Сообщить об ошибке.

Описание, атрибуты и методы объекта requests.Response.

Синтаксис:

```
import requests

# создается в результате запроса к серверу
Response = requests.get(...)
```

Параметры:

- нет.

Описание:

Объект `requests.Response` модуля `requests` содержит всю информацию ответа сервера на HTTP-запрос requests.get(), requests.post() и т.д.

Объект ответа сервера `requests.Response` генерируется после того, как библиотека `requests` получают ответ от сервера. Объект ответа `Response` содержит всю информацию, возвращаемую сервером, а также объект запроса, который создали изначально.

Атрибуты и методы объекта `Response`.

- `Response.apparent_encoding` возвращает кодировку, угаданную `chardet`,
- `Response.close()` освобождает соединение с пулом,
- `Response.content` возвращает контент в байтах,
- `Response.cookies` возвращает `cookies`, установленные сервером,
- `Response.elapsed` возвращает время, потраченное на запрос,
- `Response.encoding` устанавливает кодировку, для декодирования,
- `Response.headers` возвращает заголовки сервера,
- `Response.history` возвращает историю перенаправлений,
- `Response.is_permanent_redirect` определение постоянных редиректов,
- `Response.is_redirect` есть ли редирект,
- `Response.iter_content()` перебирает данные ответа кусками,
- `Response.iter_lines()` перебирает данные ответа, по одной строке,
- `Response.json()` возвращает ответ в виде JSON,
- `Response.links` возвращает ссылки заголовка ответа,
- `Response.next` возвращает объект `PreparedRequest`,
- `Response.ok` `True`, если `status_code` меньше 400,
- `Response.raise_for_status()` вызывает исключение `HTTPError`,
- `Response.raw` возвращает ответа в виде файлового объекта,
- `Response.reason` возвращает текстовое представление ответа,
- `Response.request` возвращает объект `PreparedRequest` запроса,
- `Response.status_code` возвращает код ответа сервера,
- `Response.text` возвращает контент ответа сервера в юникоде,
- `Response.url` возвращает URL-адрес, после перенаправлений.

- Пример работы с объектом ответа сервера `Response .

`Response.apparent_encoding`:

Атрибут `Response.apparent_encoding` возвращает кодировку, определенную сторонним модулем `chardet`.

`Response.close()`:

Метод `Response.close()` освобождает соединение с пулом. Как только этот метод был вызван, базовый необработанный объект больше не будет доступен.

Примечание: обычно не нужно вызывать явно.

`Response.content`:

Атрибут `Response.content` возвращает содержание ответа сервера, представленное в байтах.

`Response.cookies = None`:

Атрибут `Response.cookies` возвращает хранилище `CookieJar` файлов cookie, которые сервер отправил обратно.

Другими словами возвращает `cookies`, установленные сервером.

`Response.elapsed = None`:

Атрибут `Response.elapsed` возвращает время, прошедшее между отправкой запроса и получением ответа (в виде `timedelta`).

Это свойство специально измеряет время, затраченное между отправкой первого байта запроса и завершением анализа заголовков. Поэтому на него не влияет потребление содержимого ответа или значения ключевого аргумента `stream`.

`Response.encoding = None`:

Атрибут `Response.encoding` возвращает/устанавливает кодировку для декодирования контента при доступе к атрибуту `Response.text`.

`Response.headers = None`:

Атрибут `Response.headers` возвращает словарь без учета регистра, с заголовками сервера, которые он вернул во время ответа.

Например, заголовки ['content-encoding'] вернут значение заголовка ответа `Content-Encoding`.

`Response.history = None`:

Атрибут `Response.history` возвращает список объектов ответа сервера из истории запроса. Здесь окажутся все перенаправленные ответы.

Список сортируется от самого старого до самого последнего запроса.

`Response.is_permanent_redirect`:

Атрибут `Response.is_permanent_redirect` возвращает `True`, если в этом ответе одна из постоянных версий перенаправления.

`Response.is_redirect`:

Атрибут `Response.is_redirect` возвращает `True` если этот ответ является хорошо сформированным HTTP-перенаправлением, которое могло бы быть обработано автоматически (`Session.resolve_redirects`).

`Response.iter_content(chunk_size=1, decode_unicode=False)`:

Метод `Response.iter_content()` перебирает данные ответа. Когда в запросе установлен `stream=True`, то это позволяет избежать одновременного чтения содержимого в память для больших ответов.

Размер блока `chunk_size` - это количество байтов, которые он должен считывать в память. Это не обязательно длина каждого возвращаемого элемента, т.к. может иметь место декодирование.

Аргумент `chunk_size` должен иметь тип `int` или `None`. Значение `None` будет функционировать по-разному в зависимости от значения `stream`. Если `stream=True`, то будет считывать данные по мере их поступления в любом размере полученных фрагментов. Если `stream=False`, то данные возвращаются как один фрагмент.

Если аргумент `decode_unicode=True`, то содержимое будет декодировано с использованием наилучшей доступной кодировки на основе ответа.

`Response.iter_lines(chunk_size=512, decode_unicode=False, delimiter=None)`:

Метод `Response.iter_lines()` перебирает данные ответа, по одной строке за раз. Когда в запросе установлен `stream=True`, то это позволяет избежать одновременного чтения содержимого в память для больших ответов.

Обратите внимание, что этот метод не является безопасным для повторного входа.

`Response.json(**kwargs)`:

Метод `Response.json()` возвращает закодированное в `json` содержимое ответа, если таковое имеется.

Аргумент `**kwargs` это необязательные аргументы, которые принимает `json.loads`.

Возникающие исключения:

- `simplejson.JSONDecodeError` - если тело ответа не содержит действительного `json` и установлен `simplejson`.
- `json.JSONDecodeError` - если тело ответа не содержит допустимого `json` и `simplejson` не установлен.

`Response.links`:

Атрибут `Response.links` возвращает проанализированные ссылки заголовка ответа, если таковые имеются.

`Response.next`:

Атрибут `Response.next` возвращает объект подготовленного запроса `PreparedRequest` для следующего запроса в цепочке перенаправления, если таковой имеется.

`Response.ok`:

Атрибут `Response.ok` возвращает `True`, если `status_code` меньше 400, и `False`, если нет.

Этот атрибут проверяет, находится ли код состояния ответа в диапазоне от 400 до 600, чтобы проверить, была ли ошибка клиента или ошибка сервера. Если код состояния находится в диапазоне от 200 до 400, то этот атрибут вернет значение `True`.

Это не проверка кода ответа 200 OK.

`Response.raise_for_status()`:

Метод `Response.raise_for_status()` вызывает исключение `HTTPError`, если таковой произошел.

`Response.raw = None`:

Атрибут `Response.raw` возвращает представление ответа в виде файлового объекта (для расширенного использования).

Использование `Response.raw` требует, чтобы в запросе был установлен `stream=True`. Это требование не

распространяется на внутреннее использование запросов.

`Response.reason = None`:

Атрибут `Response.reason` возвращает текстовое представление ответа HTTP-статуса, например `'Not Found'` или `'OK'`.

`Response.request = None`:

Атрибут `Response.request` возвращает объект подготовленного запроса `PreparedRequest`, на который является ответом.

`Response.status_code = None`:

Атрибут `Response.status_code` целочисленный код ответа HTTP-статуса, например 404 или 200.

`Response.text`:

Атрибут `Response.text` возвращает содержание/контент ответа сервера в юникоде.

Если `Response.encoding=None`, то кодировка будет угадана с помощью модуля `chardet`.

Кодировка содержимого ответа определяется исключительно на основе HTTP-заголовков, следующих за RFC 2616. Если вы точно знаете кодировку сайта, а `Response.text` возвращает "*крокозябры*", то тогда сначала следует установить `Response.encoding` в нужную кодировку, перед тем как вызвать `Response.text`.

`Response.url = None`:

Атрибут `Response.url` окончательный URL-адреса ответа, после перенаправлений, которые делал сервер.

Пример работы с объектом ответа сервера `Response`:

```
import requests
resp = requests.get('https://httpbin.org/get',
                    headers={'user-agent': 'my-agent-0.0.1'},
                    cookies={'one': 'true'})
if resp.ok:
    print('Заголовки ответа сервера:')
    print(resp.headers)
    print('\ncookies, установленные сервером:')
    print(dict(resp.cookies))
    print('\nКонтент запрашиваемой страницы:')
    print(resp.text)
    print('\nДоступ к параметрам нашего запроса `resp.request`:')
    print('- заголовки, которые отправил запрос:')
    print(resp.request.headers)
    print('- метод нашего запроса:')
    print(resp.request.method)

# Заголовки ответа сервера:
# {'Date': 'Wed, 31 Mar 2021 09:32:48 GMT', 'Content-Type': 'application/json',
# 'Content-Length': '327', 'Connection': 'keep-alive', 'Server': 'gunicorn/19.9.0',
# 'Access-Control-Allow-Origin': '*', 'Access-Control-Allow-Credentials': 'true'}
#
# cookies, установленные сервером:
# {}
#
# Контент запрашиваемой страницы:
# {
#   "args": {},
#   "headers": {
#     "Accept": "*/*"
```

```
#     "Accept": "*/*",
#     "Accept-Encoding": "gzip, deflate",
#     "Cookie": "one=true",
#     "Host": "httpbin.org",
#     "User-Agent": "my-agent-0.0.1",
#     "X-Amzn-Trace-Id": "Root=1-606441c0-674418047cacc3e2617e6449"
#   },
#   "origin": "xxx.xxx.xxx.xxx",
#   "url": "https://httpbin.org/get"
# }
#
#
# Доступ к параметрам нашего запроса `resp.request`:
# - заголовки, которые отправил запрос:
# {'user-agent': 'my-agent-0.0.1', 'Accept-Encoding': 'gzip, deflate',
# 'Accept': '*/*', 'Connection': 'keep-alive', 'Cookie': 'one=true'}
# - метод нашего запроса:
# GET
```

Содержание раздела: