.

- Налоговый вычет 13% >
- Спикеры-практики >
- Трудоустройство >
- Учитесь сейчас, платите потом >

[Узнать больше](#)

(Внимание! При копировании материала ссылка на источник обязательна)

[@docs_python_ru](#)