

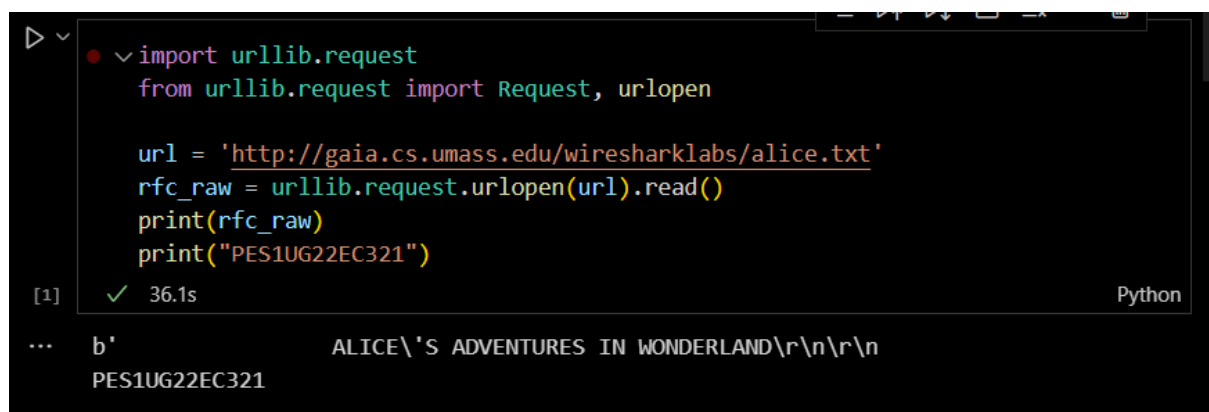
Using “urllib” library to fetch web objects and perform Wireshark analysis

Introduction: “Urllib” is a Python standard library recommended for HTTP tasks. The standard library also has a low-level module called http. Although this offers access to almost all aspects of the protocol, it has not been designed for everyday use. For making requests and receiving responses, we employ the urllib.request module. Retrieving the contents of a URL is a straightforward process when done using urllib.

Objectives: To use urllib package to extract the details such as headers and content in requests and responses under HTTP. To appreciate the advantage over socket package when dealing with HTTP. To compare and verify the results with Wireshark.

Request with urllib:

The urllib package is broken into several submodules for dealing with the different tasks that we may need to perform when working with HTTP. For making request and receiving responses, we employ the urllib.request module.



```
Python
> import urllib.request
  from urllib.request import Request, urlopen

url = 'http://gaia.cs.umass.edu/wiresharklabs/alice.txt'
rfc_raw = urllib.request.urlopen(url).read()
print(rfc_raw)
print("PES1UG22EC321")

[1] ✓ 36.1s

... b'          ALICE\'S ADVENTURES IN WONDERLAND\r\n\r\n
PES1UG22EC321
```

13	1.426531	192.168.1.5	151.101.2.132	HTTP	168 GET / HTTP/1.1
15	1.436056	151.101.2.132	192.168.1.5	HTTP	360 HTTP/1.1 302 Found

```

Hypertext Transfer Protocol
> HTTP/1.1 302 Found\r\n
  Connection: close\r\n
> Content-Length: 0\r\n
  Server: Varnish\r\n
  Retry-After: 0\r\n
  location: https://www.debian.org/\r\n
  Accept-Ranges: bytes\r\n
  Date: Sat, 02 Nov 2024 11:00:21 GMT\r\n
  Via: 1.1 varnish\r\n
  X-Served-By: cache-maa10243-MAA\r\n
  X-Cache: HIT\r\n
  X-Cache-Hits: 0\r\n
  X-Timer: S1730545222.990346,VS0,VE0\r\n
\r\n
[HTTP response 1/1]
[Time since request: 0.009525000 seconds]
[Request in frame: 13]
[Request URI: http://debian.org/]

```

```

Hypertext Transfer Protocol
> GET / HTTP/1.1\r\n
  Accept-Encoding: identity\r\n
  Host: debian.org\r\n
  User-Agent: Python-urllib/3.10\r\n
  Connection: close\r\n
\r\n
[Full request URI: http://debian.org/]
[HTTP request 1/1]
[Response in frame: 15]

```

We are sending a request and receiving a response from 'http://gaia.cs.umass.edu/wiresharklabs/alice.txt.org' using urlopen in the above example. Response.url prints the URL accessed . Readline reads the first line of the Html response

```

> print(response)
> print(response.readline())
> print(response.url)
> print(response.read(50))
> print(response.status)
> print(response.reason)
> print("PES1UG22EC321")

[4] ✓ 0.0s Python

... <http.client.HTTPResponse object at 0x0000020CA43EC040>
b''
http://gaia.cs.umass.edu/wiresharklabs/alice.txt
b''
200
OK
PES1UG22EC321

```

The response.readlines() reads the content line-by-line and response.read(50) reads the first 50 bytes of the Html response.

Status Code:

Status codes are integers that tell how the request went, different codes convey different meanings.

The status code 200 indicates the request was successful.

Handling HTTP and URL Errors

Status codes help us to see whether our response was successful or not. Any code in the 200 range indicates a success, whereas any code in either the 400 range or the 500 range

indicates failure. Status codes should always be checked so that our program can respond appropriately if something goes wrong. The urllib package helps us in checking the status codes by raising an exception if it encounters a problem.

```
try:
    urlopen('http://www.ietf.org/rfc/rfc0.txt')
except urllib.error.HTTPError as e:
    print('status', e.code)
    print('reason', e.reason)
    print('url', e.url)
    print("PES1UG22EC321")
```

[5] ✓ 1.9s Python

... status 404
reason Not Found
url <https://www.ietf.org/rfc/rfc0.txt>
PES1UG22EC321

114	1.409945	192.168.1.5	104.16.44.99	HTTP	182 GET /rfc/rfc0.txt HTTP/1.1
121	1.437218	104.16.44.99	192.168.1.5	HTTP	574 HTTP/1.1 301 Moved Permanently (text/html)

Hypertext Transfer Protocol

```
> HTTP/1.1 301 Moved Permanently\r\n
  Date: Sat, 02 Nov 2024 11:35:42 GMT\r\n
  Content-Type: text/html\r\n
> Content-Length: 167\r\n
  Connection: close\r\n
  Cache-Control: max-age=3600\r\n
  Expires: Sat, 02 Nov 2024 12:35:42 GMT\r\n
  Location: https://www.ietf.org/rfc/rfc0.txt\r\n
  Vary: Accept-Encoding\r\n
  Server: cloudflare\r\n
  CF-RAY: 8dc3d29bdb001d18-BLR\r\n
  alt-svc: h3=":443"; ma=86400\r\n
\r\n
[HTTP response 1/1]
[Time since request: 0.027273000 seconds]
\[Request in frame: 114\]
[Request URI: http://www.ietf.org/rfc/rfc0.txt]
File Data: 167 bytes
```

Line-based text data: text/html (7 lines)

```
<html>\r\n
<head><title>301 Moved Permanently</title></head>\r\n
<body>\r\n
<center><h1>301 Moved Permanently</h1></center>\r\n
<hr><center>cloudflare</center>\r\n
</body>\r\n
</html>\r\n
```

The code attempts to open a URL by using `urlopen('http://www.ietf.org/rfc/rfc0.txt')`, which targets the specified web address. If accessing the URL leads to an HTTP error, such as a 404 Not Found, 403 Forbidden, or 500 Internal Server Error, the program triggers an `HTTPError` exception. In the event of an `HTTPError`, the code catches the exception and displays relevant error information, including the HTTP status code (e.g., 404), the reason phrase corresponding to that code (e.g., "Not Found"), and the URL that caused the error.

```
urlopen('http://192.0.2.1/index.html')
21.1s
-----
TimeoutError                                Traceback (most recent call last)
File c:\Users\vijit\AppData\Local\Programs\Python\Python310\lib\urllib\request.py:1348, in AbstractHTTPHandler.do_open(self, http_class, req, **http_conn_args)
1347 try:
-> 1348     h.request(req.get_method(), req.selector, req.data, headers,
1349               encode_chunked=req.has_header('Transfer-encoding'))
1350 except OSError as err: # timeout error

File c:\Users\vijit\AppData\Local\Programs\Python\Python310\lib\http\client.py:1282, in HTTPConnection.request(self, method, url, body, headers, encode_chunked)
1281 """Send a complete request to the server."""
-> 1282 self._send_request(method, url, body, headers, encode_chunked)

File c:\Users\vijit\AppData\Local\Programs\Python\Python310\lib\http\client.py:1328, in HTTPConnection._send_request(self, method, url, body, headers, encode_chunked)
1327     body = _encode(body, 'body')
-> 1328 self.endheaders(body, encode_chunked=encode_chunked)

File c:\Users\vijit\AppData\Local\Programs\Python\Python310\lib\http\client.py:1277, in HTTPConnection.endheaders(self, message_body, encode_chunked)
1276     raise CannotSendHeader()
-> 1277 self._send_output(message_body, encode_chunked=encode_chunked)

File c:\Users\vijit\AppData\Local\Programs\Python\Python310\lib\http\client.py:1037, in HTTPConnection._send_output(self, message_body, encode_chunked)
1036 del self._buffer[:]
-> 1037 self.send(msg)
1038 if message_body is not None:
1039     ...
...
-> 1351     raise URLError(err)
1352     r = h.getresponse()
1353 except:
URLError: curl error [WinError 10060] A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed beca
```

In this instance, we have asked for index.html from the 192.0.2.1. host. The 192.0.2.0/24 IP address range is reserved to be used by documentation only, so you will never encounter a host using the preceding IP address. Hence the TCP connection times out and socket raises a timeout exception, which urllib catches, re-wraps, and re-raises for us. HTTPError occurs when the request reaches the server, but the server returns an error (e.g., 404 Not Found). URLError occurs if the request cannot connect to the server (e.g., connection timeout).

HTTP headers:

Headers are the lines of protocol-specific information that appear at the beginning of the raw message that is sent over the TCP connection. The getheaders() method returns a list of response headers, which contain metadata such as server information and content type. It displays a list of headers as (name, value) pairs.

```
response = urlopen('http://www.debian.org')
response.getheaders()
✓ 1.0s

[('Date', 'Sat, 02 Nov 2024 11:11:25 GMT'),
 ('Server', 'Apache'),
 ('Content-Location', 'index.en.html'),
 ('Vary', 'negotiate,accept-language,Accept-Encoding,cookie'),
 ('TCN', 'choice'),
 ('X-Content-Type-Options', 'nosniff'),
 ('X-Frame-Options', 'sameorigin'),
 ('Referrer-Policy', 'no-referrer'),
 ('X-Xss-Protection', '1'),
 ('Permissions-Policy', 'interest-cohort=()'),
 ('Strict-Transport-Security', 'max-age=15552000'),
 ('Upgrade', 'h2,h2c'),
 ('Connection', 'Upgrade, close'),
 ('Last-Modified', 'Sat, 02 Nov 2024 03:25:55 GMT'),
 ('ETag', '"3b38-625e59c0574d9"'),
 ('Accept-Ranges', 'bytes'),
 ('Content-Length', '15160'),
 ('Cache-Control', 'max-age=86400'),
 ('Expires', 'Sun, 03 Nov 2024 11:11:25 GMT'),
 ('X-Clacks-Overhead', 'GNU Terry Pratchett'),
 ('Content-Type', 'text/html'),
 ('Content-Language', 'en')]
```

105	2.019068	192.168.1.5	130.89.148.77	HTTP	172	GET / HTTP/1.1	
Frame 105:	172 bytes on wire (1376 bits), 172 bytes captured (1376 bits) on interface \Device\NPF_{7F...}						
Ethernet II, Src: MicroStarINT_f7:e9:a6 (d8:bb:c1:f7:e9:a6), Dst: TaicangT&WEI_67:35:e0 (5c:f9:fd:67:...							
Internet Protocol Version 4, Src: 192.168.1.5, Dst: 130.89.148.77							
Transmission Control Protocol, Src Port: 58446, Dst Port: 80, Seq: 1, Ack: 1, Len: 118							
Hypertext Transfer Protocol							
> GET / HTTP/1.1\r\n							
Accept-Encoding: identity\r\n							
Host: www.debian.org\r\n							
User-Agent: Python-urllib/3.10\r\n							
Connection: close\r\n							
\r\n							
[Full request URI: http://www.debian.org/]							
[HTTP request 1/1]							
[Response in frame: 114]							

Customizing Requests with Headers:

```

> req = Request('http://www.debian.org')
  req.add_header('Accept-Language', 'sv')
  print('Headers we added: ', req.header_items())
  print("PES1UG22EC321")
[6] ✓ 0.0s Ruby
... Headers we added: [('Accept-language', 'sv')]
PES1UG22EC321

```

The above code adds the Accept-Language header to request content in Swedish. If the server supports it, it returns the content in the requested language. It displays the first 5 lines of content in Swedish.

```

headers = {'Accept-Language': 'sv', 'Connection': 'keep-alive'}
req = Request('http://www.debian.org', headers=headers)
print('Headers we added: ', req.header_items())

response = urlopen(req)
print('Headers urlopen() added: ', req.header_items())
print('Server\'s response: ', response.readlines()[:5])
print("PES1UG22EC321")
[7] ✓ 3.5s Ruby
... Headers we added: [('Accept-language', 'sv'), ('Connection', 'keep-alive')]
Headers urlopen() added: [('Host', 'www.debian.org'), ('User-agent', 'Python-urllib')]
Server's response: [b'<!DOCTYPE html>\n', b'<html lang="sv">\n', b'<head>\n', b'
PES1UG22EC321

```

The above code is used to view the headers present in the request and creates a Request object for the Debian homepage, adding headers directly during initialization. Headers are supplied as a dictionary to the Request object constructor as the headers keyword argument. In this way we can add multiple headers in one go, by adding more entries to the dictionary.

Content Compression:

The Accept-Encoding request header and the Content-Encoding response header work together to temporarily encode the body of a response for efficient transmission over the network, primarily for compression purposes. The process begins when the client sends a request that lists acceptable encodings in the Accept-Encoding header. The server then selects a supported encoding method and applies it to the response body. Subsequently, the server sends the response back to the client, indicating the chosen encoding method in the Content-Encoding header. Finally, the client decodes the response body using the specified encoding method, enabling reduced data transfer and improved performance.

```
# Example 1: Using gzip encoding

req = Request('http://www.debian.org')
req.add_header('Accept-Encoding', 'gzip')
response = urlopen(req)
print('The encoding supported by server is ', response.getheader('Content-Encoding'))
print("PES1UG22EC321")
```

[8] ✓ 2.1s Ruby

... The encoding supported by server is gzip
PES1UG22EC321

	Time	Source	Destination	Protocol	Length	Info
245	4.330759	192.168.1.5	130.89.148.77	HTTP	168	GET / HTTP/1.1
247	4.471800	130.89.148.77	192.168.1.5	HTTP	673	HTTP/1.1 302 Found (text/html)

Hypertext Transfer Protocol

```
✓ GET / HTTP/1.1\r\n
  > [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /
    Request Version: HTTP/1.1
  Host: www.debian.org\r\n
  User-Agent: Python-urllib/3.10\r\n
  Accept-Encoding: gzip\r\n
  Connection: close\r\n
  \r\n
  [Full request URI: http://www.debian.org/]
  [HTTP request 1/1]
```

Decompressing Gzip

```
# Decompress using gzip
import gzip
content = gzip.decompress(response.read())
print('Server\'s response: ', content.splitlines()[5])
print("PES1UG22EC321")
```

✓ 0.0s Ruby

Server's response: [b'<!DOCTYPE html>', b'<html lang="en">', b'<head>', b'<meta charset="utf-8' PES1UG22EC321

```

Hypertext Transfer Protocol
  HTTP/1.1 302 Found\r\n
    > [Expert Info (Chat/Sequence): HTTP/1.1 302 Found\r\n]
      Response Version: HTTP/1.1
      Status Code: 302
      [Status Code Description: Found]
      Response Phrase: Found
    Date: Sat, 02 Nov 2024 11:57:30 GMT\r\n
    Server: Apache\r\n
    X-Content-Type-Options: nosniff\r\n
    X-Frame-Options: sameorigin\r\n
    Referrer-Policy: no-referrer\r\n
    X-Xss-Protection: 1\r\n
    Permissions-Policy: interest-cohort=()\r\n
    Location: https://www.debian.org/\r\n
  > Content-Length: 271\r\n
  Connection: close\r\n
  Content-Type: text/html; charset=iso-8859-1\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.141041000 seconds]
  [Request in frame: 245]
  [Request URI: http://www.debian.org/]
  File Data: 271 bytes
Line-based text data: text/html (9 lines)
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">\n
<html><head>\n
<title>302 Found</title>\n
</head><body>\n
<h1>Found</h1>\n
<p>The document has moved <a href="https://www.debian.org/">here</a>.</p>\n
<hr>\n
<address>Apache Server at www.debian.org Port 80</address>\n
</body></html>\n

```

The Accept-Encoding header requests gzip-compressed content. The response is decoded using the gzip module to access readable content. Displays the content-encoding type and the first few lines of decompressed content.

```

# Example 1: Using gzip encoding

req = Request('http://www.debian.org')
req.add_header('Accept-Encoding', 'gzip')
response = urlopen(req)
print('The encoding supported by server is ', response.getheader('Content-Encoding'))
print("PES1UG22EC321")

```

[8] ✓ 2.1s Ruby

```

... The encoding supported by server is  gzip
PES1UG22EC321

```

No.	Time	Source	Destination	Protocol	Length	Info
207	3.092793	192.168.1.5	130.89.148.77	HTTP	172	GET / HTTP/1.1
216	3.245095	130.89.148.77	192.168.1.5	HTTP	673	HTTP/1.1 302 Found (text/html)

```

Hypertext Transfer Protocol
  HTTP/1.1 302 Found\r\n
    > [[Expert Info (Chat/Sequence): HTTP/1.1 302 Found\r\n]
      Response Version: HTTP/1.1
      Status Code: 302
      [Status Code Description: Found]
      Response Phrase: Found
      Date: Sat, 02 Nov 2024 12:01:51 GMT\r\n
      Server: Apache\r\n
      X-Content-Type-Options: nosniff\r\n
      X-Frame-Options: sameorigin\r\n
      Referrer-Policy: no-referrer\r\n
      X-Xss-Protection: 1\r\n
      Permissions-Policy: interest-cohort=()\r\n
      Location: https://www.debian.org/\r\n
    > Content-Length: 271\r\n
      Connection: close\r\n
      Content-Type: text/html; charset=iso-8859-1\r\n
      \r\n
      [HTTP response 1/1]
      [Time since request: 0.152302000 seconds]
      [Request in frame: 207]
      [Request URI: http://www.debian.org/]
      File Data: 271 bytes
Line-based text data: text/html (9 lines)
  <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">\n
  <html><head>\n
  <title>302 Found</title>\n
  </head><body>\n
  <h1>Found</h1>\n
  <p>The document has moved <a href="https://www.debian.org/">here</a>.</p>\n
  <hr>\n
  <address>Apache Server at www.debian.org Port 80</address>\n
  </body></html>\n

```

Multiple Values

To tell the server that we can accept more than one encoding, add more values to the Accept-Encoding header and separate them by commas.

Multiple encoding Preferences

Multiple encodings are specified, with q values indicating preference. gzip is preferred, followed by deflate with lower priority, and identity with the lowest.


```
# Example 2: Sending multiple encodings with priority specified by q in [0,1]
req = Request('http://www.debian.org')
encodings = 'deflate;q=0.8, gzip, identity;q=0.0'
req.add_header('Accept-Encoding', encodings)
response = urlopen(req)
print('The encoding supported by server is ', response.getheader('Content-Encoding'))
print('The content type returned by server is ', response.getheader('Content-Type'))
print('The content length (in bytes) returned by server is ', response.getheader('Content-Length'))
print('The object was last modified at the server at ', response.getheader('Last-Modified'))
print("PES1UG22EC321")
```

✓ 2.5s

Ruby

```
The encoding supported by server is  gzip
The content type returned by server is  text/html
The content length (in bytes) returned by server is  4085
The object was last modified at the server at  Sat, 02 Nov 2024 03:25:55 GMT
PES1UG22EC321
```

Above image shows Reading Content-Type and Charset

101	1.446237	192.168.1.5	194.177.211.2...	HTTP	172	GET / HTTP/1.1
112	1.620934	194.177.211.2...	192.168.1.5	HTTP	673	HTTP/1.1 302 Found (text/html)

```
Hypertext Transfer Protocol
  HTTP/1.1 302 Found\r\n
    [Expert Info (Chat/Sequence): HTTP/1.1 302 Found\r\n]
      Response Version: HTTP/1.1
      Status Code: 302
      [Status Code Description: Found]
      Response Phrase: Found
      Date: Sat, 02 Nov 2024 12:08:12 GMT\r\n
      Server: Apache\r\n
      X-Content-Type-Options: nosniff\r\n
      X-Frame-Options: sameorigin\r\n
      Referrer-Policy: no-referrer\r\n
      X-Xss-Protection: 1\r\n
      Permissions-Policy: interest-cohort=()\r\n
      Location: https://www.debian.org/\r\n
    > Content-Length: 271\r\n
    Connection: close\r\n
    Content-Type: text/html; charset=iso-8859-1\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.174697000 seconds]
    [Request in frame: 101]
    [Request URI: http://www.debian.org/]
    File Data: 271 bytes
```

Line-based text data: text/html (9 lines)

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">\n
<html><head>\n
<title>302 Found</title>\n
</head><body>\n
<h1>Found</h1>\n
<p>The document has moved <a href="https://www.debian.org/">here</a>.</p>\n
<hr>\n
<address>Apache Server at www.debian.org Port 80</address>\n
</body></html>\n
```

The Content-Type header informs the client of the content type (e.g.,text/html). The charset parameter provides the encoding (UTF-8 in this case), which is used to decode the content.

```

# Example 3: To see the charset returned along with Content-Type
response = urlopen('http://www.python.org')
format, params = response.getheader('Content-Type').split(';')
print(format)
print(params)
charset = params.split('=')[1]
print(charset)
print('The HTML file returned by the server after decoding as per charset is given by \n', response.read().decode(charset))
print("PES1UG22EC321")

```

[12] ✓ 1.8s Ruby

```

...
text/html
charset=utf-8
utf-8
The HTML file returned by the server after decoding as per charset is given by
<!doctype html>
<!--[if lt IE 7]> <html class="no-js ie6 lt-ie7 lt-ie8 lt-ie9"> <![endif]-->
<!--[if IE 7]> <html class="no-js ie7 lt-ie8 lt-ie9"> <![endif]-->
<!--[if IE 8]> <html class="no-js ie8 lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!--><html class="no-js" lang="en" dir="ltr"> <!--<![endif]-->

<head>
  <!-- Google tag (gtag.js) -->
  <script async src="https://www.googletagmanager.com/gtag/js?id=G-TF35YF9CVH"></script>
  <script>
    window.dataLayer = window.dataLayer || [];
    function gtag(){dataLayer.push(arguments);}
    gtag('js', new Date());
    gtag('config', 'G-TF35YF9CVH');
  </script>
  <!-- Plausible.io analytics -->
  <script defer data-domain="python.org" src="https://plausible.io/js/script.js"></script>

  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  ...
</body>
</html>

PES1UG22EC321

```

51	1.382902	192.168.1.5	151.101.156.2...	HTTP	172 GET / HTTP/1.1
53	1.393694	151.101.156.2...	192.168.1.5	HTTP	445 HTTP/1.1 301 Moved Permanently

```

▼ Hypertext Transfer Protocol
  ▼ HTTP/1.1 301 Moved Permanently\r\n
    > [Expert Info (Chat/Sequence): HTTP/1.1 301 Moved Permanently\r\n]
      Response Version: HTTP/1.1
      Status Code: 301
      [Status Code Description: Moved Permanently]
      Response Phrase: Moved Permanently
    Connection: close\r\n
  > Content-Length: 0\r\n
  Server: Varnish\r\n
  Retry-After: 0\r\n
  Location: https://www.python.org/\r\n
  Accept-Ranges: bytes\r\n
  Date: Sat, 02 Nov 2024 12:12:52 GMT\r\n
  Via: 1.1 varnish\r\n
  X-Served-By: cache-maa10250-MAA\r\n
  X-Cache: HIT\r\n
  X-Cache-Hits: 0\r\n
  X-Timer: S1730549572.329229,V50,VE0\r\n
  Strict-Transport-Security: max-age=63072000; includeSubDomains; preload\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.010792000 seconds]
  [Request in frame: 51]
  [Request URI: http://www.python.org/]

```

User Agents

Any client that communicates using HTTP can be referred to as a user agent. RFC 7231 suggests that user agents should use the User-Agent header to identify themselves in every request.

```
# To see the user-agent
req = Request('http://www.python.org')
urlopen(req)
print('The user agent is ', req.get_header('User-agent'))
print("PES1UG22EC321")
```

[13] ✓ 1.5s

... The user agent is Python-urllib/3.10
PES1UG22EC321

27	1.562905	192.168.1.5	151.101.156.2...	HTTP	172	GET / HTTP/1.1
29	1.573025	151.101.156.2...	192.168.1.5	HTTP	445	HTTP/1.1 301 Moved Permanently

▼ Hypertext Transfer Protocol

▼ GET / HTTP/1.1\r\n

> [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]

Request Method: GET

Request URI: /

Request Version: HTTP/1.1

Accept-Encoding: identity\r\n

Host: www.python.org\r\n

User-Agent: Python-urllib/3.10\r\n

Connection: close\r\n

\r\n

[\[Full request URI: http://www.python.org/\]](http://www.python.org/)

[HTTP request 1/1]

[\[Response in frame: 29\]](#)

▼ Hypertext Transfer Protocol

▼ HTTP/1.1 301 Moved Permanently\r\n

> [Expert Info (Chat/Sequence): HTTP/1.1 301 Moved Permanently\r\n]

Response Version: HTTP/1.1

Status Code: 301

[\[Status Code Description: Moved Permanently\]](#)

Response Phrase: Moved Permanently

Connection: close\r\n

> Content-Length: 0\r\n

Server: Varnish\r\n

Retry-After: 0\r\n

Location: https://www.python.org/\r\n

Accept-Ranges: bytes\r\n

Date: Sat, 02 Nov 2024 12:15:01 GMT\r\n

Via: 1.1 varnish\r\n

X-Served-By: cache-maa10234-MAA\r\n

X-Cache: HIT\r\n

X-Cache-Hits: 0\r\n

X-Timer: S1730549702.856490,VS0,VE0\r\n

Strict-Transport-Security: max-age=63072000; includeSubDomains; preload\r\n

\r\n

[HTTP response 1/1]

[Time since request: 0.010120000 seconds]

[\[Request in frame: 27\]](#)

[Request URI: http://www.python.org/]

Webmasters often inspect the user agents of requests to gather insights for various purposes, such as classifying website visits for statistical analysis, blocking clients with specific user agent strings, and sending alternative versions of resources to user agents known to have issues—like bugs in CSS interpretation or lack of support for languages like JavaScript. However, these practices can hinder access to desired content. To circumvent such restrictions, users can modify their user agent to mimic that of a well-known browser, a technique referred to as spoofing.

```

# Some webmasters can block specific user agents so let us do spoofing as Firefox client
req = Request('http://www.debian.org')
req.add_header('User-Agent', 'Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20140722 Firefox/24.0 Iceweasel/24.7.0')
response = urlopen(req)
print(response.read())
print("PES1UG22EC321")

```

[14] ✓ 2.65 Ruby

... b'<!DOCTYPE html>\n<html lang="en">\n<head>\n<meta charset="utf-8">\n<title>Debian -- The Universal Operating System PES1UG22EC321

→	22	1.460789	192.168.1.5	130.89.148.77	HTTP	238	GET / HTTP/1.1
←	24	1.600205	130.89.148.77	192.168.1.5	HTTP	673	HTTP/1.1 302 Found (text/html)

Hypertext Transfer Protocol

✓ GET / HTTP/1.1\r\n

> [[Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]

Request Method: GET

Request URI: /

Request Version: HTTP/1.1

Accept-Encoding: identity\r\n

Host: www.debian.org\r\n

User-Agent: Mozilla/5.0 (X11;Linux x86_64; rv:24.0) Gecko/20140722 Firefox/24.0 Iceweasel/24 7.0\r\n

Connection: close\r\n

\r\n

[Full request URI: http://www.debian.org/]

[HTTP request 1/1]

[Response in frame: 24]

Hypertext Transfer Protocol

✓ HTTP/1.1 302 Found\r\n

> [[Expert Info (Chat/Sequence): HTTP/1.1 302 Found\r\n]

Response Version: HTTP/1.1

Status Code: 302

[Status Code Description: Found]

Response Phrase: Found

Date: Sat, 02 Nov 2024 12:19:47 GMT\r\n

Server: Apache\r\n

X-Content-Type-Options: nosniff\r\n

X-Frame-Options: sameorigin\r\n

Referrer-Policy: no-referrer\r\n

X-Xss-Protection: 1\r\n

Permissions-Policy: interest-cohort=()\r\n

Location: https://www.debian.org/\r\n

> Content-Length: 271\r\n

Connection: close\r\n

Content-Type: text/html; charset=iso-8859-1\r\n

\r\n

[HTTP response 1/1]

[Time since request: 0.139416000 seconds]

[Request in frame: 22]

[Request URI: http://www.debian.org/]

File Data: 271 bytes

Line-based text data: text/html (9 lines)

```

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">\n
<html><head>\n
<title>302 Found</title>\n
</head><body>\n
<h1>Found</h1>\n
<p>The document has moved <a href="https://www.debian.org/">here</a>.</p>\n
<hr>\n
<address>Apache Server at www.debian.org Port 80</address>\n
</body></html>\n

```

Conclusion: The python codes have been executed and the HTTP packets have been captured. In conclusion, using the urllib package to handle HTTP requests and responses provides a more efficient and user-friendly approach compared to the socket package.