

Recon With Requests



Python Mauritius UserGroup

github.com/pymug

twitter.com/pymugdotcom

linkedin.com/company/pymug/

pymug.com

Web Scraping June 23rd Meetup 2019



Python Events @PythonEvents · Jun 13



New [#Python](#) [#User](#) [#Group](#) [#Event](#): Python Mauritius UserGroup (Pymug) June Meetup (June 23, 2019 at 06:00AM - June 23, 2019 at 10:00AM GMT) - bit.ly/2XISaNY



Recon With Requests



by Abdur-Rahmaan Janhangeer ([@osdotsystem](#))

About Requests (2.0)

- HTTP for Humans™

About Requests (2.0)

- HTTP for Humans™
- Armin Ronacher, creator of Flask

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

 What's in Requests?

What's in Requests?

- Quickstart
- Advanced Usage
- Authentication Part

1. Quickstart [1]

Make a Request

Passing Parameters In URLs

Response Content

Binary Response Content

JSON Response Content

Raw Response Content

Custom Headers

1. Quickstart [2]

More complicated POST requests

POST a Multipart-Encoded File

Response Status Codes

Response Headers

Cookies

Redirection and History

Timeouts

Errors and Exceptions

2. Advanced Usage [1]

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

2. Advanced Usage [2]

Event Hooks

Custom Authentication

Streaming Requests

Proxies

Compliance

HTTP Verbs

Custom Verbs

Link Headers

Transport Adapters

Blocking Or Non-Blocking?

Header Ordering

Timeouts

3. Authentication

Basic Authentication

Digest Authentication

OAuth 1 Authentication





OAuth 2 and OpenID Connect Authentication

Other Authentication

New Forms of Authentication

Our Project

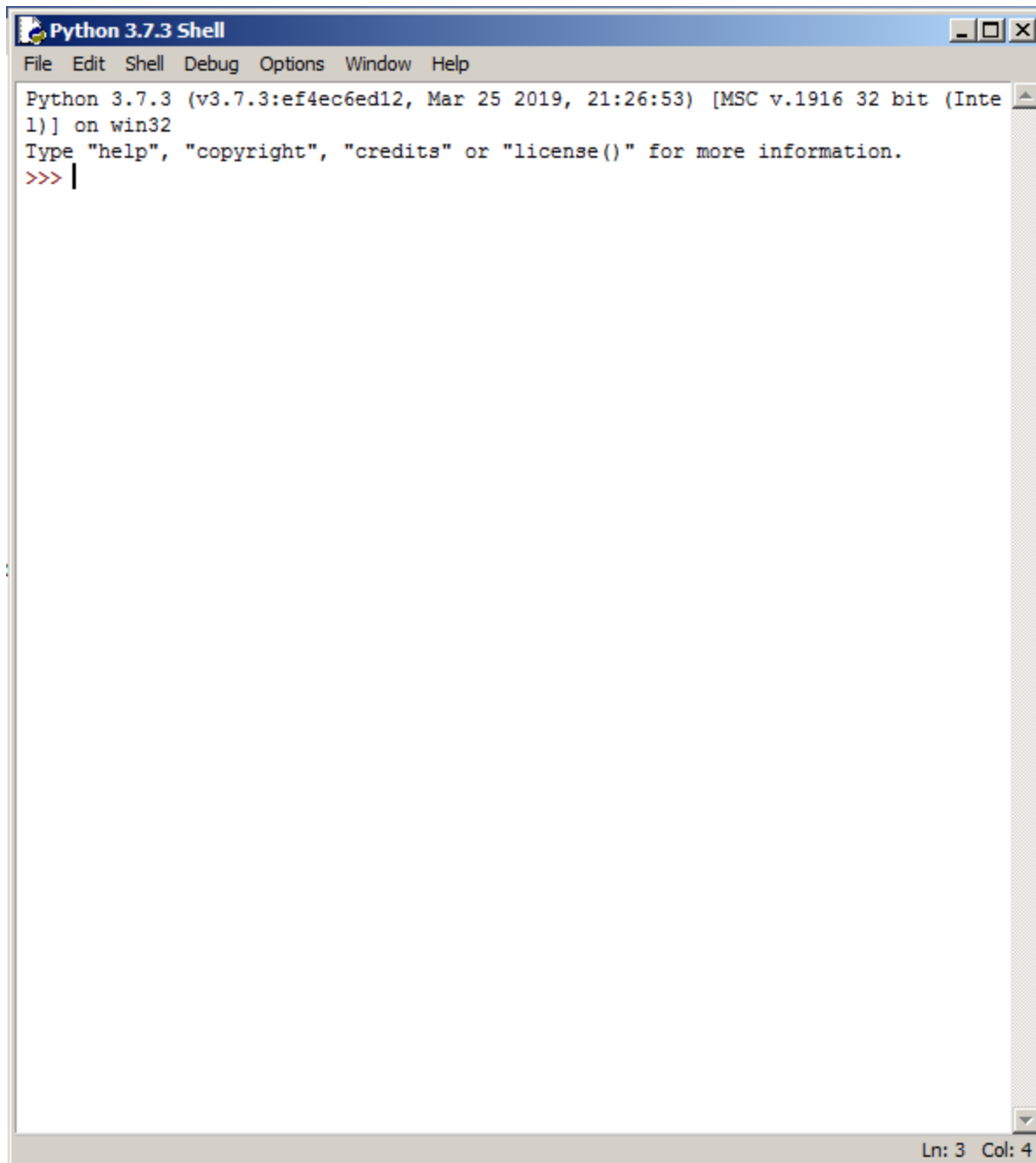
 pymug-june19/backend

-  media
-  static
-  templates
-  `app.py`

run `python app.py`


```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a
  production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 651-196-206
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

 **Our launchpad**

A screenshot of a Python 3.7.3 Shell window. The window has a title bar with the text "Python 3.7.3 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with the following items: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window is a text editor with a light gray background. It contains the following text:

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

The text is in a monospaced font. The prompt ">>>" is in red, and the cursor is a vertical line. At the bottom right of the window, there is a status bar that reads "Ln: 3 Col: 4".

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

Ln: 3 Col: 4

let's start with something simple

```
import requests  
requests.get('http://localhost:5000')
```

we get

```
<Response [200]>
```

```
*Python 3.7.3 Shell*
File Edit Shell Debug Options Window Help

Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import requests
>>> requests.get('http://localhost:5000')
<Response [200]>
>>> r = requests
>>> req = r.get('http://localhost:5000')
>>> dir(req)
['_attrs_', '__bool__', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__enter__', '__eq__', '__exit__', '__format__', '__ge__', '__getattribute__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__nonzero__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__setstate__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', '_content', '_content_consumed', '_next', 'apparent_encoding', 'close', 'connection', 'content', 'cookies', 'elapsed', 'encoding', 'headers', 'history', 'is_permanent_redirect', 'is_redirect', 'iter_content', 'iter_lines', 'json', 'links', 'next', 'ok', 'raise_for_status', 'raw', 'reason', 'request', 'status_code', 'text', 'url']
>>> req.text
'<!DOCTYPE html>\n<html>\n<head>\n\t<title></title>\n</head>\n<body>\n\t<p>landing page</p>\n</body>\n</html>'
>>>
```

Ln: 12 Col: 4

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help

Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import requests
>>> requests.get('http://localhost:5000')
<Response [200]>
>>> r = requests
>>> req = r.get('http://localhost:5000')
>>> dir(req)
['_attrs_', '__bool__', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__enter__', '__eq__', '__exit__', '__format__', '__ge__', '__getattribute__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__nonzero__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__setstate__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', 'content', 'content_consumed', 'next', 'apparent_encoding', 'close', 'connection', 'content', 'cookies', 'elapsed', 'encoding', 'headers', 'history', 'is_permanent_redirect', 'is_redirect', 'iter_content', 'iter_lines', 'json', 'links', 'next', 'ok', 'raise_for_status', 'raw', 'reason', 'request', 'status_code', 'text', 'url']
>>> req.text
'<!DOCTYPE html>\n<html>\n<head>\n\t<title></title>\n</head>\n<body>\n\t<p>landing page</p>\n</body>\n</html>'
>>> req.raw
<urllib3.response.HTTPResponse object at 0x026A7490>
>>> req.links
{}
>>> req.encoding
'utf-8'
>>> req.headers
{'Content-Type': 'text/html; charset=utf-8', 'Content-Length': '98', 'Server': 'Werkzeug/0.15.4 Python/3.7.3', 'Date': 'Thu, 20 Jun 2019 09:39:58 GMT'}
>>> req.status_code
200
>>> req.reason
'OK'
>>> req.request
<PreparedRequest [GET]>
>>> req.url
'http://localhost:5000/'
>>>
```

```
>>> req.content
b'<!DOCTYPE html>\n<html>\n<head>\n\t<title></title>\n</head>\n<body>\n\t<p>land
ing page</p>\n</body>\n</html>'
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(req.content, 'html.parser')
>>> soup.text
'\n\n\n\n\nlanding page\n\n'
>>> soup.text.strip('\n')
'landing page'
>>> soup.find('body').text
'\nlanding page\n'
>>> |
```

---S

Getting Json

we defined `r = requests` before

```
>>> req = r.get('http://localhost:5000/info')
>>> req.json()
{'date': None, 'host': 'localhost:5000', 'host url':
'http://localhost:5000/',
'ip': '127.0.0.1', 'path': '/info?', 'scheme': 'http'}
>>> info = req.json()
>>> info['scheme']
'http'
>>>
```

try with `/all_info`



Passing Parameters

let's get a simple greet message

```
>>> req = r.get('http://localhost:5000/greet')  
>>> req.text  
'hi user'
```

but we also have this url

```
http://localhost:5000/greet\_by\_name?name=mauritius
```

which gives

```
hi user mauritius
```

to pass parameters we do

```
>>> req = r.get('http://localhost:5000/greet_by_name',  
               params={'name': 'mee'})  
>>> req.text  
'hi user mee'
```



Post Requests

let's see the login page

```
>>> req = r.get('http://localhost:5000/login')
```

finding all forms

```
>>> soup = BeautifulSoup(req.content, 'html.parser')
>>> soup.find_all('form')
[<form action="/verify_login" method="post">
    user <br/>
    <input name="name" type="text"/><br/>
    password <br/>
    <input name="password" type="password"/> <br/>

    <input type="submit" value="enter"/>
</form>,
<form action="/verify_cookie_login" method="post">
    user <br/>
    <input name="name" type="text"/><br/>
    password <br/>
    <input name="password" type="password"/> <br/>
    <input type="submit" value="enter"/>
</form>]
```

```
<form action="/verify_login" method="post">
    user <br/>
    <input name="name" type=""/><br/>
    password <br/>
    <input name="password" type="password"/> <br/>

<input type="submit" value="enter"/>
</form>
```

we see we must send name - value and password_value to
/verify_login

```
>>> req = r.post('http://localhost:5000/verify_login',
                  data={'name': 'arj', 'password': '1234'})
>>> req.text
"great! you're in"
```



Cookies

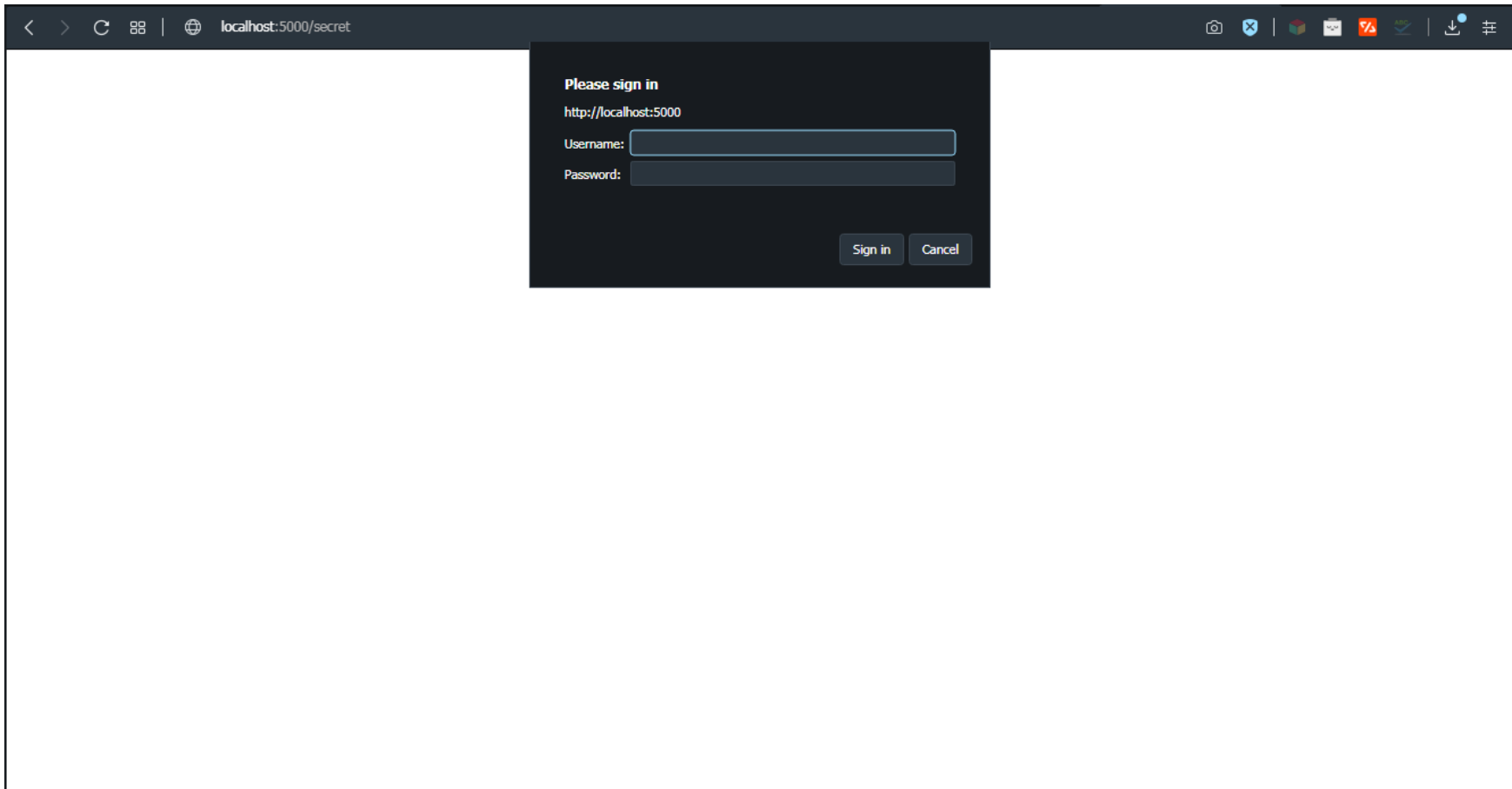
http://localhost:5000/verify_cookie_login

is authenticated by cookie

```
>>> s = requests.Session()
>>> req = s.post('http://localhost:5000/verify_cookie_login',
    data={'name': 'arj', 'password': '1234'})
>>> req.text
'<!DOCTYPE html>\n<html>\n<head>
\n\t<title></title>\n</head>\n<body>\n\t<p class="info">
there is no secret cookie</p>\n</body>\n</html>'
```


Basic Auth

basic auth is presented like that



```
>>> from requests.auth import HTTPBasicAuth
>>> auth = HTTPBasicAuth('arj', '1234')
>>> req = r.post(url="http://localhost:5000/secret", auth=
auth)
>>> req.text
'<!DOCTYPE html>\n<html>\n<head>\n\t<title>
</title>\n</head>\n<body>\n\tthere you
are!\n</body>\n</html>'
>>>
```

 Upload files

html

```
<form action="/upload_service" method = "POST"
      enctype="multipart/form-data">
  <input type="file" name="file_to_upload" />
  <input type="submit"/>
</form>
```

requests

```
>>> files = {'file': open('<path_to>/pymug-
june19/backend/pylogo.png', 'rb')}
>>> req = r.post("http://localhost:5000/upload_service",
    files=files)
>>> req.text
'file uploaded successfully'
```