LECTURE 34

REQUESTING URLS IN PYTHON

MCS 260 Fall 2020

Emily Dumas

REMINDERS

- Worksheet 12 available (download data in advance)
- Quiz 12 will be posted tomorrow
- Read Project 4 description
- Project 4 proposals ASAP, due Nov 16

INTERNET LAYER CAKE

Application	Retrieve http://example.com/
Transport	Transmit GET / to 93.184.216.34
Network	Deliver this packet to 93.184.216.34
Link	Send this ethernet frame to the router
Physical	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** (an associative array), 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 — redirection; more action required (e.g. moved)
- 4xx — client error; your request has a problem
- 5xx — server error; cannot handle this valid request

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 binary 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:

- `res.status` — the status code
- `res.geturl()` — returns the final URL (maybe not the one requested, if redirection used)
- `res.read()` — returns the payload as bytes
- `res.headers` — dict-like object storing the HTTP headers (not HTML header!)
- `res.headers.get_content_charset()` — Return payload encoding, if known

USING AN API

`urllib.request.urlopen` is a great way to fetch data from 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 10, 2020.

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

REVISION HISTORY

- 2020-11-11 Added link to HTTP status code list
- 2020-11-10 Initial publication