Web Python Lesson 1: HTTP and APIs

vanderbi.lt/py

Steve Baskauf

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

APIs vs. web scraping

- An **API** (**application programming interface**) is designed by the information provider to make structured data available to web users
- APIs usually make information available in JSON

- **Web scraping** is necessary when the information provider doesn't intend or doesn't bother to make structured data easily available.
- Web scraping is only practical if there is consistent format within or among web pages in a site.
- Web scraping extracts information from HTML

URI vs. URL

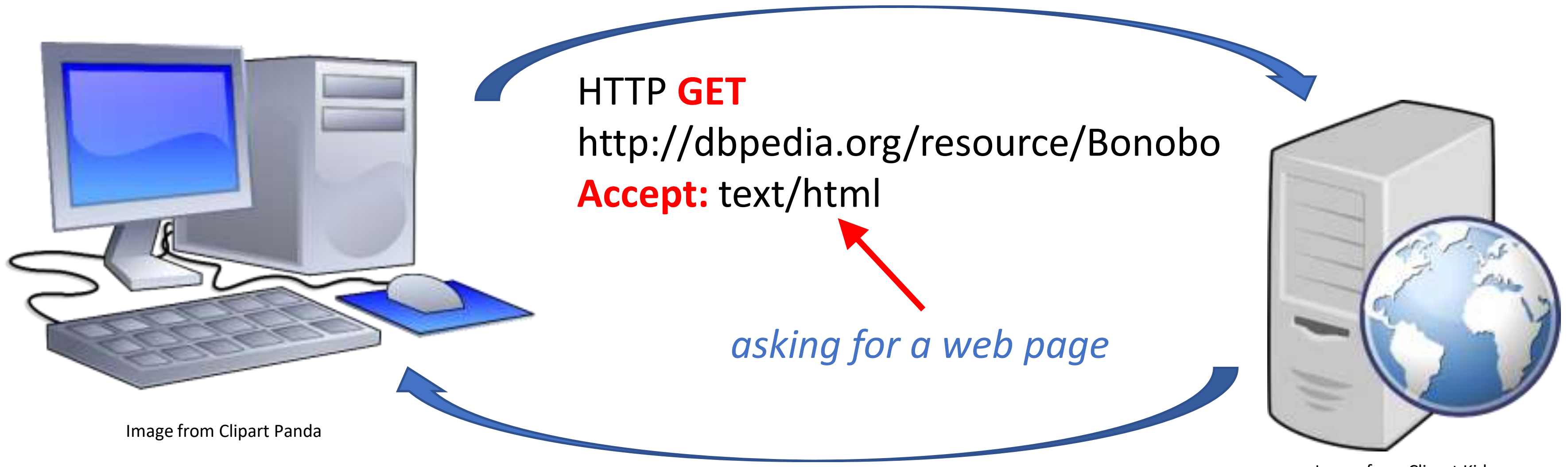
- A URI is a Uniform Resource Identifier
- A URI is a globally unique identifier for anything

- A URL is a Uniform Resource Locator
- A URL is a subset of URIs that will actually retrieve a file.

- When I say URI, you can generally think "URL"

HTTP protocol

Hypertext Transfer Protocol (HTTP), used to carry out an interaction across the Internet. mediated by. Retrieving information using HTTP GET is called "dereferencing a URI". (People also say "resolving" a URI.)



HTTP **GET**
http://dbpedia.org/resource/Bonobo
Accept: text/html

asking for a web page

Image from Clipart Panda

Image from Clipart Kid

Client software

(a.k.a. the "machine")

In this case, the client is a web browser. It displays the returned body as a web page.

HTTP **Status:** 200 OK

Body:

```
<?xml version="1.0" encoding="UTF-8" ?><!DOCTYPE html PUBLIC "-//W3C//DTD
XHTML+RDFa 1.0//EN" "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd"><html
xmlns="http://www.w3.org/1999/xhtml"
xmlns:dbpprop="http://dbpedia.org/property/"
...
```

Web server

Dereferencing a URI in a browser

- Paste the URI from the Jupyter notebook:
  `http://bioimages.vanderbilt.edu/pages/contributors.htm`
  into a browser and request.

Pieces of HTTP

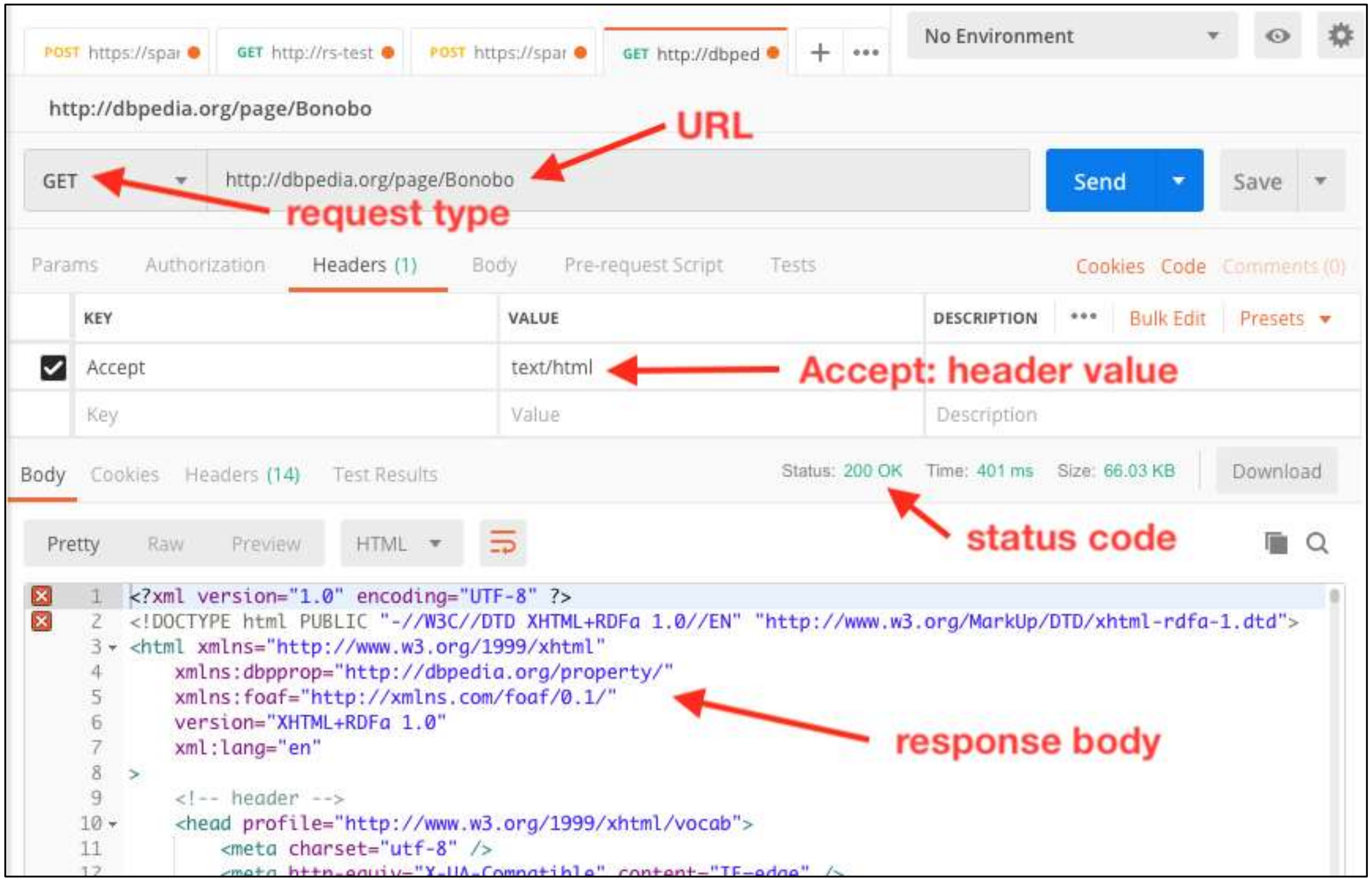
- Request type: GET, POST, and others
- Status code (response from server): 200, 30x, 404
- Response headers from server:
  - Content-Type (media type being sent)
  - Content-Length (bytes)
  - Date
- Body (text from server)

- Paste the URI from the Jupyter notebook:

  `http://bioimages.vanderbilt.edu/pages/contributors.htm`

  into Postman and Send.

Exploring HTTP with Postman



http://dbpedia.org/page/Bonobo

GET http://dbpedia.org/page/Bonobo

KEY	VALUE
Accept	text/html

Status: 200 OK Time: 401 ms Size: 66.03 KB

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN" "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:dbpprop="http://dbpedia.org/property/"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    version="XHTML+RDFa 1.0"
    xml:lang="en"
>
    <!-- header -->
    <head profile="http://www.w3.org/1999/xhtml/vocab">
        <meta charset="utf-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

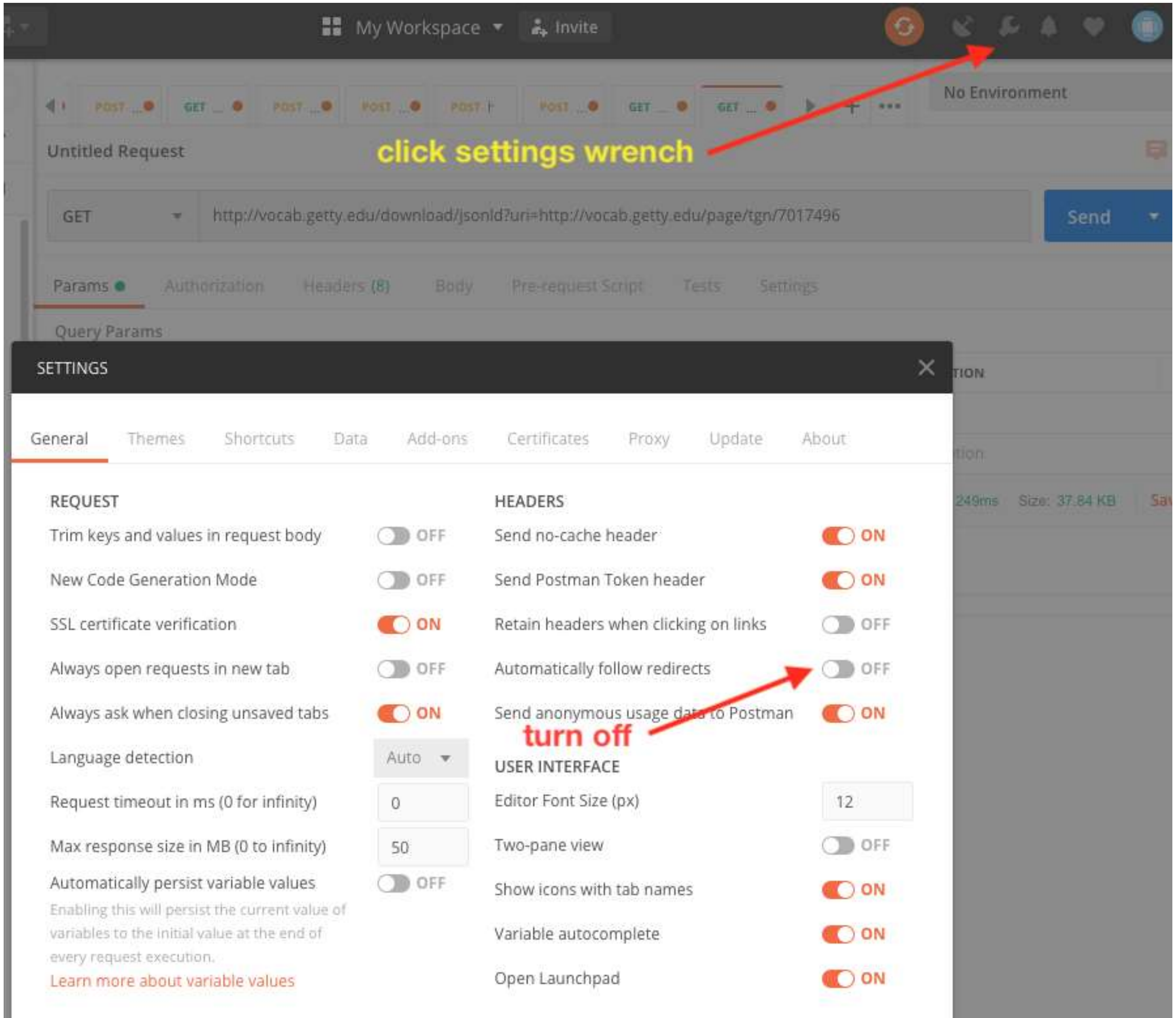
What we are sending to the server

What the server is sending back to us

Exploring HTTP with Python

- Run the first script on the Jupyter notebook

Turn off automatic redirects



SETTINGS

General | Themes | Shortcuts | Data | Add-ons | Certificates | Proxy | Update | About

REQUEST

Trim keys and values in request body	OFF
New Code Generation Mode	OFF
SSL certificate verification	ON
Always open requests in new tab	OFF
Always ask when closing unsaved tabs	ON
Language detection	Auto
Request timeout in ms (0 for infinity)	0
Max response size in MB (0 to infinity)	50
Automatically persist variable values	OFF

Enabling this will persist the current value of variables to the initial value at the end of every request execution.
Learn more about variable values

HEADERS

Send no-cache header	ON
Send Postman Token header	ON
Retain headers when clicking on links	OFF
Automatically follow redirects	OFF
Send anonymous usage data to Postman	ON

USER INTERFACE

Editor Font Size (px)	12
Two-pane view	OFF
Show icons with tab names	ON
Variable autocomplete	ON
Open Launchpad	ON

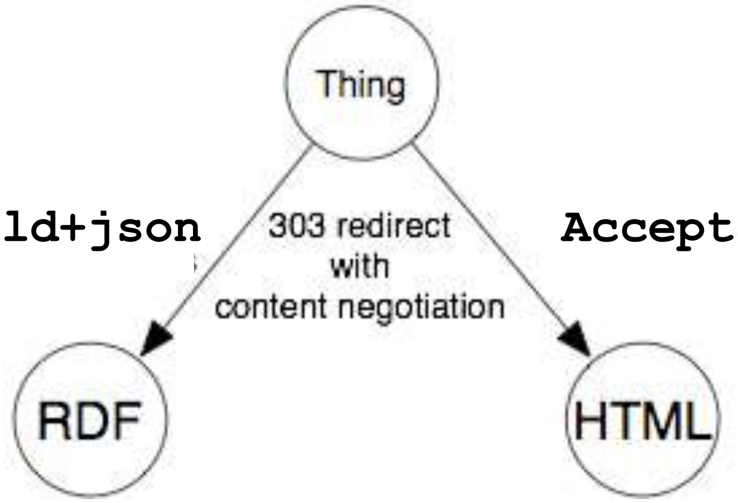
Content negotiation: diagram

- We can send request headers to the server with information about what we want it to send
- An example is an **`Accept`** header for our preferred media type

`http://dbpedia.org/resource/Bonobo`

`Accept: application/ld+json`

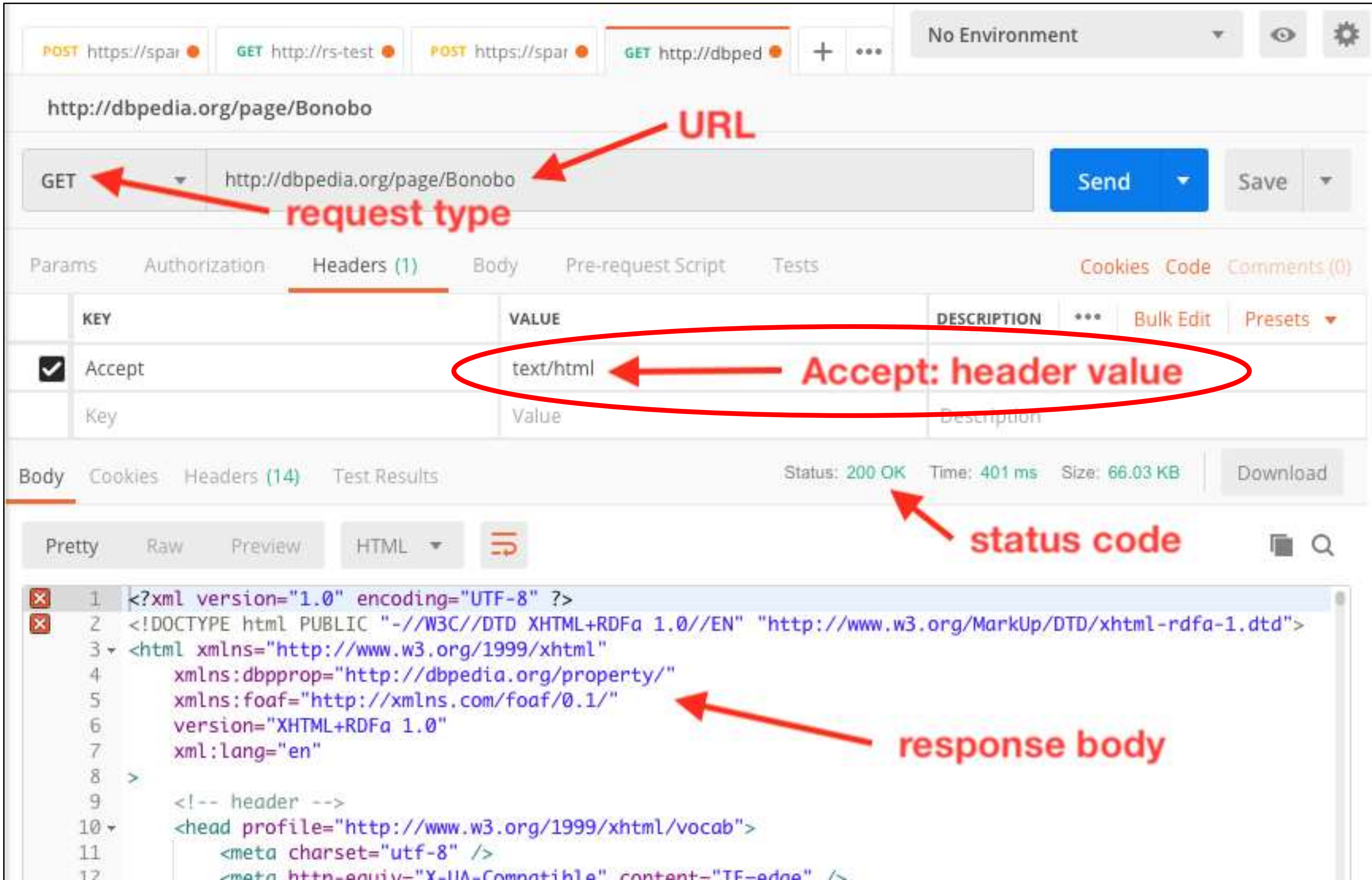
`Accept: text/html`



`http://dbpedia.org/data/Bonobo.jsonld`

`http://dbpedia.org/page/Bonobo`

Exploring HTTP with Postman



```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN" "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:dbpprop="http://dbpedia.org/property/"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    version="XHTML+RDFa 1.0"
    xml:lang="en"
>
    <!-- header -->
    <head profile="http://www.w3.org/1999/xhtml/vocab">
        <meta charset="utf-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

Content negotiation: Postman

- Set URL to **`http://dbpedia.org/resource/Bonobo`**
- Set Accept header to **`text/html`** and Send
- Examine status code, Location response header, and body
- Set URL to **`http://dbpedia.org/page/Bonobo`** and Send
- Examine status code, Location response header, and body

- Repeat starting with Accept header **`application/ld+json`**

Postman settings

- Turn automatic redirects back on
- Unless something goes wrong, we don't care that much about how we get to the final URL

Content negotiation with Python

- Run the second script in the Jupyter notebook
- Not every website or API supports **`Accept`** headers
- Sometimes the desired media type is requested directly as part of the URL
- Usually the server correctly identifies the **`Content-Type`** that it's sending (but not always)

Using data from GitHub

- A lot of data are now available from GitHub
- Example: heights of U.S. presidents rendered as a table: https://github.com/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/data/president_heights.csv
- To use in Python, we need the raw data. Click the Raw button, then copy the URL.
- Example script in Jupyter notebook.

What's an API?

- An API is a website that provides access to structured data (usually JSON, sometimes XML)
- An **endpoint** is a URL that returns data about a particular kind of thing (known as a "**resource**").
- A simple API may have only one endpoint.
- More complex APIs may have many endpoints that allow you to get information about a variety of things
- It's impossible to know what an endpoint does without a "developer guide" or "API reference".

API etiquette

- Do not harvest the entire dataset. Ask the provider for a **data dump** if you want the whole thing.
- Do not **repeatedly hit** the API at high speed. Use the **`.sleep()`** function from the **`time`** module to limit the rate of your requests, e.g.

```
time.sleep(1)
```

for a one second delay.

- Most APIs will require you to use **paging** to receive a reasonable amount of data in one call.
- Abusing an API often will result in you being **blocked** at least for a time.
- Test your script in a **sandbox** (if one is available) before using the real API.

International Space Station API

- This simple API only does one thing: return the latitude and longitude of the current position of the ISS.
- The script uses the **`.json()`** method to turn the JSON response text into a Python data structure.
  - Notice that valid JSON requires keys and values to be in double quotes.
  - The Python data structure in this case is a dictionary, which can have keys and values in single or double quotes.

Options for sorting through JSON

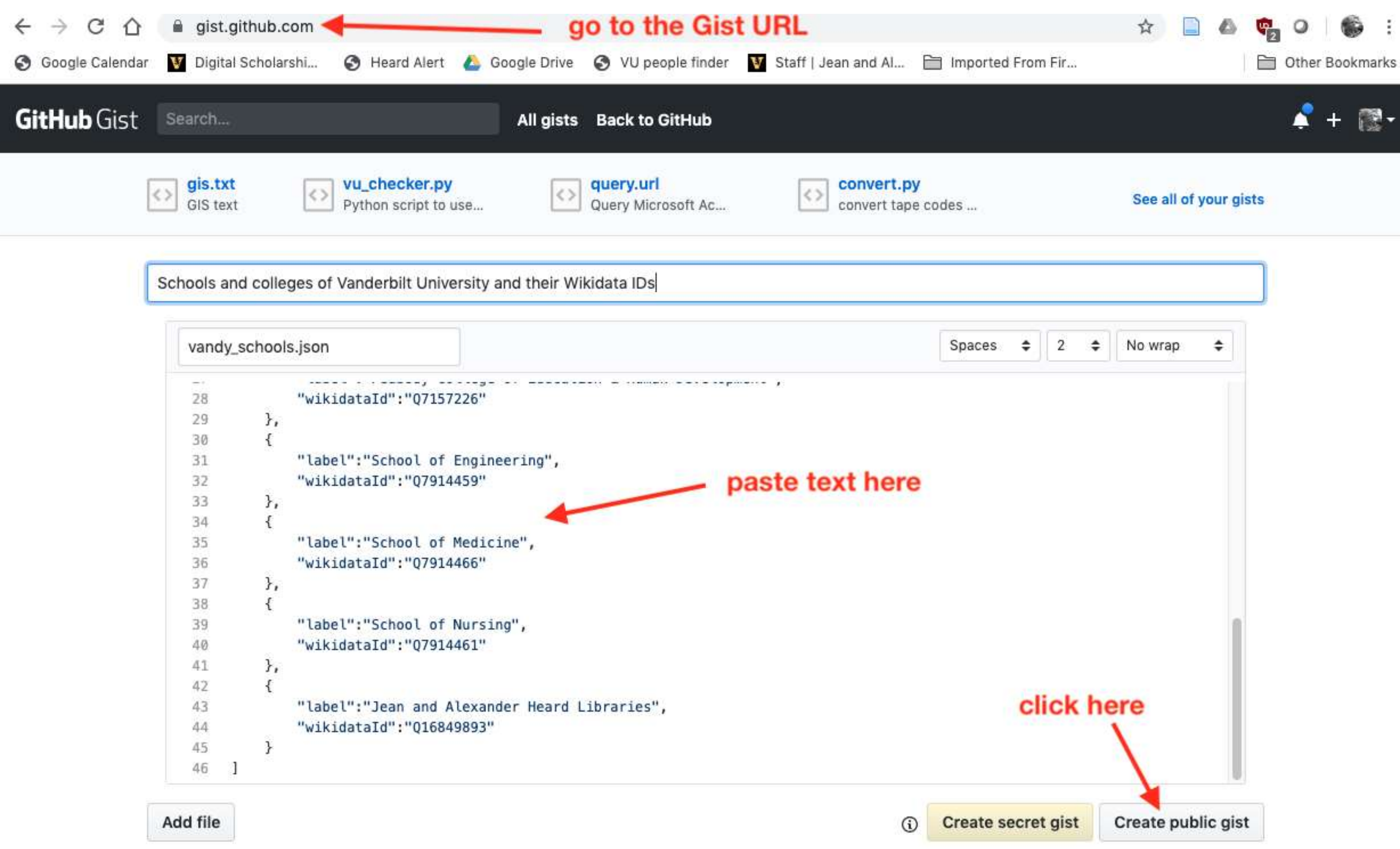
1. VS Code
   - paste the JSON string
   - save with a **`.json`** file extension
   - right click, select **`Format Document`**
2. JSON Editor Online (good if file is huge)
   - https://jsoneditoronline.org/
   - past JSON on left
   - click rightward arrow and expand or collapse
3. Use **`json.dumps()`** in your Python code
   - **`indent`** argument sets indentation spacing
   - **`sort_keys`** alphabetizes keys within JSON objects
   - See code example in Jupyter notebook

Putting data into a GitHub Gist

- Easiest way to give access to data for others to use in Python
- GitHub account required
- After creating the Gist, click the raw button
- Copy the URL to use in your code

- Technical note: all GitHub raw files are served as **`Content-Type = text/plain`** regardless of actual file type
- Some applications will have problems with incorrectly identified media types.

Putting data in a GitHub Gist



Schools and colleges of Vanderbilt University and their Wikidata IDs

vandy_schools.json

Spaces: 2, No wrap

```
        "wikidataId":"Q7157226"
    },
    {
        "label":"School of Engineering",
        "wikidataId":"Q7914459"
    },
    {
        "label":"School of Medicine",
        "wikidataId":"Q7914466"
    },
    {
        "label":"School of Nursing",
        "wikidataId":"Q7914461"
    },
    {
        "label":"Jean and Alexander Heard Libraries",
        "wikidataId":"Q16849893"
    }
]
```

For next week:

- Create a GitHub account if you don't already have one.