

REQUEST MODULE

Requests is an elegant and simple HTTP library for Python, built for human beings.

Requests allows you to send HTTP/1.1 requests extremely easily. 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](#).

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
- 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 3.8+, and runs great on PyPy.

```
r = requests.get('https://api.github.com/events')
```

Now, we have a Response object called r. We can get all the information we need from this object.

Requests' simple API means that all forms of HTTP request are as obvious. For example, this is how you make an HTTP POST request:

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

HTTP request types: PUT, DELETE, HEAD and OPTIONS? These are all just as simple:

```
r = requests.put('https://httpbin.org/put', data={'key': 'value'})
```

```
r = requests.delete('https://httpbin.org/delete')
```

```
r = requests.head('https://httpbin.org/get')
```

```
r = requests.options('https://httpbin.org/get')
```

Passing Parameters In URLs

You often want to send some sort of data in the URL's query string. If you were constructing the URL by hand, this data would be given as key/value pairs in the URL after a question mark, e.g. `httpbin.org/get?key=val`. Requests allows you to provide these arguments as a dictionary of strings, using the `params` keyword argument. As an example, if you wanted to pass `key1=value1` and `key2=value2` to `httpbin.org/get`, you would use the following code:

```
payload = {'key1': 'value1', 'key2': 'value2'}
```

```
r = requests.get('https://httpbin.org/get', params=payload)
```

You can see that the URL has been correctly encoded by printing the URL:

```
print(r.url)
```

```
https://httpbin.org/get?key2=value2&key1=value1
```

```
import requests
# r = requests.get('https://api.github.com/events')
# print(r.cookies)
payload = {'Name': 'Jhon', 'Password': 'Prajwal'}
r = requests.get('https://httpbin.org/get', params=payload)
print(r.url)
```

<https://httpbin.org/get?Name=Jhon&Password=Prajwal>

Note that any dictionary key whose value is None will not be added to the URL's query string.

You can also pass a list of items as a value:

```
payload = {'key1': 'value1', 'key2': ['value2', 'value3']}
r = requests.get('https://httpbin.org/get', params=payload)
print(r.url)
```

<https://httpbin.org/get?key1=value1&key2=value2&key2=value3>

Response Content

We can read the content of the server's response. Consider the GitHub timeline again:

```
import requests
```

```
r = requests.get('https://api.github.com/events')
```

```
r.text
```

```
'[{"repository":{"open_issues":0,"url":"https://github.com/...
```

Requests will automatically decode content from the server. Most unicode charsets are seamlessly decoded.