

# Python's Requests Library (Guide)

## Section 1 of 3

- Getting Started with Requests
- The GET Request
- Processing Responses
  - Status Codes
  - Content and Header

# A Few Details About Requests

HTTP for Humans™

- Created by Kenneth Reitz
- <http://docs.python-requests.org/en/master/>

# About Python's Requests Library

Why learn about requests?

- De facto standard for making HTTP requests in Python
- Beautifully Simple API
- Abstracts the complicated parts of HTTP
- Focuses on Interaction with Services

# About Python's Requests Library

What do I need for this tutorial?

- A general knowledge of HTTP (Hypertext Transfer Protocol)
  - [https://www.w3schools.com/tags/ref\\_httpmethods.asp](https://www.w3schools.com/tags/ref_httpmethods.asp)

# The GET Request

Indicates you are trying to retrieve data from a specified resource

- Testing it with GitHub's Root Rest API
  - <https://api.github.com>
- Returns a Response
  - An object containing results of the request

# Status Codes

Issued by a server in response to a client request.

A few examples.

- 1xx Information
- 2xx Success - request was received, understood and accepted
- 3xx Redirection
- 4xx Client Errors
- 5xx Server Errors

More info at [https://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_status\\_code](https://en.wikipedia.org/wiki/List_of_HTTP_status_code)

# End of Section 1 of 3

1. Getting Started with Requests
2. The GET Request
3. Processing Responses
- ▶ **4. Query String Parameters**
- 5. Request Headers**
- 6. Other HTTP Methods**
- 7. The Message Body**
- 8. Inspecting Your Request**

# Welcome to Section 2 of 3

1. Getting Started with Requests
2. The GET Request
3. Processing Responses
- ▶ 4. **Query String Parameters**
5. **Request Headers**
6. **Other HTTP Methods**
7. **The Message Body**
8. **Inspecting Your Request**



# The Query String Parameters

Part of a URL which assigns values to specified parameters

- Commonly added to the base URL
  - <https://api.github.com/search/repositories?q={query}>

Other examples

- commit\_search\_url :  
[https://api.github.com/search/commits?q={query}&page,per\\_page,sort,order](https://api.github.com/search/commits?q={query}&page,per_page,sort,order)
- user\_search\_url :  
[https://api.github.com/search/users?q={query}&page,per\\_page,sort,order](https://api.github.com/search/users?q={query}&page,per_page,sort,order)

# The Query String Parameters

Can be composed of a series of arguments

- The series is separated by a plus sign
  - argument1+argument2+argument3...

# The Request Headers

Further customize your requests

- Pass a dictionary of HTTP headers to `get()`
- Accept
  - Specify what is acceptable for the response
    - Limit to only certain media/types
- Authorization - discussed in the section 3

# Other HTTP Methods

Requests provides methods for each of these

- POST
- PUT
- DELETE
- HEAD
- PATCH
- OPTIONS

We will be testing these using Kenneth Reitz httpbin service

<https://httpbin.org/>(name of method)

# The Message Body

How POST, PUT and PATCH pass data


- Different from query string parameters
- Payload is passed through the parameter, `data=`
- `data=` can take
  - Dictionary
  - List of tuples
  - Bytes / file-like objects
- For JSON use the parameter, `json=`
  - Requests will serialize the data, and add the Content-Type header

# Inspecting Your Request

The requests library prepares the request before sending

- Examples of items prepared
  - Validates headers
  - Serializes JSON content
- To view the PreparedRequest use the method `.request`
  - `response.request.url`
  - `response.request.body`

# End of Section 2 of 3

1. Getting Started with Requests
2. The GET Request
3. Processing Responses
4. Query String Parameters
5. Request Headers
6. Other HTTP Methods
7. The Message Body
8. Inspecting Your Request
-  9. **Authentication**
10. **SSL Certificate Verification**
11. **Performance**

# Welcome to Section 3 of 3

1. Getting Started with Requests
2. The GET Request
3. Processing Responses
4. Query String Parameters
5. Request Headers
6. Other HTTP Methods
7. The Message Body
8. Inspecting Your Request
- ▶ **9. Authentication**
- 10. SSL Certificate Verification**
- 11. Performance**



# Authentication

Passing credentials to a server

- Passed through the `Authorization:` header
- Requests parameter is `auth=`
  - username and password are passed as a tuple
  - Defaults to HTTP Basic authentication scheme
  - [https://en.wikipedia.org/wiki/Basic\\_access\\_authentication](https://en.wikipedia.org/wiki/Basic_access_authentication)

# SSL Certificate Verification

Requests verifies the target server's SSL Certificate by default

- A package is included when installing `requests` called `certifi`

# Performance

Keeping your application running smoothly

## 11. Performance

a. Timeouts



b. The Session Object

c. Max Retries

# Timeouts

How long to wait for a response before moving on

- Default - wait indefinitely
- Timeout parameter `timeout=`
  - Amount of time in seconds
  - Value can be an integer or float
- A tuple can be used
  - First value is how long to establish connection
  - Second value is how long to wait for the response

# The Session Object

Allows certain parameters to persist across multiple requests


- The underlying TCP connection will be reused = Performance!
- A session object has all the same methods as the main Requests API

# Max Retries

By default requests will not retry after a failed request

- A Transport Adapters can be mounted to a session
  - Allows a set of custom configurations per service
  - Parameter for `max_retries=`

# End of Section 3 of 3

1. Getting Started with Requests
2. The GET Request
3. Processing Responses
4. Query String Parameters
5. Request Headers
6. Other HTTP Methods
7. The Message Body
8. Inspecting Your Request
9. Authentication
10. SSL Certificate Verification
11. Performance
-  12. **Conclusion**