Working with Web APIs

CS111 Computer Programming

Department of Computer Science
Wellesley College

The Web is full of data

Examples of interesting web data

- Weather
- Sensor data (earthquakes, sea levels, pollution, etc.)
- Sport scores
- Movie schedules
- Song lyrics
- Yelp reviews about businesses
- Facebook comments, tweets, Instagram photos, TikTok videos
- Stock markets
- Economic development
- Public opinion surveys
- YouTube videos
- Books
- Etc.

Common features of such data

Continuous change

Too big to store in one place

Massive Data Centers

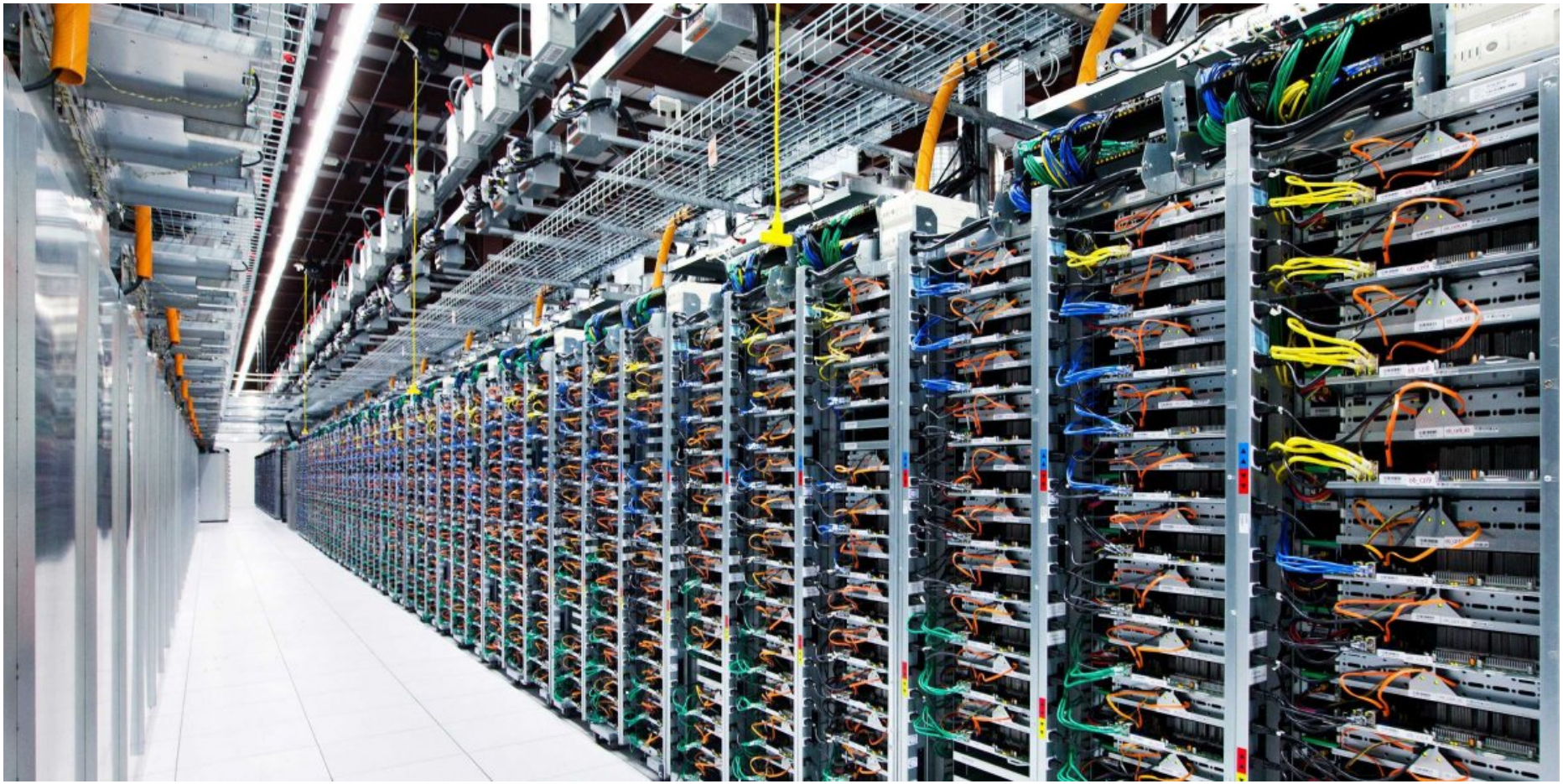
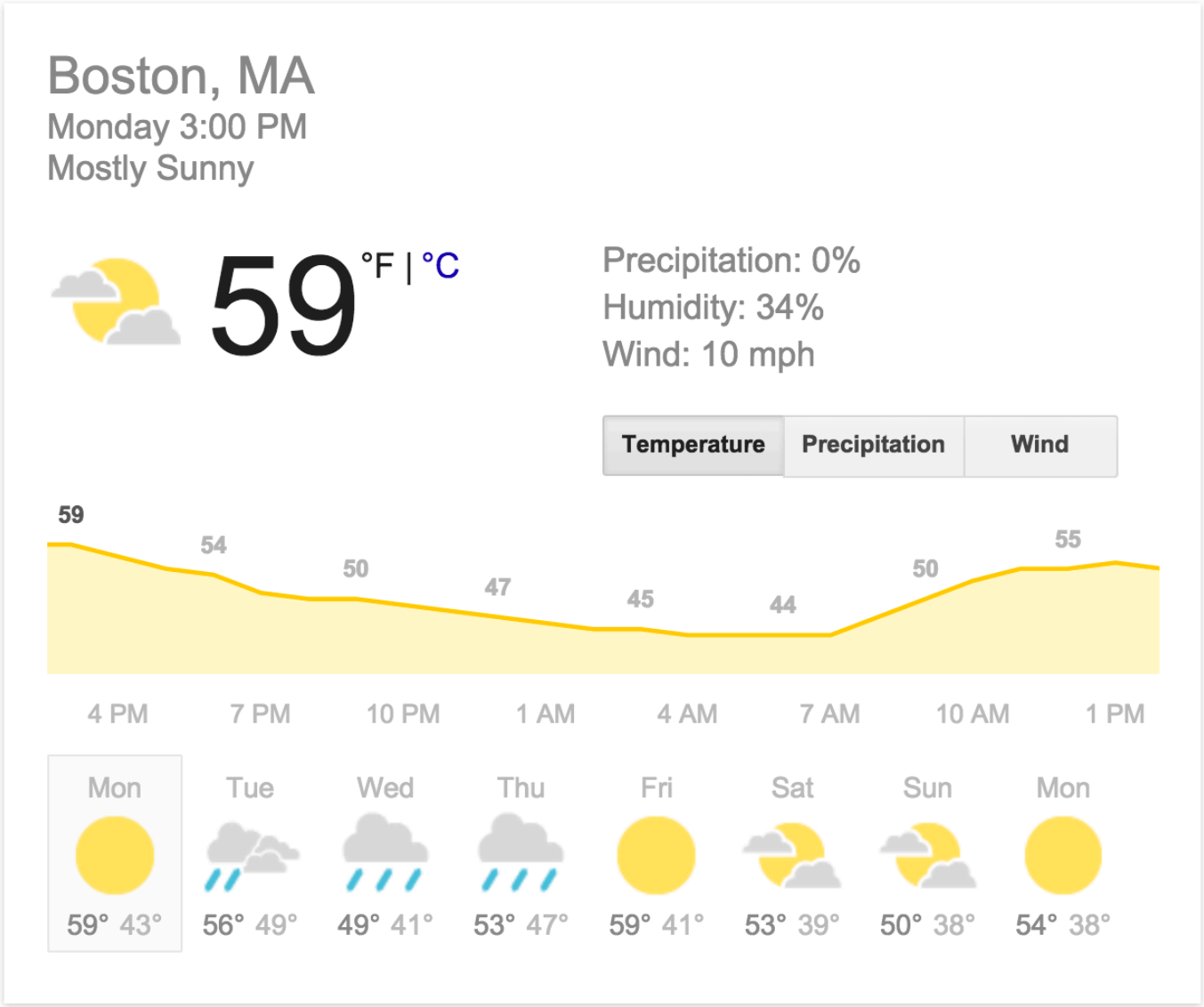


Image: http://www.businessinsider.com/google-data-centers-2014-10

Characteristics of Web Data

- Data is stored in remote computers connected to the Internet.
- Companies such as Google and Facebook operate facilities known as "data centers". [See previous slide]
- Data is often stored in **machine-readable** format, such as JSON or XML or within a database.
- Data can be displayed in a **human-readable** format such as a web page that can be viewed with a browser.

Human-Readable Data



Weather forecast on Google

Machine-Readable Data

Weather data from the Weather Underground API


```
"observation_location": {
    "full": "North End waterfront Boston, Boston, Massachusetts",
    "city": "North End waterfront Boston, Boston",
    "state": "Massachusetts",
    "country": "US",
    "country_iso3166": "US",
    "latitude": "42.365311",
    "longitude": "-71.051521",
    "elevation": "39 ft"
},
"estimated": {},
"station_id": "KMABOSTO124",
"observation_time": "Last Updated on November 9, 6:26 PM EST",
"observation_time_rfc822": "Mon, 09 Nov 2015 18:26:10 -0500",
"observation_epoch": "1447111570",
"local_time_rfc822": "Mon, 09 Nov 2015 18:26:12 -0500",
"local_epoch": "1447111572",
"local_tz_short": "EST",
"local_tz_long": "America/New_York",
"local_tz_offset": "-0500",
"weather": "Partly Cloudy",
"temperature_string": "55.8 F (13.2 C)",
"temp_f": 55.8,
"temp_c": 13.2,
"relative_humidity": "38%",
"wind_string": "From the SSW at 4.9 MPH Gusting to 7.4 MPH",
"wind_dir": "SSW",
"wind_degrees": 201,
"wind_mph": 4.9,
"wind_gust_mph": "7.4",
"wind_kph": 7.9,
"wind_gust_kph": "11.9",
"pressure_mb": "1028",
"pressure_in": "30.35",
"pressure_trend": "+",
"dewpoint_string": "31 F (-1 C)",
"dewpoint_f": 31,
"dewpoint_c": -1,
```

What is an API?

What is an API (link)

Bridging the Gap


```
"observation_location": {
   "full": "North End waterfront Boston, Boston, Massachusetts",
   "city": "North End waterfront Boston, Boston",
   "state": "Massachusetts",
   "country": "US",
   "country_iso3166": "US",
   "latitude": "42.365311",
   "longitude": "-71.051521",
   "elevation": "39 ft"
},
"estimated": {},
"station_id": "KMABOSTO124",
"observation_time": "Last Updated on November 9, 6:26 PM EST",
"observation_time_rfc822": "Mon, 09 Nov 2015 18:26:10 -0500",
"observation_epoch": "1447111570",
"local_time_rfc822": "Mon, 09 Nov 2015 18:26:12 -0500",
"local_epoch": "1447111572",
"local_tz_short": "EST",
"local_tz_long": "America/New_York",
"local_tz_offset": "-0500",
"weather": "Partly Cloudy",
"temperature_string": "55.8 F (13.2 C)",
"temp_f": 55.8,
"temp_c": 13.2,
"relative_humidity": "38%",
"wind_string": "From the SSW at 4.9 MPH Gusting to 7.4 MPH",
"wind_dir": "SSW",
"wind_degrees": 201,
"wind_mph": 4.9,
"wind_gust_mph": "7.4",
"wind_kph": 7.9,
"wind_gust_kph": "11.9",
"pressure_mb": "1028",
"pressure_in": "30.35",
"pressure_trend": "+",
"dewpoint_string": "31 F (-1 C)",
"dewpoint_f": 31,
"dewpoint_c": -1,
```


One way to bridge the gap: Write Python programs that generate HTML pages.

In this lecture, we will only look at how to take the data from APIs and process them, but not create pages.

Things we need to know (at high-level)

- What is the Internet? [slides:]
- What is the Web (WWW)? [slides:]
- What are URLs?
- What is an HTTP request?
- How to send HTTP requests via Python?
- How to read the response send by the API server?
- How to manipulate the JSON results to extract what we need?

What is a URL?

- URL = Universal Resource Locator
- Specifies the location of a web resource (web page, image, sound file, movie, etc.) in a remote server on the Internet.
- Also known as a web address.

http://cs111.wellesley.edu/content/info/simple.html

protocol	domain name host server	path	file

Detour: Your own space on the server

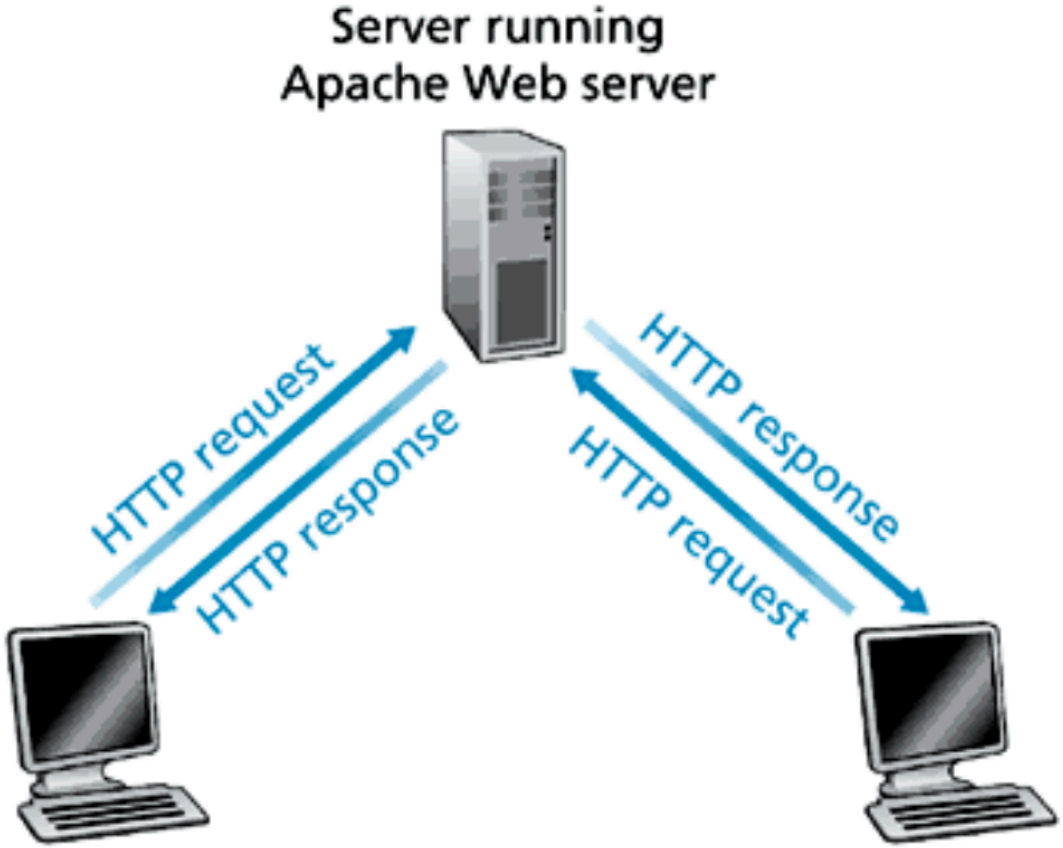
- Each of you has a folder in our CS server.
- Type the following URL in your web browser (by using your personal account name):

```
http://cs.wellesley.edu/~yourAccountName/
```

To do in (or after) class:

1. Open the file test_page.html (in lecture folder) with an editor (Atom).
2. Make changes to the page (add your name and your favorite things)
3. Using Cyberduck, upload the file to the `public_html` folder in your CS server account (by dragging the file).
4. On the browser, go to your own URL and check the new page.

HTTP = HyperTextTransferProtocol



A PC with an Internet Explorer

A Mac with Safari

An HTTP Request

User requests the page:

http://cs111.wellesley.edu/content/info/simple.html

Browser prepares and sends the following message to server:

```
GET /content/info/simple.html HTTP/1.1
Host: cs111.wellesley.edu
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML,
537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
```

An HTTP Response

header

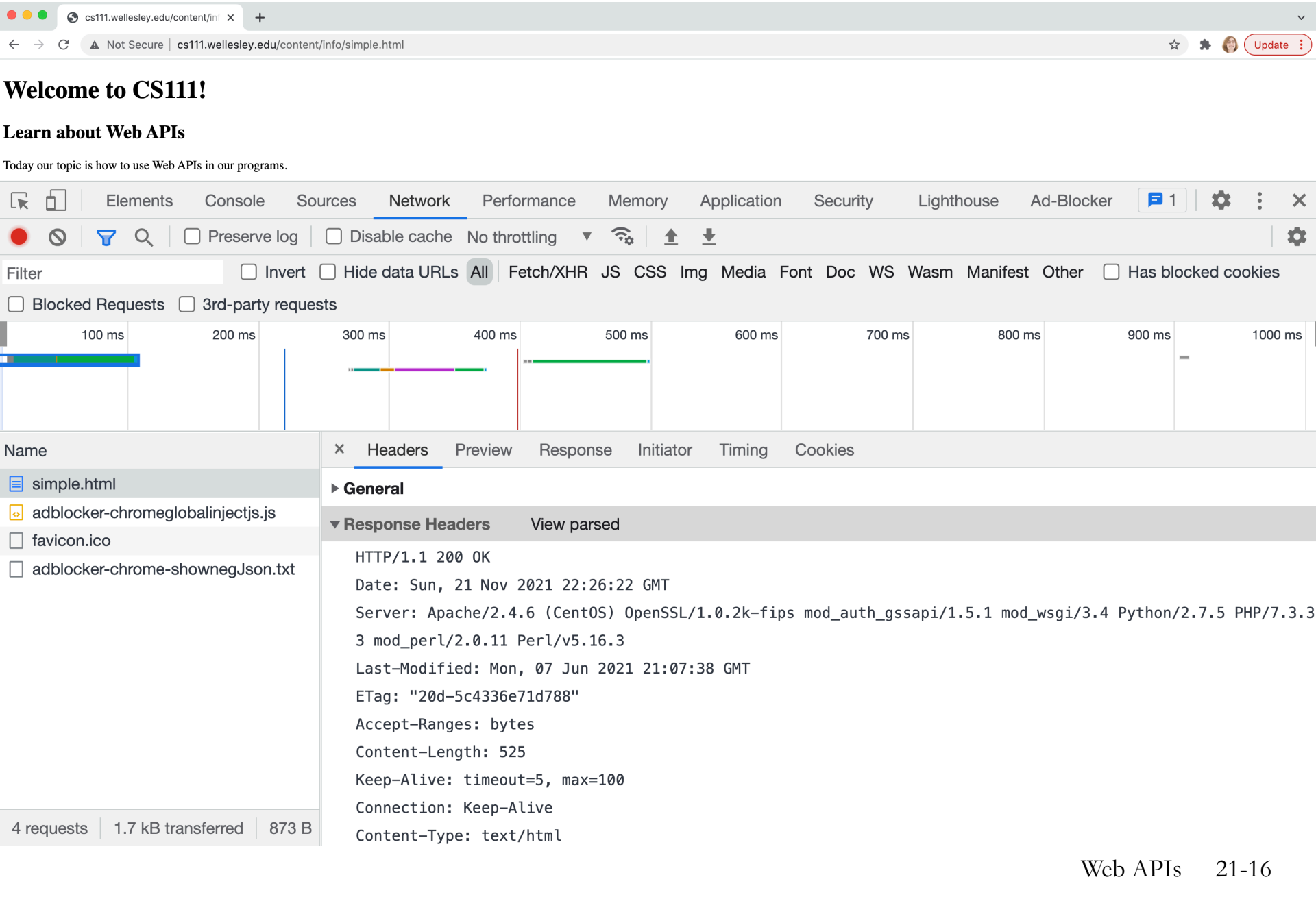
```
HTTP/1.1 200 OK
Date: Tue, 15 Nov 2016 14:39:54 GMT
Server: Apache/2.2.15 (Red Hat)
Last-Modified: Mon, 14 Nov 2016 19:55:37 GMT
ETag: "cd7748-1f1-541483945f7a5"
Accept-Ranges: bytes
Content-Length: 497
Connection: close
Content-Type: text/html; charset=UTF-8
```

content

```
<!DOCTYPE html>
<html lang="">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title></title>
</head>

<body>
  <h1>Welcome to CS111!</h1>
  <h2>Learn about Web APIs</h2>
...
```

Inspecting the browser



Using Python's `requests` module


```
import requests
httpResponse =
requests.get("http://cs111.wellesley.edu/content/info/simple.html")

In [5]: httpResponse
Out[5]: <Response [200]>
In [6]: httpResponse.status_code
Out[6]: 200
In [7]: httpResponse.headers
Out[7]: {'Content-Length': '525', 'Accept-Ranges':
'bytes', 'Server': 'Apache/2.2.15 (Red Hat)', 'Last-
Modified': 'Mon, 07 Jun 2021 21:07:38 GMT',
'Connection': 'close', 'ETag': '"20d-5c4336e71d788"',
'Date': 'Wed, 06 Dec 2017 16:16:27 GMT', 'Content-
Type': 'text/html; charset=UTF-8'}
```

Using Python's `requests` module


```
In [8]: httpResponse.content
Out[8]: b'<!DOCTYPE html>\n<html lang="">\n<head>\n <meta
charset="UTF-8">\n <meta name="viewport" content="width=device-
width, initial-scale=1.0">\n <title></title>\n</head>\n\n<body>\n
<h1>Welcome to CS111!</h1>\n <h2>Learn about Web APIs</h2>\n \n
<p>Today our topic is how to use Web APIs in our programs.</p>\n \n
<p>Some examples of Web APIs are:</p>\n \n <ol>\n
<li>OpenWeatherMap API</li>\n <li>Google Books API</li>\n
<li>Google Maps API</li>\n <li>Twitter API</li>\n <li>Facebook
Graph API</li>\n\n </ol>\n</body>\n</html>\n'
In [9]: type(httpResponse.content)
Out[9]: bytes
In [10]: type(httpResponse.text)
Out[10]: str
```

This form of getting the content is useful when dealing with text files.

Web Files vs. Web Data

Not everything on the web is stored as files on a server (HTML, PNG, etc.) Some of the information resides in databases and we can send API requests to get it.

This will be done through Web API requests.
API = Application Programming Interface

Example 1: Get the lyrics of a song

In the notebook, we show an example of communicating with a simple API that retrieves song lyrics. The URL for the request is shown here as well:

```
https://api.lyrics.ovh/v1/artist/title
```

The part in red doesn't change, meanwhile, we need to provide values for the two strings in blue, for example:

```
https://api.lyrics.ovh/v1/coldplay/yellow
```

In the notebook there are activities about using this API.

Example 2: Get book information

A second API shown in the notebook is the Open Library API from Internet Archive (https://openlibrary.org/developers/api). This API is very extensive, but we show two examples.

```
https://openlibrary.org/isbn/9780140328721.json
```

This first example returns data about a book with a given ISBN (International Standard Book Number) in JSON format.

```
https://openlibrary.org/authors/OL34184A.json
```

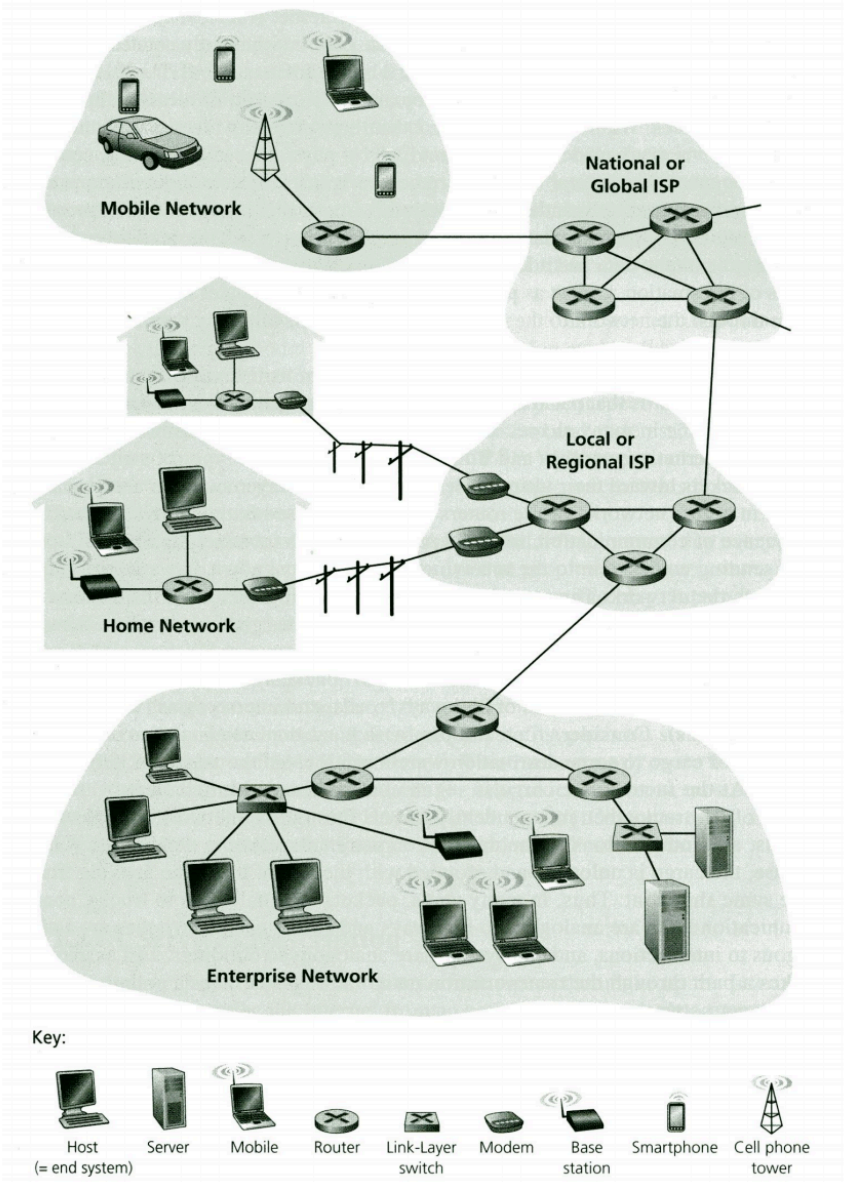
This second example returns data about the author with the provided Open Library id number.

What is the Internet?
What is the Web?
How are they related?

What is the Internet?

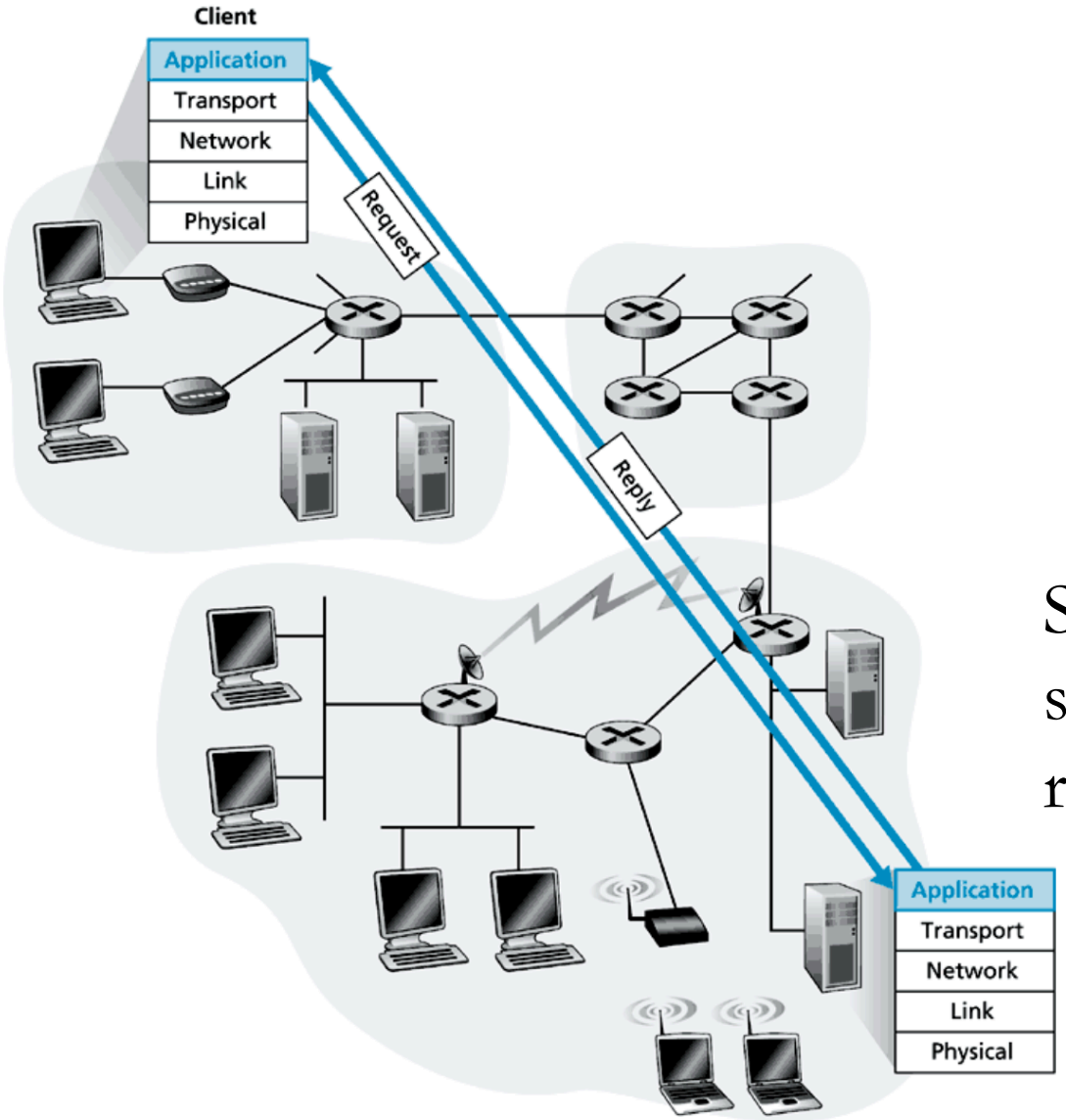
A system of **inter**connected computer **net**works that link together billion of devices using the TCP/IP communication protocols.

Take CS242 Computer Networks to learn more about TCP/IP.



Clients and Servers

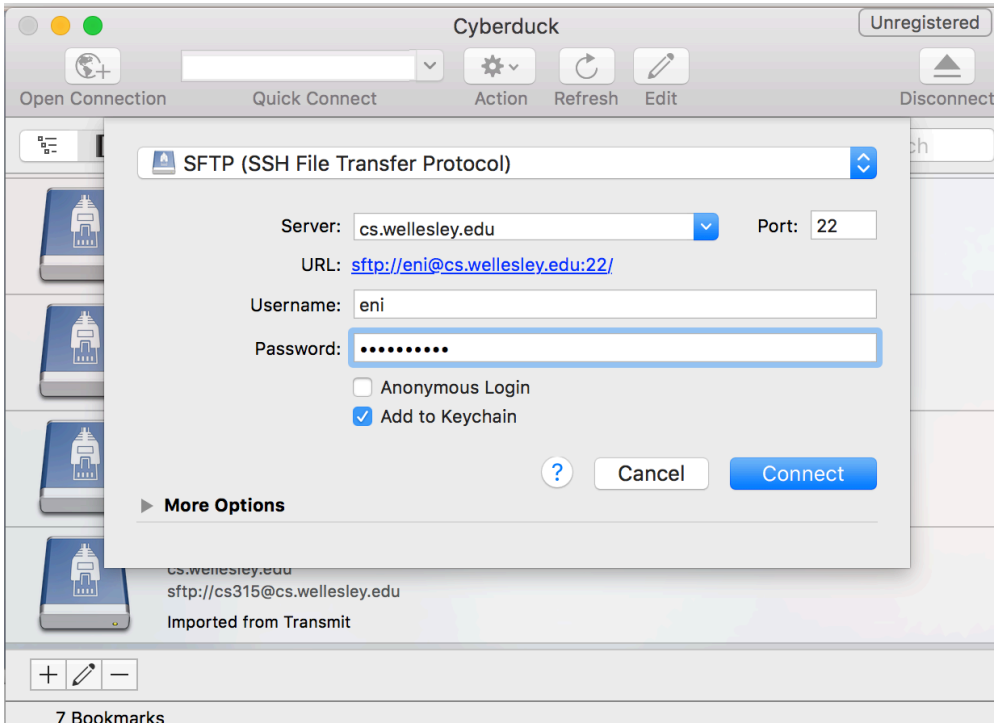
Client sends a request



Server sends a reply

Example: File Transfer

Client: You and your CyberDuck application.



Server: cs.wellesley.edu

Request: Here is a file to save in my cs server account.
Response: Got it.

Example: WWW

Client: You and your browser.

CS111
COMPUTER PROGRAMMING & PROBLEM SOLVING

Home
Schedule
Instructors & Tutors
Help Hours
Course info...
Reference...

Fall 2021 Schedule

Key: Today | Lecture | Lab | Pset Out | Pset Due | *Info* | Quiz | Exam

If a deadline conflicts with a religious holiday you observe (that's not listed), please contact your instructor to discuss accommodations.

September

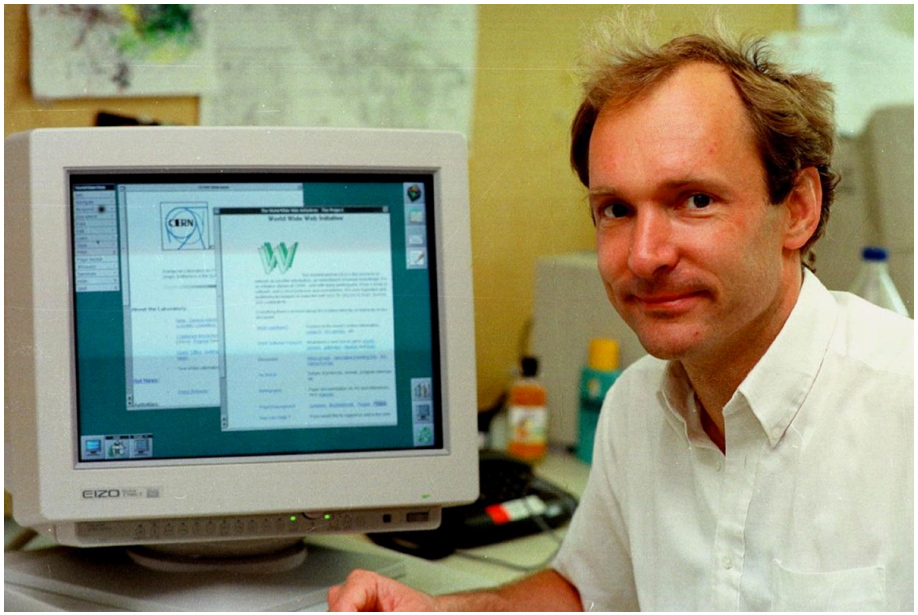
SUN	MON	TUE	WED	THU	FRI	SAT
5	6	7	8 *First day of classes* Lab 1: Workflow, Thonny and Python ps0 out ps01 out	9 Lab 1: Workflow, Thonny and Python Lec 1: Course Overview and Big Ideas ps0 due by 7:00pm	10 Lec 1: Course Overview and Big Ideas Course policies quiz due	11
12	13 Lec 2: The Python Language	14 Lec 2: The Python Language ps01 due ps02 out	15 Lab 2: Tracing & Debugging	16 Lab 2: Tracing & Debugging Lec 3: Functions 1	17 Lec 3: Functions 1 Quiz 1 (Python basics) due	18
19	20 Lec 4: Functions II	21 Lec 4: Functions II ps02 due ps03 out	22 Lab 3: Functions	23 Lab 3: Functions Lec 5: Functions III + Connection Topic I	24 Lec 5: Functions III + Connection Topic I Quiz 2 (Functions) due	25

Server: cs111.wellesley.edu

Request: Show me the schedule page.

Response: Here you go.

WWW = World Wide Web



Tim Berners-Lee is the inventor of the WWW (1989), **an application** that runs on the Internet.

He created:

- URLs,
- the HTTP protocol,
- the HTML language

He didn't patent his technology, he put it on the Internet for free, so that other people could build upon it.

He was awarded the ACM A.M. Turing Award in 2016 for contributions to computer science.

Internet vs. WWW

- The Internet is the **physical** network of computers all over the world.
- The World Wide Web is a **virtual** network of websites connected by hyperlinks (or links).
- The Web is only one of the many applications that "run" on the Internet.
- The Web uses the HTTP (Hyper Text Transfer Protocol) to allow clients and servers to communicate.
- A client: Chrome, Safari, Firefox, Internet Explorer.
- Servers: `nytimes.com`, `facebook.com`, `cs.wellesley.edu`

Summary

1. Internet is composed of hardware and software components that allow computers to connect to one another across the world.
2. The WWW (Web) is only one of the many applications that use internet. Other applications are SFTP (through CyberDuck), email, VoiceOverIP (e.g. Skype) and many others.
3. The HTTP protocol establishes how a client (your browser) and a server (e.g. cs.wellesley.edu) should communicate via messages.
4. The usual way of accessing information on the Web is via a browser, however, we can use Python's module **`requests`** to ask for pages and data from servers on the Internet.
5. Web APIs specify how we can formulate requests to these servers. Such requests are mostly URLs that contain several parts.
6. We can think of a request URL as function invocation: it contains a function name to call and it specifies named arguments that are passed to the function.
7. The results that we receive from an API request are usually in the form of JSON strings. We can use Python to work with the data once we convert it to a Python objet (list of lists, dictionary of dictionaries, etc.).