[301] Web 1

Tyler Caraza-Harter

Learning Objectives Today

Network basics

- IP addresses
- host/domain names
- client/server and request/response

HTTP basics

- URLs
- GET/POST/etc
- headers
- status codes

Requests modules

- downloading data with requests.get
- remote calls with requests.post

Learning Objectives Today

Motivation

Networking Basics

HTTP (Hypertext Transfer Protocol)

Requests Module

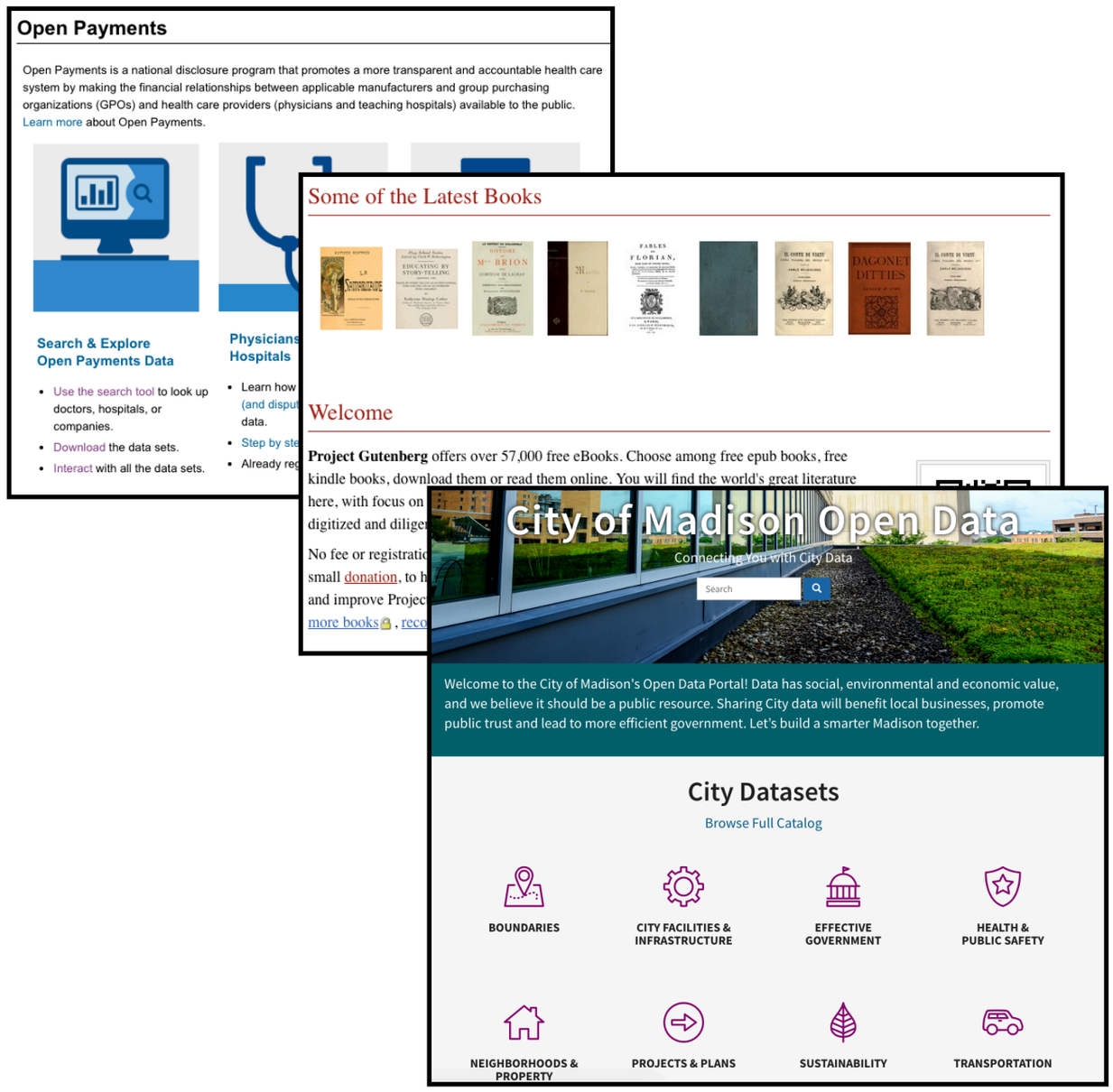
Data Science and the Internet

There are tons of online sources of data

- Examples: https://tyler.caraza-harter.com/cs301/fall18/datasets.html

Wide range of topics

- healthcare
- roads and city planning
- astronomy
- population
- business
- entertainment
- education
- etc



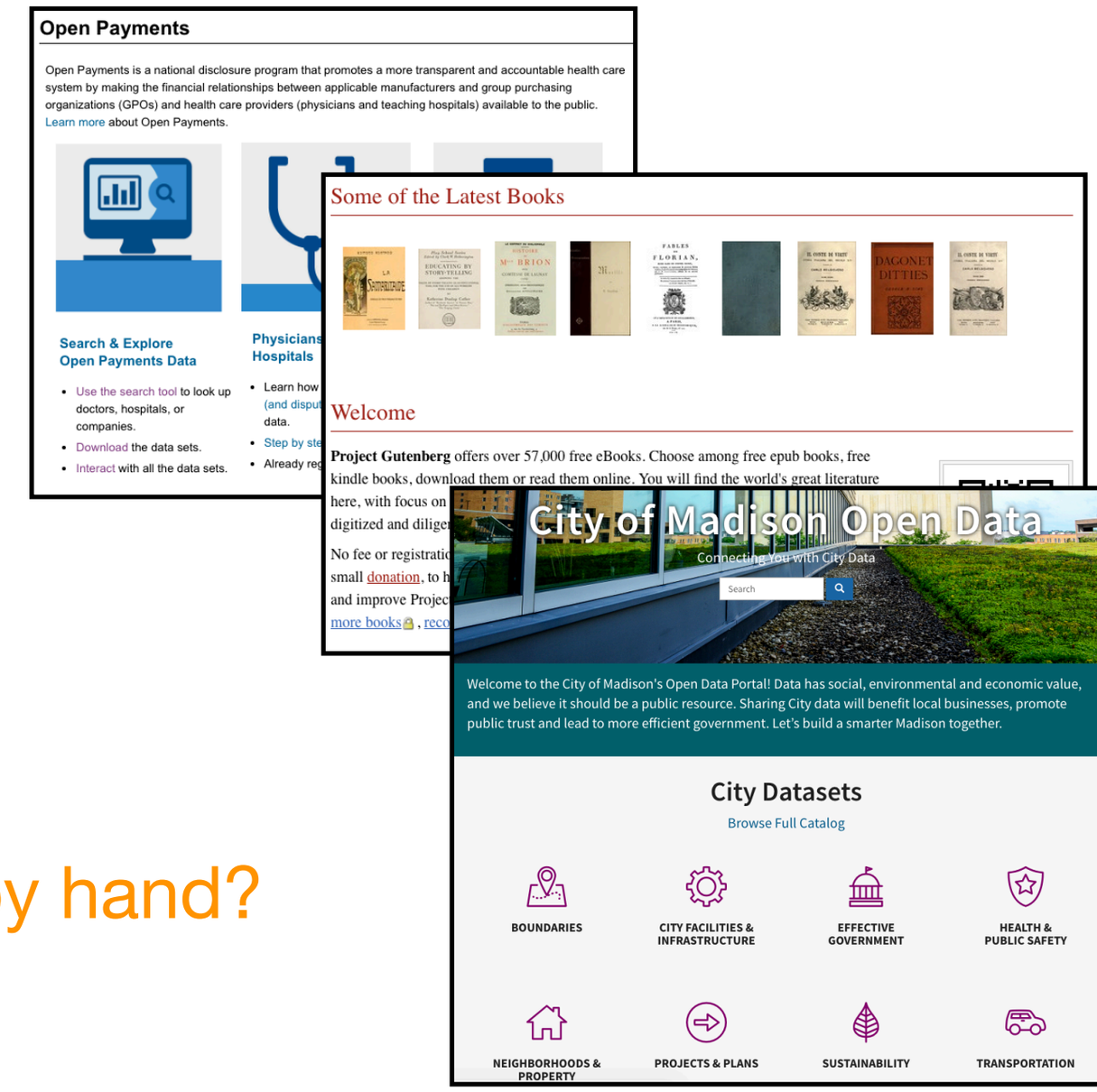
Data Science and the Internet

There are tons of online sources of data

- Examples: https://tyler.caraza-harter.com/cs301/fall18/datasets.html

Wide range of topics

- healthcare
- roads and city planning
- astronomy
- population
- business
- entertainment
- education
- etc

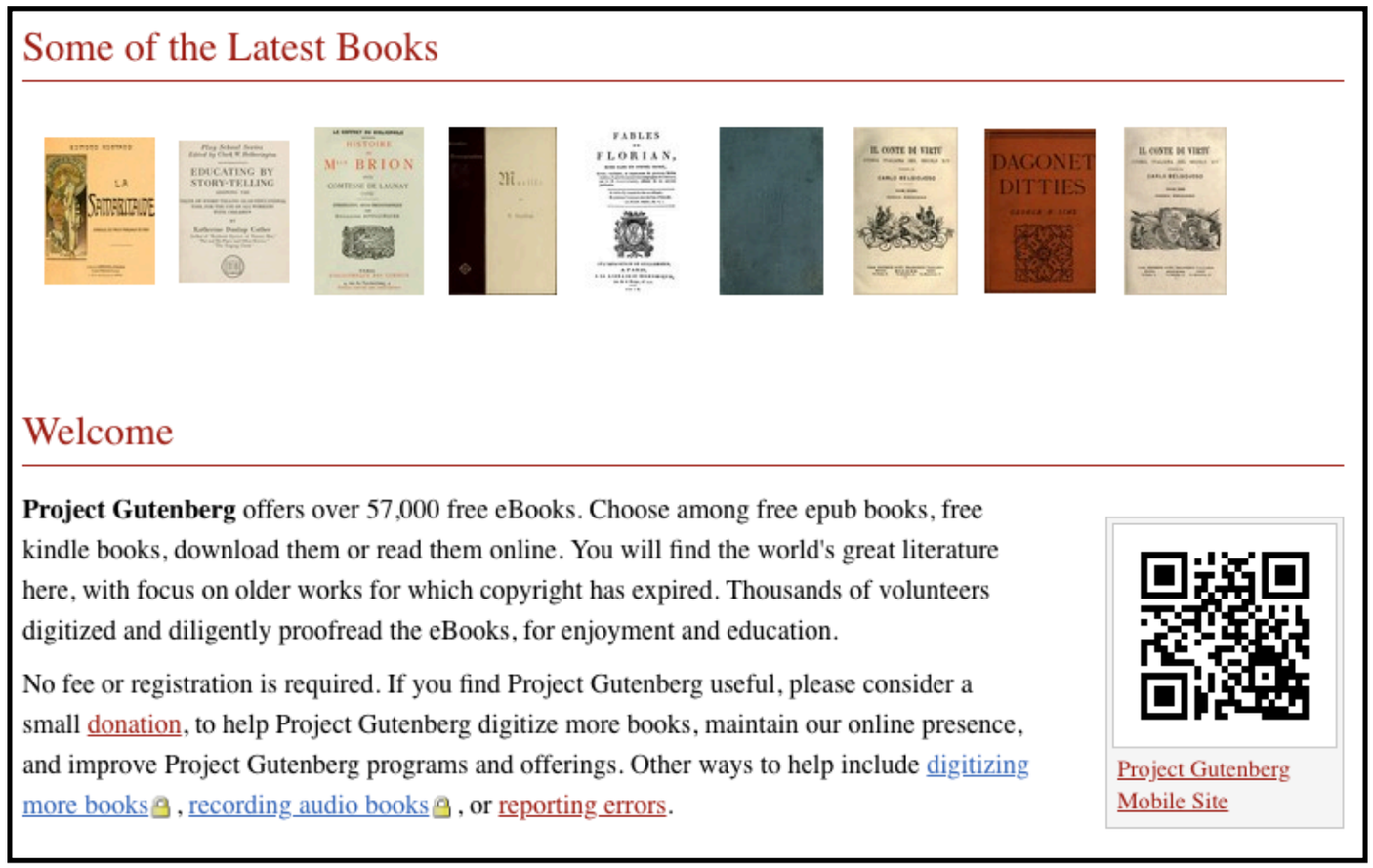


Why not just download data by hand?

Motivation 1: too much data

What if you're analyzing language trends over time?

- Dataset: Project Gutenberg has 57K free books
- Too much work to download one by one

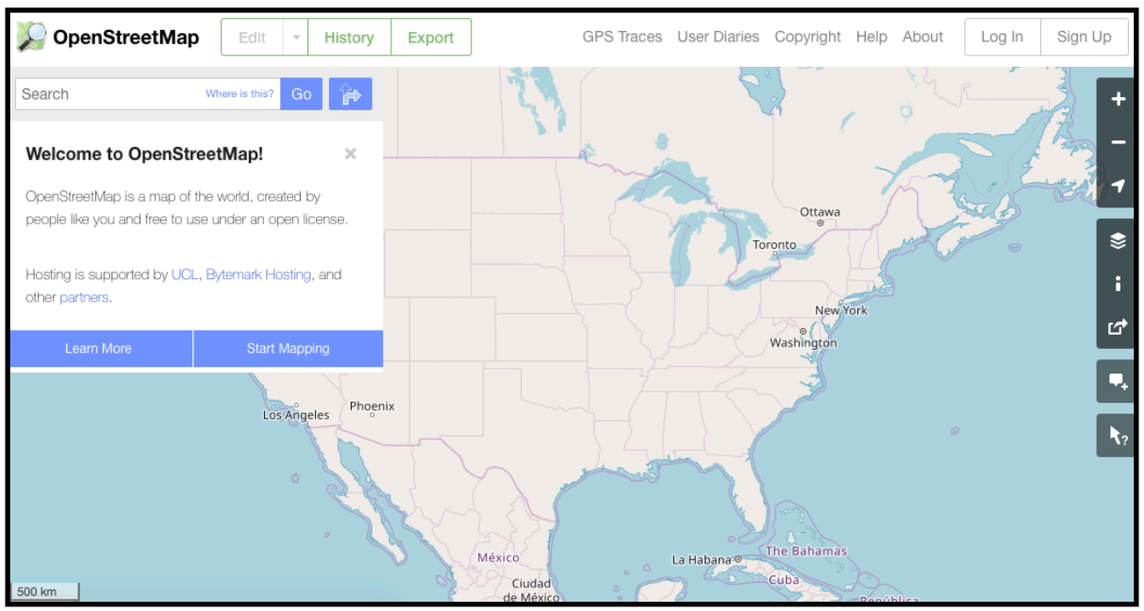


Motivation 2: data doesn't always come in files

Many datasets are difficult to download complete

Instead, you can make function calls to servers
(we'll learn how) to grab specific data

- Dataset: OpenStreetMap
- You issue calls to get specific data:
  1. specify latitude/longitude rectangle
  2. specify structures of interest (e.g., bike paths)



Learning Objectives Today

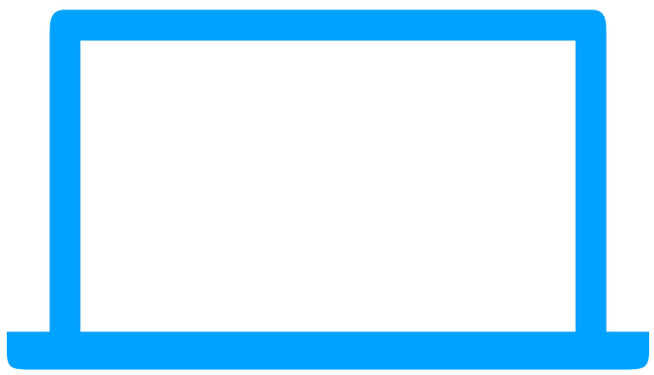
Motivation

Networking Basics

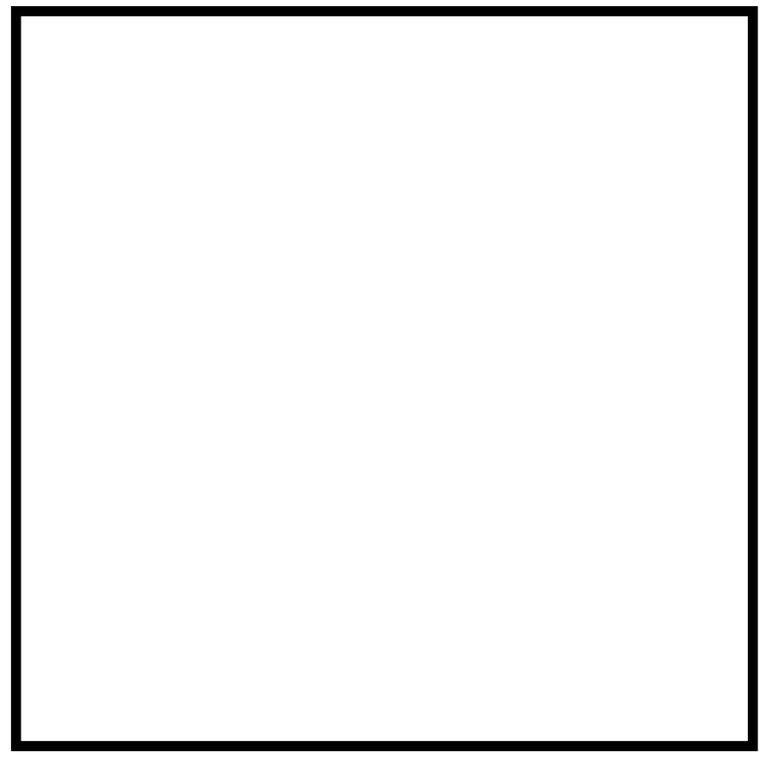
HTTP (Hypertext Transfer Protocol)

Requests Module

Networking Basics



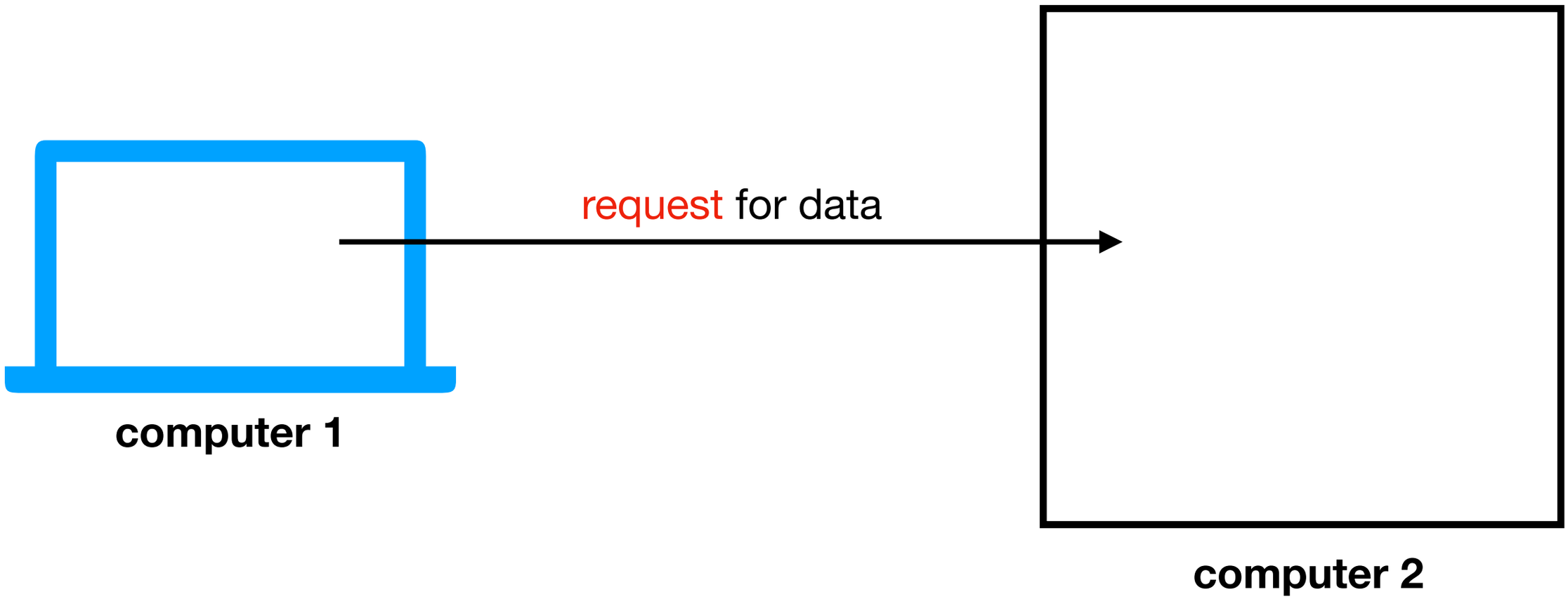
computer 1



computer 2

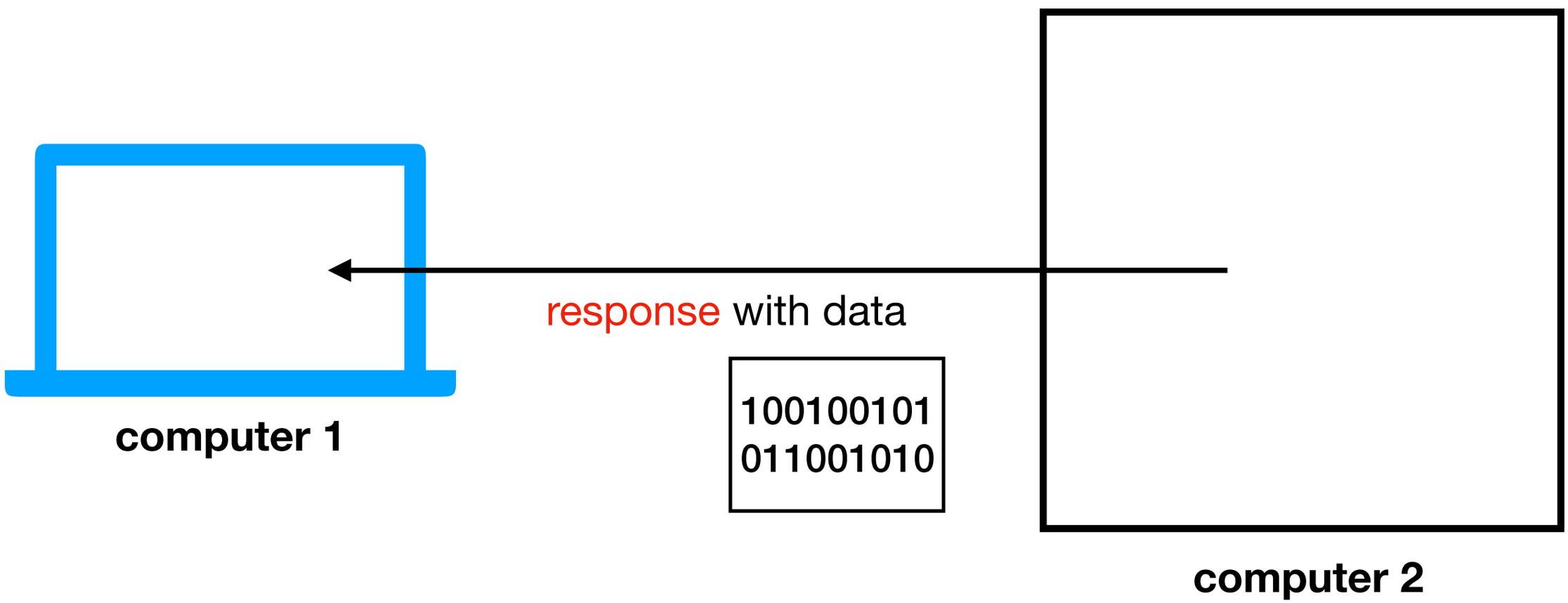
Computers communicate over a network (e.g., the Internet)
by sending messages to each other

Networking Basics



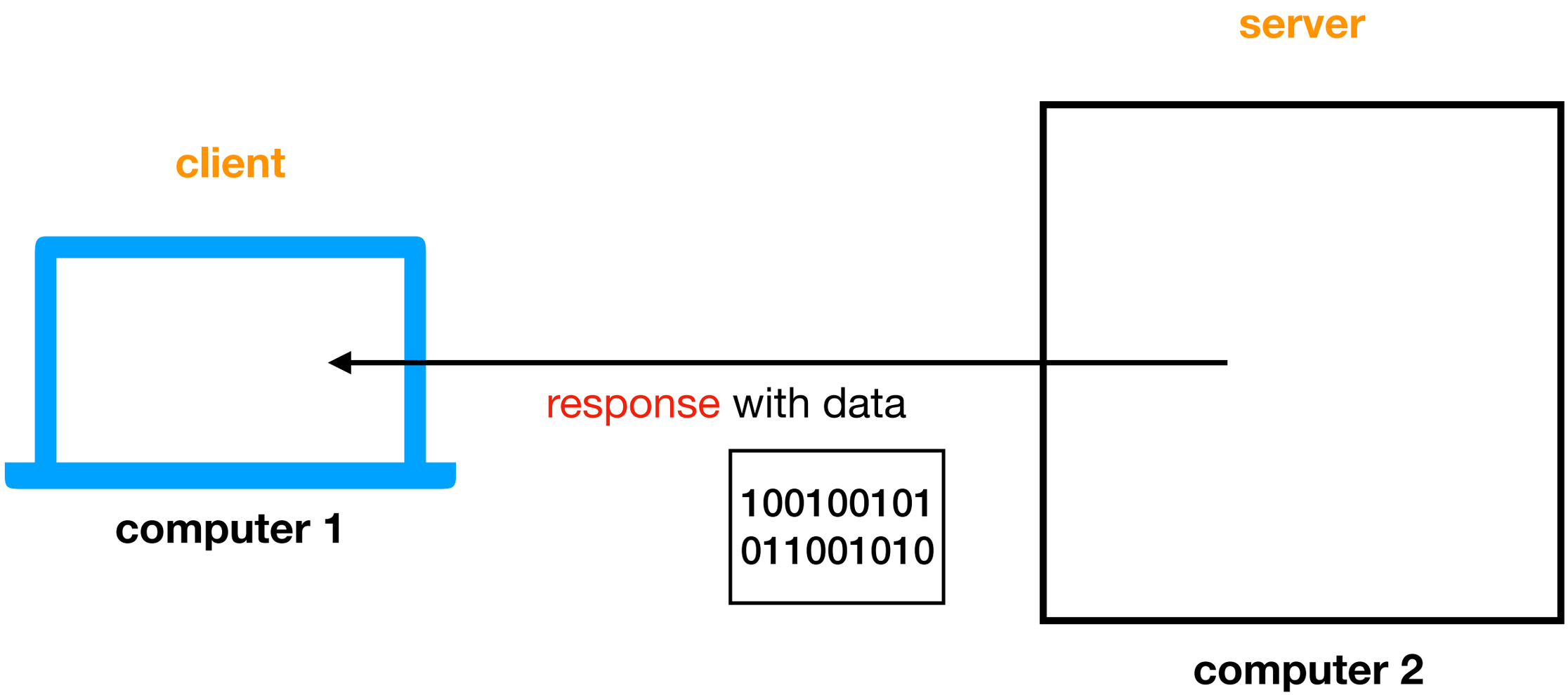
Computers communicate over a network (e.g., the Internet)
by sending messages to each other

Networking Basics



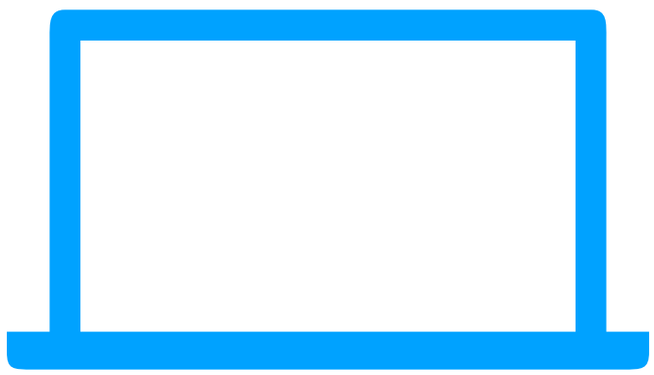
Computers communicate over a network (e.g., the Internet)
by sending messages to each other

Networking Basics

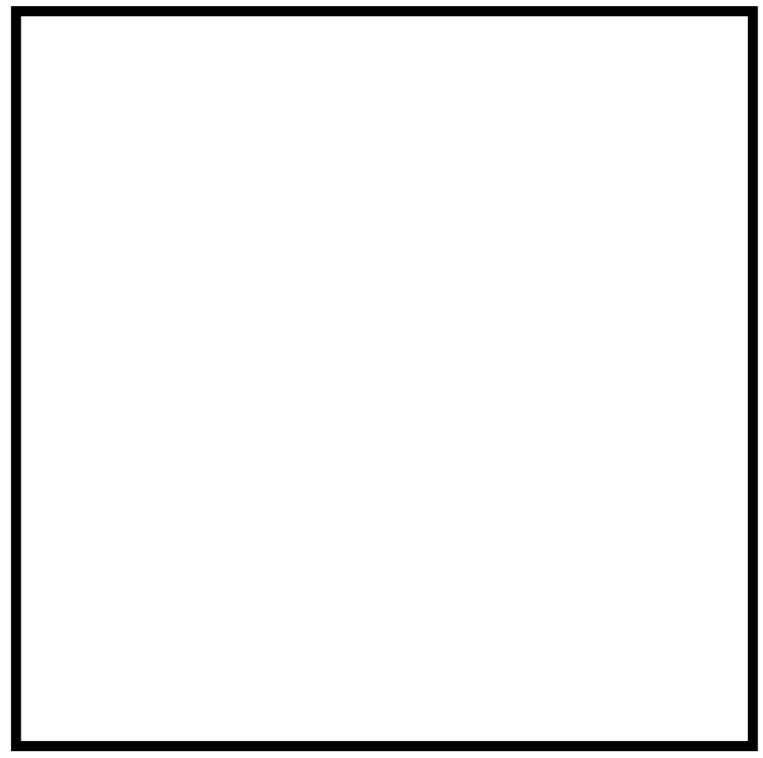


Computers communicate over a network (e.g., the Internet)
by sending messages to each other

Networking Basics



computer 1



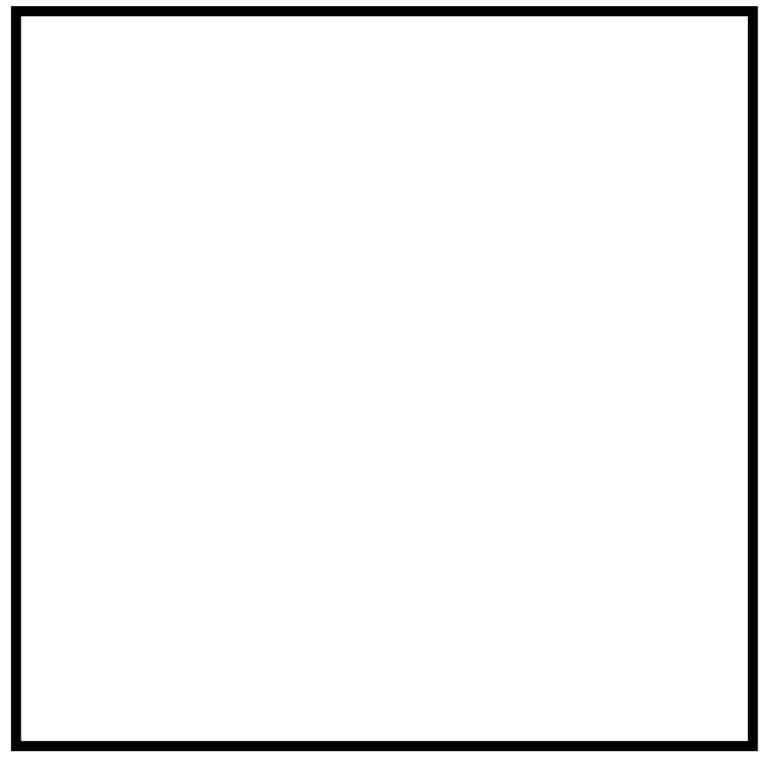
computer 2

Challenge: there are millions of computers.

How do we indicate which machine should get our request?

Internet Protocol

computer 1



computer 2

Solution: every machine* has an IP address (Internet Protocol).
Requests are sent to a specific IP address.

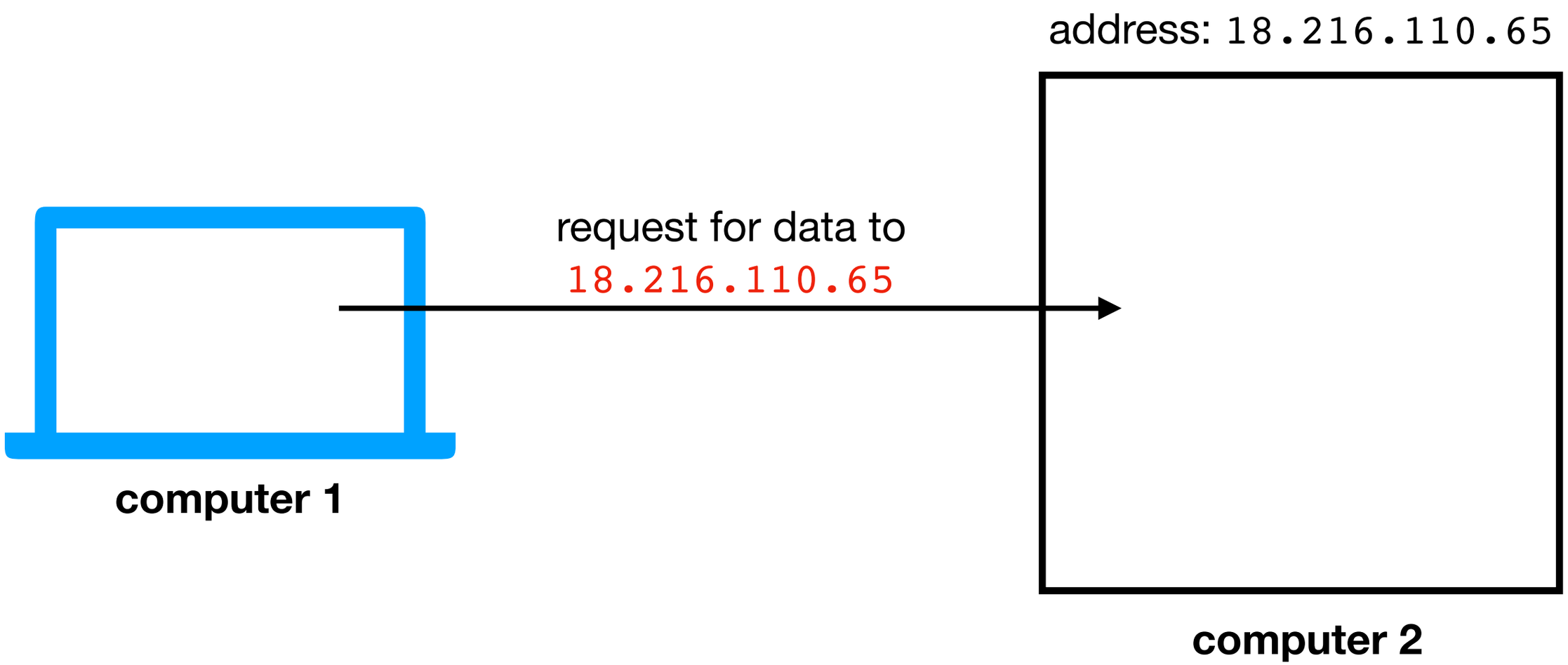
*some machines have more multiple addresses

Internet Protocol

Solution: every machine* has an IP address (Internet Protocol).
Requests are sent to a specific IP address.

*some machines have more multiple addresses

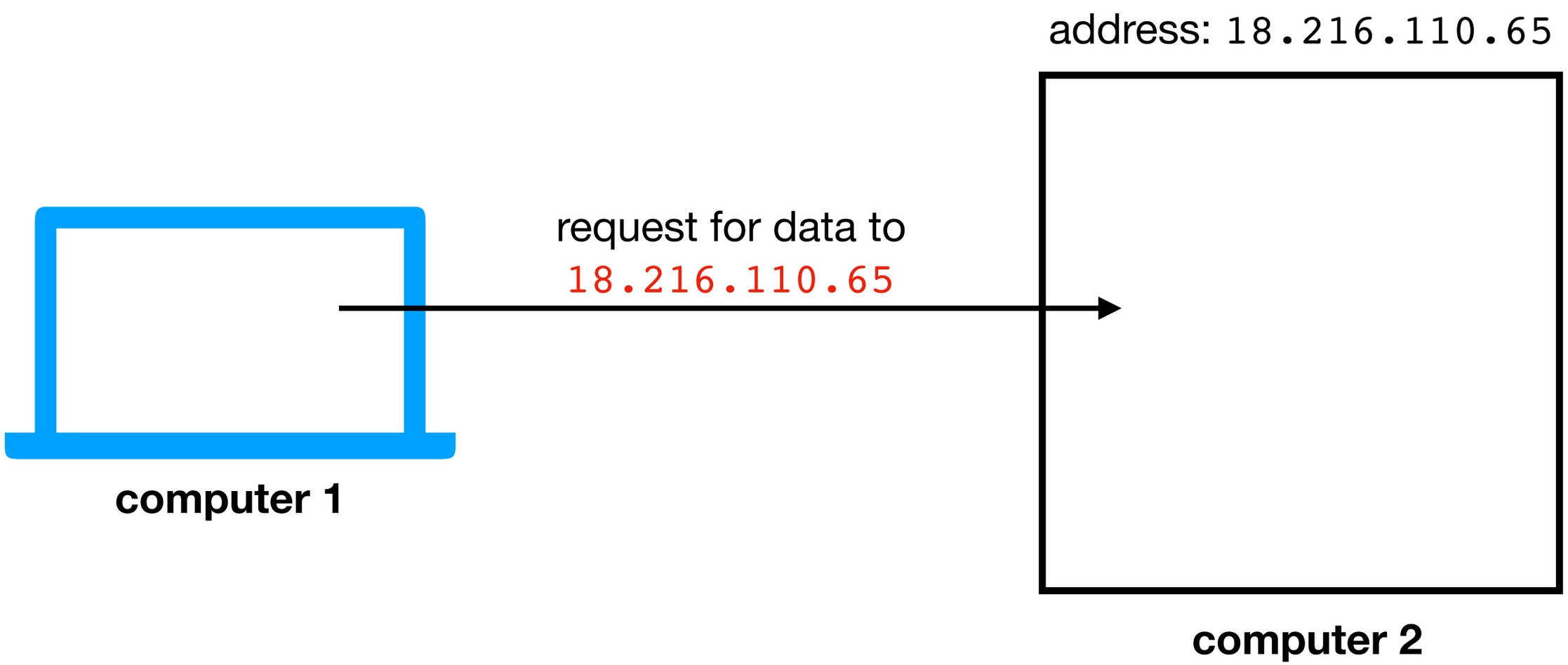
Internet Protocol



Solution: every machine* has an IP address (Internet Protocol).
Requests are sent to a specific IP address.

*some machines have more multiple addresses

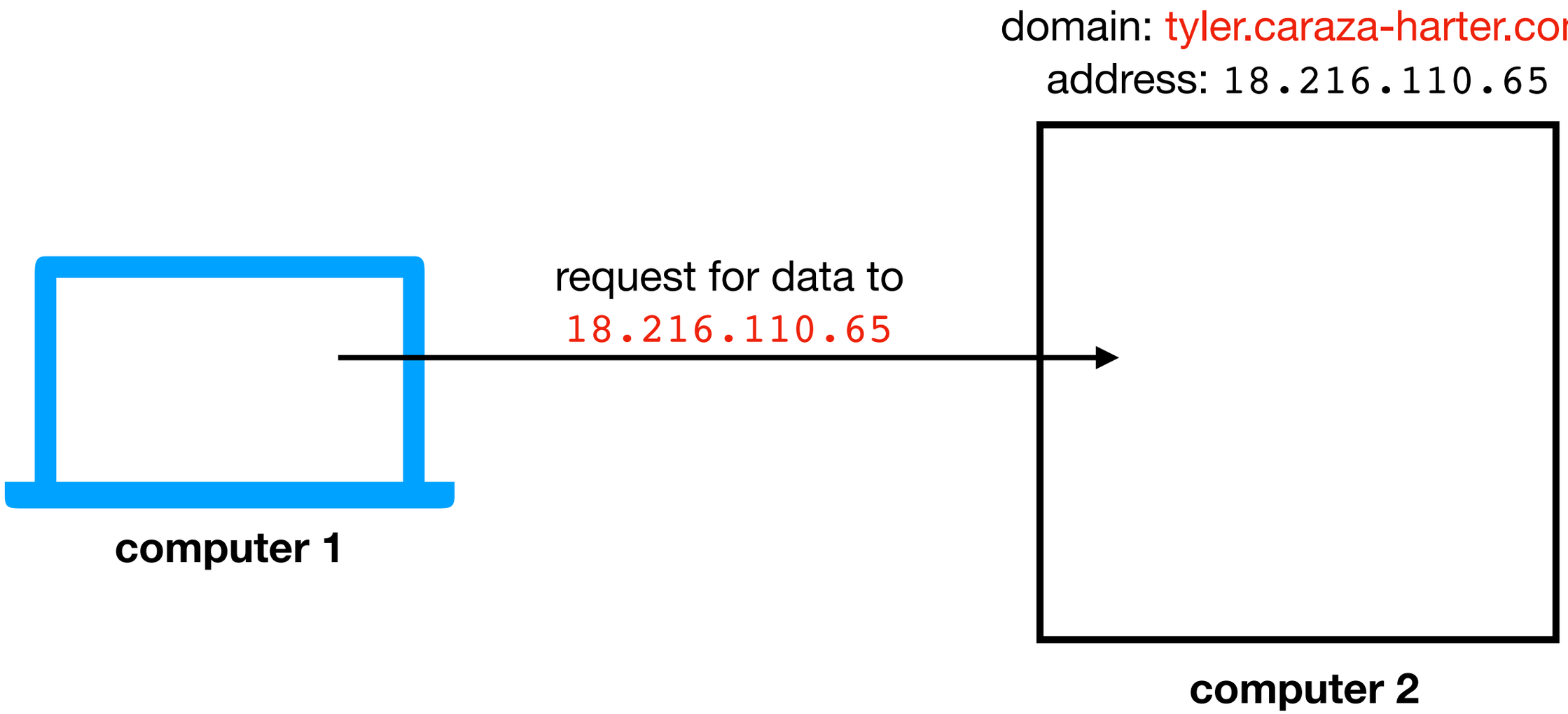
Internet Protocol



Challenge: it's hard to remember IP addresses.

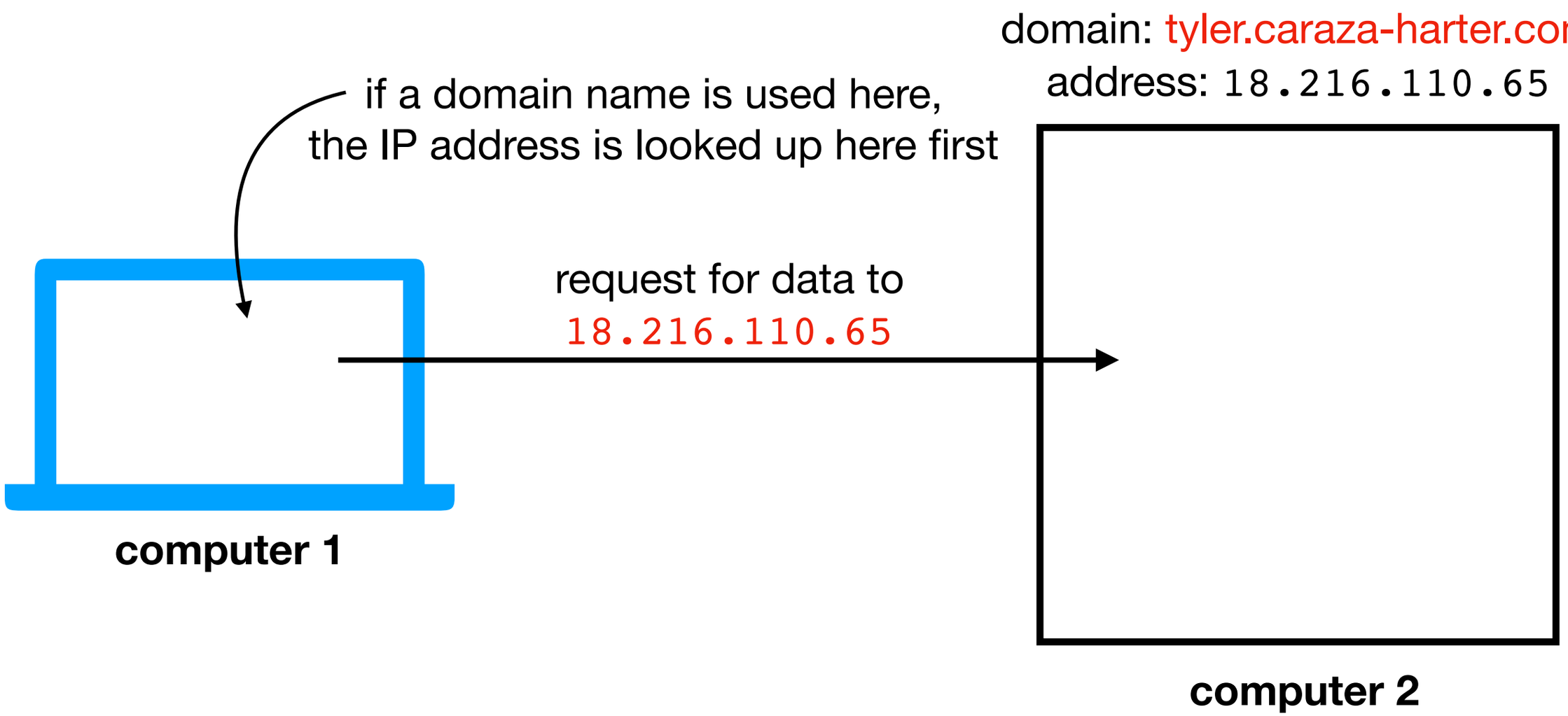
Imagine you had to type a number instead of www.google.com!

Domain Names



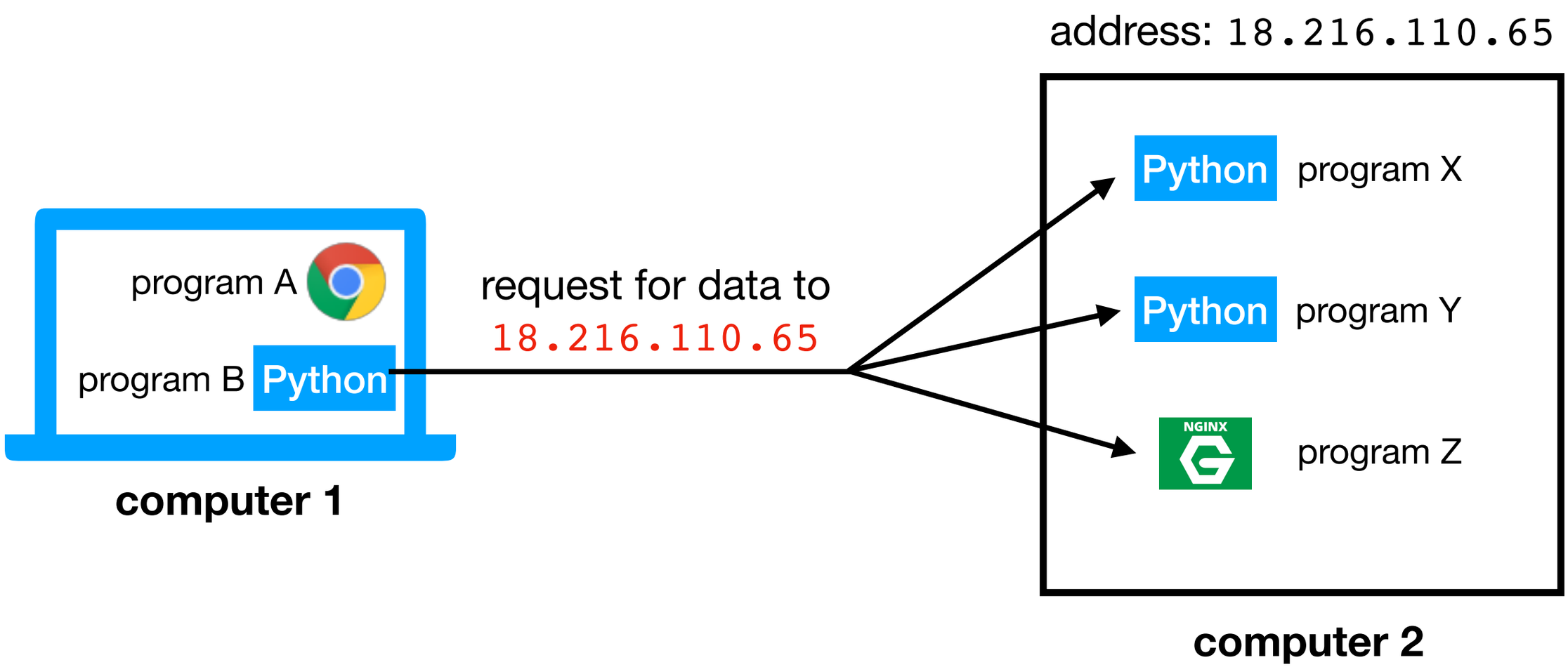
Solution: use “nicknames” (called domain names)
for IP addresses of machines that serve data

Domain Names



Solution: use “nicknames” (called domain names)
for IP addresses of machines that serve data

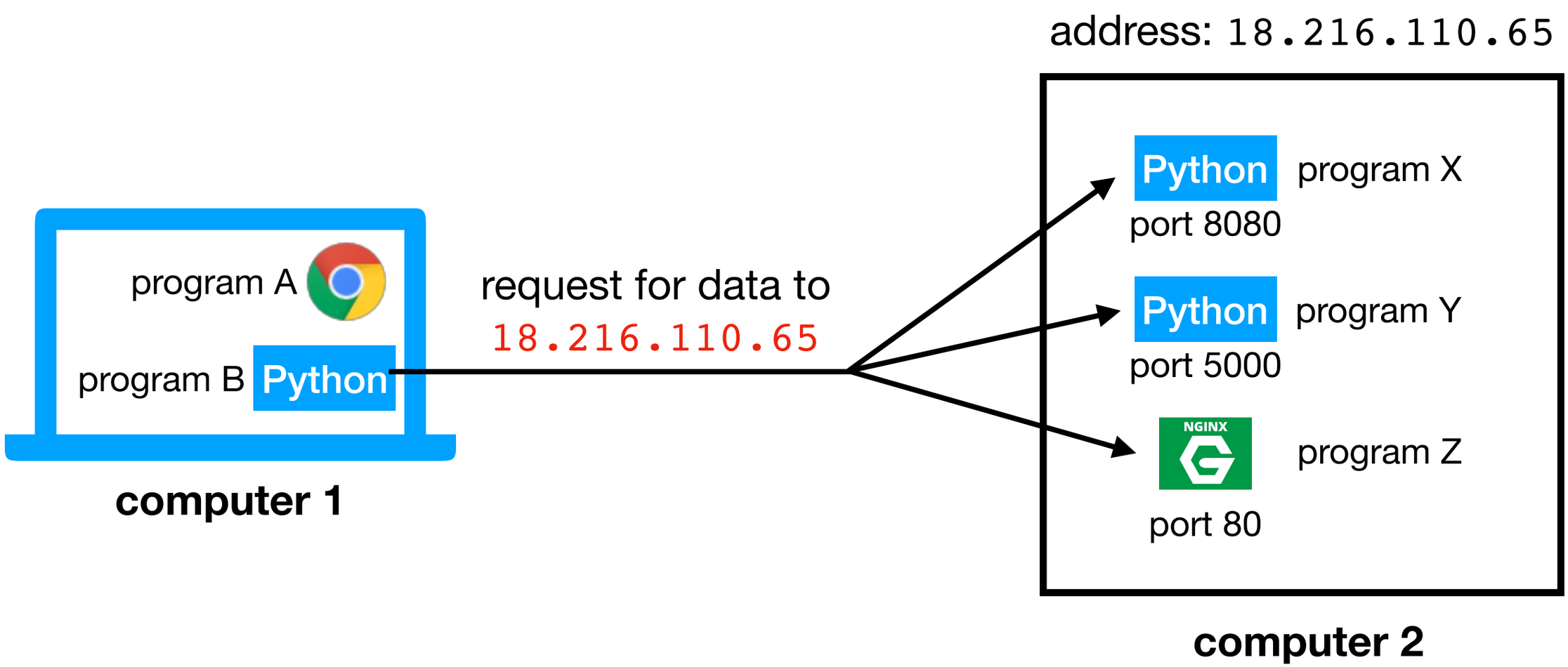
Port Numbers



Challenge: there may be multiple programs running on each computer.

How do we get the messages to the right program?

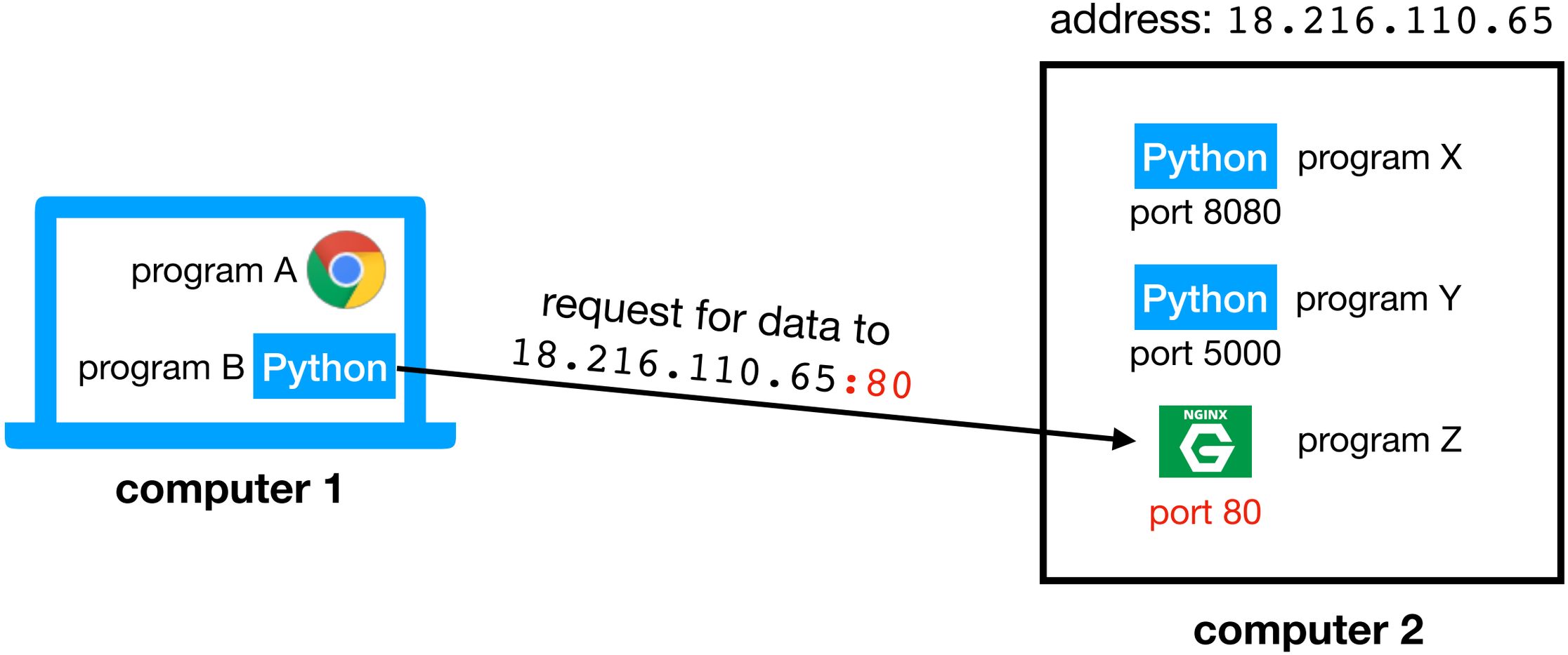
Port Numbers



Solution: give each program a unique ID (called a “port number”)

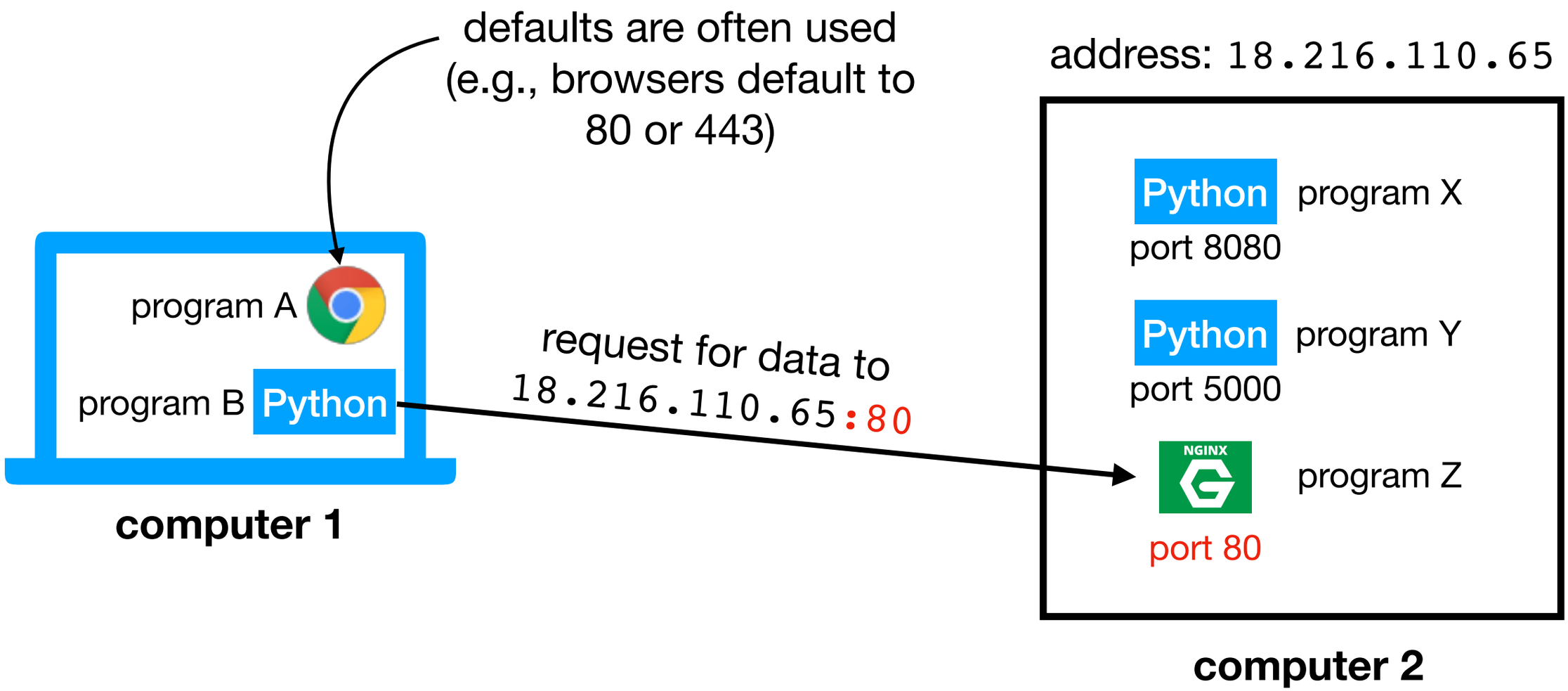
(like apartment numbers)

Port Numbers

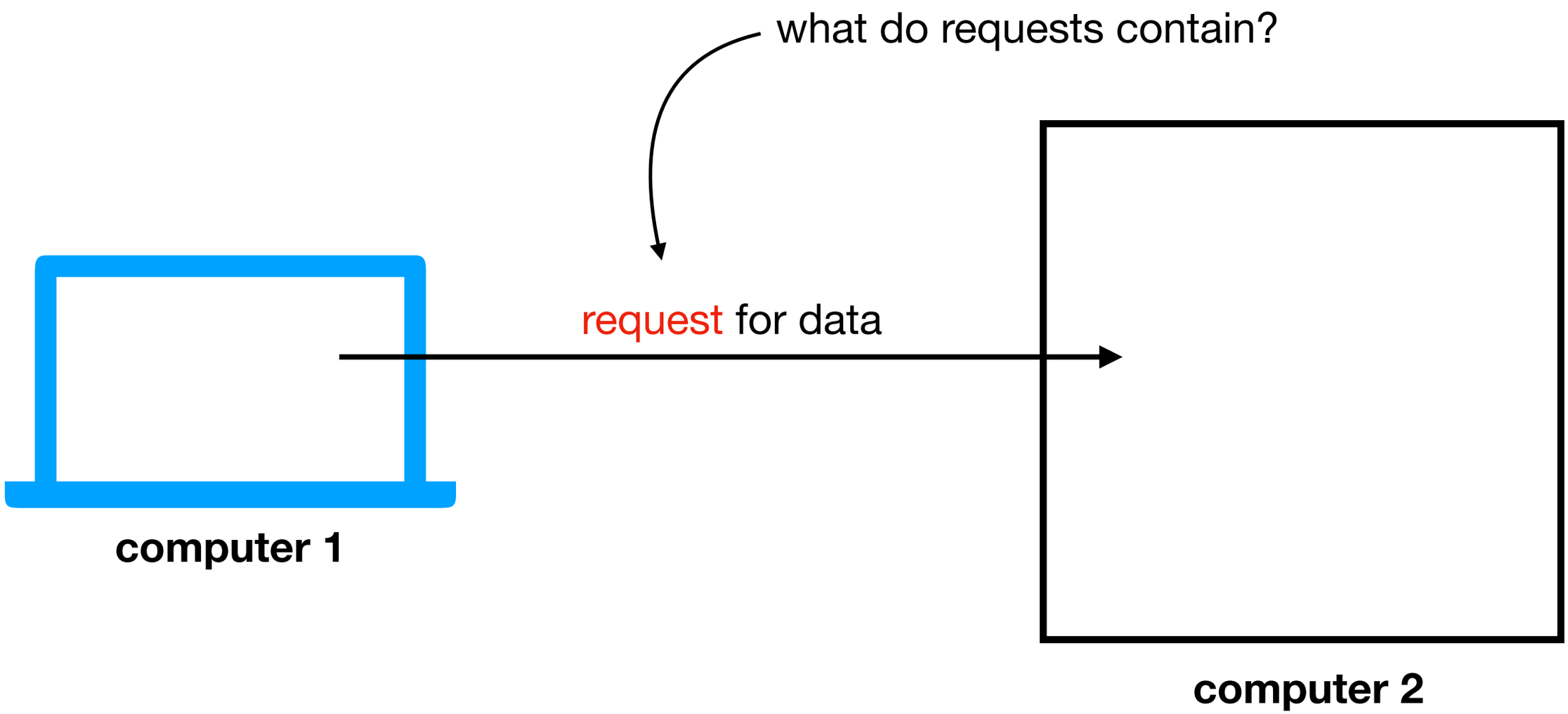


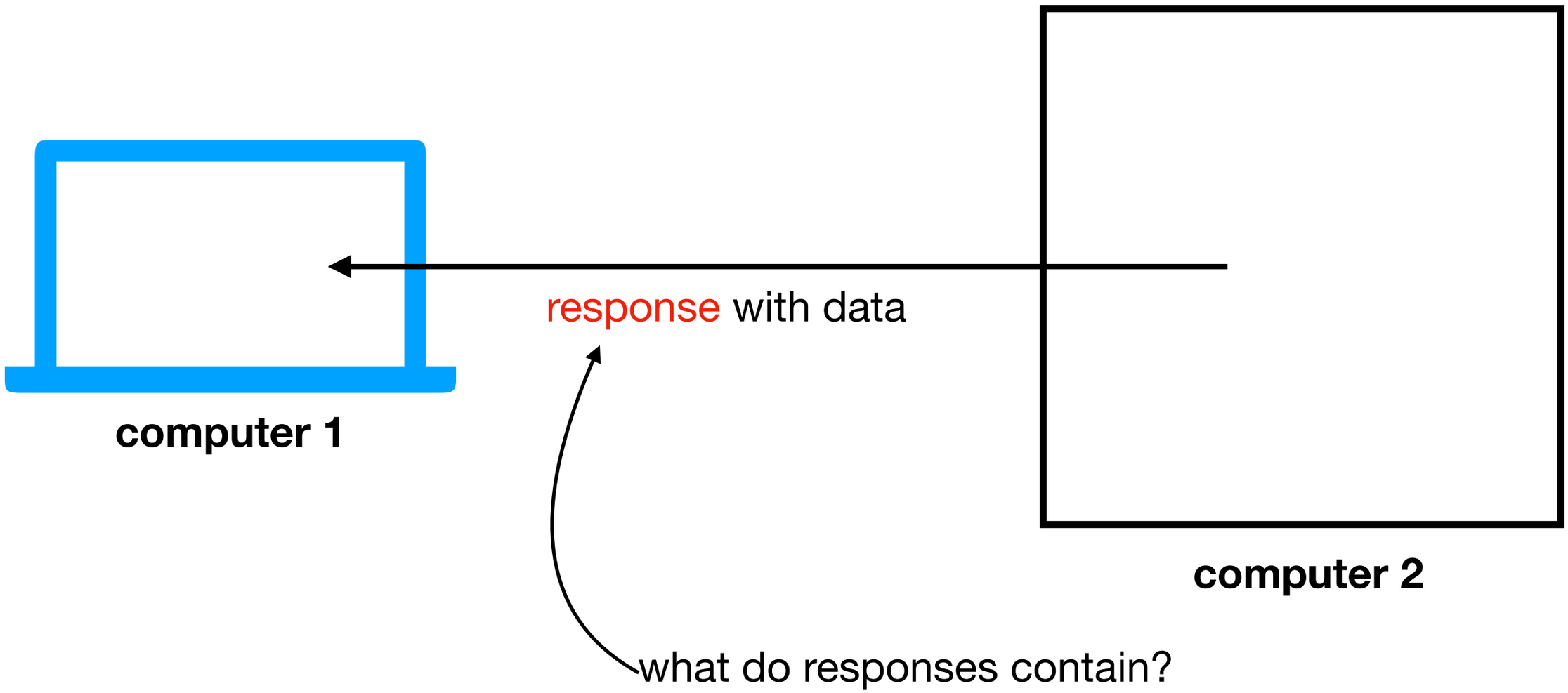
Solution: specify port number in request

Port Numbers



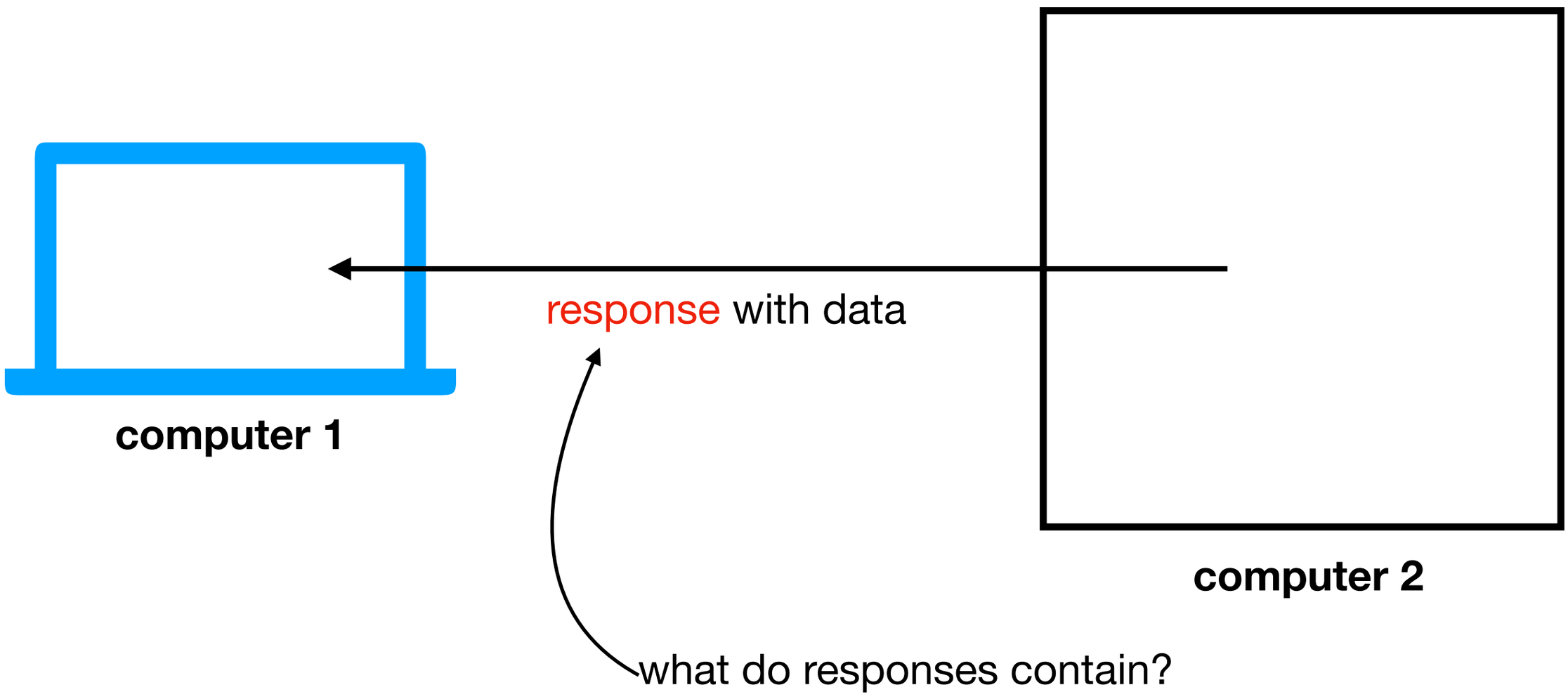
Solution: specify port number in request


what do requests contain?
request for data
computer 1
computer 2


response with data
computer 1
computer 2
what do responses contain?

depends on application! (video chat, web browsing, etc)

we'll only consider web applications for this semester



Learning Objectives Today

Motivation

Networking Basics

HTTP (Hypertext Transfer Protocol)

Requests Module

HTTP

Protocol for communicating web data

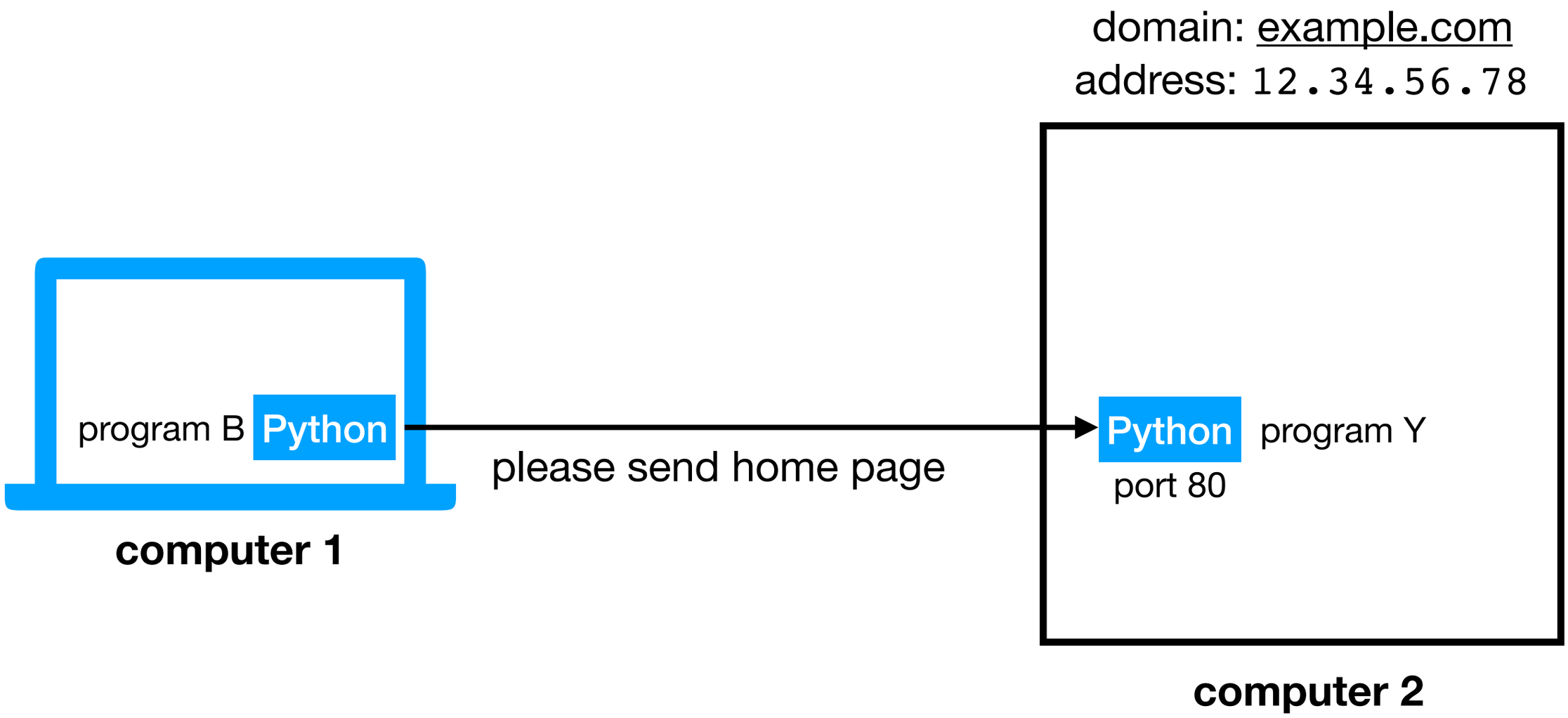
- downloading a specific webpage, image, etc

Note: we won't talk about HTTPS today, which is HTTP with encryption

HTTP

Protocol for communicating web data

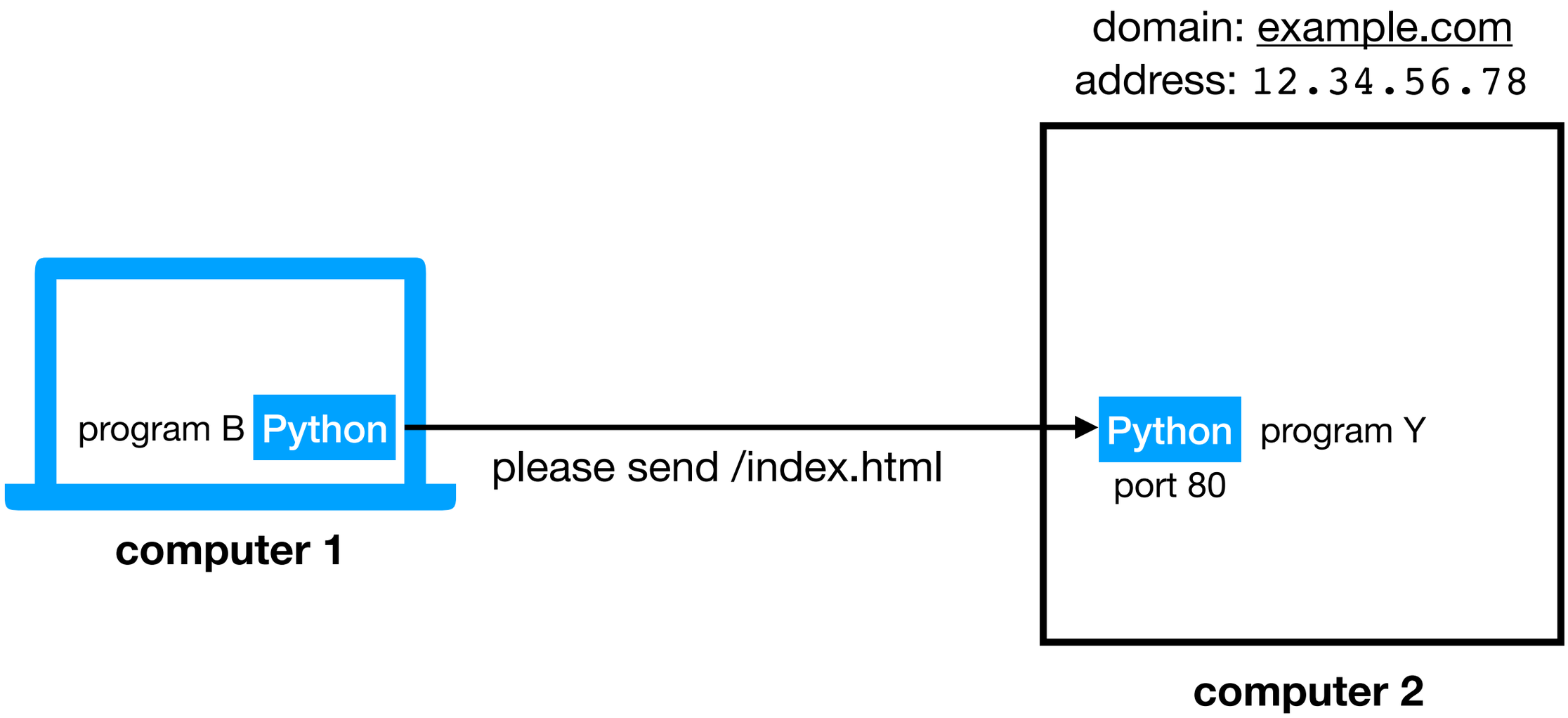
- downloading a specific webpage, image, etc



HTTP

Protocol for communicating web data

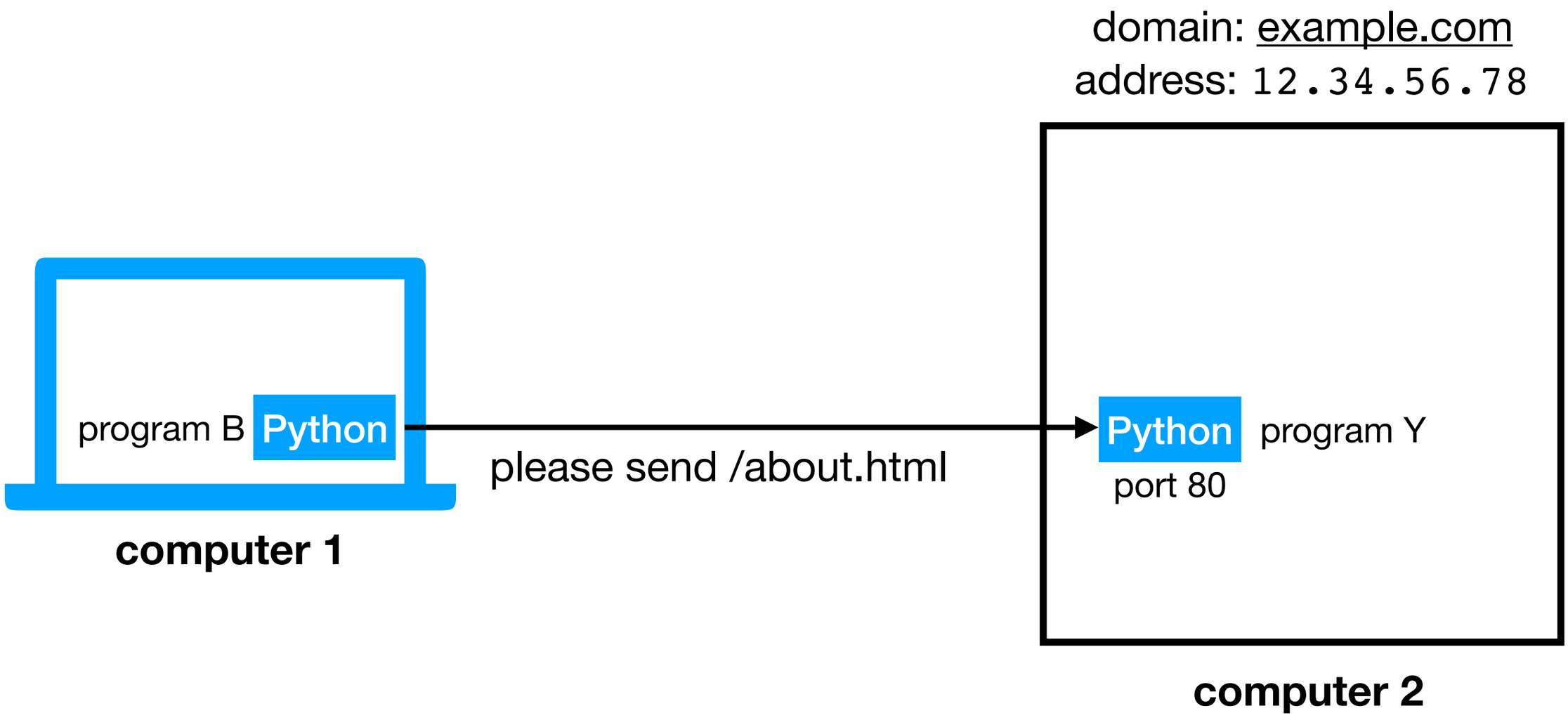
- downloading a specific webpage, image, etc



HTTP

Protocol for communicating web data

- downloading a specific webpage, image, etc



HTTP

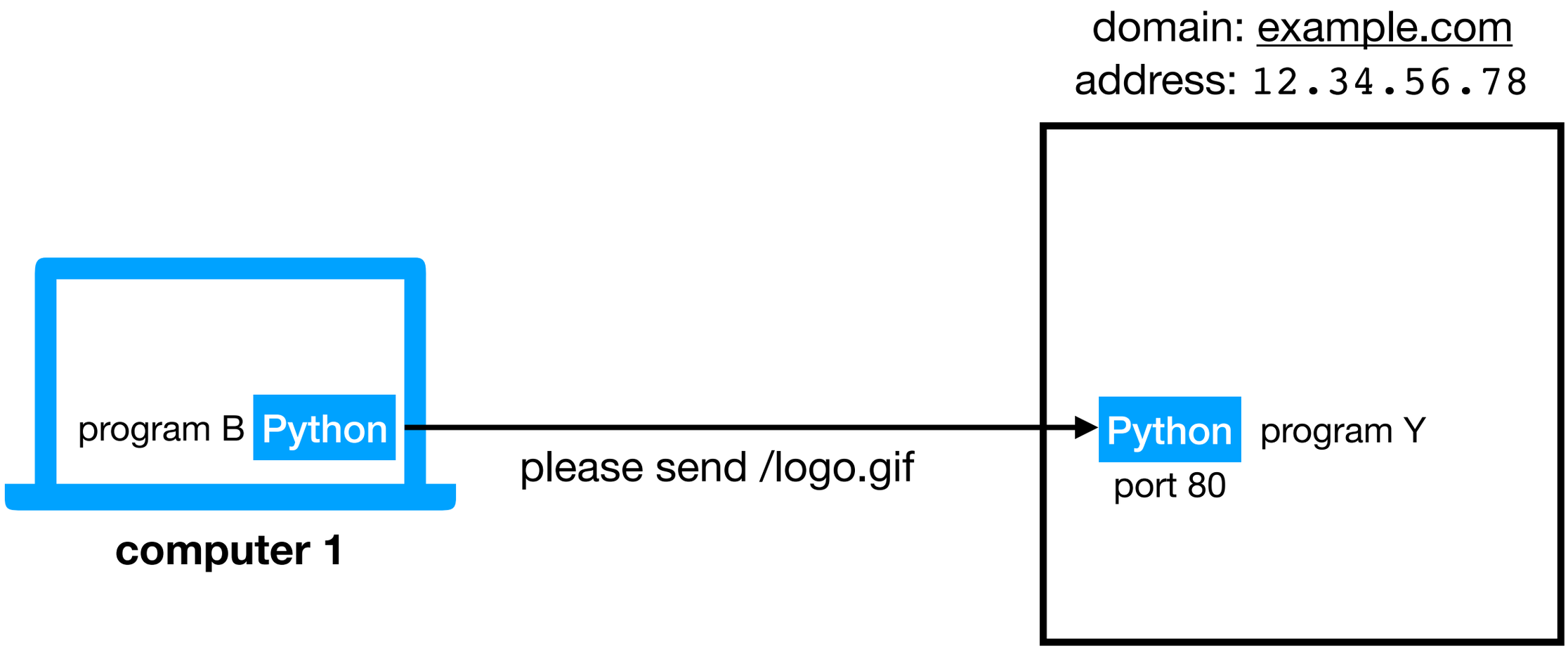
Protocol for communicating web data

- downloading a specific webpage, image, etc

HTTP

Protocol for communicating web data

- downloading a specific webpage, image, etc



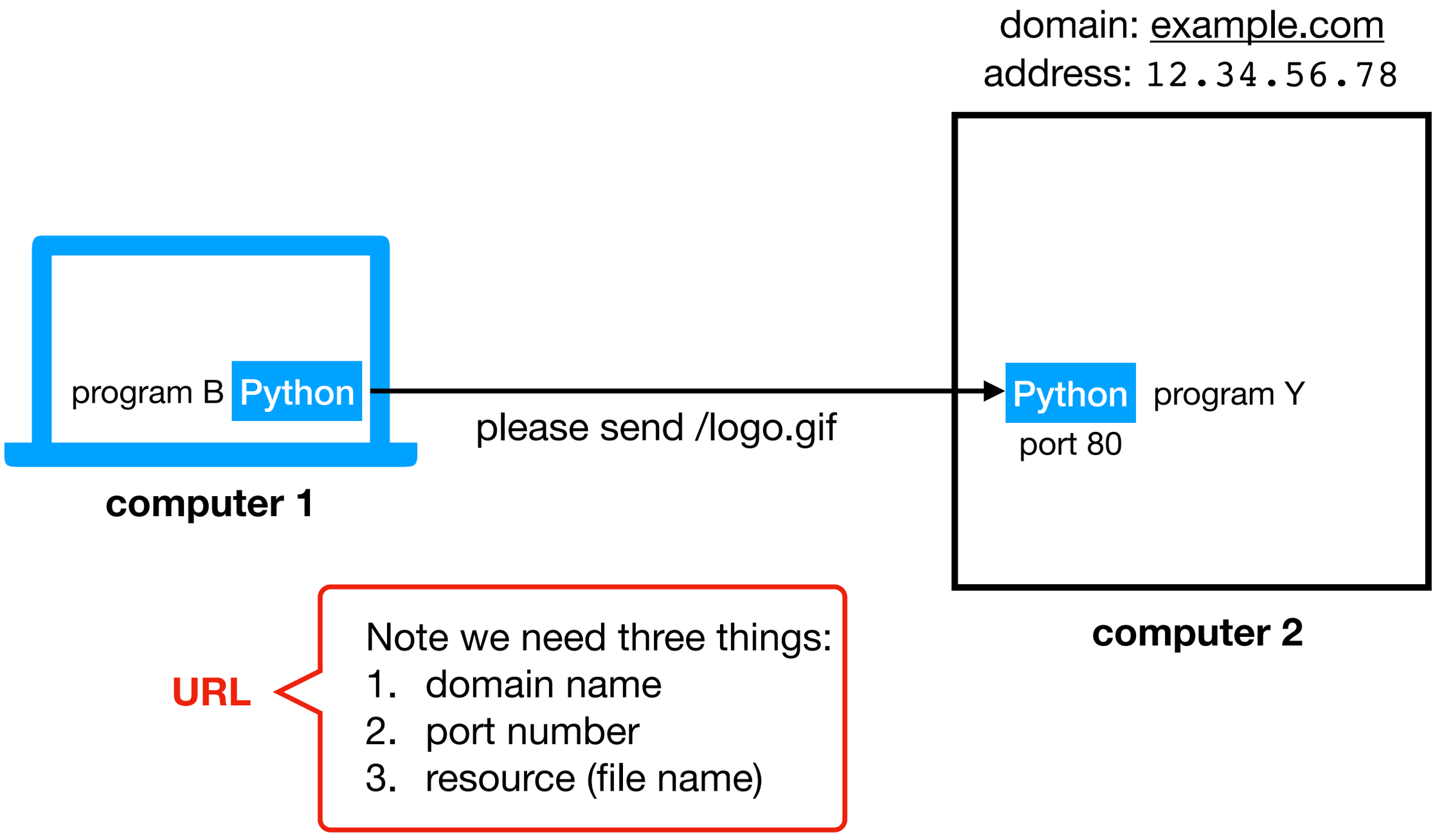
Note we need three things:

1. domain name
2. port number
3. resource (file name)

HTTP

Protocol for communicating web data

- downloading a specific webpage, image, etc



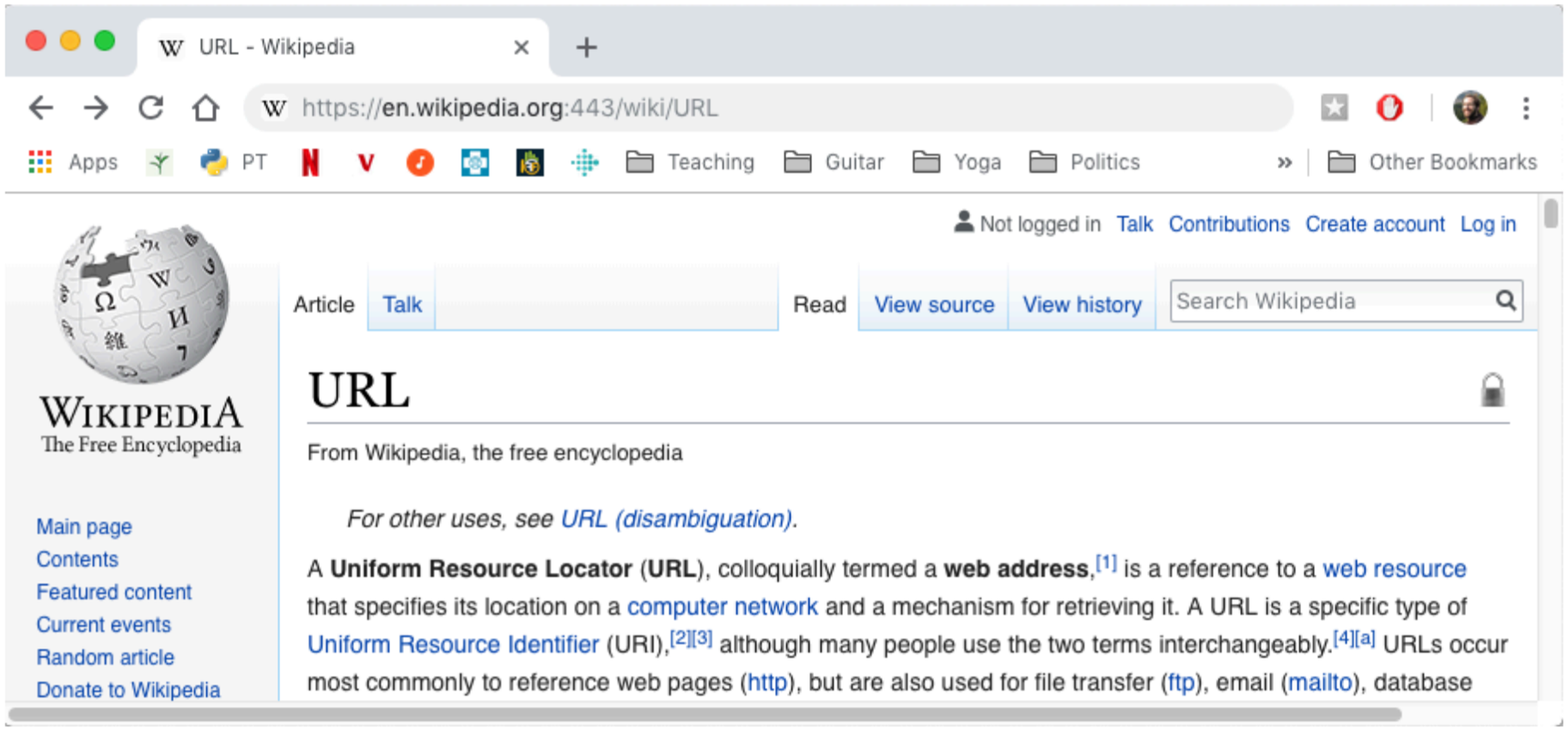
URL

Note we need three things:

1. domain name
2. port number
3. resource (file name)

URLs

https://en.wikipedia.org:443/wiki/URL



URL

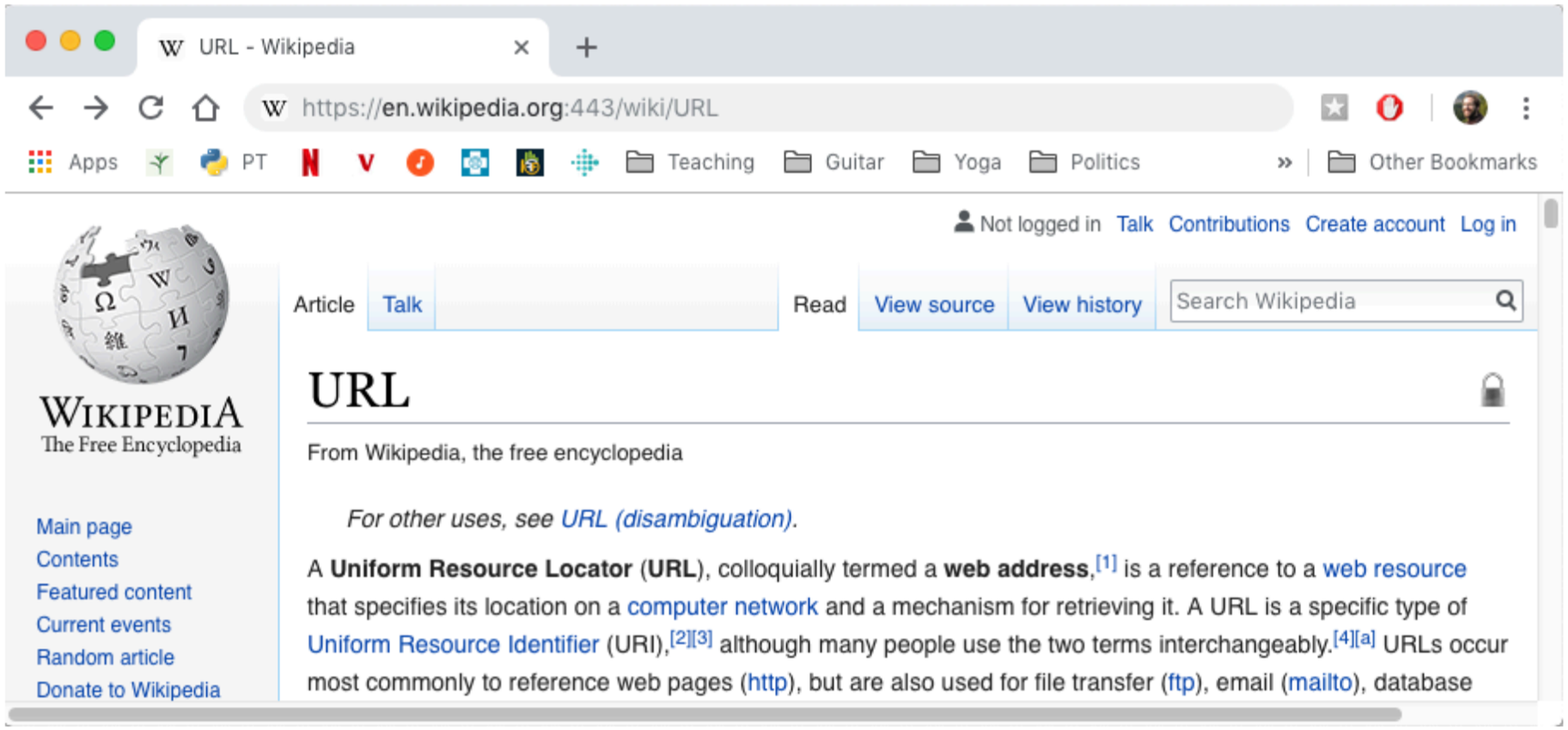
Note we need three things:

1. domain name
2. port number
3. resource (file name)

URLs

domain name

https://en.wikipedia.org:443/wiki/URL



URL

Note we need three things:

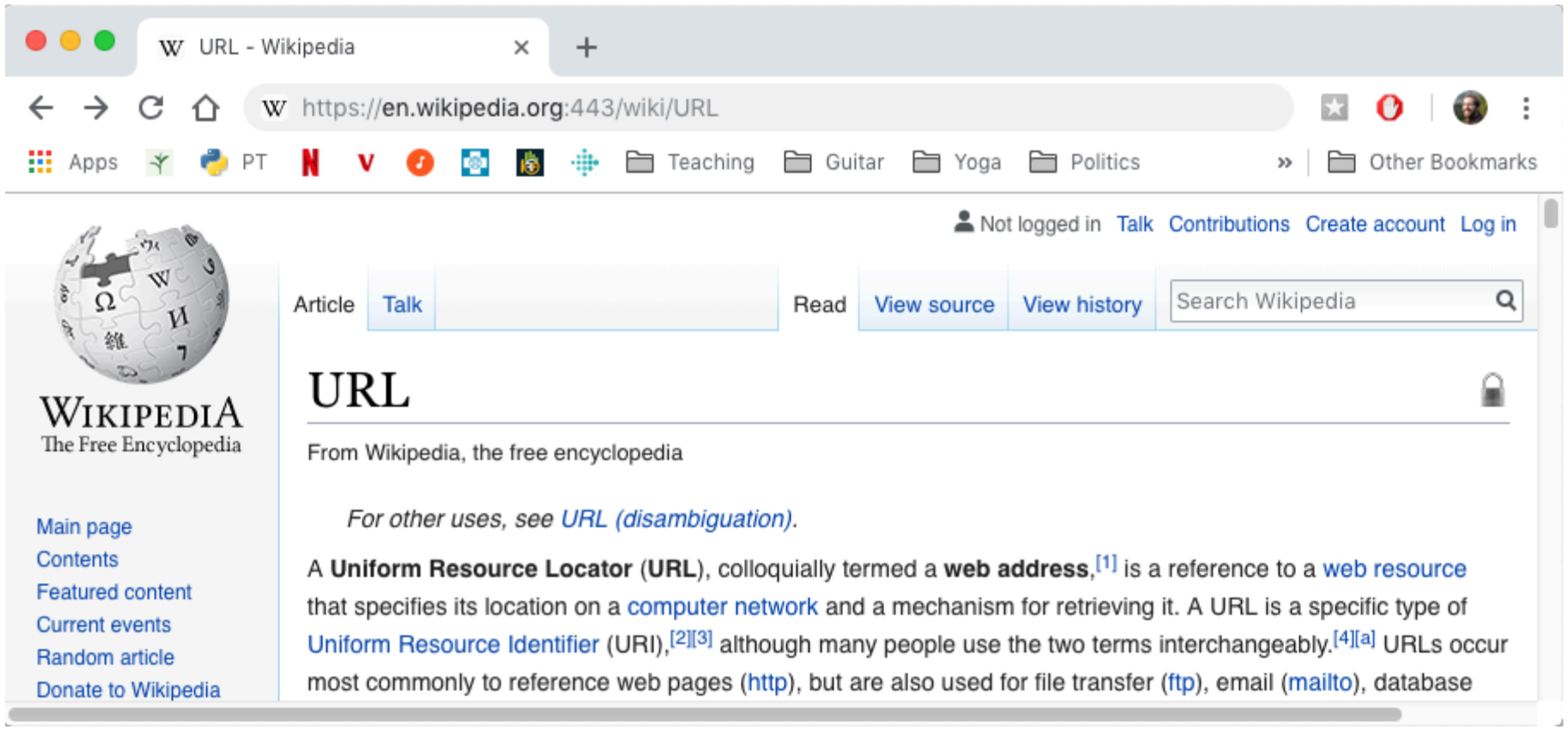
1. domain name
2. port number
3. resource (file name)

URLs

domain name

https://en.wikipedia.org:443/wiki/URL

port



URL

Note we need three things:

1. domain name
2. port number
3. resource (file name)

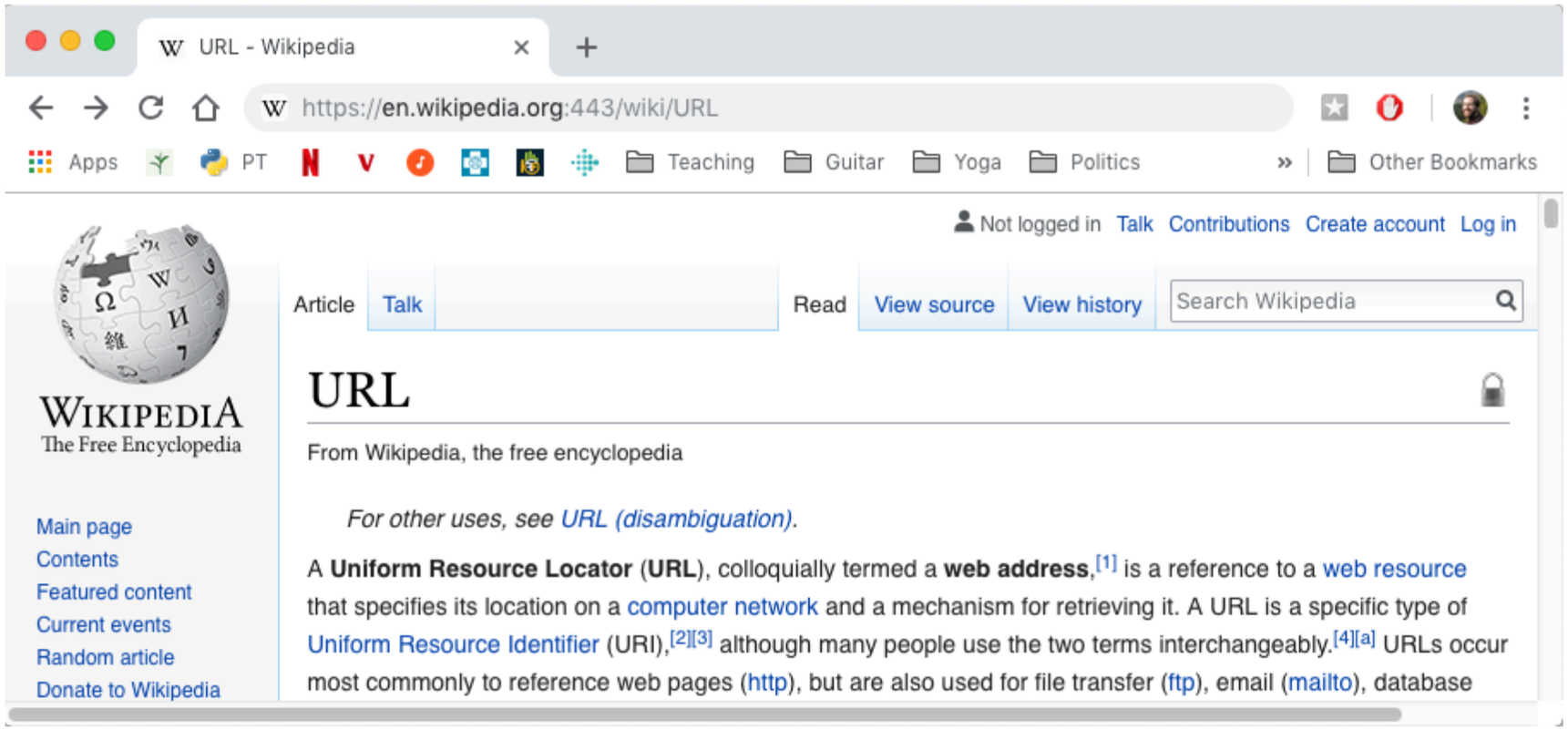
URLs

domain name

resource

`https://en.wikipedia.org:443/wiki/URL`

port



URL

Note we need three things:

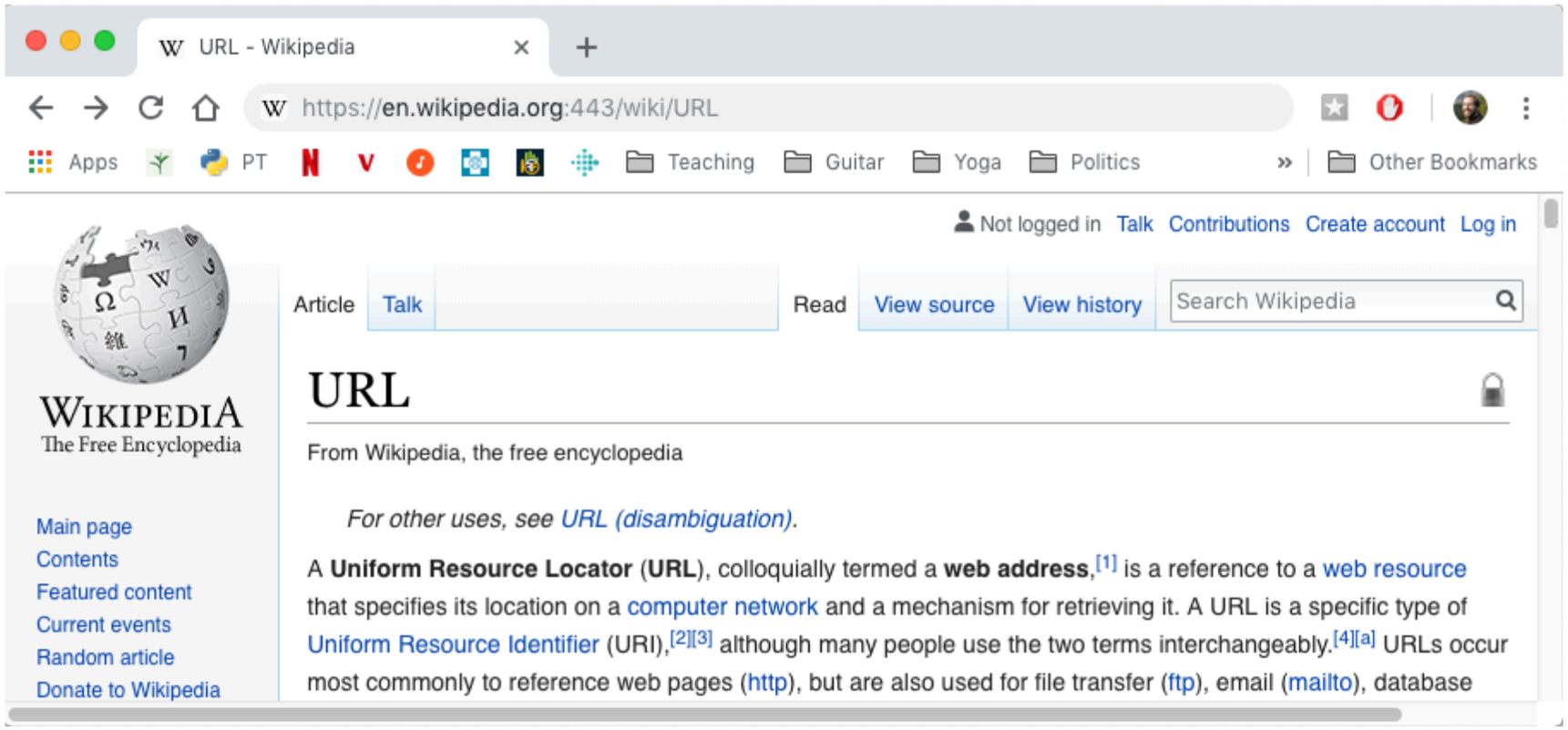
1. domain name
2. port number
3. resource (file name)

URLs

domain name resource

https://en.wikipedia.org/wiki/URL

port would have defaulted to 443 if not specified



URL

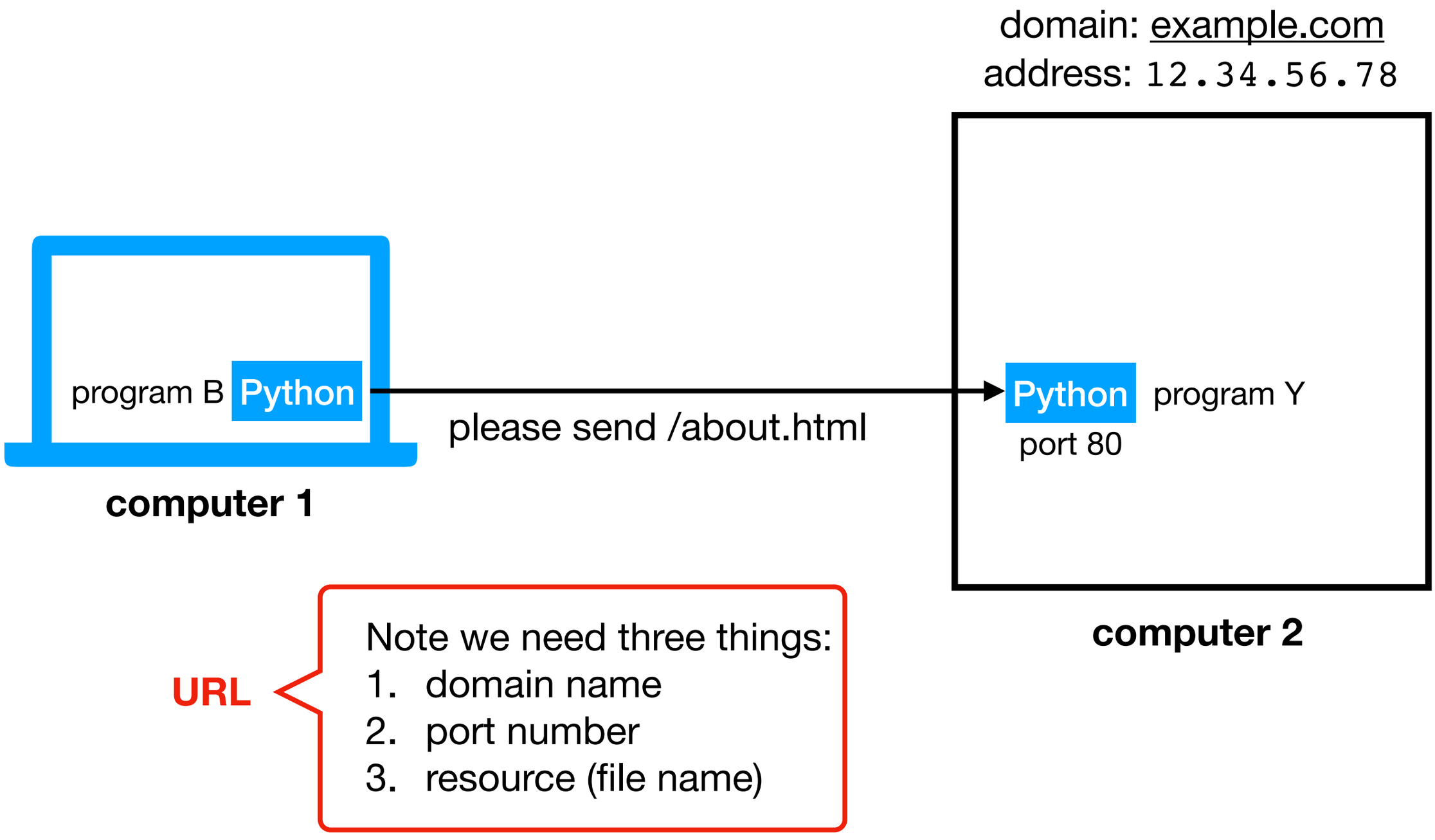
Note we need three things:

1. domain name
2. port number
3. resource (file name)

HTTP

Protocol for communicating web data

- downloading a specific webpage, image, etc



URL

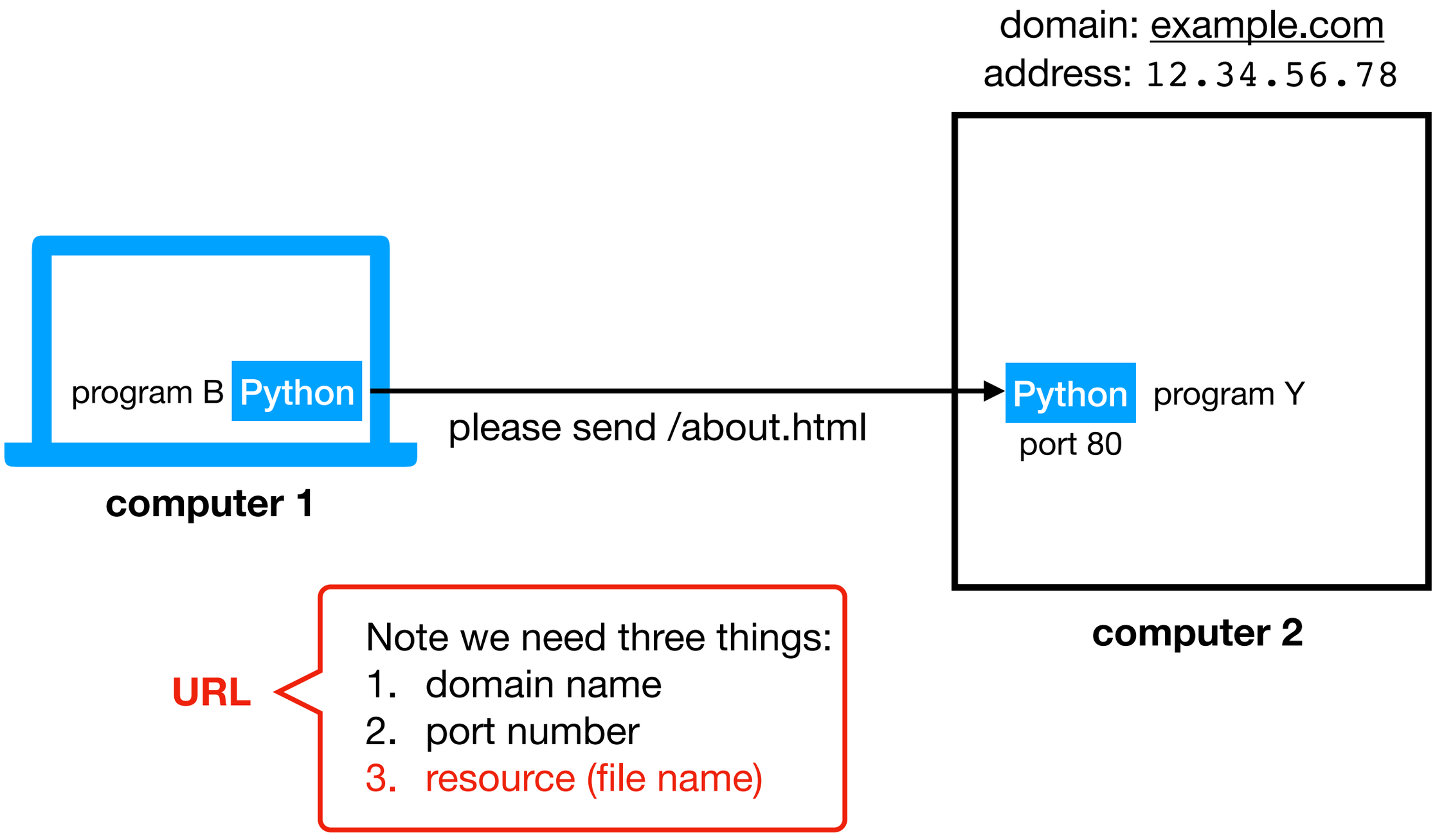
Note we need three things:

1. domain name
2. port number
3. resource (file name)

HTTP

Protocol for communicating web data

- downloading a specific webpage, image, etc



URL

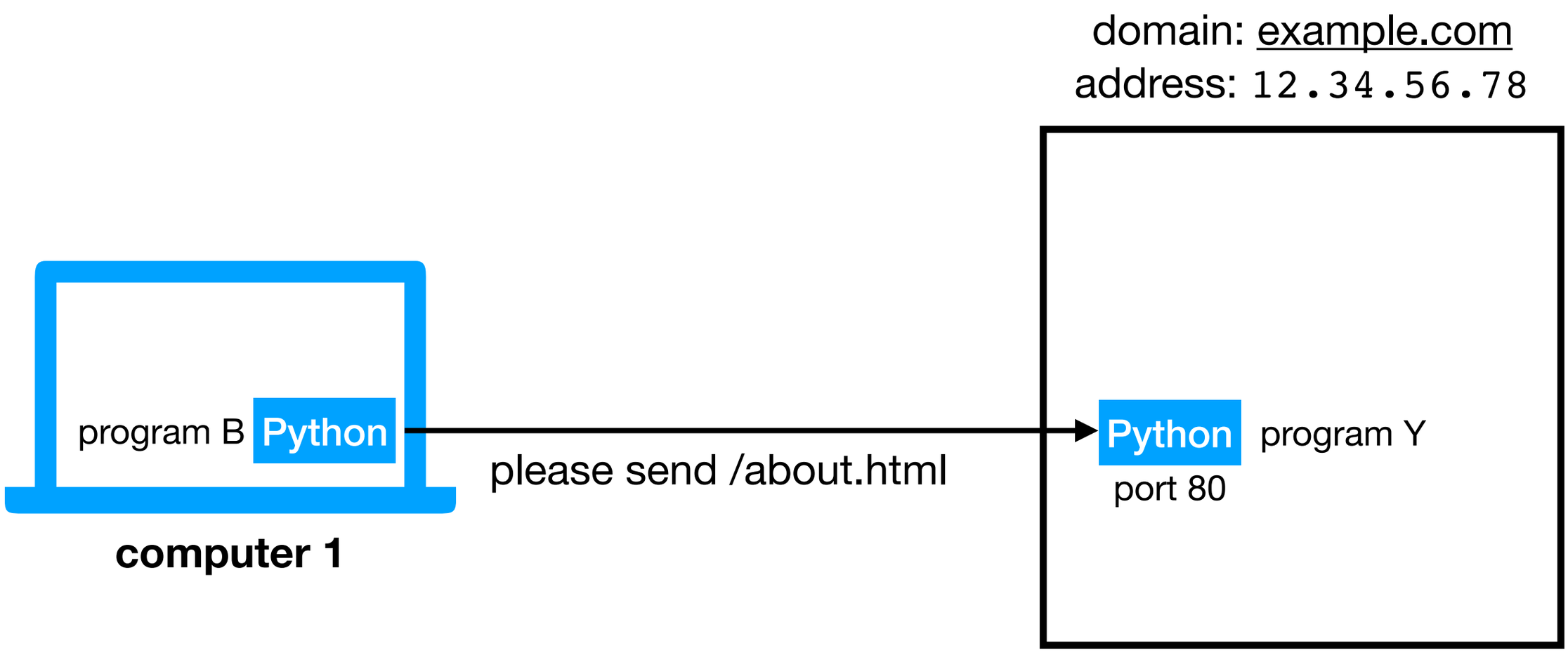
Note we need three things:

1. domain name
2. port number
3. resource (file name)

HTTP

Protocol for communicating web data

- downloading a specific webpage, image, etc



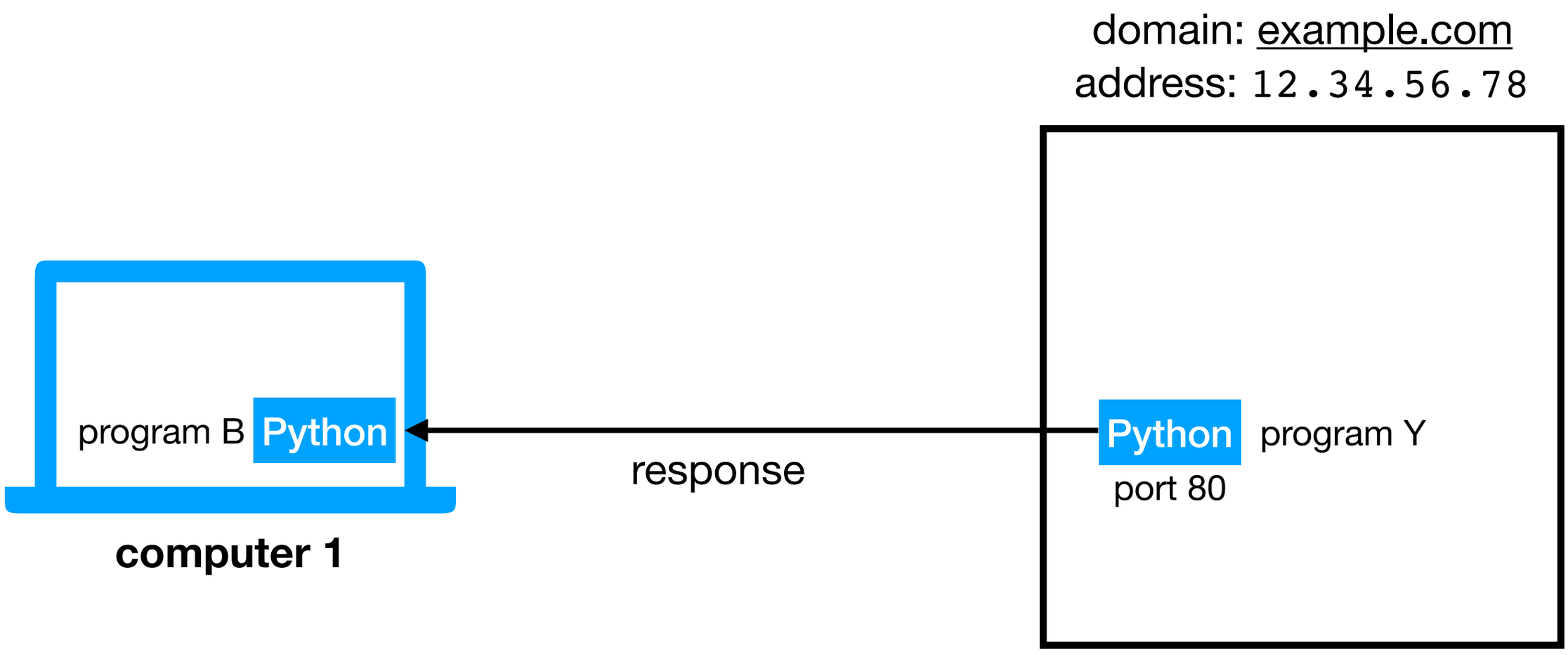
HTTP Request:

```
GET /about.html HTTP/1.1
Host: example.com
User-Agent: ...
Accept: */*
```

HTTP

Protocol for communicating web data

- downloading a specific webpage, image, etc



HTTP Response:

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74
Server: Werkzeug/0.14.1 Python/3.6.6
Date: Sun, 11 Nov 2018 17:00:29 GMT

all the contents
```

Request and Response

HTTP Request:

```
GET /about.html HTTP/1.1
Host: example.com
User-Agent: ...
Accept: */*
```

HTTP Response:

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74
Server: Werkzeug/0.14.1 Python/3.6.6
Date: Sun, 11 Nov 2018 17:00:29 GMT

all the contents
```

There are **LOTS** of details here we don't care about right now

Request and Response

we want the about.html page

HTTP Request:

```
GET /about.html HTTP/1.1
Host: example.com
User-Agent: ...
Accept: */*
```

HTTP Response:

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74
Server: Werkzeug/0.14.1 Python/3.6.6
Date: Sun, 11 Nov 2018 17:00:29 GMT

all the contents
```

data in about.html

There are **LOTS** of details here we don't care about right now

Request and Response

we want the about.html page

HTTP Request:

```
GET /about.html HTTP/1.1
Host: example.com
User-Agent: ...
Accept: */*
```

status code. 200 is good. 404, 500, others are various errors or other more complicated states

HTTP Response:

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74
Server: Werkzeug/0.14.1 Python/3.6.6
Date: Sun, 11 Nov 2018 17:00:29 GMT

all the contents
```

data in about.html

There are **LOTS** of details here we don't care about right now

method. *GET* is simple download. *POST* means we are uploading data as part of our request. We won't talk about others today.

we want the about.html page

HTTP Request:

```
GET /about.html HTTP/1.1
Host: example.com
User-Agent: ...
Accept: */*
```

status code. 200 is good. 404, 500, others are various errors or other more complicated states

HTTP Response:

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74
Server: Werkzeug/0.14.1 Python/3.6.6
Date: Sun, 11 Nov 2018 17:00:29 GMT

all the contents
```

data in about.html

There are **LOTS** of details here we don't care about right now

Learning Objectives Today

Motivation

Networking Basics

HTTP (Hypertext Transfer Protocol)

Requests Module

Requests module

Purpose

- easily send requests to a server and parse the response
- *“HTTP for Humans™”*

Installation

- comes with Anaconda
- otherwise run this:

```
pip install requests
```

Using it

- just import:

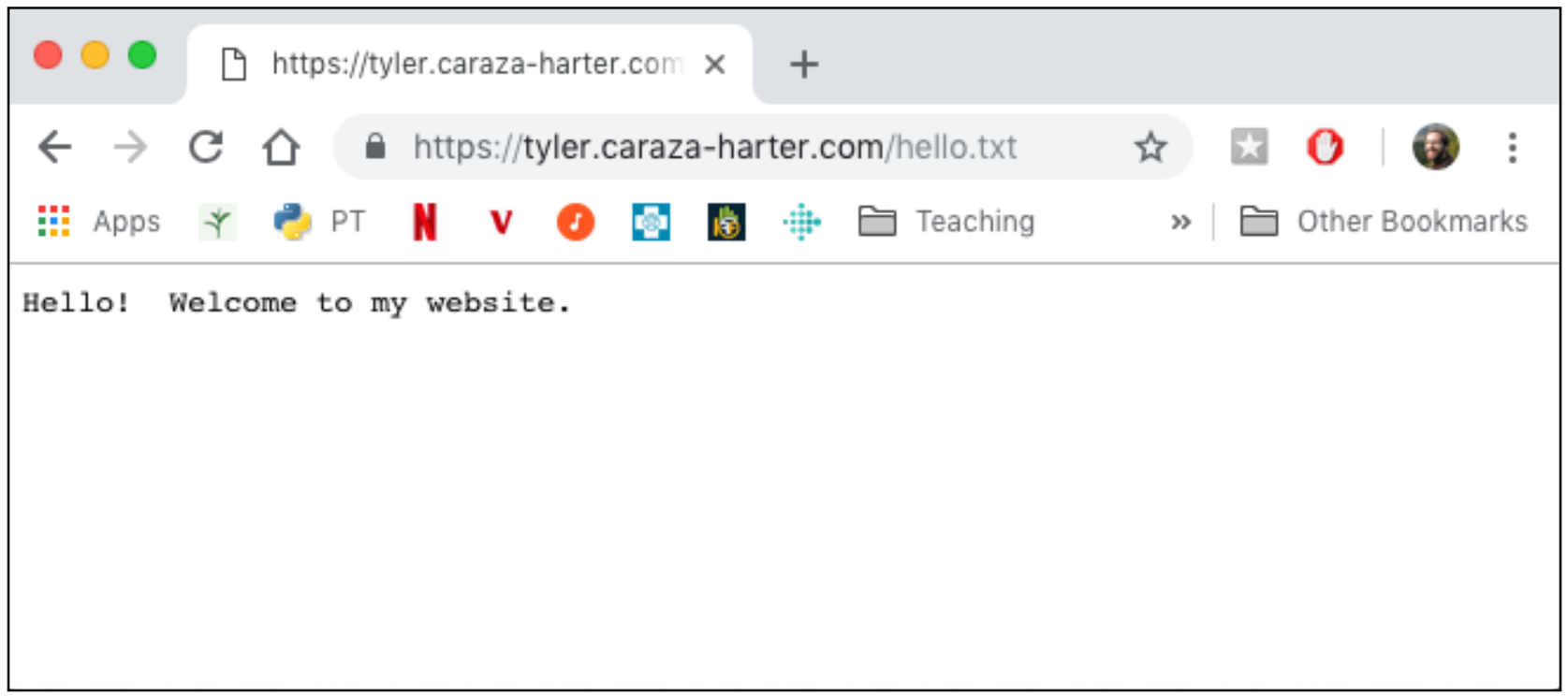
```
import requests
```

GET Request

```
import requests

url = “https://tyler.caraza-harter.com/hello.txt"

requests.get(url)
```



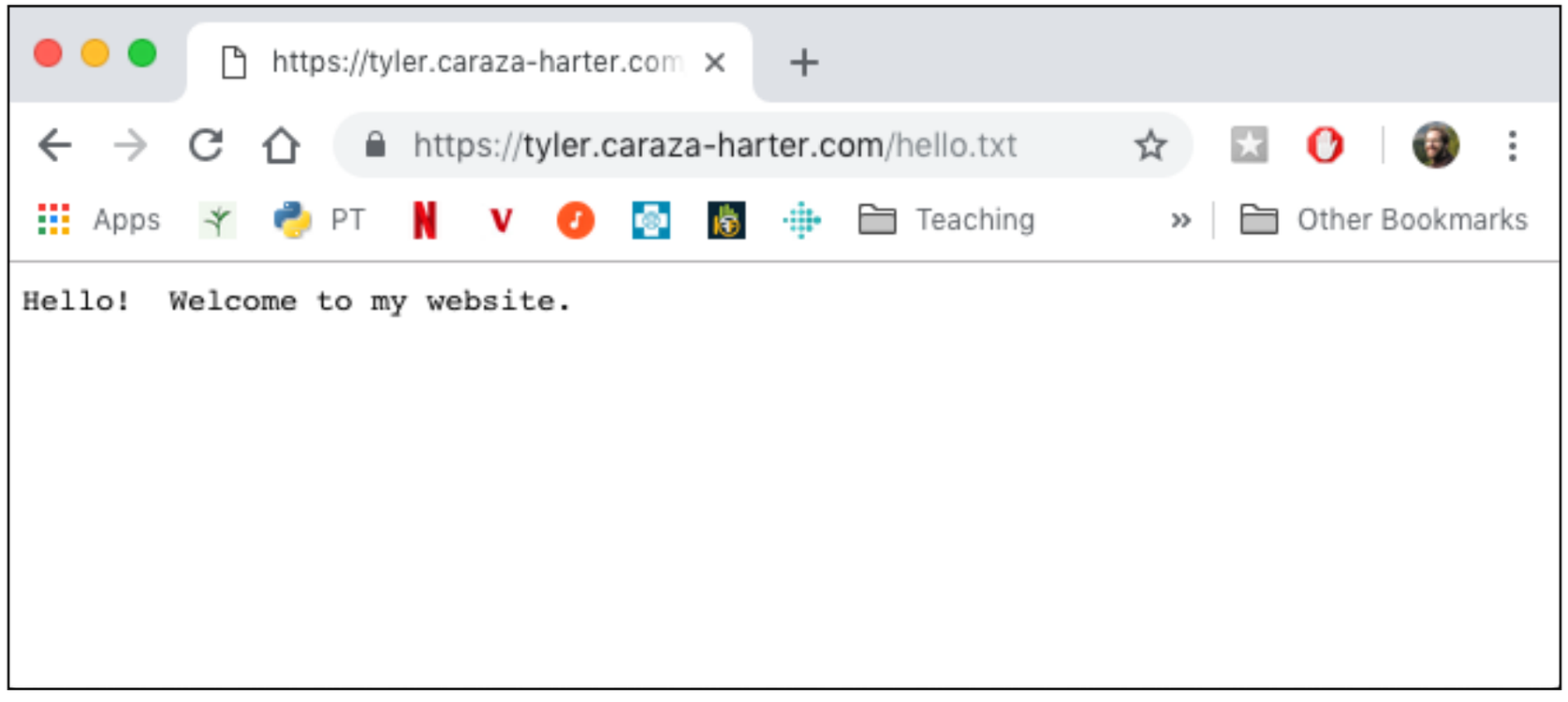
GET Request

```
import requests

url = “https://tyler.caraza-harter.com/hello.txt"

requests.get(url)
```

sends a **GET** request to tyler.caraza-harter.com, asking for the contents of the **/hello.txt** page



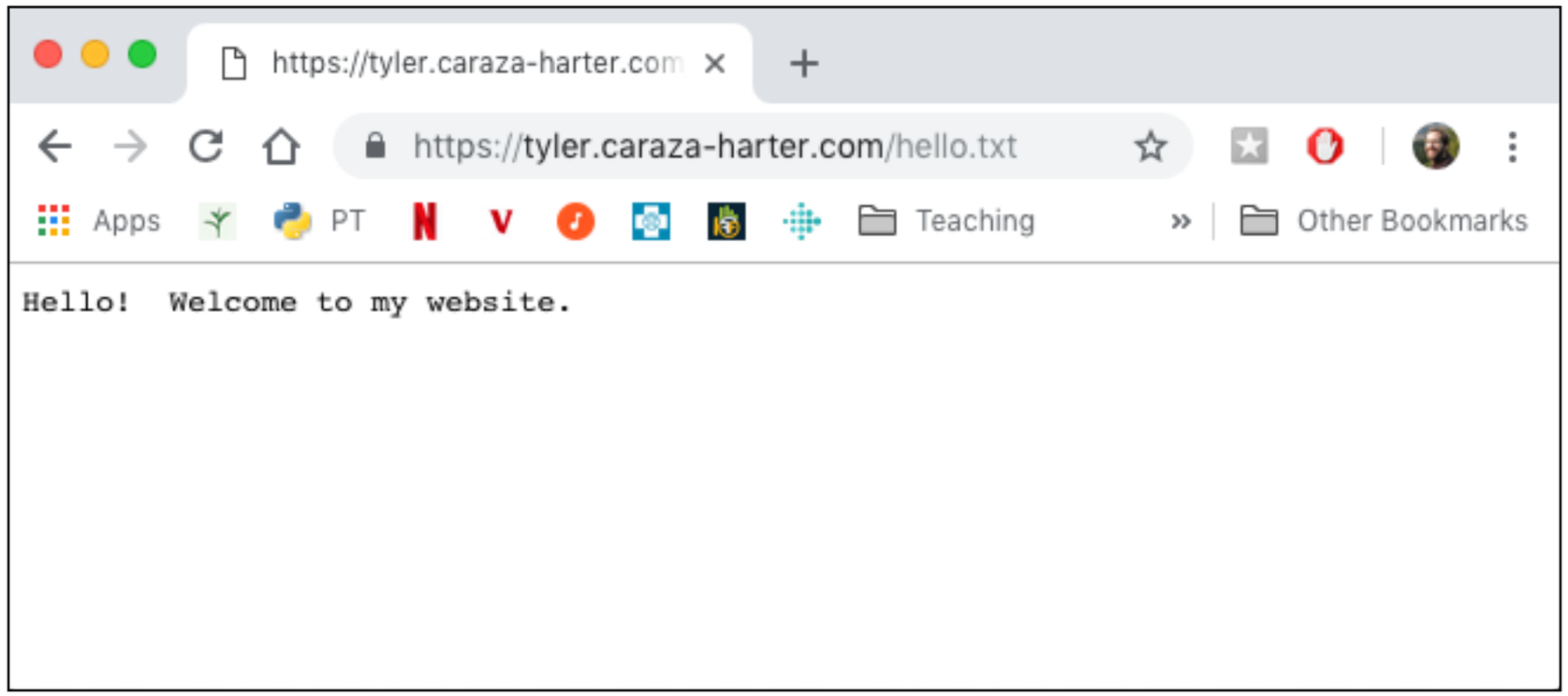
GET Request

```
import requests

url = “https://tyler.caraza-harter.com/hello.txt"

resp = requests.get(url)
```

put response from tyler.caraza-harter.com in the resp variable



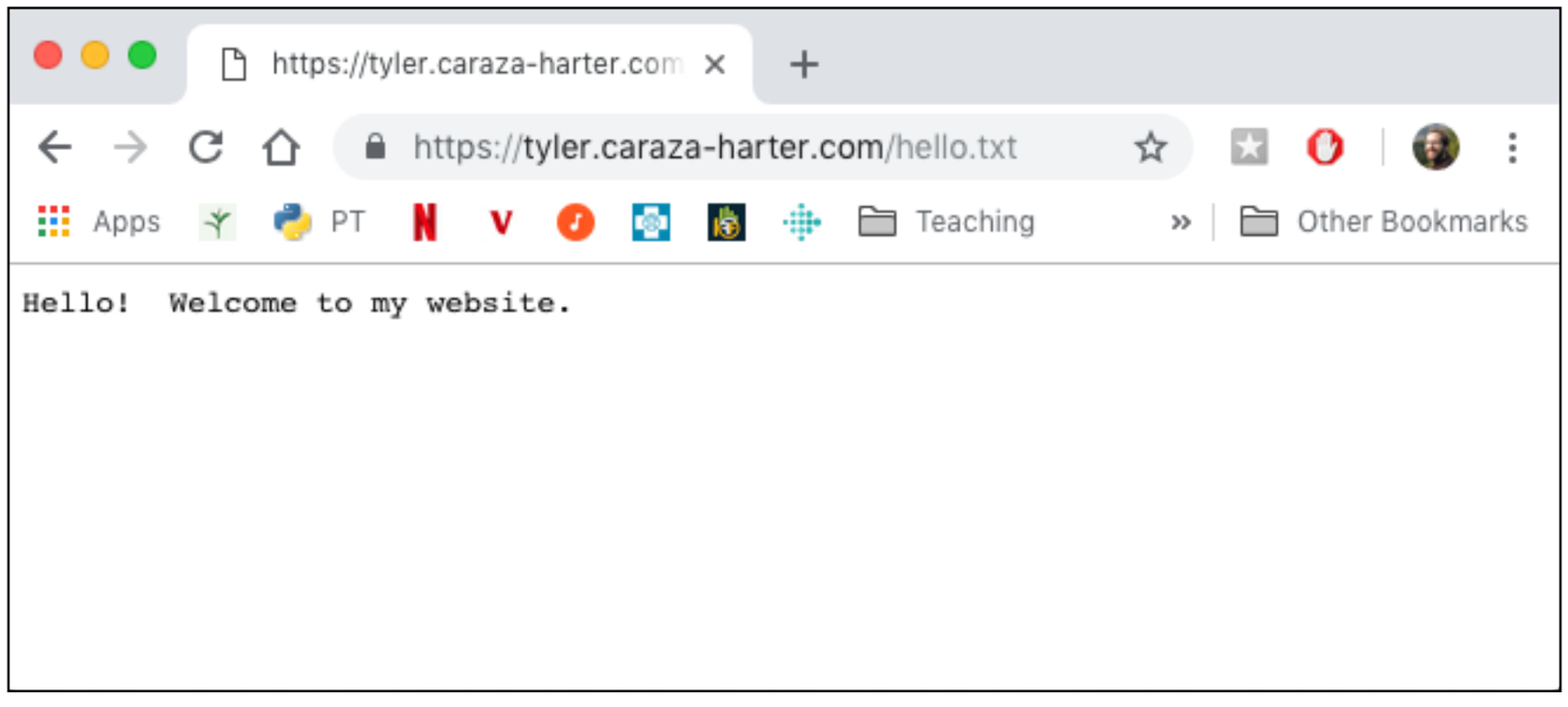
GET Request

```
import requests

url = "https://tyler.caraza-harter.com/hello.txt"

resp = requests.get(url)

# make sure we got 200 (success) back
assert(resp.status_code == 200)
```



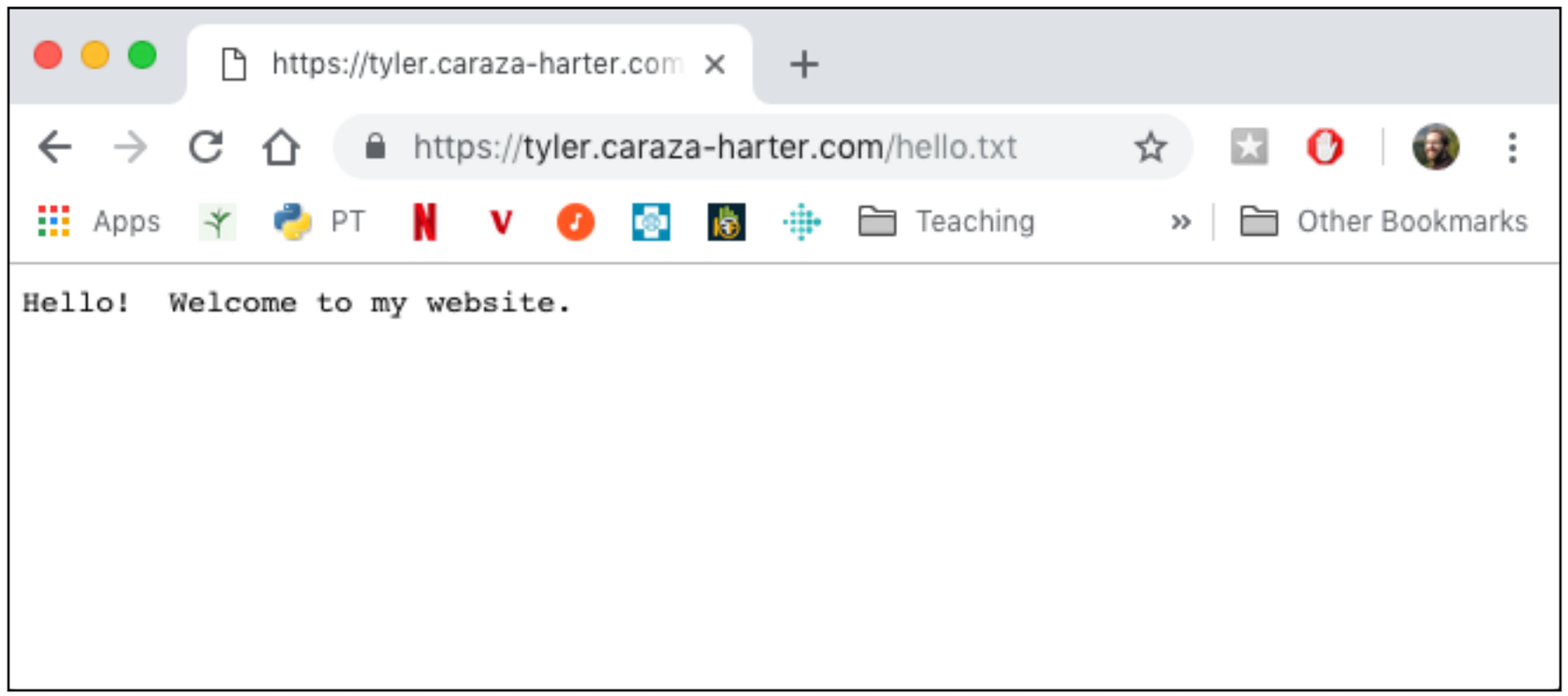
GET Request

```
import requests

url = "https://tyler.caraza-harter.com/hello.txt"

resp = requests.get(url)

resp.raise_for_status() # shortcut
```



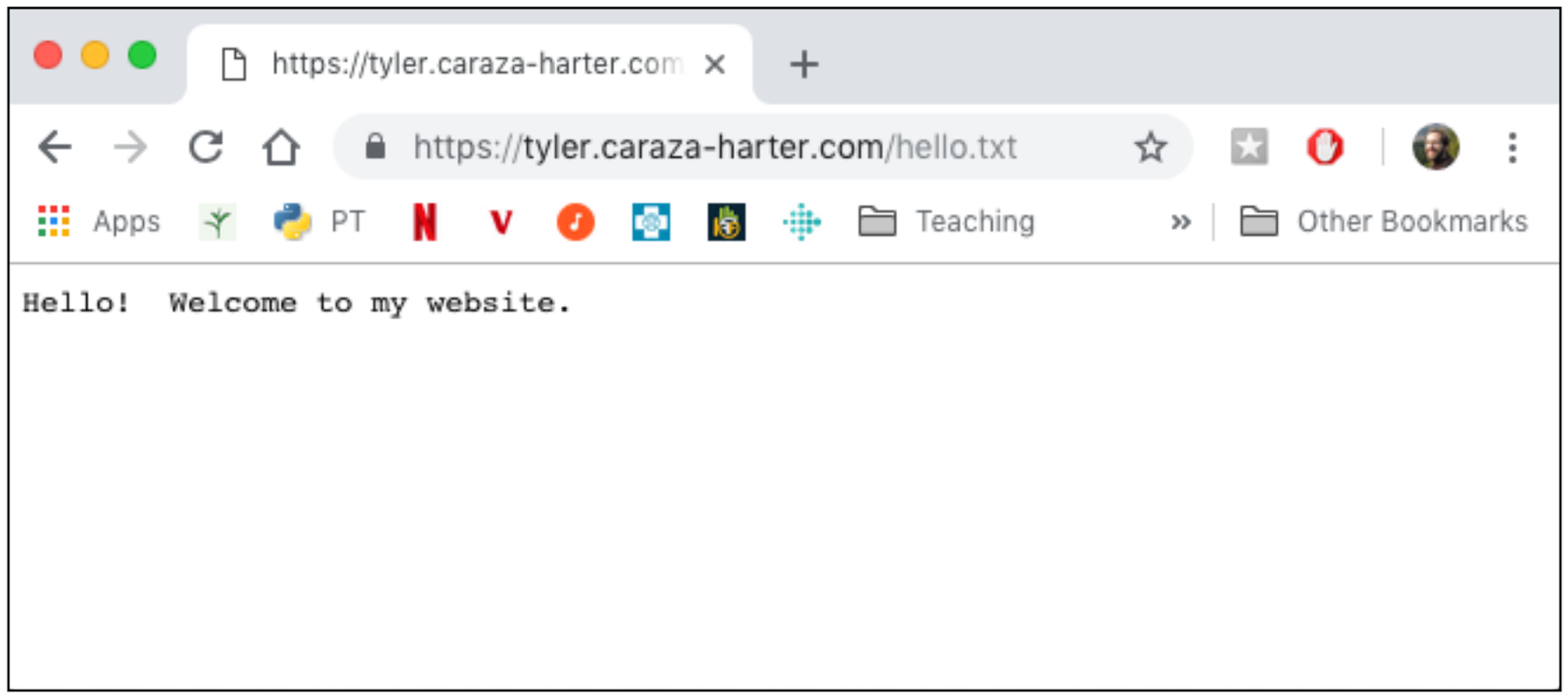
GET Request

```
import requests

url = "https://tyler.caraza-harter.com/hello.txt"

resp = requests.get(url)

resp.raise_for_status() # shortcut
print(resp.text) # “Hello! Welcome to my website.”
```

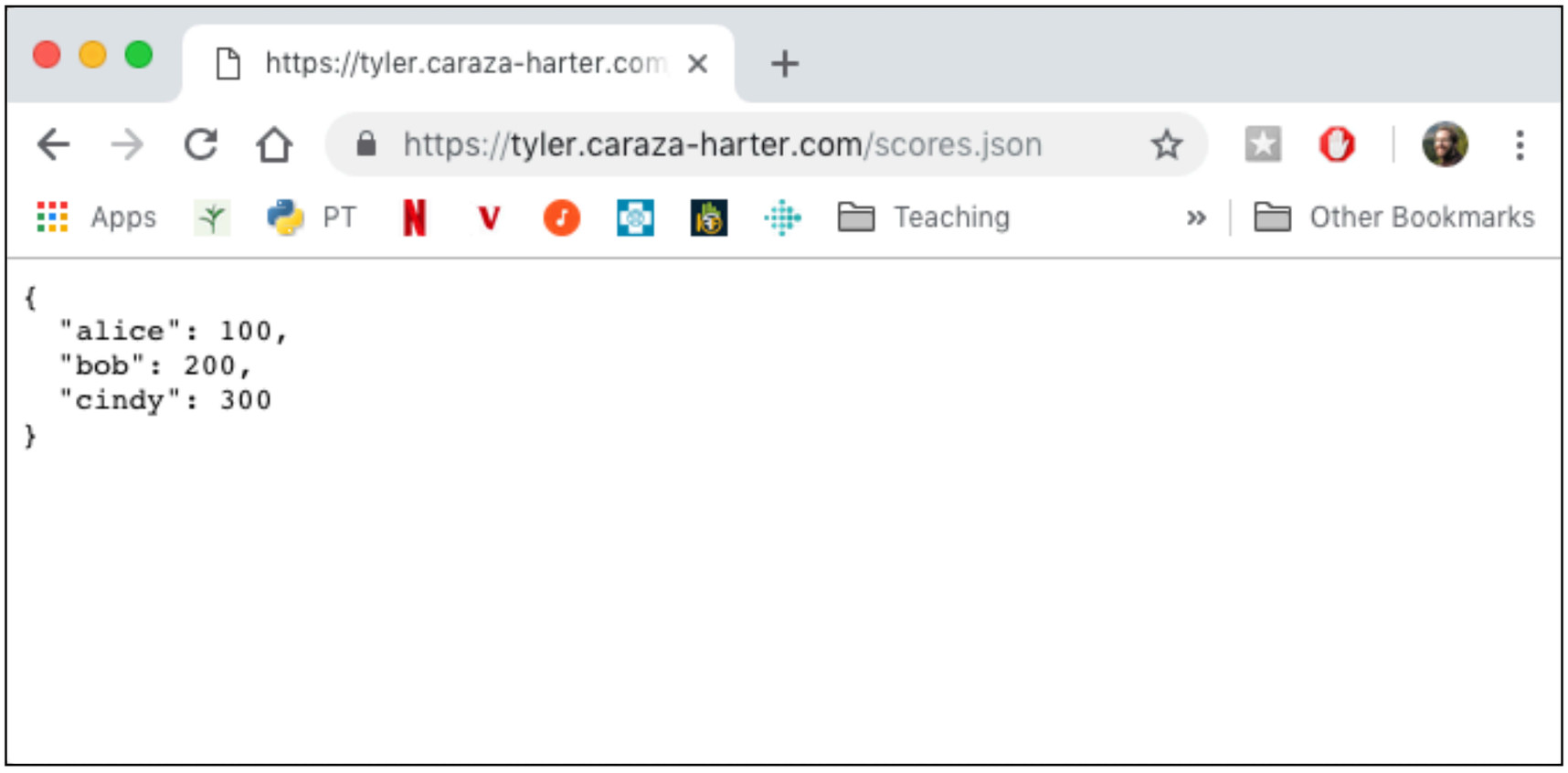


JSON Responses

```
import requests, json

url = “https://tyler.caraza-harter.com/scores.json”
resp = requests.get(url)

scores = json.loads(resp.text)
```

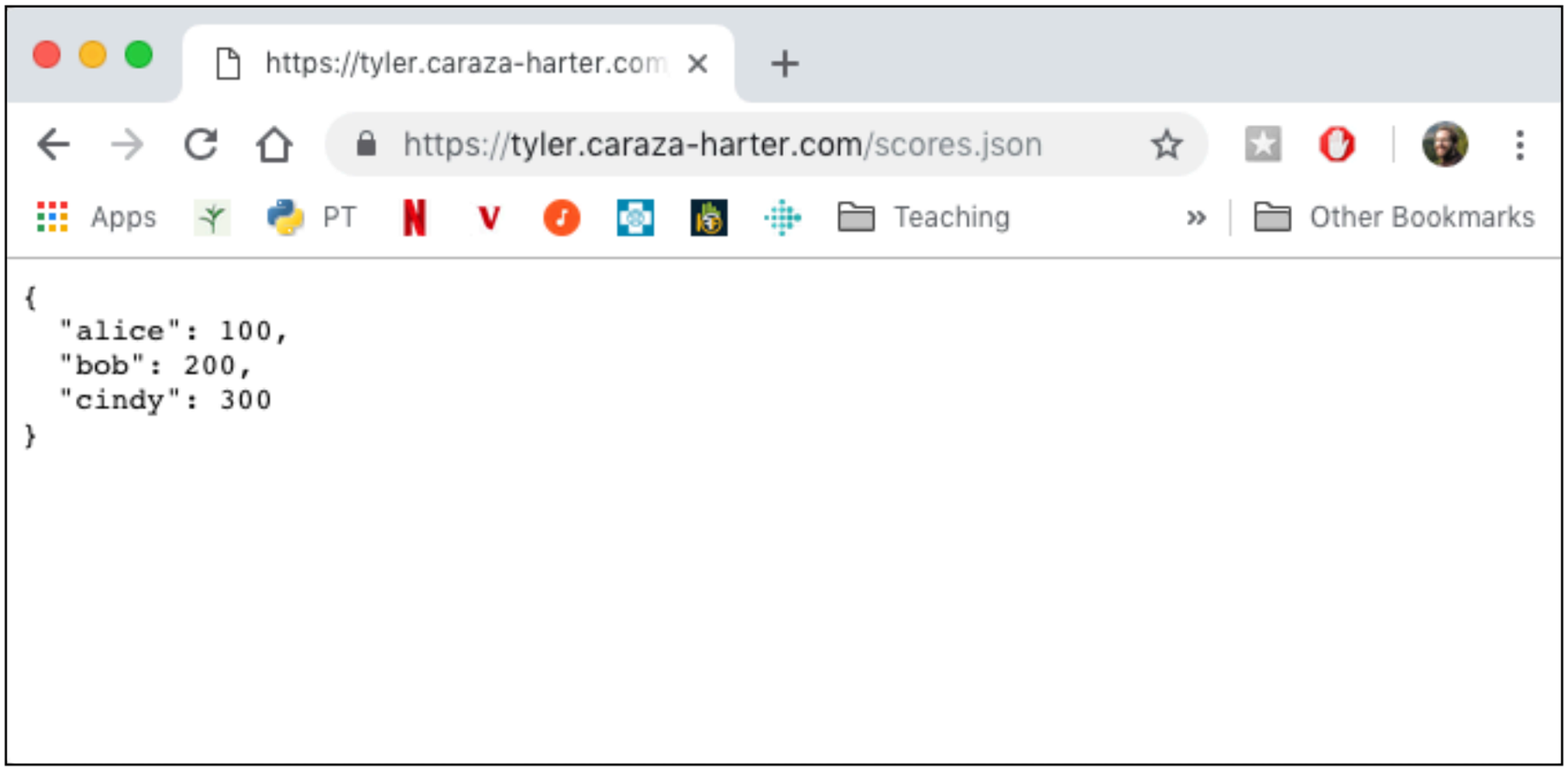


JSON Responses

```
import requests, json

url = “https://tyler.caraza-harter.com/scores.json"
resp = requests.get(url)

scores = json.loads(resp.text)
scores = resp.json() # shortcut
```



Demo 1: State Populations

Goal: fetch population data for all states and provide summary stats

Input:

- List of state files: https://tyler.caraza-harter.com/cs301/fall18/materials/code/lec-28/state_files.txt
- The 50 JSON files

Output:

- Stats about population: mean, max, min, etc

```
In [19]: df.describe().astype(int)
```

Out[19]:

	2000	2010	2015
count	50	50	50
mean	5616996	6162876	6364951
std	6185579	6848235	7152085
min	493782	563626	584304
25%	1735533	1833004	1857308
50%	4026890	4436369	4530803
75%	6281944	6680312	6986155
max	33871648	37253956	38792291

POST Request

```
import requests

url = “…”

requests.post(url, json={“alice”: 100})
```

POST Request

```
import requests

url = “…”

requests.post(url, json={“alice”: 100})
```

post function automatically converts these
Python structures to JSON and sends it in the request

Demo 2: Simple Messaging

Goal: provide application to add messages to a thread and view the whole thread

Input:

- message group ID
- message to add

Output:

- All the messages in a group

Examples:

```
import requests
url = "http://18.216.110.65"
requests.post(url + '/send_message', json={"group": "1", "message": "test"}).text
requests.post(url + '/read_messages', json={"group": "1"}).json()
```