**Python Requests Tutorial**

Requests library is one of the integral parts of Python for making HTTP requests to a specified URL. Whether it be REST APIs or Web Scrapping, requests is must to be learned for proceeding further with these technologies. When one makes a request to a URI, it returns a response. Python requests provides inbuilt functionalities for managing both the request and response.

**Contents**

* [Why learn Python requests module?](https://www.geeksforgeeks.org/python-requests-tutorial/#why-learn)
* [Installing Requests](https://www.geeksforgeeks.org/python-requests-tutorial/#install-requests)
* [Making a Request](https://www.geeksforgeeks.org/python-requests-tutorial/#making-a-request)
* [Http Request Methods](https://www.geeksforgeeks.org/python-requests-tutorial/#http-request-methods)
* [Response object](https://www.geeksforgeeks.org/python-requests-tutorial/#response-object)
* [Response Methods](https://www.geeksforgeeks.org/python-requests-tutorial/#response-methods)
* [Authentication using Python Requests](https://www.geeksforgeeks.org/python-requests-tutorial/#authentication-using-python-requests)
* [SSL Certificate Verification](https://www.geeksforgeeks.org/python-requests-tutorial/#ssl-certificate-verification)
* [Session Objects](https://www.geeksforgeeks.org/python-requests-tutorial/#session-objects)

**Why learn Python requests module?**

* Requests is an Apache2 Licensed HTTP library, that allows to send HTTP/1.1 requests using Python.
* To play with web, Python Requests is must. Whether it be hitting APIs, downloading entire facebook pages, and much more cool stuff, one will have to make a request to the URL.
* Requests play a major role is dealing with [REST APIs](https://www.geeksforgeeks.org/rest-api-introduction/), and [Web Scrapping](https://www.geeksforgeeks.org/introduction-to-web-scraping/).
* Checkout an Example Python Script using Requests and Web Scrapping – [Implementing Web Scraping in Python with BeautifulSoup](https://www.geeksforgeeks.org/implementing-web-scraping-python-beautiful-soup/)

[**Recent Articles on Requests !!**](https://www.geeksforgeeks.org/tag/python-requests/)

**Installing Requests**

Requests installation depends on type of operating system on eis using, the basic command anywhere would be to open a command terminal and run,

pip install requests

The basic method for installation of requests on any operating system is to grab the base files and install requests manually and Requests is actively developed on GitHub, where the code is always available. For code – [visit here](https://github.com/psf/requests).  
You can either clone the public repository:

git clone git://github.com/psf/requests.git

Once you have a copy of the source, you can embed it in your own Python package, or install it into your site-packages easily:

cd requests

pip install **.**

For more checkout – [How to install requests in Python – For windows, linux, mac](https://www.geeksforgeeks.org/how-to-install-requests-in-python-for-windows-linux-mac/)

**Making a Request**

Python requests module has several built-in methods to make Http requests to specified URI using GET, POST, PUT, PATCH or HEAD requests. A Http request is meant to either retrieve data from a specified URI or to push data to a server. It works as a request-response protocol between a client and a server. Let’s demonstrate how to make a GET request to an endpoint.  
GET method is used to retrieve information from the given server using a given URI. The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the ‘?’ character.  
For example:

https://www.google.com/search?q=hello

**How to make GET request through Python Requests**

Python’s requests module provides in-built method called **get()** for making a GET request to a specified URI.

**Syntax –**

requests.get(url, params={key: value}, args)

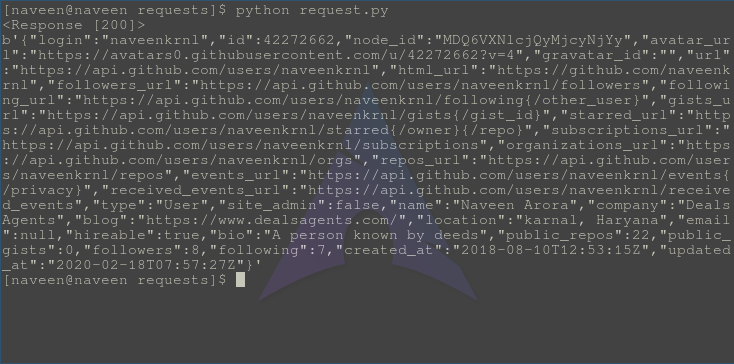
**Example –**

Let’s try making a request to github’s APIs for example purposes.

|  |
| --- |
| import requests    # Making a GET request  r = requests.get('<https://api.github.com/users/naveenkrnl>')    # check status code for response received  # success code - 200  print(r)    # print content of request  print(r.content) |

save this file as request.py and through terminal run,

python request.py

**Output –**  


For more, visit [GET method – Python requests](https://www.geeksforgeeks.org/get-method-python-requests/)

**Http Request Methods –**

| **Method** | **Description** |
| --- | --- |
| [GET](https://www.geeksforgeeks.org/get-method-python-requests/) | GET method is used to retrieve information from the given server using a given URI. |
| [POST](https://www.geeksforgeeks.org/post-method-python-requests/) | POST request method requests that a web server accepts the data enclosed in the body of the request message, most likely for storing it |
| [PUT](https://www.geeksforgeeks.org/put-method-python-requests/) | The PUT method requests that the enclosed entity be stored under the supplied URI. If the URI refers to an already existing resource, it is modified and if the URI does not point to an existing resource, then the server can create the resource with that URI. |
| [DELETE](https://www.geeksforgeeks.org/delete-method-python-requests/) | The DELETE method deletes the specified resource |
| [HEAD](https://www.geeksforgeeks.org/head-method-python-requests/) | The HEAD method asks for a response identical to that of a GET request, but without the response body. |
| [PATCH](https://www.geeksforgeeks.org/patch-method-python-requests/) | It is used for modify capabilities. The PATCH request only needs to contain the changes to the resource, not the complete resource |

**Response object**

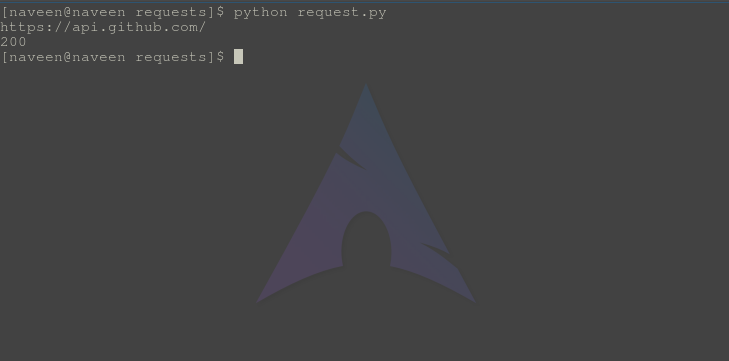
When one makes a request to a URI, it returns a response. This Response object in terms of python is returned by requests.method(), method being – get, post, put, etc. Response is a powerful object with lots of functions and attributes that assist in normalizing data or creating ideal portions of code. For example, **response.status\_code** returns the status code from the headers itself, and one can check if the request was processed successfully or not.  
Response object can be used to imply lots of features, methods, and functionalities.

**Example :**

|  |
| --- |
| # import requests module  import requests    # Making a get request  response = requests.get('<https://api.github.com/>')    # print request object  print(response.url)    # print status code  print(response.status\_code) |

Save this file as request.py, and run using below command

Python request.py

  
Status code 200 indicates that request was made successfully.

**Response Methods**

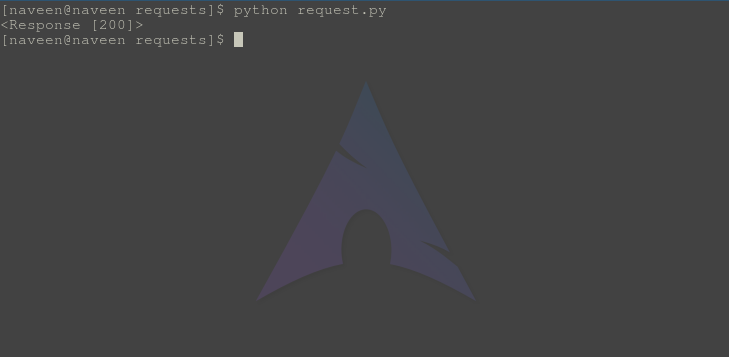
| **Method** | **Description** |
| --- | --- |
| [response.headers](https://www.geeksforgeeks.org/response-headers-python-requests/) | response.headers returns a dictionary of response headers. |
| [response.encoding](https://www.geeksforgeeks.org/response-encoding-python-requests/) | response.encoding returns the encoding used to decode response.content. |
| [response.elapsed](https://geeksforgeeks.org/response-elapsed-python-requests/) | response.elapsed returns a timedelta object with the time elapsed from sending the request to the arrival of the response. |
| [response.close()](https://www.geeksforgeeks.org/response-close-python-requests/) | response.close() closes the connection to the server. |
| [response.content](https://www.geeksforgeeks.org/response-content-python-requests/) | response.content returns the content of the response, in bytes. |
| [response.cookies](https://www.geeksforgeeks.org/response-cookies-python-requests/) | response.cookies returns a CookieJar object with the cookies sent back from the server. |
| [response.history](https://www.geeksforgeeks.org/response-history-python-requests/) | response.history returns a list of response objects holding the history of request (url). |
| [response.is\_permanent\_redirect](https://www.geeksforgeeks.org/response-is_permanent_redirect-python-requests/) | response.is\_permanent\_redirect returns True if the response is the permanent redirected url, otherwise False. |
| [response.is\_redirect](https://www.geeksforgeeks.org/response-is_redirect-python-requests/) | response.is\_redirect returns True if the response was redirected, otherwise False. |
| [response.iter\_content()](https://www.geeksforgeeks.org/response-iter_content-python-requests/) | response.iter\_content() iterates over the response.content. |
| [response.json()](https://www.geeksforgeeks.org/response-json-python-requests/) | response.json() returns a JSON object of the result (if the result was written in JSON format, if not it raises an error). |
| [response.url](https://www.geeksforgeeks.org/response-url-python-requests/) | response.url returns the URL of the response. |
| [response.text](https://www.geeksforgeeks.org/response-text-python-requests/) | response.text returns the content of the response, in unicode. |
| [response.status\_code](https://www.geeksforgeeks.org/response-status_code-python-requests/) | response.status\_code returns a number that indicates the status (200 is OK, 404 is Not Found). |
| [response.request](https://www.geeksforgeeks.org/response-request-python-requests/) | response.request returns the request object that requested this response. |
| [response.reason](https://www.geeksforgeeks.org/response-reason-python-requests/) | response.reason returns a text corresponding to the status code. |
| [response.raise\_for\_status()](https://www.geeksforgeeks.org/response-raise_for_status-python-requests/) | response.raise\_for\_status() returns an HTTPError object if an error has occurred during the process. |
| [response.ok](https://www.geeksforgeeks.org/response-ok-python-requests/) | response.ok returns True if status\_code is less than 200, otherwise False. |
| [response.links](https://www.geeksforgeeks.org/response-links-python-requests/) | response.links returns the header links. |

**Authentication using Python Requests**

Authentication refers to giving a user permissions to access a particular resource. Since, everyone can’t be allowed to access data from every URL, one would require authentication primarily. To achieve this authentication, typically one provides authentication data through Authorization header or a custom header defined by server.

**Example –**

|  |
| --- |
| # import requests module  import requests  from requests.auth import HTTPBasicAuth    # Making a get request  response = requests.get('<https://api.github.com> / user, ',              auth = HTTPBasicAuth('user', 'pass'))    # print request object  print(response) |

Replace “user” and “pass” with your username and password. It will authenticate the request and return a response 200 or else it will return error 403.  
  
For more visit – [Authentication using Python requests](https://www.geeksforgeeks.org/authentication-using-python-requests/)

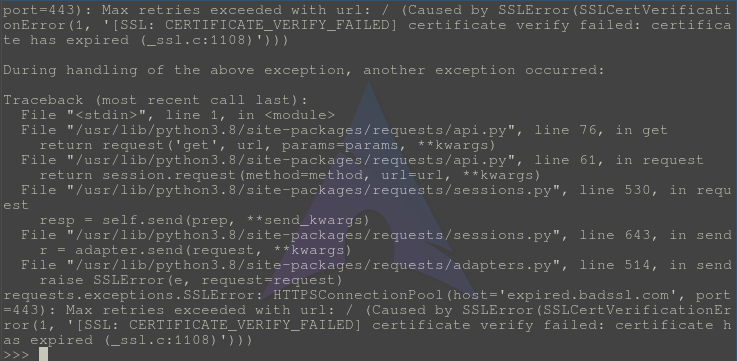
**SSL Certificate Verification**

Requests verifies SSL certificates for HTTPS requests, just like a web browser. SSL Certificates are small data files that digitally bind a cryptographic key to an organization’s details. Often, an website with a SSL certificate is termed as secure website. By default, SSL verification is enabled, and Requests will throw a SSLError if it’s unable to verify the certificate.

**Disable SSL certificate verification**

Let us try to access a website with an invalid SSL certificate, using Python requests

|  |
| --- |
| # import requests module  import requests    # Making a get request  response = requests.get('<https://expired.badssl.com/>')    # print request object  print(response) |

**Output :-**  
  
This website doesn’t have SSL setup so it raises this error.  
one can also pass the link to the certificate for validation via python requests only.

|  |
| --- |
| # import requests module  import requests    # Making a get request  response = requests.get('<https://github.com>', verify ='/path/to/certfile')    # print request object  print(response) |

This would work in case the path provided is correct for SSL certificate for github.com.  
For more visit- [SSL Certificate Verification – Python requests](https://www.geeksforgeeks.org/ssl-certificate-verification-python-requests/)

**Session Objects**

Session object allows one to persist certain parameters across requests. It also persists cookies across all requests made from the Session instance and will use urllib3’s connection pooling. So if several requests are being made to the same host, the underlying TCP connection will be reused, which can result in a significant performance increase. A session object all the methods as of requests.

**Using Session Objects**

Let us illustrate use of session objects by setting a cookie to a url and then making a request again to check if cookie is set.

|  |
| --- |
| # import requests module  import requests    # create a session object  s = requests.Session()    # make a get request  s.get('<https://httpbin.org/cookies/set/sessioncookie/123456789>')    # again make a get request  r = s.get('<https://httpbin.org/cookies>')    # check if cookie is still set  print(r.text) |

**Output:**  
