

# LECTURE 34

## REQUESTING URLS IN PYTHON

MCS 260 Fall 2021

David Dumas

# REMINDERS

- Homework 12 will be posted tomorrow
- Read Project 4 description
- Project 4 proposals due by Nov 17
  - But please give yourself more work time by doing it sooner if possible
- Project 4 is due on Fri Dec 3 at 6:00pm Central
- No office hours or synchronous lecture Friday
  - I'll post a lecture video and slide set

# INTERNET LAYER CAKE

<b>Application</b>	Retrieve <a href="http://example.com/">http://example.com/</a>
<b>Transport</b>	Transmit GET / to 93.184.216.34
<b>Network</b>	Deliver this packet to 93.184.216.34
<b>Link</b>	Send this ethernet frame to the router
<b>Physical</b>	Change voltages on these wires...

# TODAY

We'll discuss making **Application-level network requests in Python**.

We focus specifically on retrieving data (documents, etc.) from a Uniform Resource Locator or URL.

The `urllib` module in Python supports this. It is primarily focused on HTTP, HTTPS, and local files.

# HTTP REQUEST TYPES

HTTP allows many types of requests. For example:

- **GET** — Ask for the resource. Most common.
- **POST** — Submit data to the resource.
- **PUT** — Submit data that should replace the resource.

Today we'll only use GET.

# HTTP RESPONSE

Response consists of a numeric **status code**, some **headers** (`key: value` pairs, one per line), then a **payload**.

E.g. GET a web page, the HTML will be in the payload.

There are **lots of codes**; first digit gives category:

- 2xx — success
- 3xx — what you want is somewhere else
- 4xx — error (server thinks it's your fault)
- 5xx — error (server's fault)

# PARTS OF A HTTP RESPONSE

Response to GET `http://example.com/`

```
HTTP/1.1 200 OK
Age: 309829
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Mon, 19 Apr 2021 03:40:44 GMT
Expires: Mon, 26 Apr 2021 03:40:44 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECS (ord/572F)
Vary: Accept-Encoding
Content-Length: 1256
```

```
<!doctype html>
<html>
<head>
  <title>Example Domain</title>
  .
  .
  .
```

# PARTS OF A HTTP RESPONSE

Response to GET `http://example.com/`

**Status line** with response code

HTTP/1.1 200 OK

Age: 309829

Cache-Control: max-age=604800

Content-Type: text/html; charset=UTF-8

Date: Mon, 19 Apr 2021 03:40:44 GMT

Expires: Mon, 26 Apr 2021 03:40:44 GMT

Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT

Server: ECS (ord/572F)

Vary: Accept-Encoding

Content-Length: 1256

**Headers**

"key: value", one per line

Required blank line

<!doctype html>

<html>

<head>

    <title>Example Domain</title>

    .

    .

    .

**Body**

A sequence of bytes



# BASIC URLLIB USAGE

Import `urllib.request` to get the most convenient functions for loading URLs.

Call `urllib.request.urlopen(url)` to open the URL `url` using GET. It returns a response object.

Response objects behave like read-only files, and should be closed with `.close()`.

If a 4xx or 5xx response is received, or if contacting the host fails, a `urllib.error.URLError` exception is raised.

# RESPONSE OBJECTS

A HTTP response object `res` has:

# BYTES AND STRINGS

Often the payload is meant to be a string, but you will always receive it as bytes.

To recover that string from the `bytes` object returned by `res.read()`, you need to call the `.decode(...)` method, e.g.

```
enc = res.headers.get_content_charset() # probably "UTF-8"  
response_string = res.read().decode(enc) # bytes -> str
```

# APIS

An **application programming interface** or **API** is a structured way for computer programs to talk to each other.

APIs often use the network, and often use HTTP.

Some are available freely to anyone.

# USING AN API

`urllib.request.urlopen` is a great way to fetch data from HTTP APIs.

Example for today: A free dice rolling JSON API\* by Steve Brazier at `roll.diceapi.com`.

## Examples:

- `http://roll.diceapi.com/json/d6` — roll one six-sided die
- `http://roll.diceapi.com/json/3d6` — roll three six-sided dice
- `http://roll.diceapi.com/json/4d12` — roll four twelve-sided dice

\* This API could disappear at any moment. It worked on November 9, 2021.

# URL PARAMETERS

HTTP GET requests can send an associative array of parameters. For example, to send the dictionary `{"name": "David", "apple": "McIntosh"}` to `http://example.com/` the URL would be

```
http://example.com/?name=David&apple=McIntosh
```

The parameter list begins with `?` and has `&` between `name=value` pairs. It gets tricky when values or names have spaces, but `urllib.parse.urlencode` can convert a dictionary to a suitable string.

# CAT FACTS

The domain `cat-fact.herokuapp.com` hosts an API\* created by CS undergrad student Alex Wohlbruck for retrieving facts about cats (and other animals). E.g.

- `https://cat-fact.herokuapp.com/facts/random?amount=2` — two random facts about cats
- `https://cat-fact.herokuapp.com/facts/random?animal_type=dog&amount=1` — one random fact about dogs

\* This API could disappear at any moment. It worked on November 10, 2020.

# REFERENCES

- The [urllib documentation](#) is quite nice, especially the examples in each section, e.g.
  - [Examples of using urllib.request](#)
- A big list of free/open APIs, mostly JSON-based

# REVISION HISTORY

- 2021-11-10 Initial publication