

Requests: HTTP for Humans

Fork me on GitHub

Release v2.18.1. ([Installation](#))

license Apache 2.0 wheel yes python 2.6, 2.7, 3.3, 3.4, 3.5, 3.6 codecov 87% Say Thanks!

Requests is the only Non-GMO HTTP library for Python, safe for human consumption.

Warning: Recreational use of the Python standard library for HTTP may result in dangerous side-effects, including: security vulnerabilities, verbose code, reinventing the wheel, constantly reading documentation, depression, headaches, or even death.

Behold, the power of Requests:

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
u'{"type": "User" ... '
>>> r.json()
{'u'private_gists': 419, u'total_private_repos': 77, ...}
```

See [similar code, sans Requests](#).

Requests allows you to send *organic, grass-fed* HTTP/1.1 requests, without the need for manual labor. There's no need to manually add query strings to your URLs, or to form-encode your POST data. Keep-alive and HTTP connection pooling are 100% automatic, thanks to [urllib3](#).

User Testimonials

Twitter, Spotify, Microsoft, Amazon, Lyft, BuzzFeed, Reddit, The NSA, Her Majesty's Government, Google, Twilio, Runscope, Mozilla, Heroku, PayPal, NPR, Obama for America, Transifex, Native Instruments, The Washington Post, SoundCloud, Kippt, Sony, and Federal U.S. Institutions that prefer to be unnamed claim to use Requests internally.

Armin Ronacher—

Requests is the perfect example how beautiful an API can be with the right level of abstraction.

Matt DeBoard—

I'm going to get Kenneth Reitz's Python requests module tattooed on my body, somehow. The whole thing.

Daniel Greenfeld—

Nuked a 1200 LOC spaghetti code library with 10 lines of code thanks to Kenneth Reitz's request library. Today has been AWESOME.

Kenny Meyers—

Python HTTP: When in doubt, or when not in doubt, use Requests. Beautiful, simple, Pythonic.

Requests is one of the most downloaded Python packages of all time, pulling in over 11,000,000 downloads every month. All the cool kids are doing it!

Beloved Features

Requests is ready for today's web.

- Keep-Alive & Connection Pooling
- International Domains and URLs
- Sessions with Cookie Persistence
- Browser-style SSL Verification
- Automatic Content Decoding

 v: latest ▼

- Basic/Digest Authentication
- Elegant Key/Value Cookies
- Automatic Decompression
- Unicode Response Bodies
- HTTP(S) Proxy Support
- Multipart File Uploads
- Streaming Downloads
- Connection Timeouts
- Chunked Requests
- `.netrc` Support

Requests officially supports Python 2.6–2.7 & 3.3–3.7, and runs great on PyPy.

The User Guide

This part of the documentation, which is mostly prose, begins with some background information about Requests, then focuses on step-by-step instructions for getting the most out of Requests.

- [Introduction](#)
 - [Philosophy](#)
 - [Apache2 License](#)
 - [Requests License](#)
- Installation of Requests
 - `$ pip install requests`
 - [Get the Source Code](#)
- Quickstart
 - [Make a Request](#)
 - [Passing Parameters In URLs](#)
 - [Response Content](#)
 - [Binary Response Content](#)
 - [JSON Response Content](#)
 - [Raw Response Content](#)
 - [Custom Headers](#)
 - [More complicated POST requests](#)
 - [POST a Multipart-Encoded File](#)
 - [Response Status Codes](#)
 - [Response Headers](#)
 - [Cookies](#)
 - [Redirection and History](#)
 - [Timeouts](#)
 - [Errors and Exceptions](#)
- [Advanced Usage](#)
 - [Session Objects](#)
 - [Request and Response Objects](#)
 - [Prepared Requests](#)
 - [SSL Cert Verification](#)
 - [Client Side Certificates](#)
 - [CA Certificates](#)
 - [Body Content Workflow](#)
 - [Keep-Alive](#)
 - [Streaming Uploads](#)
 - [Chunk-Encoded Requests](#)
 - [POST Multiple Multipart-Encoded Files](#)
 - [Event Hooks](#)

- [Custom Authentication](#)
- [Streaming Requests](#)
- [Proxies](#)
- [Compliance](#)
- [HTTP Verbs](#)
- [Custom Verbs](#)
- [Link Headers](#)
- [Transport Adapters](#)
- [Blocking Or Non-Blocking?](#)
- [Header Ordering](#)
- [Timeouts](#)
- [Authentication](#)
 - [Basic Authentication](#)
 - [Digest Authentication](#)
 - [OAuth 1 Authentication](#)
 - [OAuth 2 and OpenID Connect Authentication](#)
 - [Other Authentication](#)
 - [New Forms of Authentication](#)

The Community Guide

This part of the documentation, which is mostly prose, details the Requests ecosystem and community.

- [Frequently Asked Questions](#)
- [Recommended Packages and Extensions](#)
- [Integrations](#)
- [Articles & Talks](#)
- [Support](#)
- [Vulnerability Disclosure](#)
- [Community Updates](#)
- [Release and Version History](#)
- [Release Process and Rules](#)

The API Documentation / Guide

If you are looking for information on a specific function, class, or method, this part of the documentation is for you.

- [Developer Interface](#)
 - [Main Interface](#)
 - [Exceptions](#)
 - [Request Sessions](#)
 - [Lower-Level Classes](#)
 - [Lower-Lower-Level Classes](#)
 - [Authentication](#)
 - [Encodings](#)
 - [Cookies](#)
 - [Status Code Lookup](#)
 - [Migrating to 1.x](#)
 - [Migrating to 2.x](#)

The Contributor Guide

If you want to contribute to the project, this part of the documentation is for you.

- [Contributor's Guide](#)

- Be Cordial
- [Get Early Feedback](#)
- Contribution Suitability
- Code Contributions
 - Steps for Submitting Code
 - [Code Review](#)
 - [New Contributors](#)
 - Kenneth Reitz's Code Style™
- Documentation Contributions
- Bug Reports
- [Feature Requests](#)
- [Development Philosophy](#)
 - Management Style
 - [Values](#)
 - Semantic Versioning
 - Standard Library?
 - [Linux Distro Packages](#)
- [How to Help](#)
 - Feature Freeze
 - [Development Dependencies](#)
 - Runtime Environments
- [Authors](#)
 - [Keepers of the Four Crystals](#)
 - [Urllib3](#)
 - Patches and Suggestions

There are no more guides. You are now guideless. Good luck.