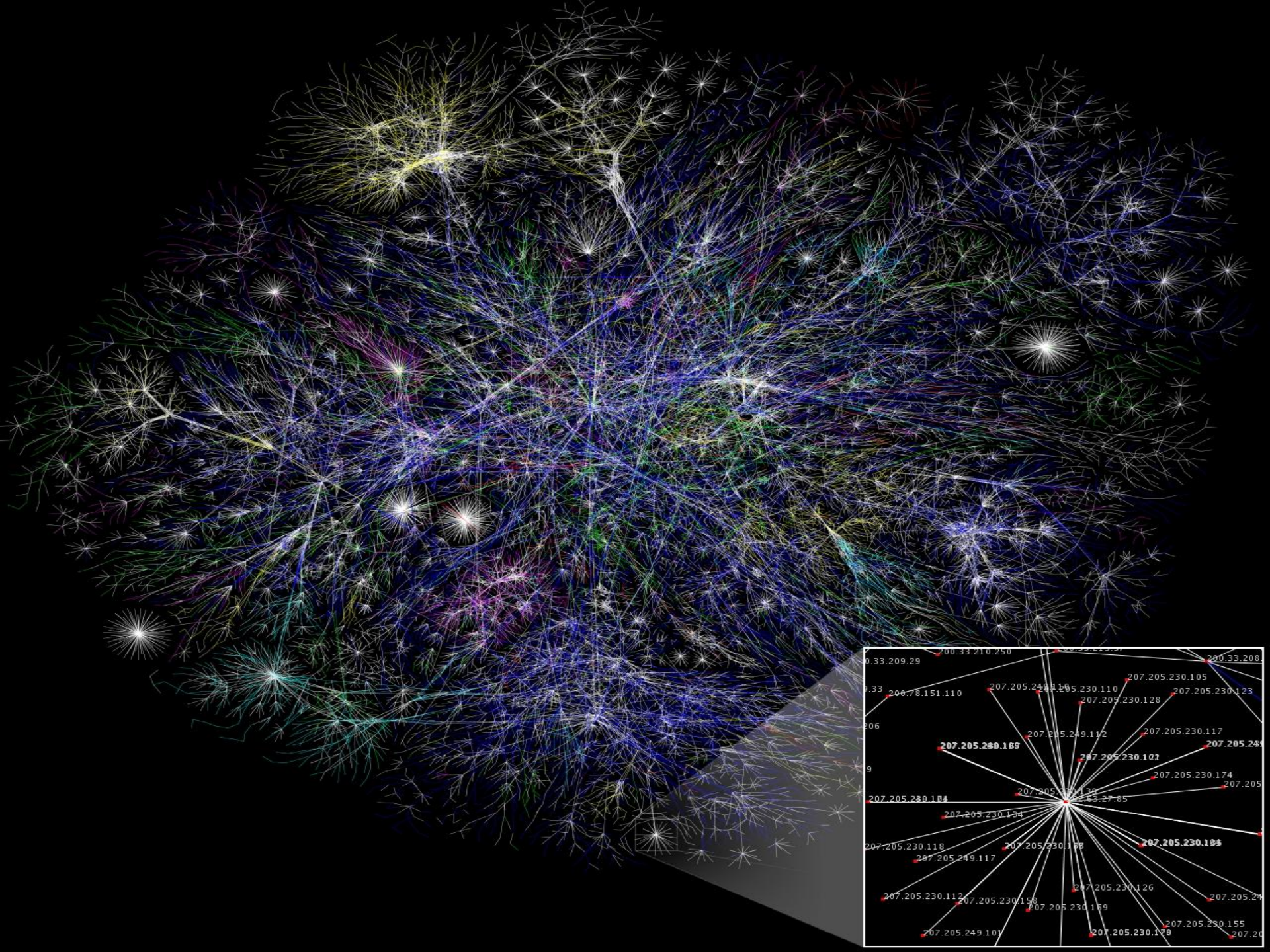
Web, Urllib

Кратко о том, что и как в инете, и причем здесь Питон.

Общие сведения

- **Интернет** — всемирная система объединённых компьютерных сетей, построенная на использовании протокола IP и маршрутизации пакетов данных. Интернет образует глобальное информационное пространство, служит физической основой для Всемирной паутины и множества других систем (протоколов) передачи данных.

- **Всемирная паутина** — распределенная система, предоставляющая доступ к связанным между собой документам, расположенным на различных компьютерах, подключенных к Интернету. Всемирную паутину образуют миллионы web-серверов.


200.33.210.250
0.33.209.29
200.78.151.110
207.205.230.105
207.205.230.110
207.205.230.123
207.205.230.128
207.205.249.112
207.205.230.117
207.205.230.174
207.205.230.134
207.205.230.118
207.205.249.117
207.205.230.126
207.205.230.112
207.205.230.158
207.205.230.159
207.205.230.155
207.205.249.101

- Большинство ресурсов всемирной паутины представляет собой гипертекст. Гипертекстовые документы, размещаемые во всемирной паутине, называются web-страницами.

- Для загрузки и просмотра web-страниц используются специальные программы — браузеры.

- Для облегчения создания, хранения и отображения гипертекста во Всемирной паутине традиционно используется язык HTML (англ. *HyperText Markup Language*), язык разметки гипертекста.

- После того, как HTML-файл становится доступен веб-серверу, его начинают называть «веб-страницей».

- *В целом, Всемирная паутина стоит на «трёх китах»: HTTP, HTML и URL.*

- **Семантическая паутина** — это направление развития Всемирной паутины — это направление развития Всемирной паутины, целью которого является представление информации в виде, пригодном для машинной обработки.

- Семантическая паутина работает параллельно с обычной Паутиной и на её основе, используя протокол HTTPСемантическая паутина работает параллельно с обычной Паутиной и на её основе, используя протокол HTTP и идентификаторы ресурсов URI.

- **URI** (англ. ***Uniform Resource Identifier***) — унифицированный (единообразный) идентификатор ресурса.

- URI — это последовательность символов, идентифицирующая абстрактный или физический ресурс.

- Структура URI очень гибка, синтаксис не сложен. В базовом виде URI представляется как:
- **<схема>:<идентификатор-в-зависимости-от-схемы>**

- Самый известный пример URI — это URL.
- http://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BA%D1%80%D0%

Ключевые принципы

- Интернет состоит из многих тысяч корпоративных, научных, правительственных и домашних компьютерныхИнтернет состоит из многих тысяч корпоративных, научных, правительственных и домашних компьютерных сетей. Объединение сетей разной архитектуры и топологииИнтернет состоит из многих тысяч корпоративных, научных, правительственных и домашних компьютерных сетей. Объединение сетей разной архитектуры и топологии стало возможно благодаря протоколу IPИнтернет состоит из многих тысяч корпоративных, научных, правительственных и домашних компьютерных сетей. Объединение сетей разной архитектуры и топологии стало возможно благодаря

- Протокол **IP** используется для негарантированной доставки данных, разделяемых на так называемые пакеты используется для негарантированной доставки данных, разделяемых на так называемые пакеты от одного узла сети к другому.

- **IP-пакет** — форматированный блок информации — форматированный блок информации, передаваемый по вычислительной сети — форматированный блок информации, передаваемый по вычислительной сети. Соединения вычислительных сетей, которые не поддерживают пакеты, такие как традиционные

- **Transmission Control Protocol (TCP)** — один из основных сетевых протоколов — один из основных сетевых протоколов Интернета, предназначенный для управления передачей данных — один из основных сетевых протоколов Интернета, предназначенный для управления передачей данных в сетях и подсетях TCP/IP.

- TCP — это транспортный механизм, предоставляющий поток данных, с предварительной установкой соединения, за счёт этого дающий уверенность в достоверности получаемых данных, осуществляет повторный запрос данных в случае потери данных и устраняет

- **Сетевой протокол** — набор правил, позволяющий осуществлять соединение и обмен данными между двумя и более включёнными в сеть устройствами.

- Систему протоколов Интернет называют «стеком протоколов TCP/IP».

- Протокол IP был специально создан агностическим в отношении физических каналов связи.

- На стыках сетей специальные маршрутизаторыНа стыках сетей специальные маршрутизаторы (программные или аппаратные) занимаются автоматической сортировкой и перенаправлением пакетов данных, исходя из IP-адресов получателей этих пакетов.

- **Сетевая модель OSI (ЭМВОС)** — абстрактная сетевая модель — абстрактная сетевая модель для коммуникаций и разработки сетевых протоколов — абстрактная сетевая модель для коммуникаций и разработки сетевых протоколов. Предлагает взгляд на компьютерную сеть с точки зрения

Модель OSI		
Тип данных	Уровень	Функции
Данные	7. Прикладной уровень	Доступ к сетевым службам
	6. Уровень представления	Представление и кодирование данных
	5. Сеансовый уровень	Управление сеансом связи
Сегменты	4. Транспортный	Прямая связь между конечными пунктами и надежность
Пакеты	3. Сетевой	Определение маршрута и логическая адресация
Кадры	2. Канальный	Физическая адресация
Биты	1. Физический уровень	Работа со средой передачи, сигналами и двоичными данными

Уровень OSI	Протоколы, примерно соответствующие уровню OSI
Прикладной	BGP, DNS, FTP, HTTP, HTTPS, IMAP, LDAP, POP3, SNMP, SMTP, SSH, Telnet, XMPP (Jabber)
Сеансовый / Представления	SSL, TLS
Транспортный	TCP, UDP
Сетевой	EIGRP, ICMP, IGMP, IP, IS-IS, OSPF, RIP
Канальный	Arcnet, ATM, Ethernet, Frame relay, HDLC, PPP, L2TP, SLIP, Token ring

HTTP

- **HTTP** (*HyperText Transfer Protocol*) — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов).

- **Основой** HTTP является технология «клиент-сервер».
- Предполагается существование:

- Потребителей (клиентов), которые инициируют соединение и посылают запрос.

- Поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

- **Метод HTTP** — последовательность из любых символов, кроме управляющих и разделителей, указывающая на основную операцию над ресурсом.
- Список методов:
- **GET, HEAD, POST,**
- PUT, PATCH, DELETE, TRACE, LINK, UNLINK, CONNECT, OPTIONS.

- GET - используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс.
- HEAD - Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело.
- POST - Применяется для передачи пользовательских данных заданному ресурсу.

httplib

- Этот модуль определяет классы, реализующие HTTP и HTTPS протоколы.
- Обычно он не используется непосредственно, его классы и методы использует библиотека urllib для обработки URL адресов, которые используют HTTP и HTTPS.

Классы httplib

- *class* httplib.HTTPSConnection(...)
- *class* httplib.HTTPResponse(...)
- *class* httplib.HTTPMessage
- (Provides utility functions to deal with HTTP Headers.)

- Также в этом модуле много объектов класса Exception.

- *class* httplib.HTTPConnection(*host*[, *port*[, *strict*[, *timeout*[, *source_address*]]]])

  (предоставляет один сеанс с HTTP сервером.)

- h3 = httplib.HTTPConnection('www.google.ru', 80)
- h3 = httplib.HTTPConnection('www.google.ru’, 80, timeout=10)

- HTTPConnection.request(*method*, *url*[, *body*[, *headers*]])
- request(…) отправляет запрос серверу используя HTTP метод method и селектор url.

- HTTPConnection.getresponse() вызывается после request(…) для получения ответа от сервера. Возвращает объект класса HTTPResponse

- HTTPConnection.close() прекращает связь с сервером.

- *class* **httplib.HTTPResponse**(*sock*[, *debuglevel=0*][, *strict=0*])

  Объект - то, что возвращается после успешного соединения.

  Не инициализируется пользователем.
- HTTPResponse.getheaders()

  Возвращает список из пар (header, value). Этот список дает информацию о сайте и сервере.

Пример программы с использованием httlib:

```
import httplib
conn = httplib.HTTPConnection("www.python.org")
conn.request("GET", "/index.html")
j = conn.getresponse()
print j.getheaders()
```

Urllib

- Этот модуль предоставляет средства высокого уровня для чтения сетевых ресурсов, используя различные протоколы.

- Определенные в этом модуле средства позволяют обращаться к ресурсам через proxy- сервер, не требующий аутентификации.

- **Аутентификация**— проверка принадлежности субъекту доступа предъявленного им идентификатора; подтверждение подлинности.

- (Не путать с авторизацией и идентификацией)

- Стоит отметить, что при работе с WWW используется в основном протокол HTTP, однако WWW охватывает не только HTTP,

  но и многие другие протоколы

  (FTP, gopher, HTTPS и т.п.).

Средства urllib

- **urlopen**(url [, data])
- Создает и возвращает объект, реализующий чтение ресурса url.
- Использует HTTP метод GET по умолчанию. Чтобы использовался метод POST необходимо указать строку data с данными в формате ‘application/x-www-form-urlencoded’.

- **urlencode**(dict) dict – словарь.
- Возвращает строку с данными dict в формате ‘application/x-www-form-urlencoded’. Возвращаемая строка состоит из фрагментов ‘key=value’, разделенных ‘&’, где key и value преобразуются с помощью функции quote_plus().

- **quote_plus**(string [, safe]) – заменяет спец. Символы в строке string на последовательности вида ‘%xx’
- Преобразованию не подвергаются буквы, цифры и символы ‘_’ , ‘,’ , ‘.’ и ‘-’. Пробелы заменяются на ‘+’. Если + в строке safe, то заменяет на ‘%2b’.

- read(), readline(), readlines(), fileno() и close() реализуют чтение ресурса.
- info() возвращает информацию о ресурсе.
- geturl() возвращает истинную информацию о ресурсе.
- Пример:
- import urllib
- params = urllib.urlencode({’spam’ : 1, ’eggs’ : 2, ’bacon’: 0})
- f = urllib.urlopen("http://www.musi-cal.com/cgi-bin/query?" +params)
- print f.read()
- #выведет код страницы.

Модуль urlparse

- Этот модуль определяет средства для разбиения Url на компоненты, конструирования URL из компонент и преобразования относительных URL в абсолютные (RFC 1808).

Средства urlparse

- **urlparse**(urlstring [, default_scheme [, allow_fragments]])

- Разбивает URL на части, и возвращает tuple из
- 6 элементов. Идентификатор протокола, положение в сети, путь, параметры, строка запроса и идентификатор фрагмента.

- ’scheme://netloc/path;parameters?query#fragment’
- выдает ‘(scheme, netloc, path, parameters, query, fragment)’.

- **urlunparse**(tuple)
- Восстанавливает по tuple url адрес.

- **urljoin**(base, rel_url [, allow_fragments])
- Конструирует и возвращает полный url адрес.

- Всё программное обеспечение для работы с протоколом HTTP разделяется на три больших категории:

- **Серверы** как основные поставщики услуг хранения и обработки информации (обработка запросов).

- **Клиенты** — конечные потребители услуг сервера (отправка запроса).

- **Прокси** для выполнения транспортных служб.

Прокси-сервер

- **Прокси-сервер** — служба в компьютерных сетях — служба в компьютерных сетях, позволяющая клиентам выполнять косвенные запросы к другим сетевым службам.

- Сначала клиент подключается к прокси-серверу и запрашивает какой-либо ресурс (например, e-mailСначала клиент подключается к прокси-серверу и запрашивает какой-либо ресурс (например, e-mail), расположенный на другом сервере.

- Затем прокси-сервер либо подключается к указанному серверу и получает ресурс у него, либо возвращает ресурс из собственного кэша (в случаях, если прокси имеет свой кэш).

Виды прокси-серверов

- **Прозрачный прокси** — схема связи, при которой трафик, или его часть, перенаправляется на прокси-сервер неявно (средствами маршрутизатора).

- **Обратный прокси** — прокси-сервер, который в отличие от прямого, ретранслирует запросы клиентов из внешней сети на один или несколько серверов, логически расположенных во внутренней сети.

```
import urllib2
uri = "http://www.python.org"
http_proxy_server = "someproxyserver.com" http_proxy_port = "3128"
http_proxy_realm = http_proxy_server
# Worked in his (limited) testing environment.
http_proxy_user = "username"
http_proxy_passwd = "password"
# Next line = "http://username:password@someproxyserver.com:3128"
http_proxy_full_auth_string = "http://%s:%s@%s:%s" % (http_proxy_user,
http_proxy_passwd, http_proxy_server, http_proxy_port)
def open_url_no_proxy():
    urllib2.urlopen(uri)
    print "Apparent success without proxy server!"
def open_url_installed_opener():
    proxy_handler = urllib2.ProxyHandler({"http":http_proxy_full_auth_string})
    opener = urllib2.build_opener(proxy_handler)
    urllib2.install_opener(opener)
    urllib2.urlopen(uri)
print "Apparent success through proxy server!“
if __name__ == "__main__": open_url_no_proxy()        open_url_installed_opener()
```

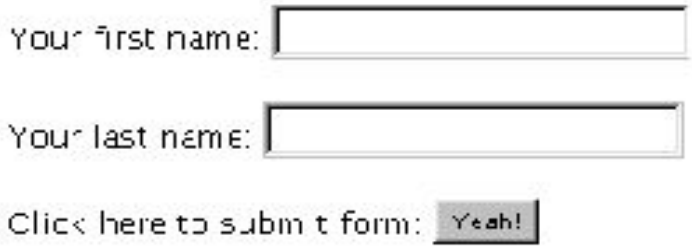
Example for using urllib2.urlopen() with a proxy server requiring authentication

Модуль CGI

- CGI- программа вызывается HTTP-сервером, обычно для обработки данных, переданных пользователем через элементы ‘FORM’ и ‘ISINDEX’ языка HTML.

- Модуль cgi берет на себя заботу обо всех возможных способах передачи данных и предоставляет их программе через простой интерфейс.

A typical HTML form




```
<form method="POST" action="http://host.com/cgi-bin/test.py">
   <p>Your first name: <input type="text" name="firstname">
   <p>Your last name: <input type="text" name="lastname">
   <p>Click here to submit form: <input type="submit" value="Yeah!">
   <input type="hidden" name="session" value="1f9a2">
</form>
```

A typical CGI script

```
#!/usr/local/bin/python
import cgi

def main():
    print "Content-type: text/html\n"
    form = cgi.FieldStorage()          # parse query
    if form.has_key("firstname") and form["firstname"].value != "":
        print "<h1>Hello", form["firstname"].value, "</h1>"
    else:
        print "<h1>Error! Please enter first name.</h1>"

main()
```

CGI script structure

- **Check form fields**
  - use cgi.FieldStorage class to parse query
    - takes care of decoding, handles GET and POST
    - "foo=ab+cd%21ef&bar=spam" -->
      {'foo': 'ab cd!ef', 'bar': 'spam'} # (well, actually, ...)
- **Perform action**
  - this is up to you!
  - database interfaces available
- **Generate HTTP + HTML output**
  - print statements are simplest
  - template solutions available

Средства cgi

- **FieldStorage**([**keyword_args])
- При инициализации его без аргументов происходит обработка данных со стандартного потока ввода и /или из переменных окружения в соответствии со стандартом CGI.()

- Конструктор класса принимает след. Аргументы:
- fp – альтернативный файловый объект
- Headers – отобр- е ин. о HTTP заголовках
- и т.д.
- И для остальных сущ. значение по умолчанию.

- Атрибуты класса FieldStorage
- name
- filename
- Value
- file
- type
- Headers
- и др.

```
import cgi
print """\
Content-Type: text/html
<html>
<body>"""
form = cgi.FieldStorage()
if form.has_key("name") and form.has_key("addr"):
    print """\
<p>Имя: %s</p>
<p>Адрес: %s</p>"""
else:
    print """\
<h1>Ошибка</h1>
<p>Введите пожалуйста имя и адрес</p>"""
print """\
</body>
</html>"""
```

Cookie

- **Куки** — небольшой фрагмент данных, созданный веб-сервером — небольшой фрагмент данных, созданный веб-сервером или веб-страницей и хранимый на компьютере — небольшой фрагмент данных, созданный веб-сервером или веб-страницей и хранимый на компьютере пользователя в виде файла, который веб-клиент (обычно веб-браузер — небольшой фрагмент данных, созданный веб-сервером или веб-страницей и хранимый на компьютере пользователя в виде файла, который веб-клиент (обычно веб-браузер)

- В техническом плане куки представляют собой фрагменты данных, изначально отправляемых веб-сервером браузеру. При каждом последующем посещении сайта браузер пересылает их обратно серверу.

GET /index.html
HTTP/1.1
Host: www.example.org

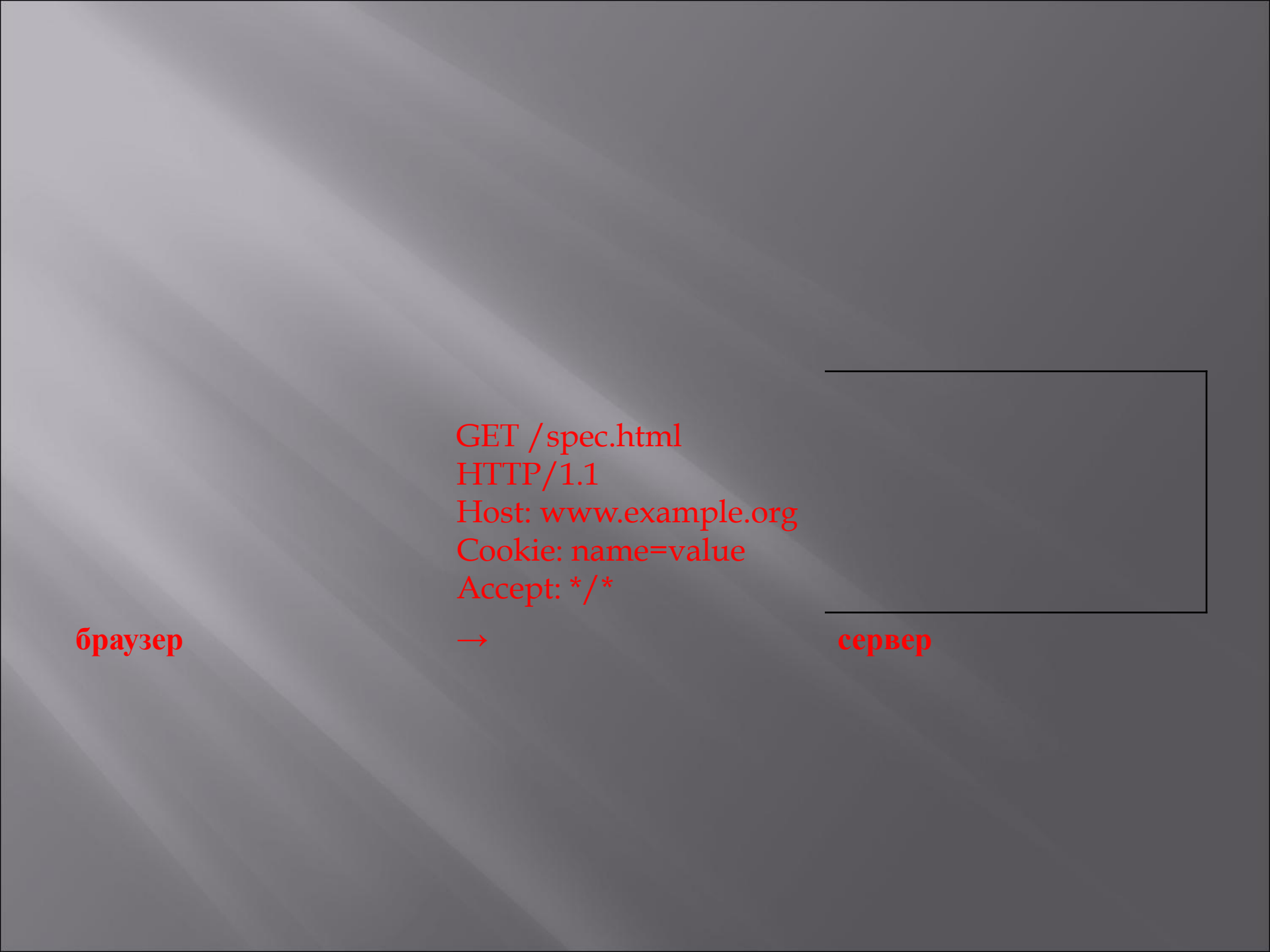
браузер → **сервер**

- Сервер отвечает

HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: name=value

(содержимое страницы)

браузер ← **сервер**


GET /spec.html
HTTP/1.1
Host: www.example.org
Cookie: name=value
Accept: */*
→
браузер
сервер

Пример:

- **import** http.cookiejar, urllib.request
- cj = http.cookiejar.CookieJar()
- opener = urllib.request.build_opener(urllib.request.HTTPCooki essor(cj))
- r = opener.open("http://example.com/")

I am just asking…