

http.cookies — HTTP Cookies

Purpose: Defines classes for parsing and creating HTTP cookie headers.

The `http.cookies` module implements a parser for cookies that is mostly [RFC 2109](#) compliant. The implementation is a little less strict than the standard because MSIE 3.0x does not support the entire standard.

Creating and Setting a Cookie

Cookies are used as state management for browser-based application, and as such are usually set by the server to be stored and returned by the client. The most trivial example of creating a cookie sets a single name-value pair.

```
# http_cookies_setheaders.py

from http import cookies

c = cookies.SimpleCookie()
c['mycookie'] = 'cookie_value'
print(c)
```

The output is a valid Set-Cookie header ready to be passed to the client as part of the HTTP response.

```
$ python3 http_cookies_setheaders.py

Set-Cookie: mycookie=cookie_value
```

Morsels

It is also possible to control the other aspects of a cookie, such as the expiration, path, and domain. In fact, all of the RFC attributes for cookies can be managed through the `Morsel` object representing the cookie value.

```
# http_cookies_Morsel.py

from http import cookies
import datetime

def show_cookie(c):
    print(c)
    for key, morsel in c.items():
        print()
        print('key =', morsel.key)
        print('  value =', morsel.value)
        print('  coded_value =', morsel.coded_value)
        for name in morsel.keys():
            if morsel[name]:
                print('  {} = {}'.format(name, morsel[name]))

c = cookies.SimpleCookie()

# A cookie with a value that has to be encoded
# to fit into the header
c['encoded_value_cookie'] = '"cookie,value;"'
c['encoded_value_cookie']['comment'] = 'Has escaped punctuation'

# A cookie that only applies to part of a site
c['restricted_cookie'] = 'cookie_value'
c['restricted_cookie']['path'] = '/sub/path'
c['restricted_cookie']['domain'] = 'PyMOTW'
c['restricted_cookie']['secure'] = True
```

```

# A cookie that expires in 5 minutes
c['with_max_age'] = 'expires in 5 minutes'
c['with_max_age']['max-age'] = 300 # seconds

# A cookie that expires at a specific time
c['expires_at_time'] = 'cookie_value'
time_to_live = datetime.timedelta(hours=1)
expires = (datetime.datetime(2009, 2, 14, 18, 30, 14) +
           time_to_live)

# Date format: Wdy, DD-Mon-YY HH:MM:SS GMT
expires_at_time = expires.strftime('%a, %d %b %Y %H:%M:%S')
c['expires_at_time']['expires'] = expires_at_time

show_cookie(c)

```

This example includes two different methods for setting stored cookies that expire. One sets the max-age to a number of seconds, the other sets expires to a date and time when the cookie should be discarded.

```

$ python3 http_cookies_Morsel.py

Set-Cookie: encoded_value_cookie="\\"cookie\054value\073\\"";
Comment="Has escaped punctuation"
Set-Cookie: expires_at_time=cookie_value; expires=Sat, 14 Feb
2009 19:30:14
Set-Cookie: restricted_cookie=cookie_value; Domain=PyMOTW;
Path=/sub/path; Secure
Set-Cookie: with_max_age="expires in 5 minutes"; Max-Age=300

key = encoded_value_cookie
value = "cookie,value;"
coded_value = "\\"cookie\054value\073\\"
comment = Has escaped punctuation

key = restricted_cookie
value = cookie_value
coded_value = cookie_value
path = /sub/path
domain = PyMOTW
secure = True

key = with_max_age
value = expires in 5 minutes
coded_value = "expires in 5 minutes"
max-age = 300

key = expires_at_time
value = cookie_value
coded_value = cookie_value
expires = Sat, 14 Feb 2009 19:30:14

```

Both the Cookie and Morsel objects act like dictionaries. A Morsel responds to a fixed set of keys:

- expires
- path
- comment
- domain
- max-age
- secure
- version

The keys for a Cookie instance are the names of the individual cookies being stored. That information is also available from the key attribute of the Morsel.

Encoded Values

The cookie header needs values to be encoded so they can be parsed properly.

```

# http_cookies_coded_value.py

```

```

from http import cookies

c = cookies.SimpleCookie()
c['integer'] = 5
c['with_quotes'] = 'He said, "Hello, World!'"

for name in ['integer', 'with_quotes']:
    print(c[name].key)
    print(' {} '.format(c[name]))
    print(' value={!r}'.format(c[name].value))
    print(' coded_value={!r}'.format(c[name].coded_value))
    print()

```

Morsel.value is always the decoded value of the cookie, while Morsel.coded_value is always the representation to be used for transmitting the value to the client. Both values are always strings. Values saved to a cookie that are not strings are converted automatically.

```

$ python3 http_cookies_coded_value.py

integer
Set-Cookie: integer=5
value='5'
coded_value='5'

with_quotes
Set-Cookie: with_quotes="He said\054 \"Hello\054 World!\\""
value='He said, "Hello, World!'"
coded_value='"He said\\054 \\\"Hello\\054 World!\\\""'

```

Receiving and Parsing Cookie Headers

Once the Set-Cookie headers are received by the client, it will return those cookies to the server on subsequent requests using a Cookie header. An incoming Cookie header string may contain several cookie values, separated by semicolons (;).

```

Cookie: integer=5; with_quotes="He said, \"Hello, World!\\""

```

Depending on the web server and framework, cookies are either available directly from the headers or the HTTP_COOKIE environment variable.

```

# http_cookies_parse.py

from http import cookies

HTTP_COOKIE = '; '.join([
    r'integer=5',
    r'with_quotes="He said, \"Hello, World!\\""',
])

print('From constructor:')
c = cookies.SimpleCookie(HTTP_COOKIE)
print(c)

print()
print('From load():')
c = cookies.SimpleCookie()
c.load(HTTP_COOKIE)
print(c)

```

To decode them, pass the string without the header prefix to SimpleCookie when instantiating it, or use the load() method.

```

$ python3 http_cookies_parse.py

From constructor:
Set-Cookie: integer=5
Set-Cookie: with_quotes="He said, \"Hello, World!\\""

```

```
From load():
Set-Cookie: integer=5
Set-Cookie: with_quotes="He said, \"Hello, World!\""
```

Alternative Output Formats

Besides using the Set-Cookie header, servers may deliver JavaScript that adds cookies to a client. SimpleCookie and Morsel provide JavaScript output via the `js_output()` method.

```
# http_cookies_js_output.py

from http import cookies
import textwrap

c = cookies.SimpleCookie()
c['mycookie'] = 'cookie_value'
c['another_cookie'] = 'second value'
js_text = c.js_output()
print(textwrap.dedent(js_text).lstrip())
```

The result is a complete script tag with statements to set the cookies.

```
$ python3 http_cookies_js_output.py

<script type="text/javascript">
<!-- begin hiding
document.cookie = "another_cookie=\"second value\"";
// end hiding -->
</script>

<script type="text/javascript">
<!-- begin hiding
document.cookie = "mycookie=cookie_value";
// end hiding -->
</script>
```

See also

- [Standard library documentation for http.cookies](#)
- `http.cookiejar` – The `cookielib` module, for working with cookies on the client-side.
- [RFC 2109](#) – HTTP State Management Mechanism

Quick Links

- [Creating and Setting a Cookie](#)
- [Morsels](#)
- [Encoded Values](#)
- [Receiving and Parsing Cookie Headers](#)
- [Alternative Output Formats](#)

This page was last updated 2018-12-09.

Navigation

- [http.server](#) — Base Classes for Implementing Web Servers
- [webbrowser](#) — Displays web pages





[Get the book](#)

The output from all the example programs from PyMOTW-3 has been generated with Python 3.7.1, unless otherwise noted. Some of the features described here may not be available in earlier versions of Python.

Looking for [examples for Python 2?](#)

This Site

-  [Module Index](#)
-  [Index](#)



© Copyright 2019, Doug Hellmann



Other Writing

-  [Blog](#)
-  [The Python Standard Library By Example](#)