

# [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

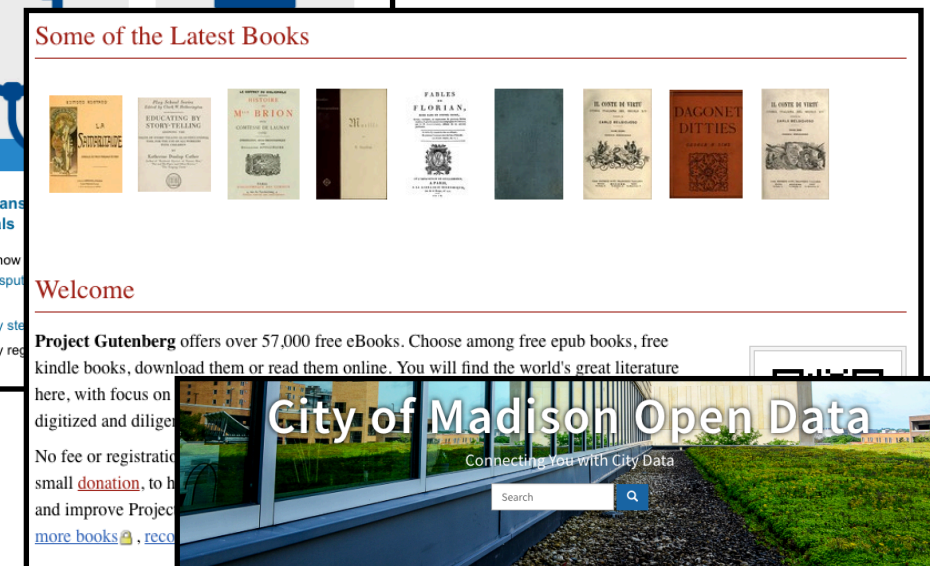
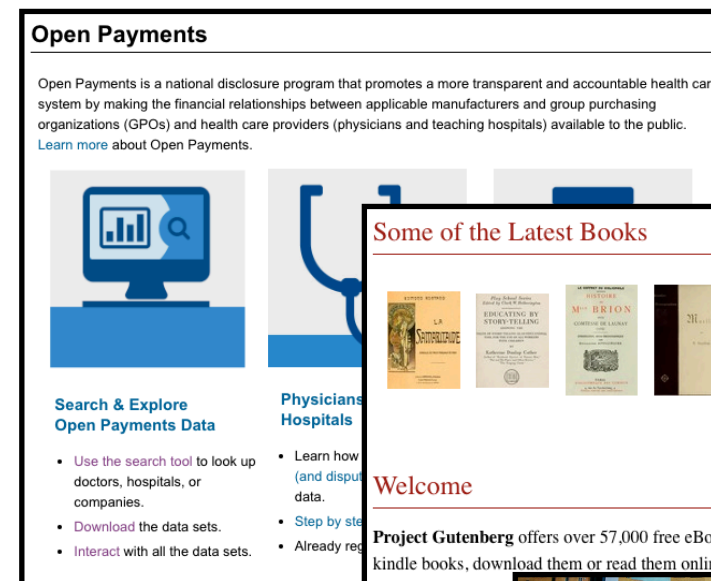
# Data Science and the Internet

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



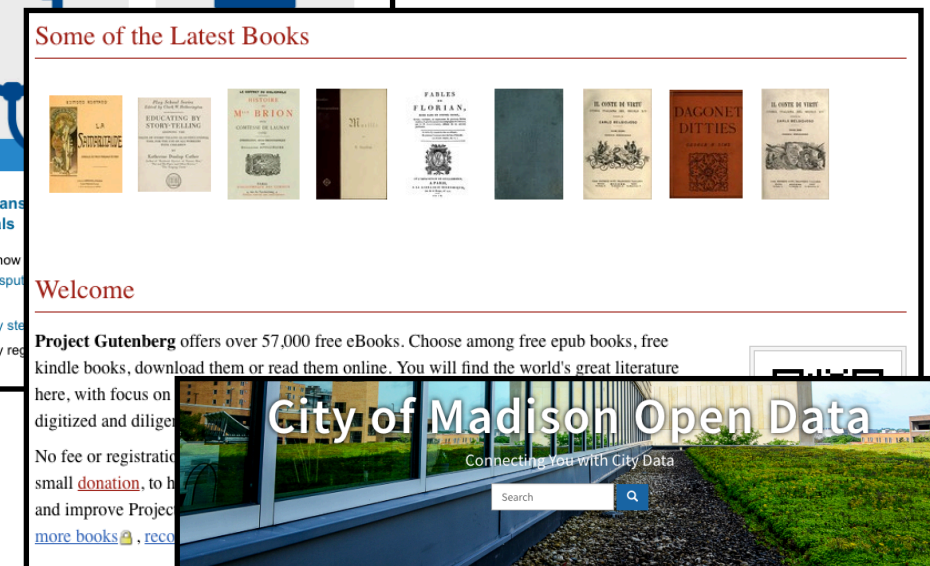
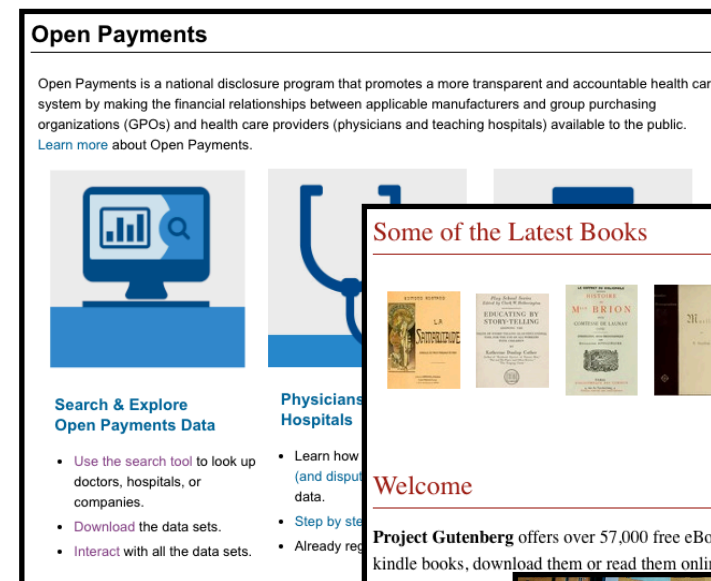
# Data Science and the Internet

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

## Some of the Latest Books



## Welcome

**Project Gutenberg** offers over 57,000 free eBooks. Choose among free epub books, free kindle books, download them or read them online. You will find the world's great literature here, with focus on older works for which copyright has expired. Thousands of volunteers digitized and diligently proofread the eBooks, for enjoyment and education.

No fee or registration is required. If you find Project Gutenberg useful, please consider a small [donation](#), to help Project Gutenberg digitize more books, maintain our online presence, and improve Project Gutenberg programs and offerings. Other ways to help include [digitizing more books](#) 📄, [recording audio books](#) 🎧, or [reporting errors](#).



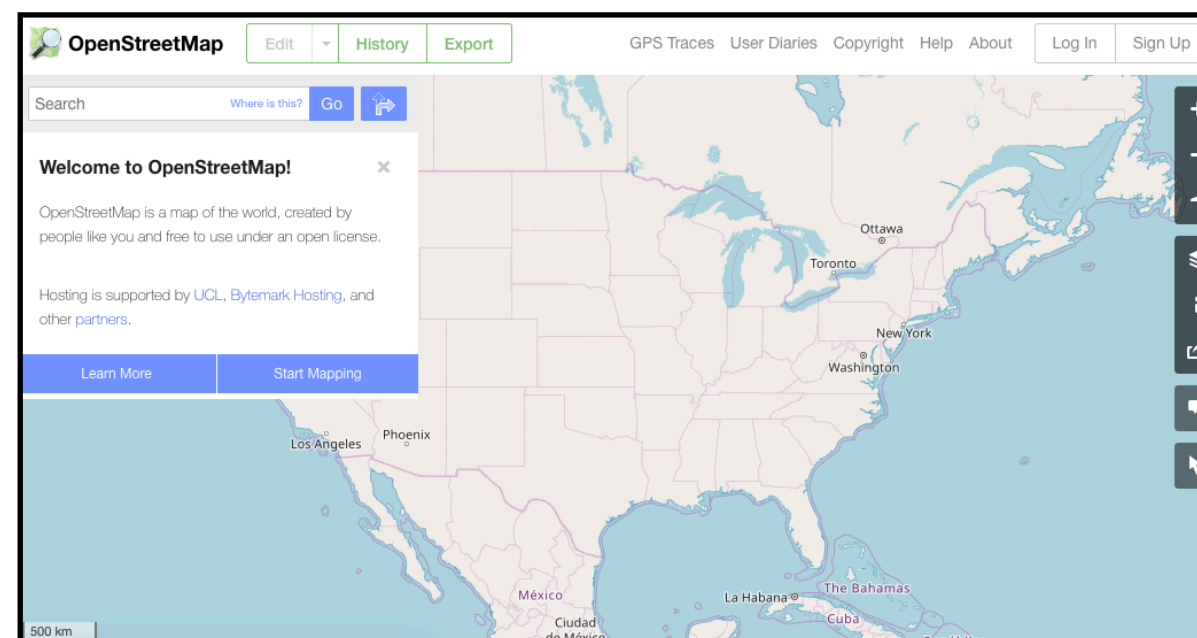
[Project Gutenberg](#)  
[Mobile Site](#)

# 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

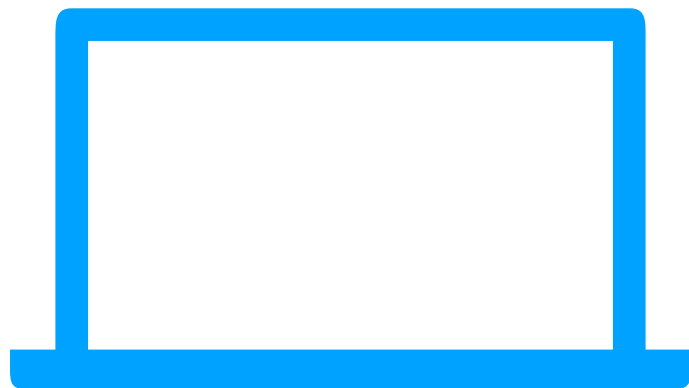
Networking Basics

HTTP (Hypertext Transfer Protocol)

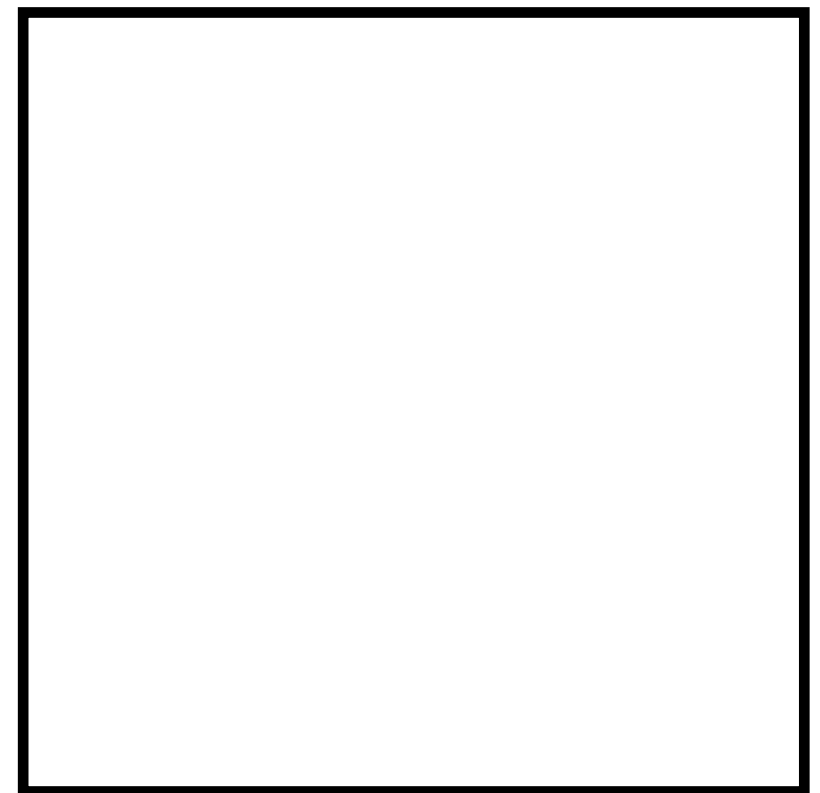
Requests Module



# Networking Basics



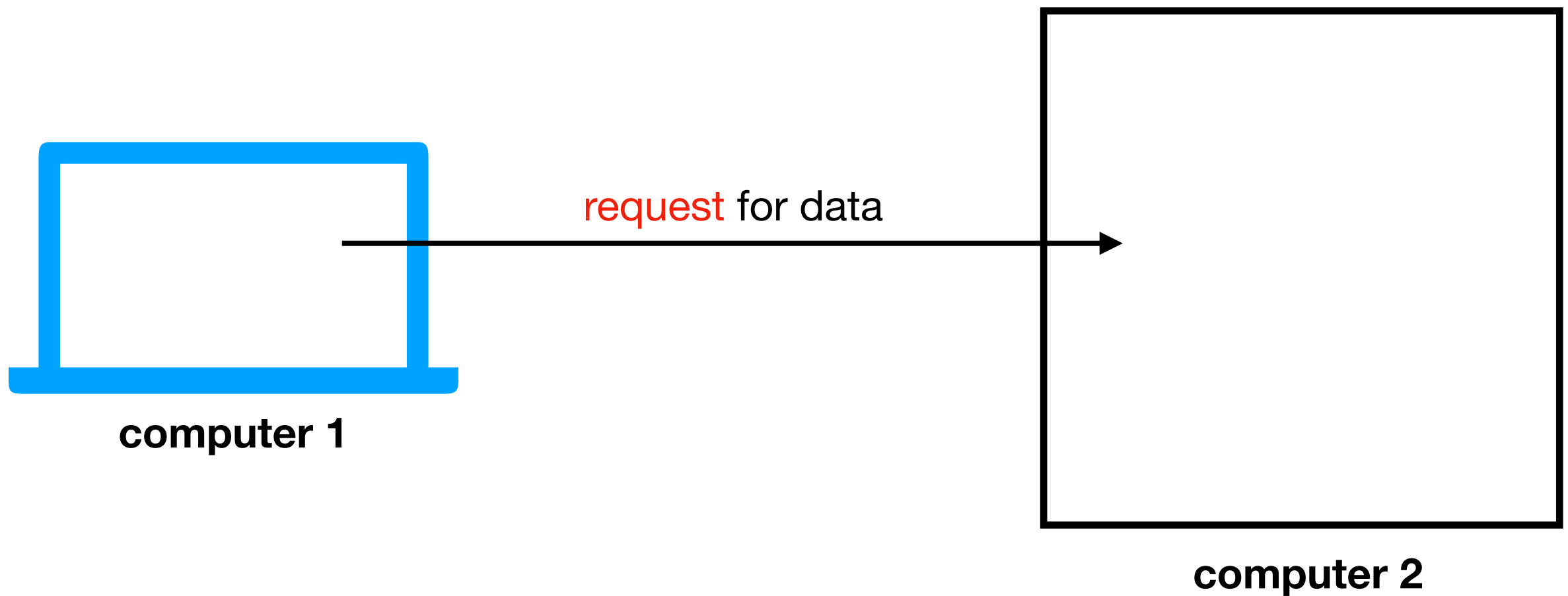
**computer 1**



**computer 2**

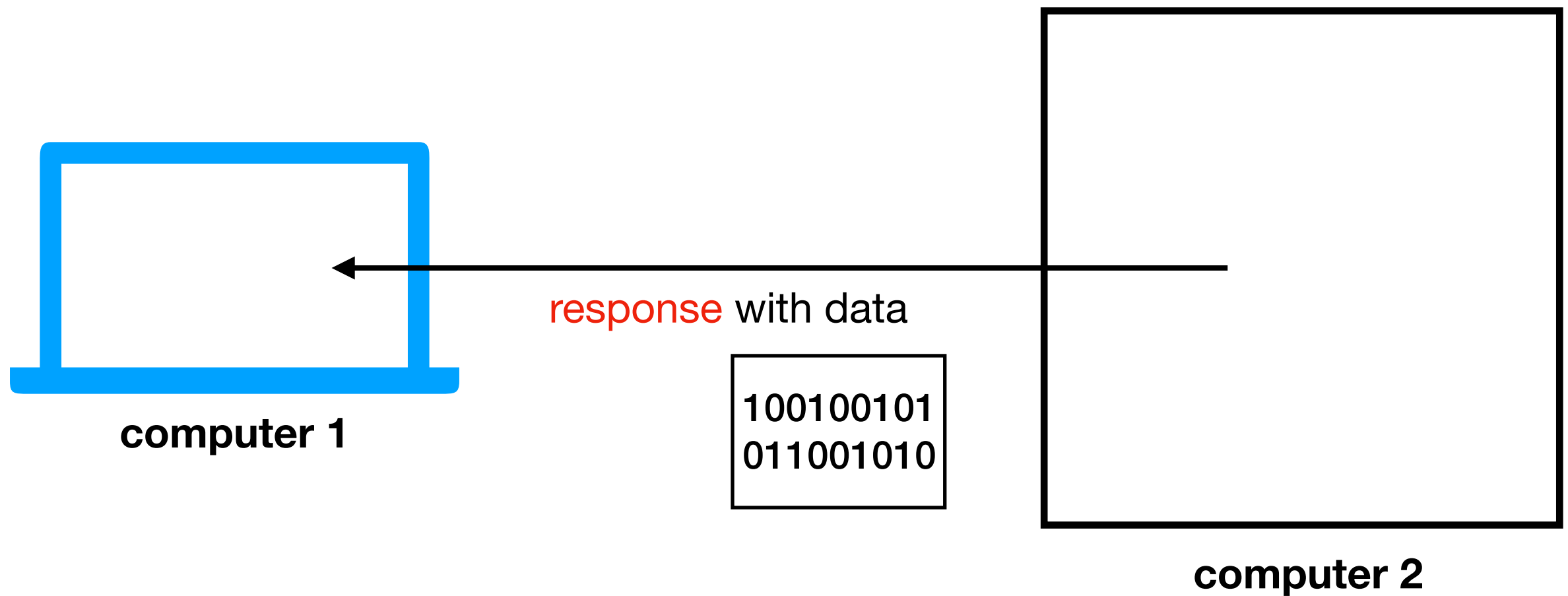
Computers communicate over a network (e.g., the Internet)  
by sending messages to each other

# Networking Basics



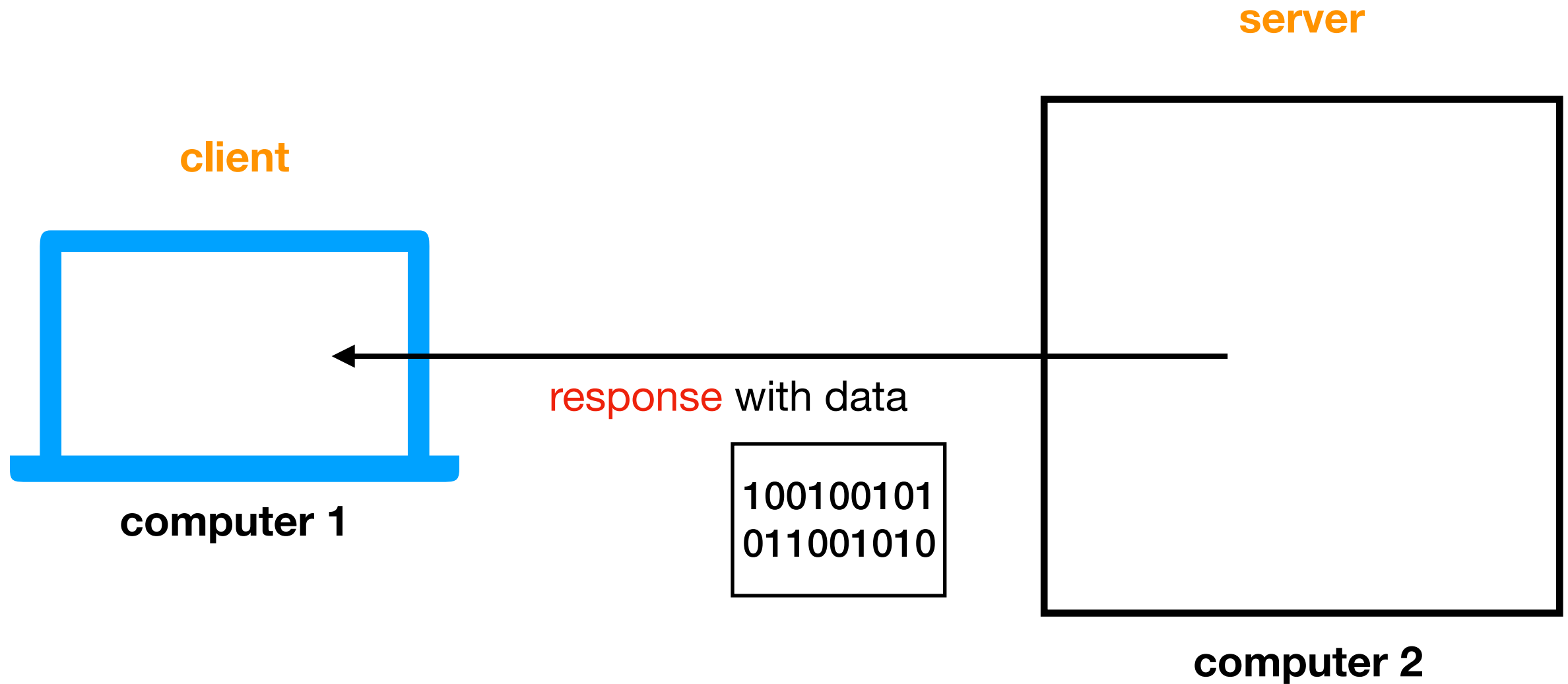
Computers communicate over a network (e.g., the Internet)  
by sending messages to each other

# Networking Basics



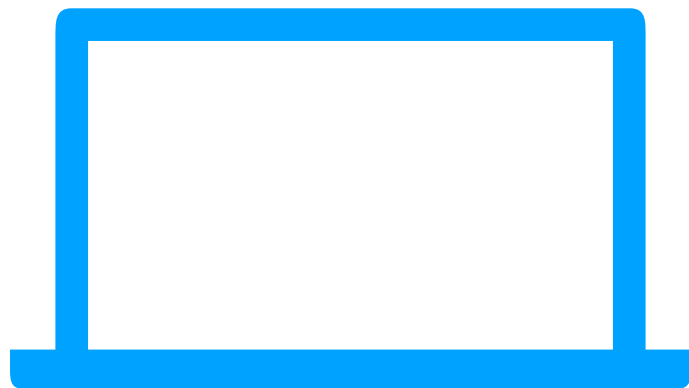
Computers communicate over a network (e.g., the Internet)  
by sending messages to each other

# Networking Basics

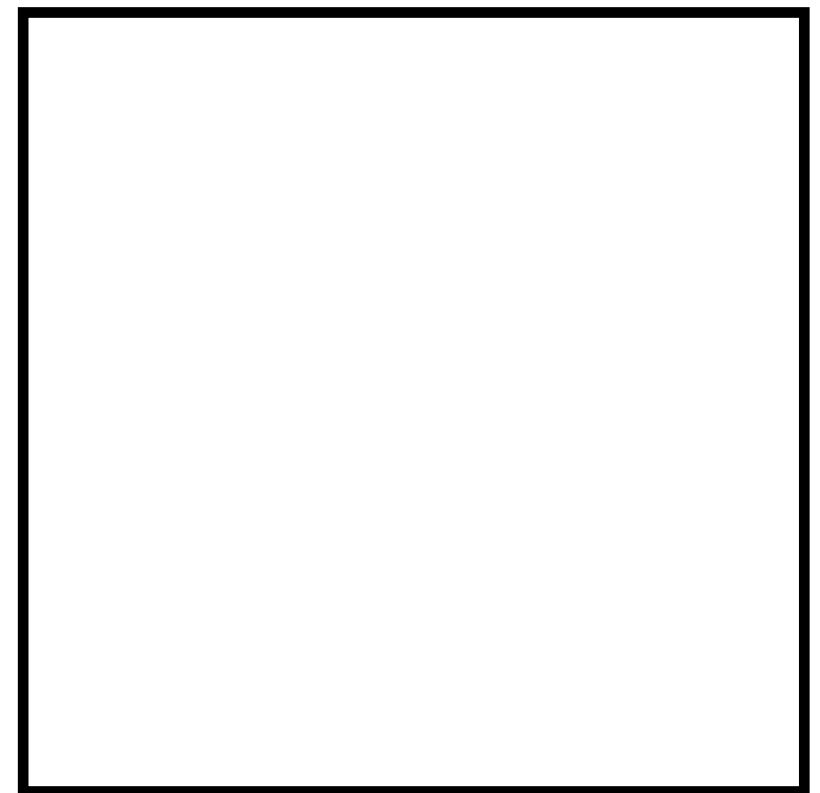


Computers communicate over a network (e.g., the Internet)  
by sending messages to each other

# Networking Basics



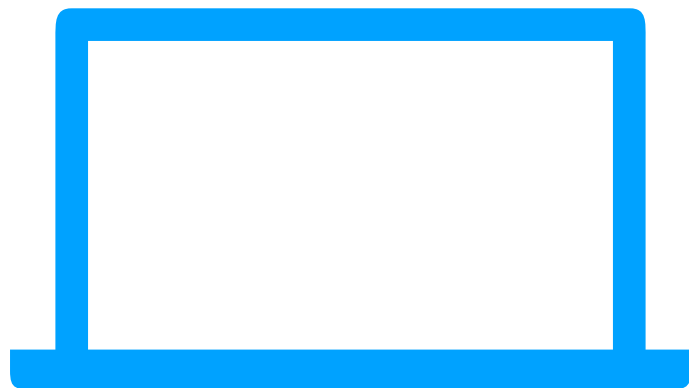
computer 1



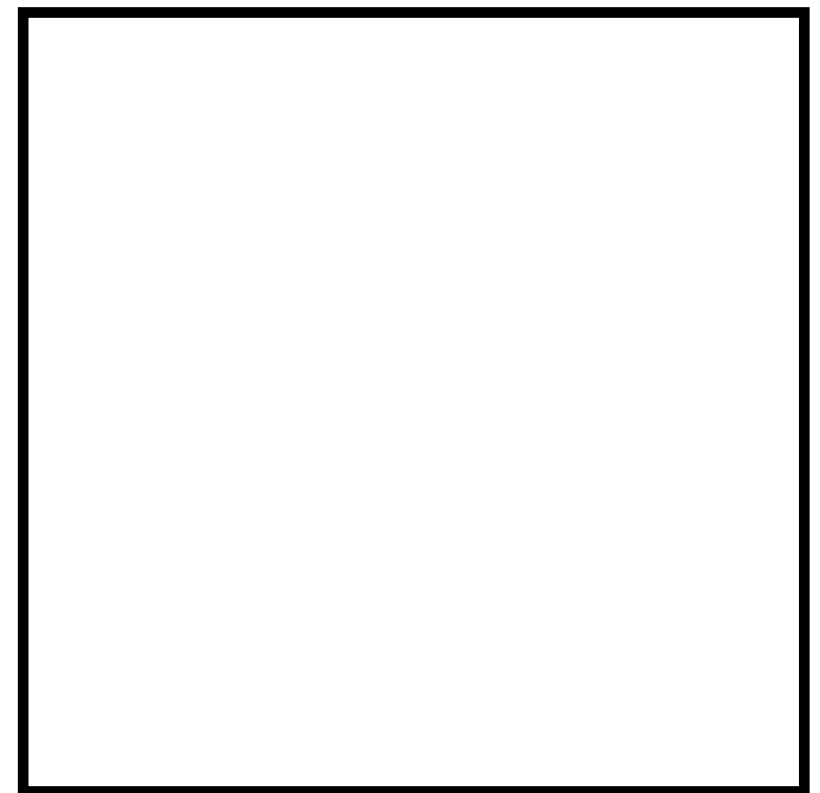
computer 2

**Challenge:** there are millions of computers.  
How do we indicate which machine should get our request?

# Internet Protocol



**computer 1**

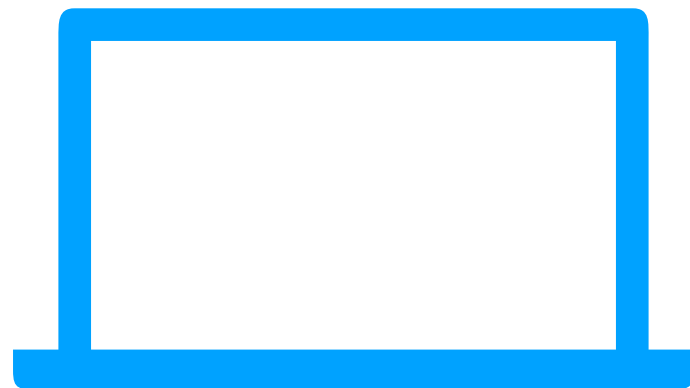


**computer 2**

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

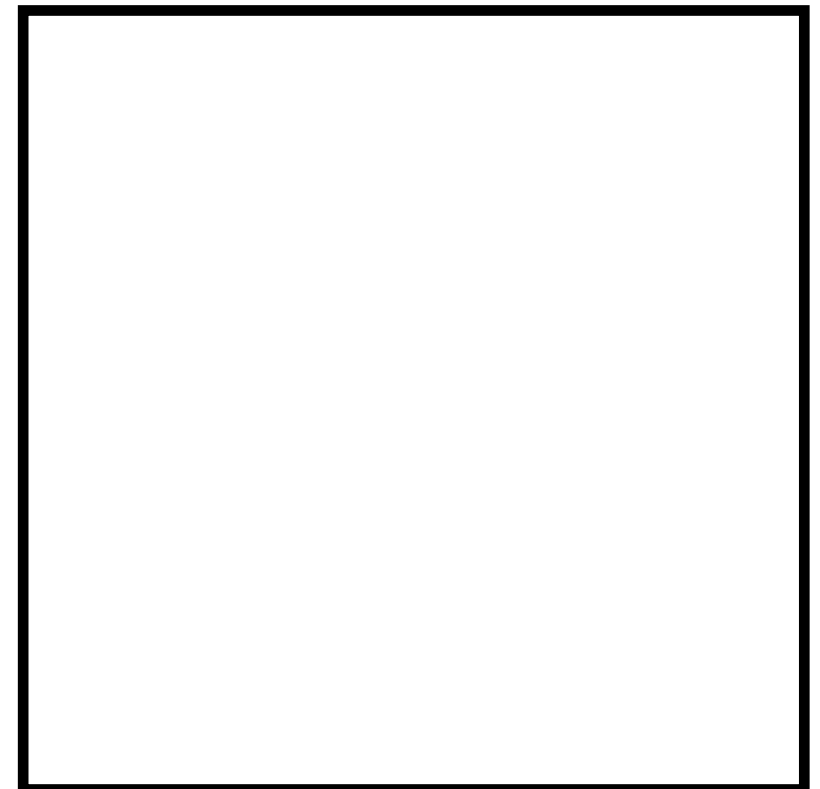
\*some machines have more multiple addresses

# Internet Protocol



computer 1

address: 18.216.110.65

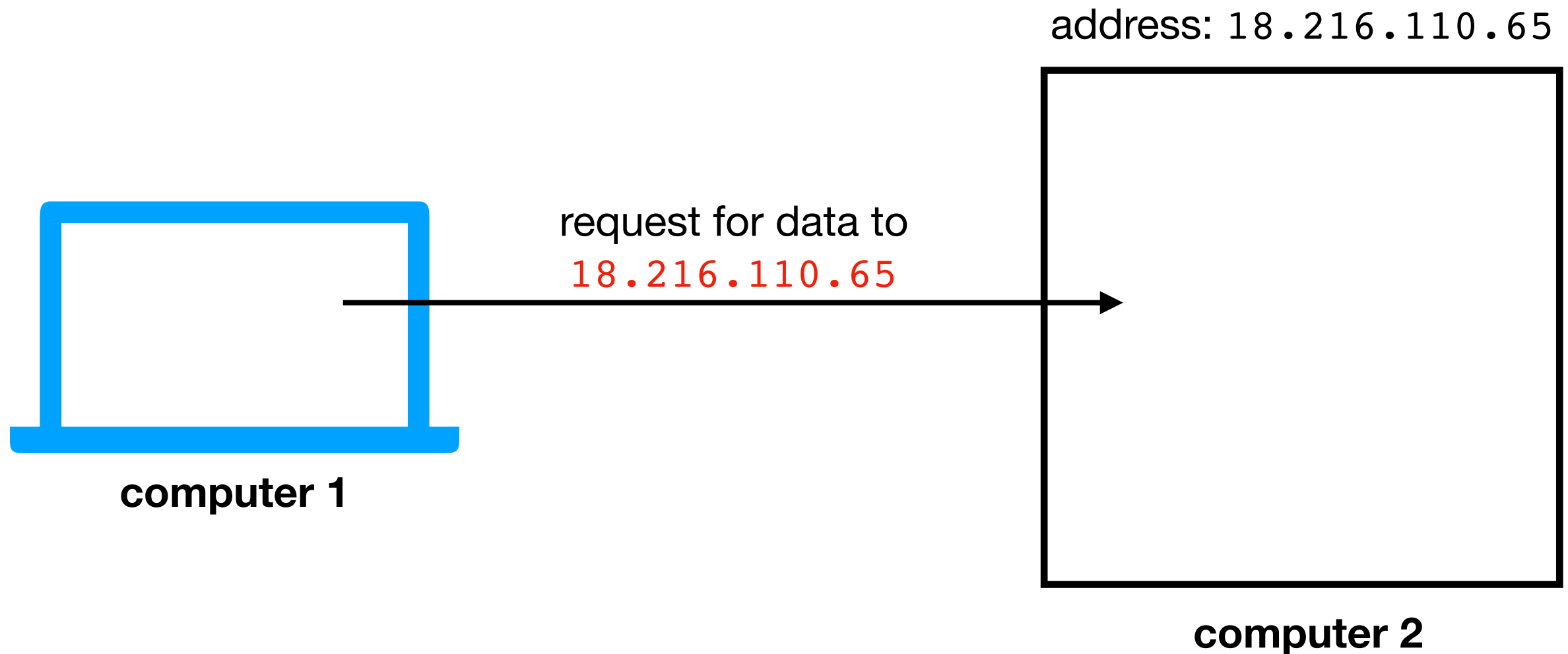


computer 2

**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

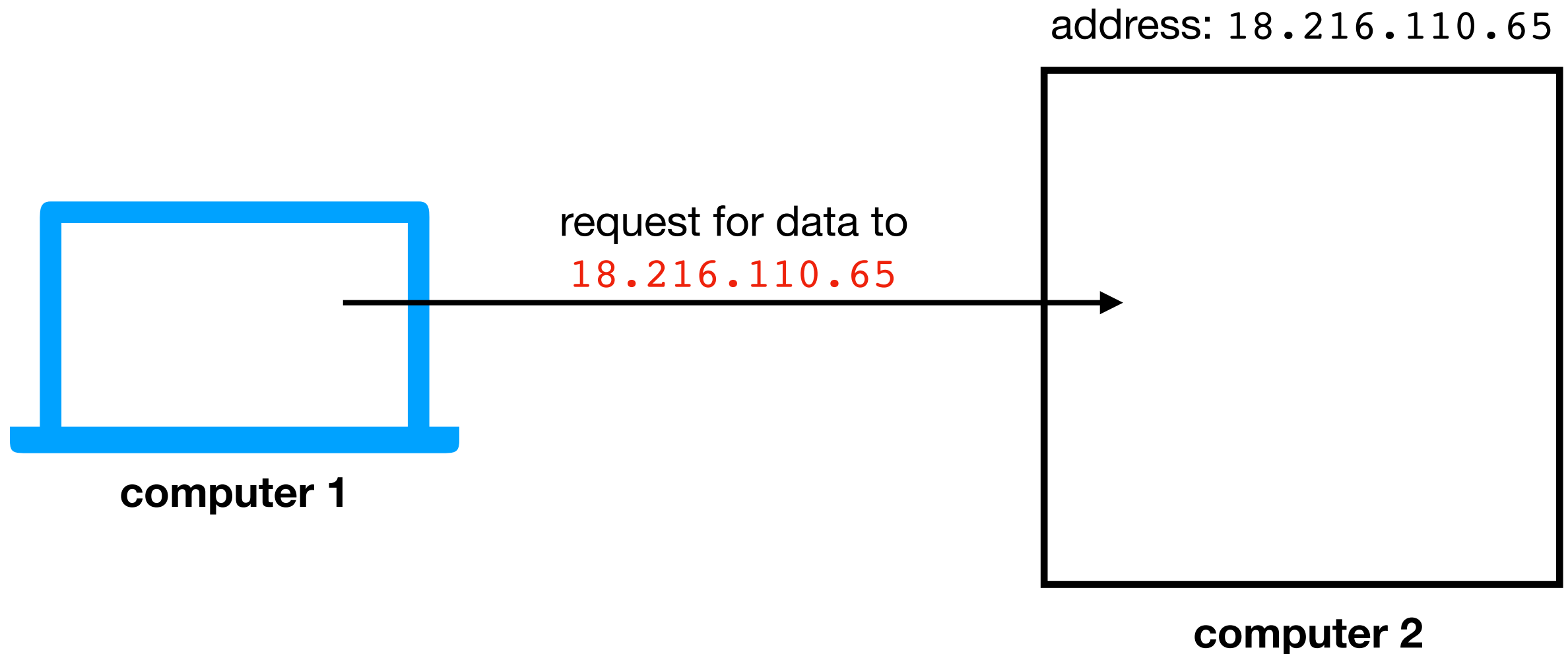


**Solution:** every machine\* has an IP address (Internet Protocol).  
Requests are sent to a specific IP address.

\*some machines have more multiple addresses

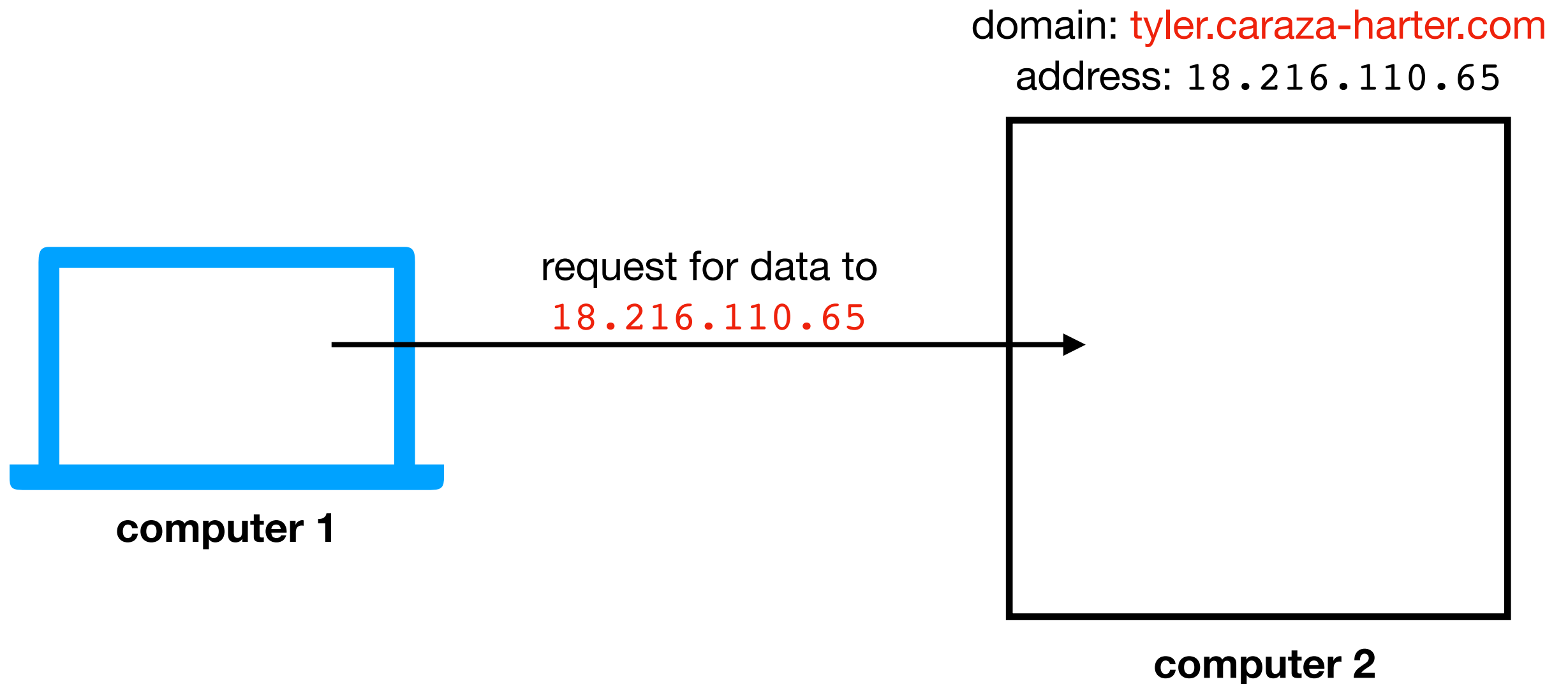


# Internet Protocol



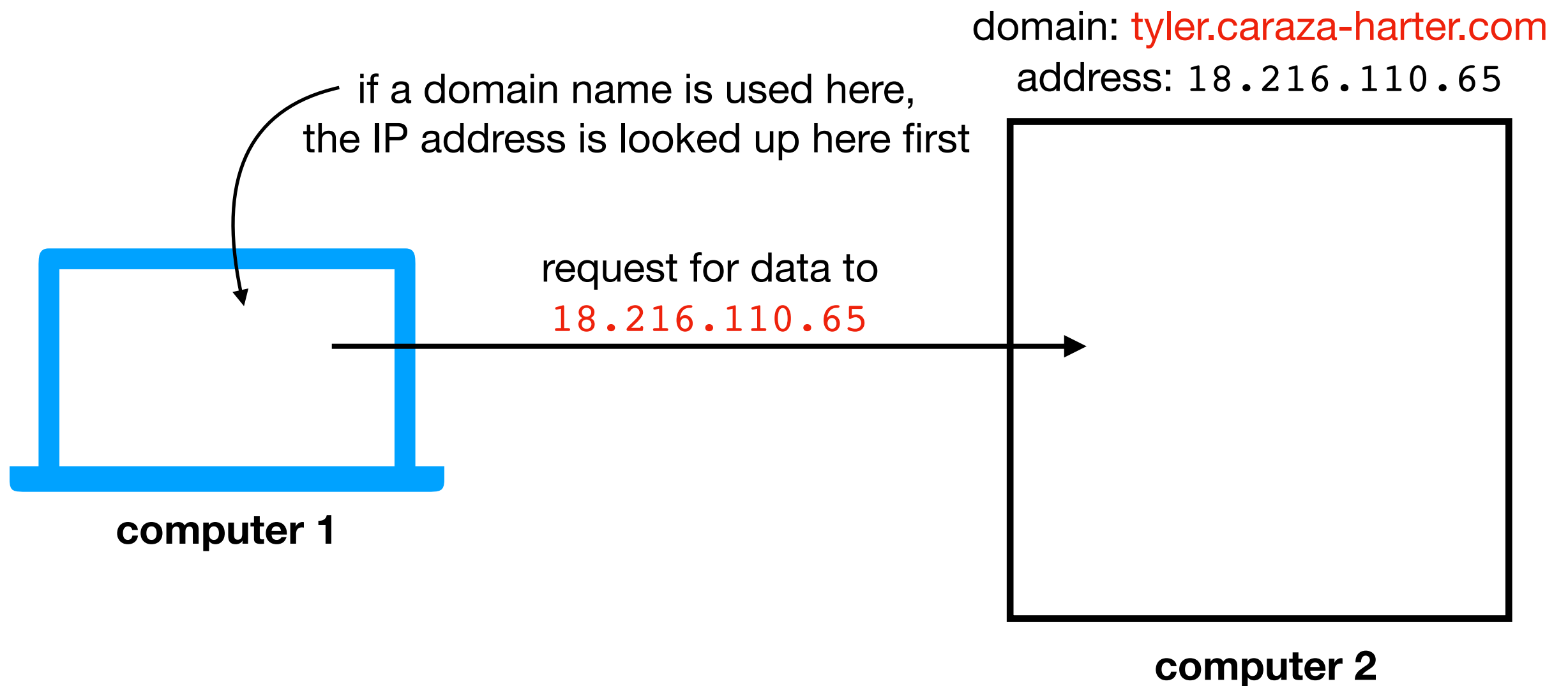
**Challenge:** it's hard to remember IP addresses.  
Imagine you had to type a number instead of [www.google.com](http://www.google.com)!

# Domain Names



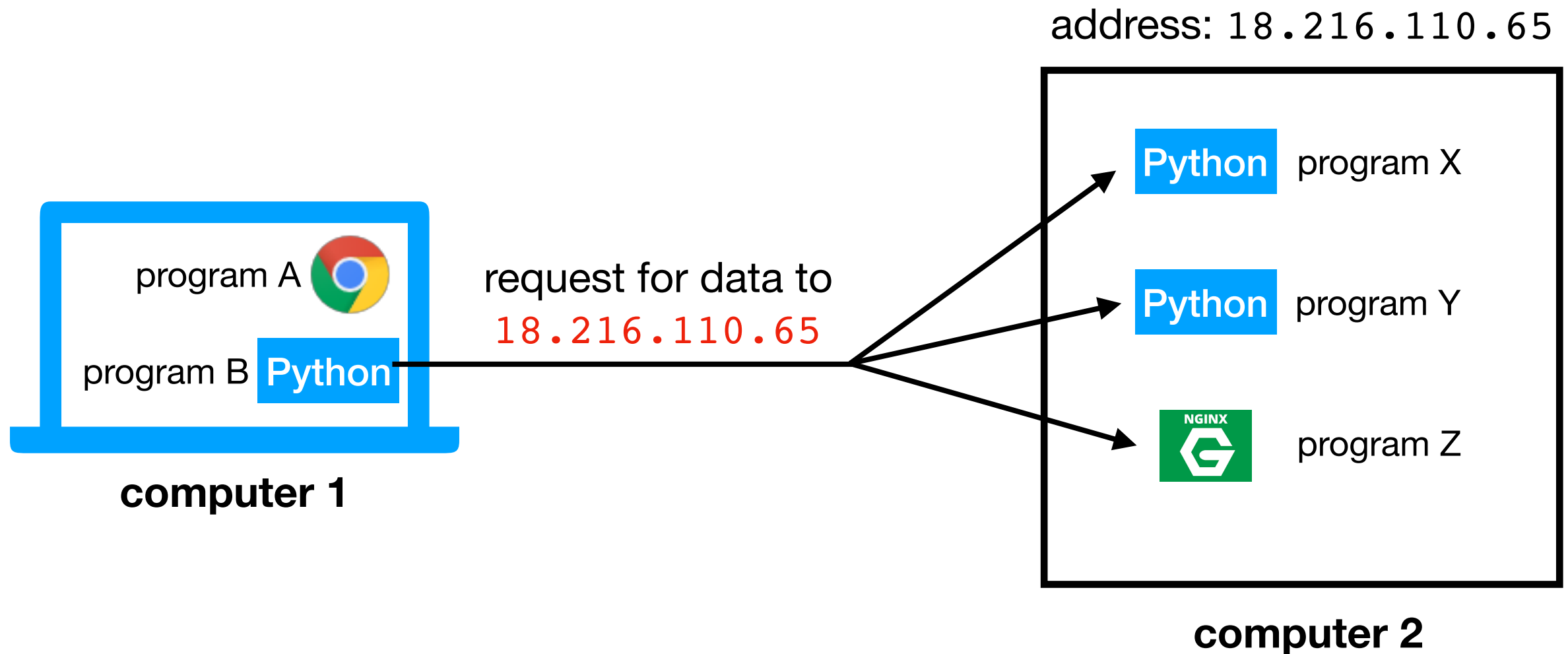
**Solution:** use “nicknames” (called domain names)  
for IP addresses of machines that serve data

# Domain Names



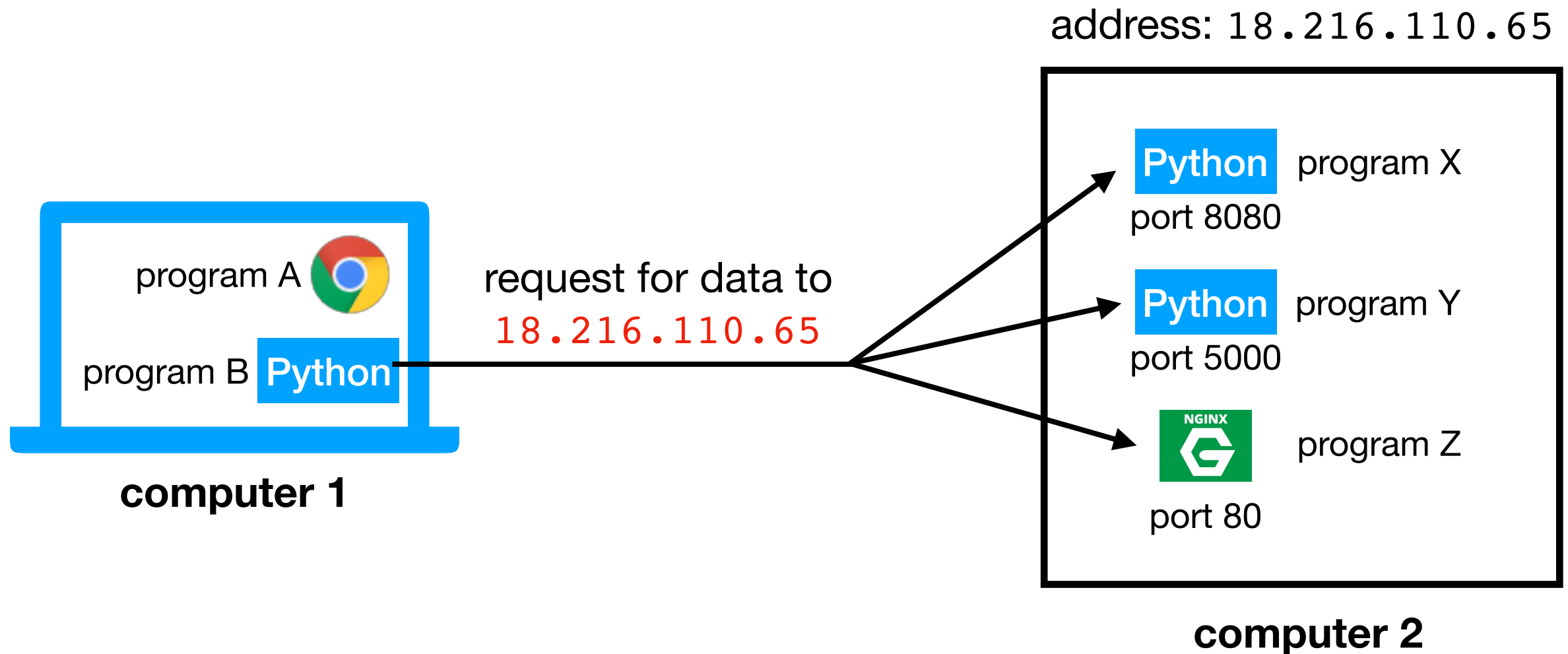
**Solution:** use “nicknames” (called domain names)  
for IP addresses of machines that serve data

# Port Numbers



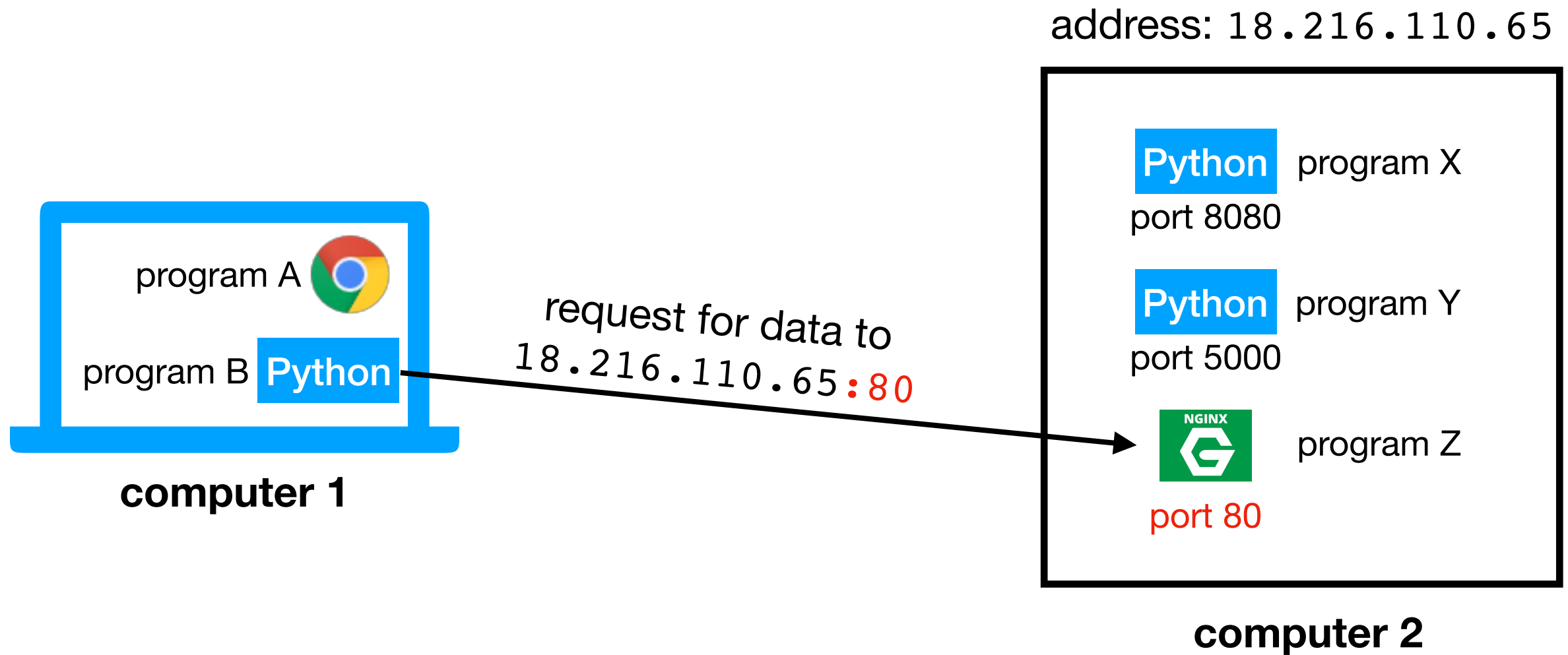
**Challenge:** there may be multiple programs running on each computer.  
How do we get the messages to the right program?

# Port Numbers



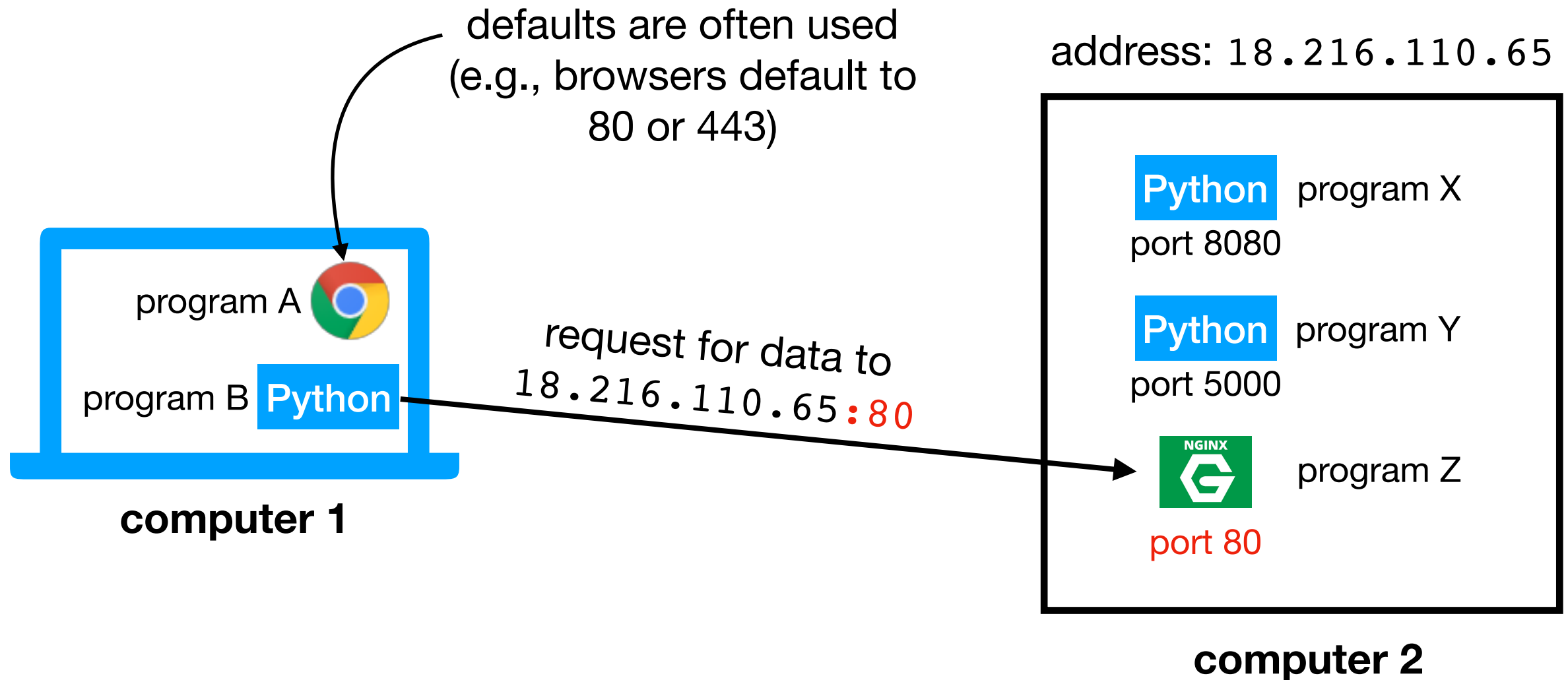
**Solution:** give each program a unique ID (called a “port number”)

# Port Numbers

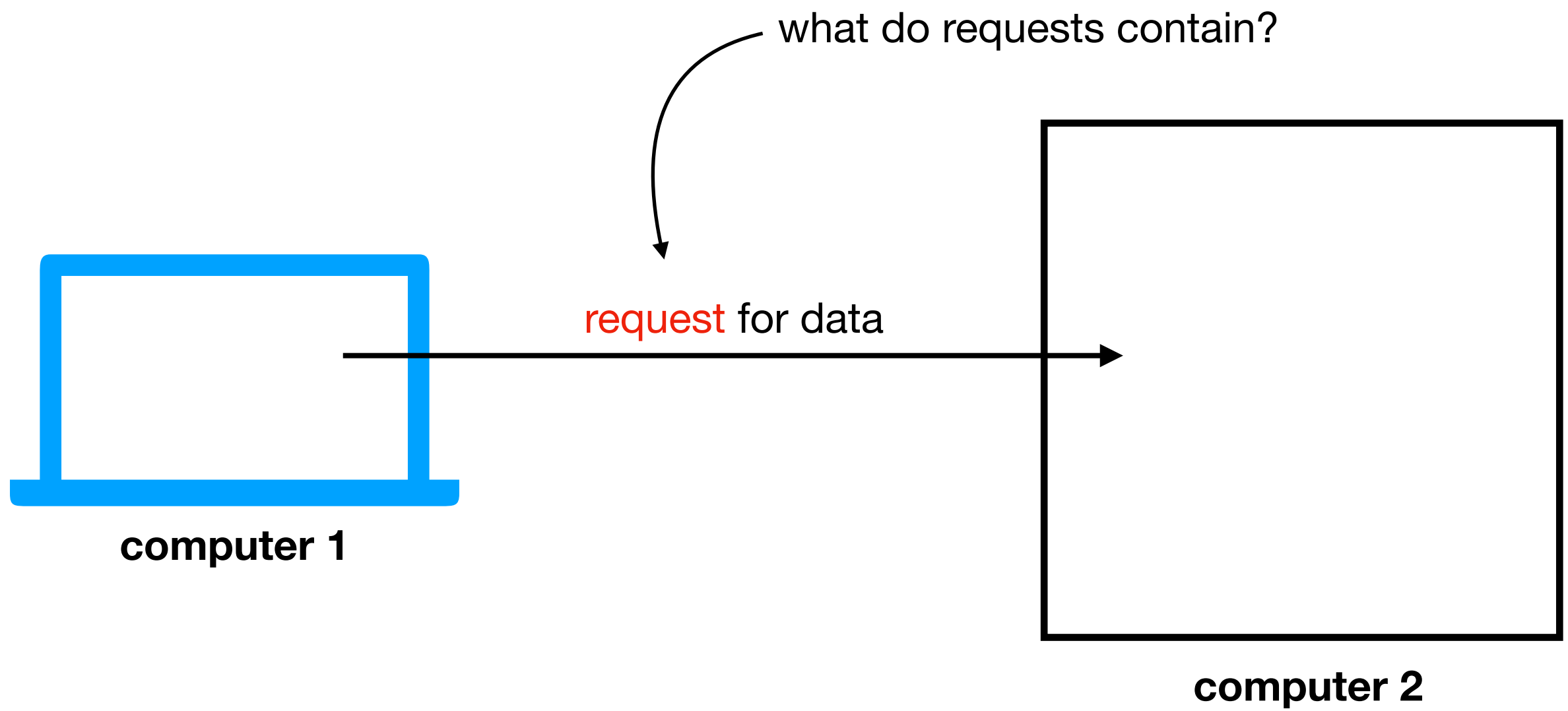


**Solution:** specify port number in request

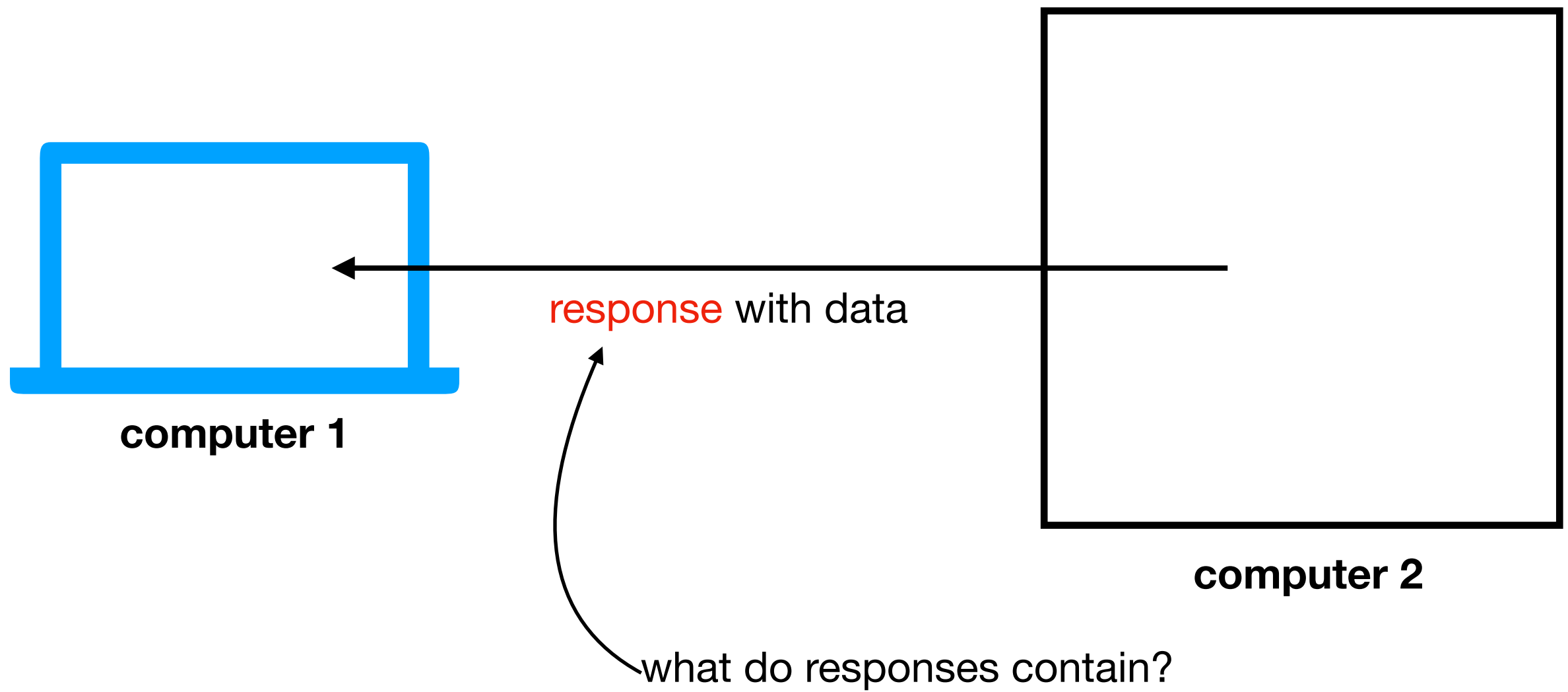
# Port Numbers



**Solution:** specify port number in request

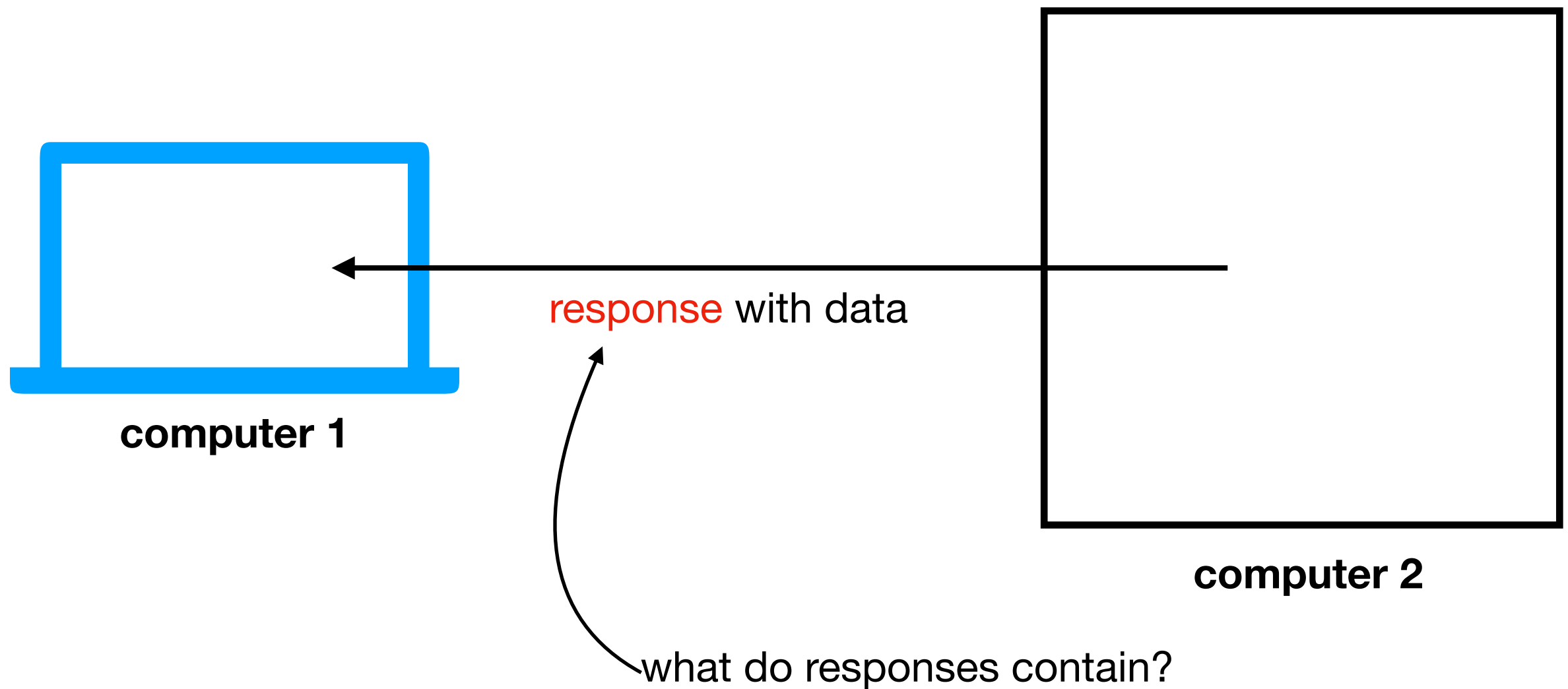






depends on application! (video chat, web browsing, etc)

we'll only consider **web applications** for this semester



# Learning Objectives Today

Motivation

Networking Basics

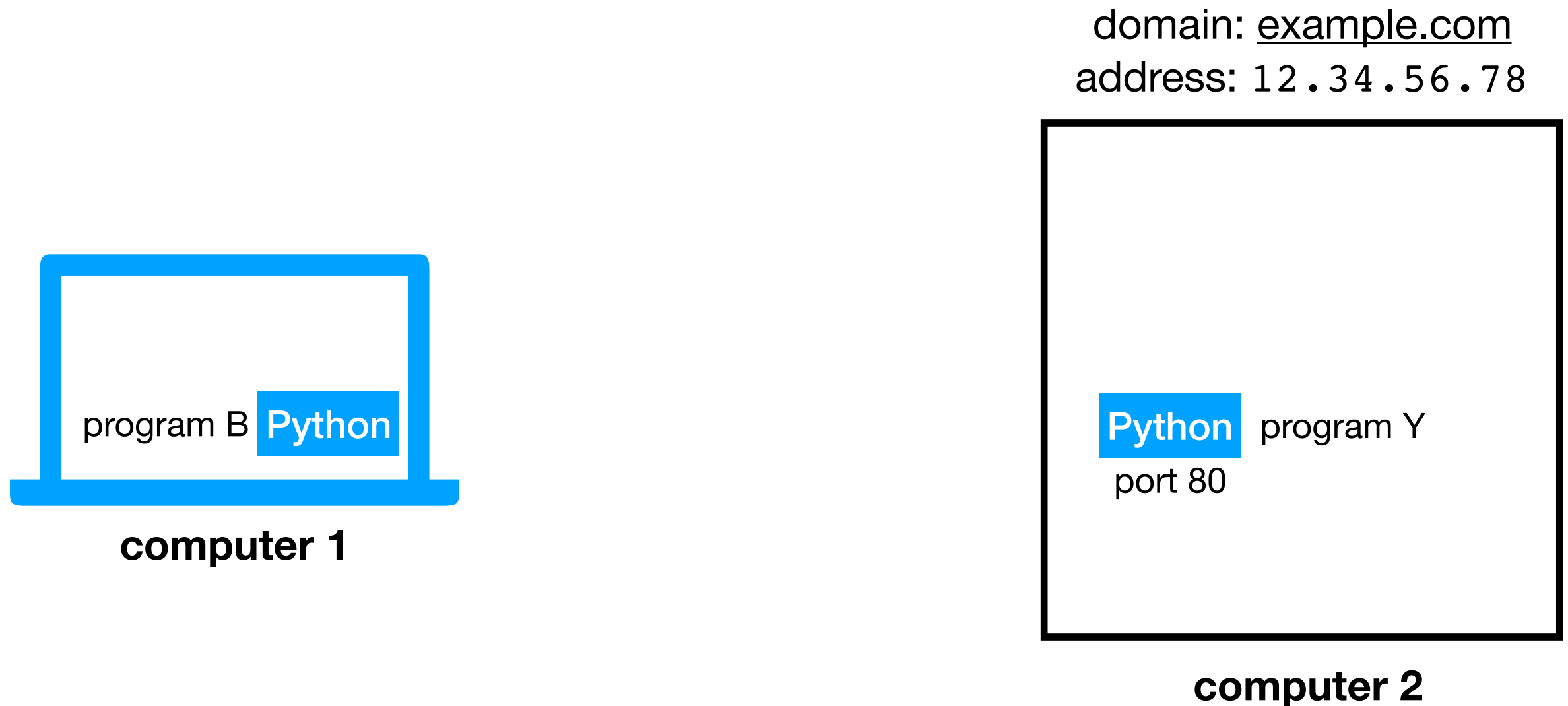
HTTP (Hypertext Transfer Protocol)

Requests Module

# HTTP

Protocol for communicating web data

- downloading a specific webpage, image, etc

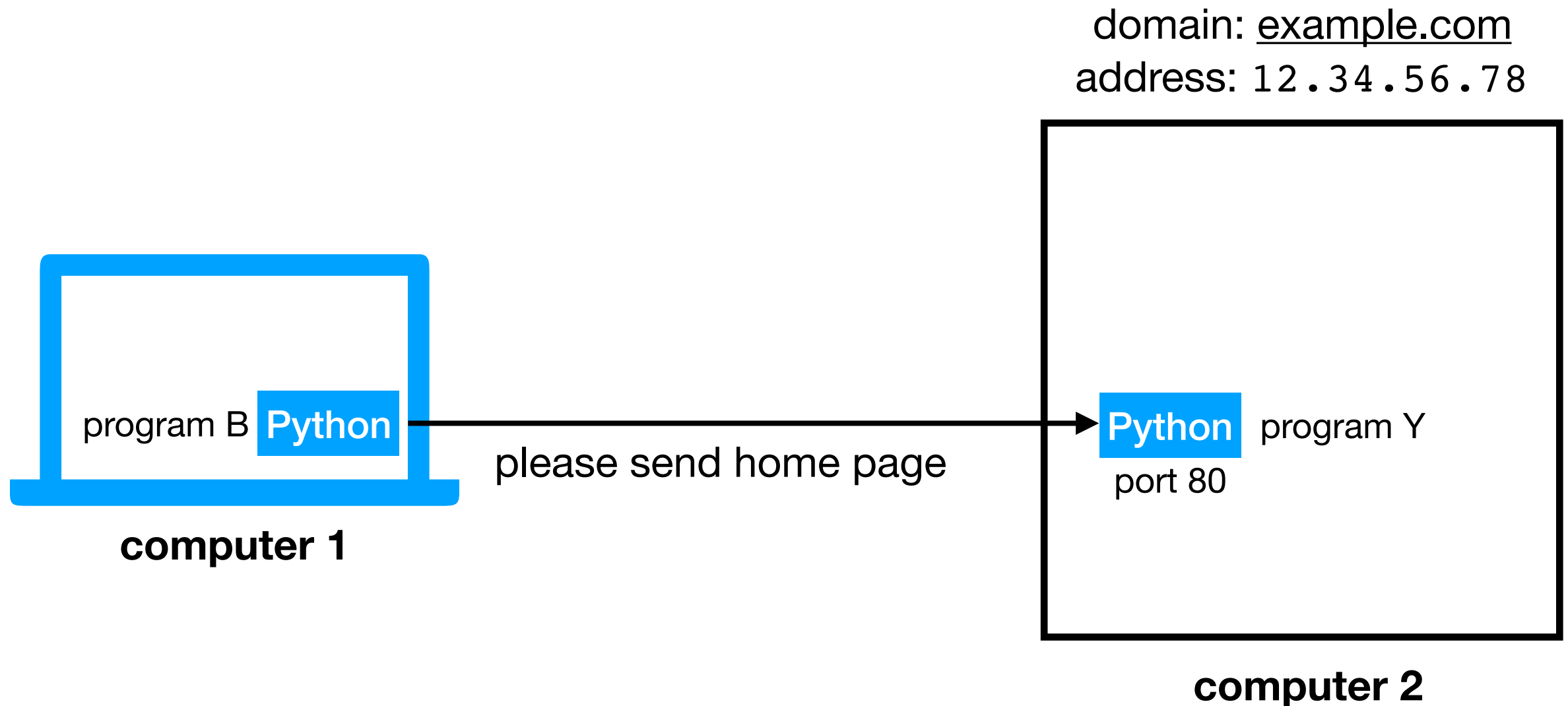


**Note:** we won't talk about HTTPS today, which is HTTP with encryption

# HTTP

Protocol for communicating web data

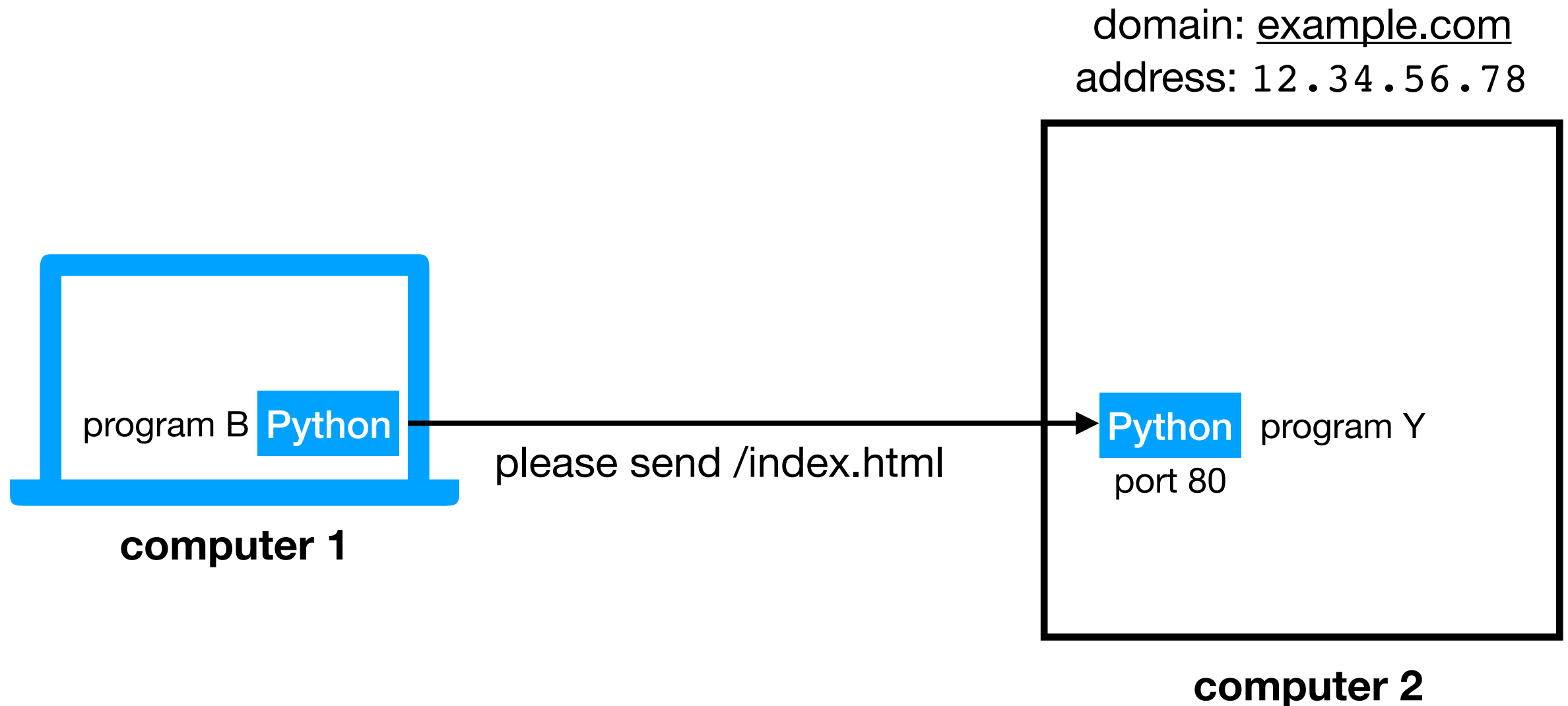
- downloading a specific webpage, image, etc



# HTTP

Protocol for communicating web data

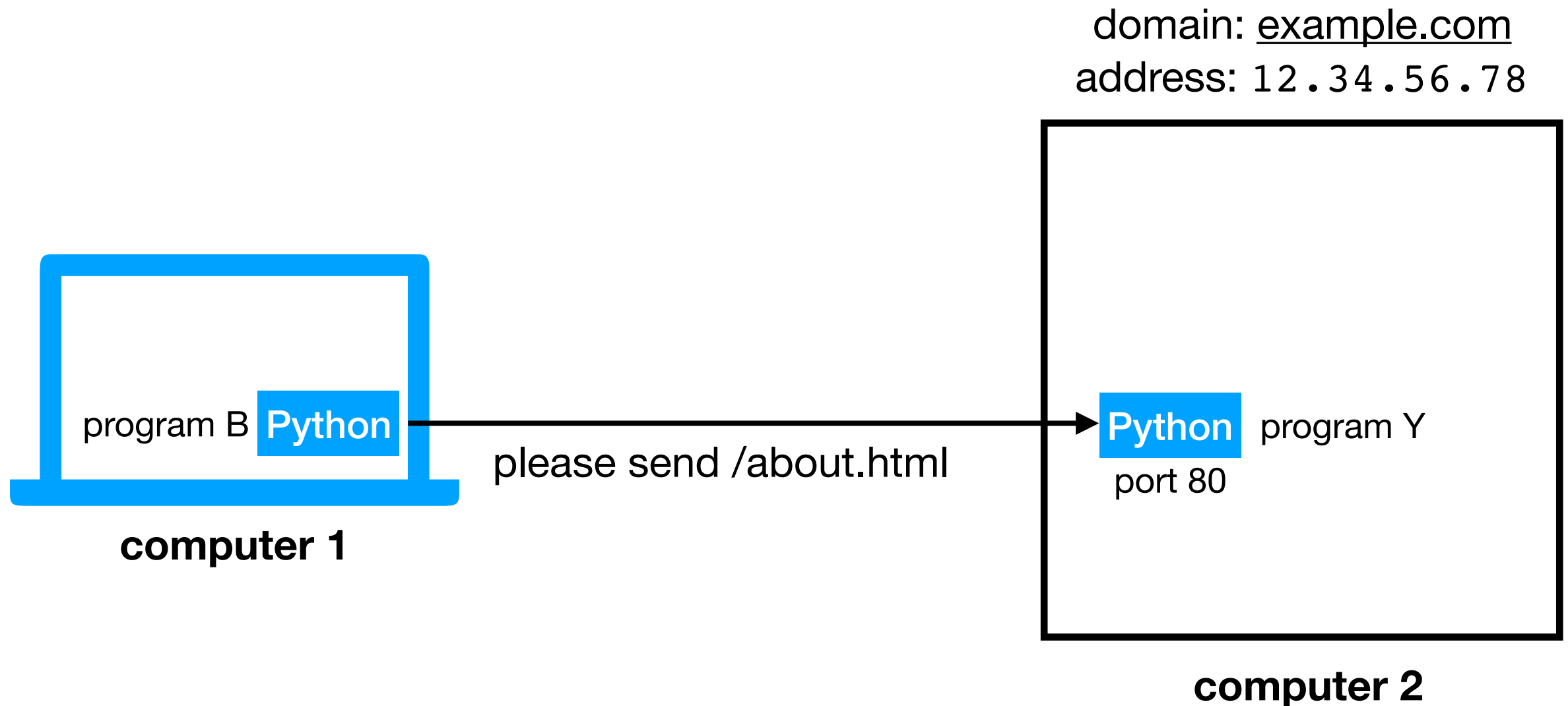
- downloading a specific webpage, image, etc



# HTTP

Protocol for communicating web data

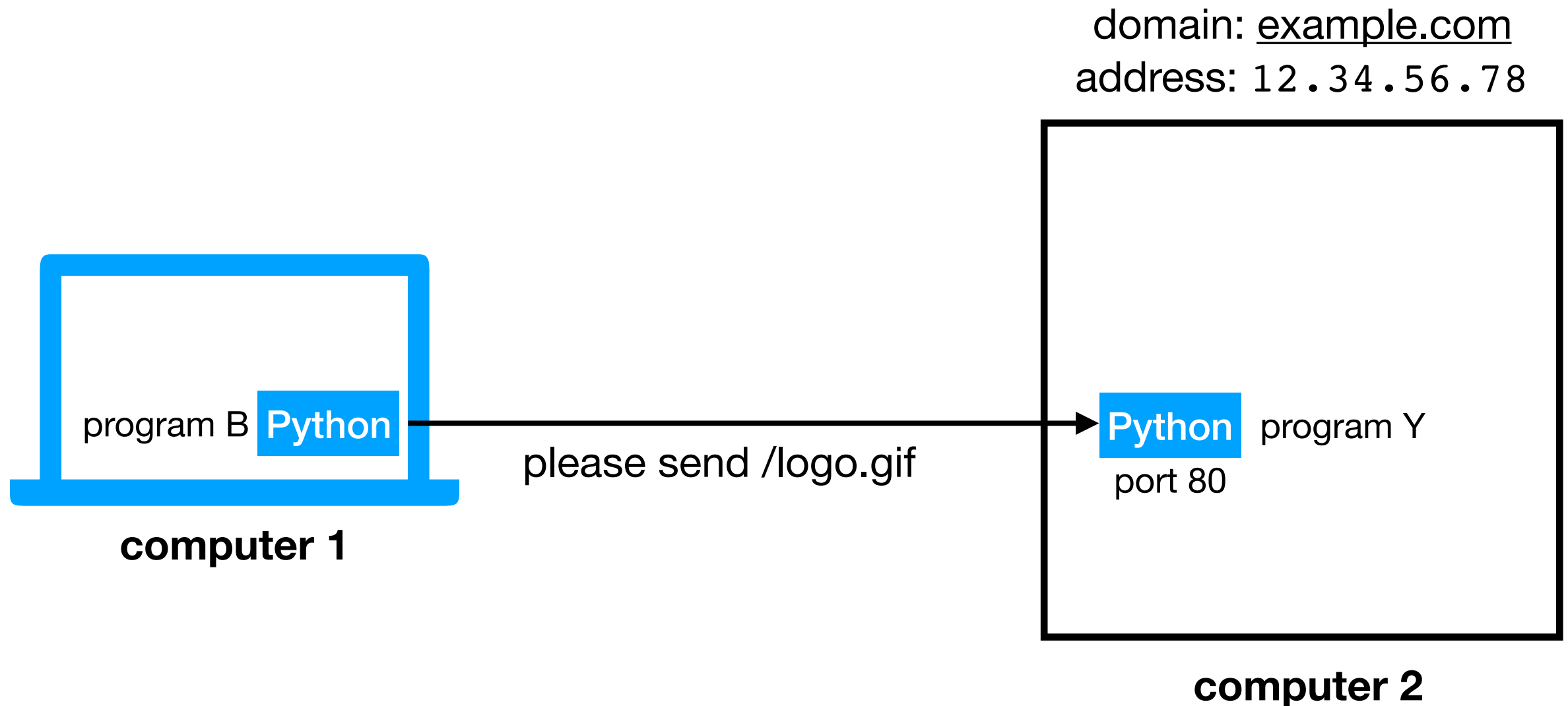
- downloading a specific webpage, image, etc



# HTTP

Protocol for communicating web data

- downloading a specific webpage, image, etc

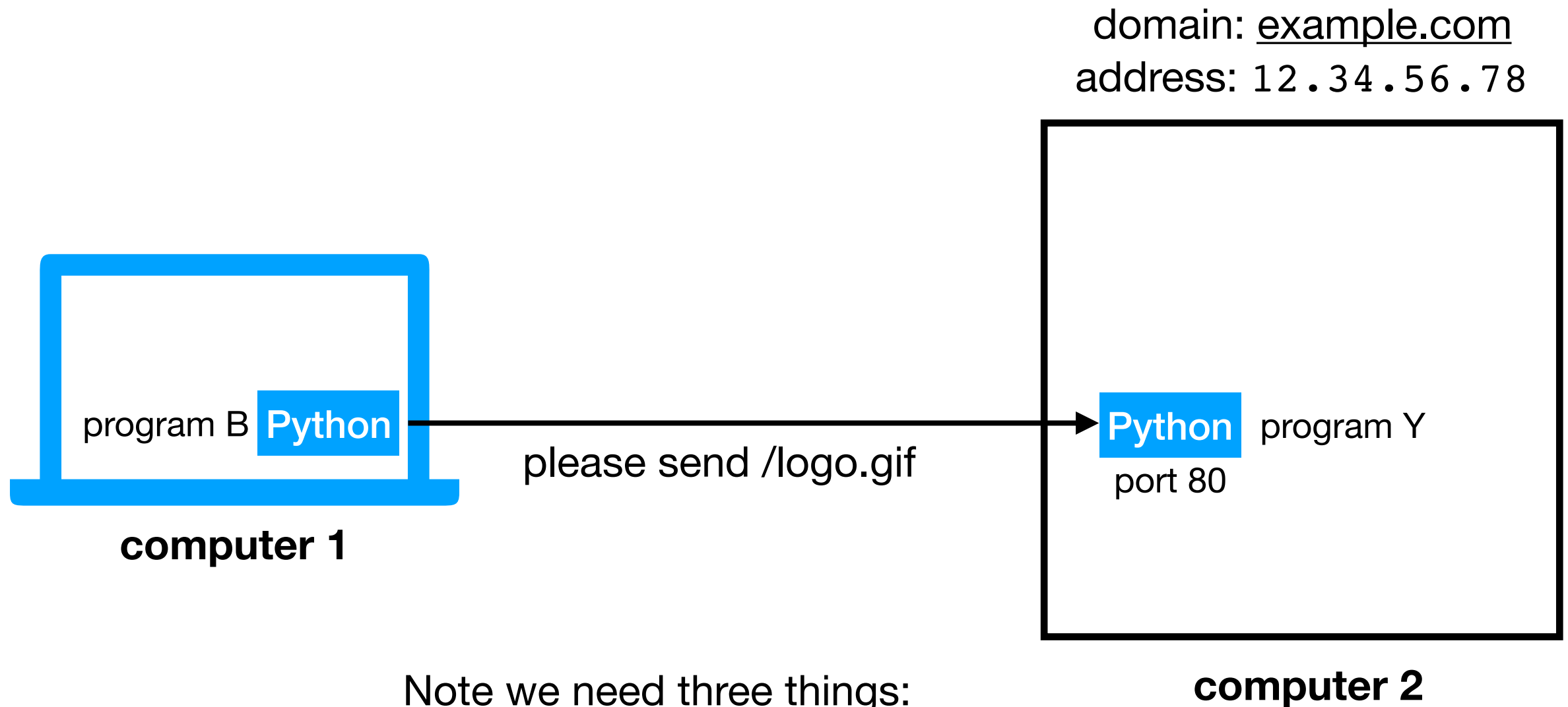




# HTTP

Protocol for communicating web data

- downloading a specific webpage, image, etc



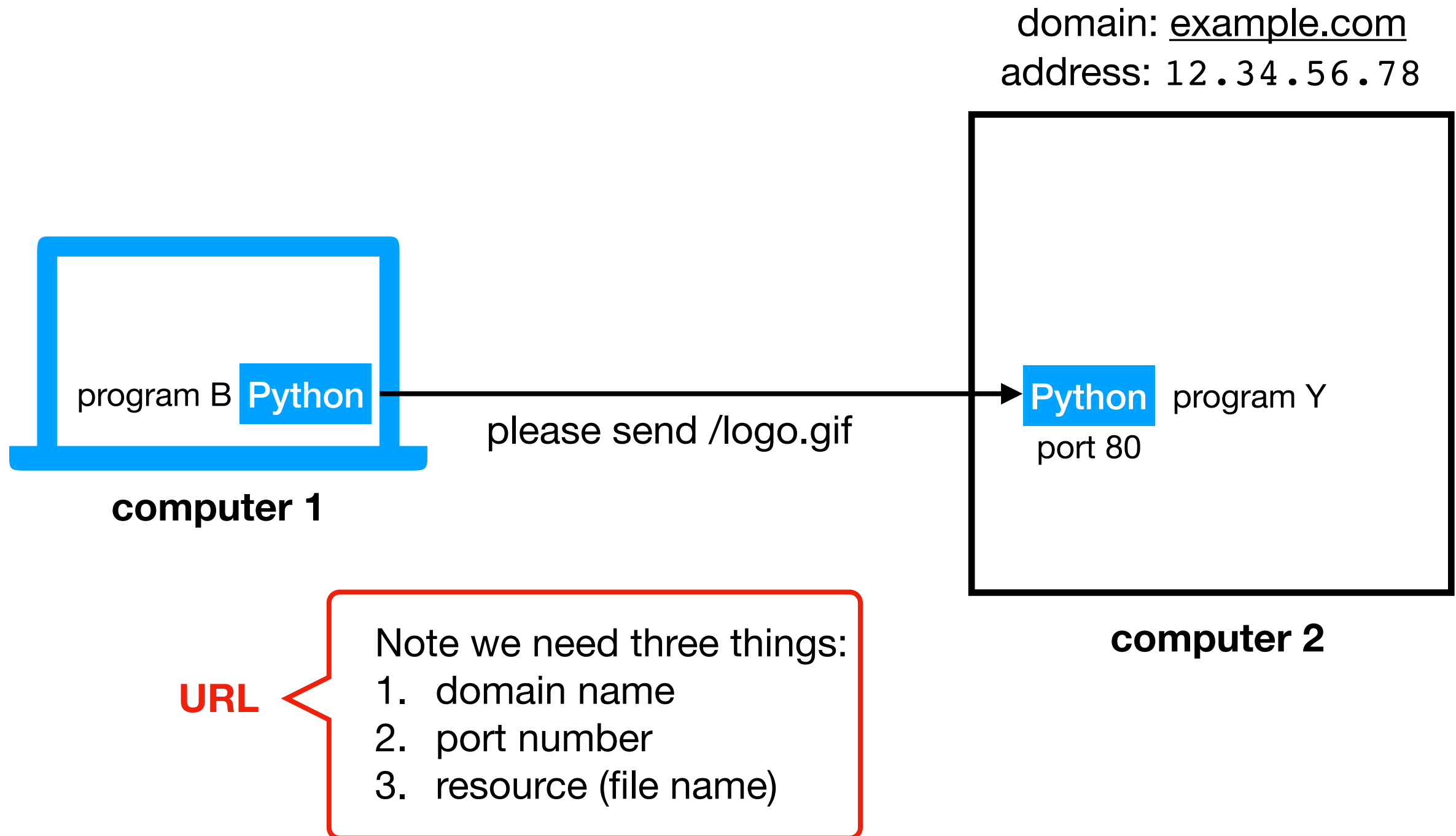
Note we need three things:

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

# HTTP

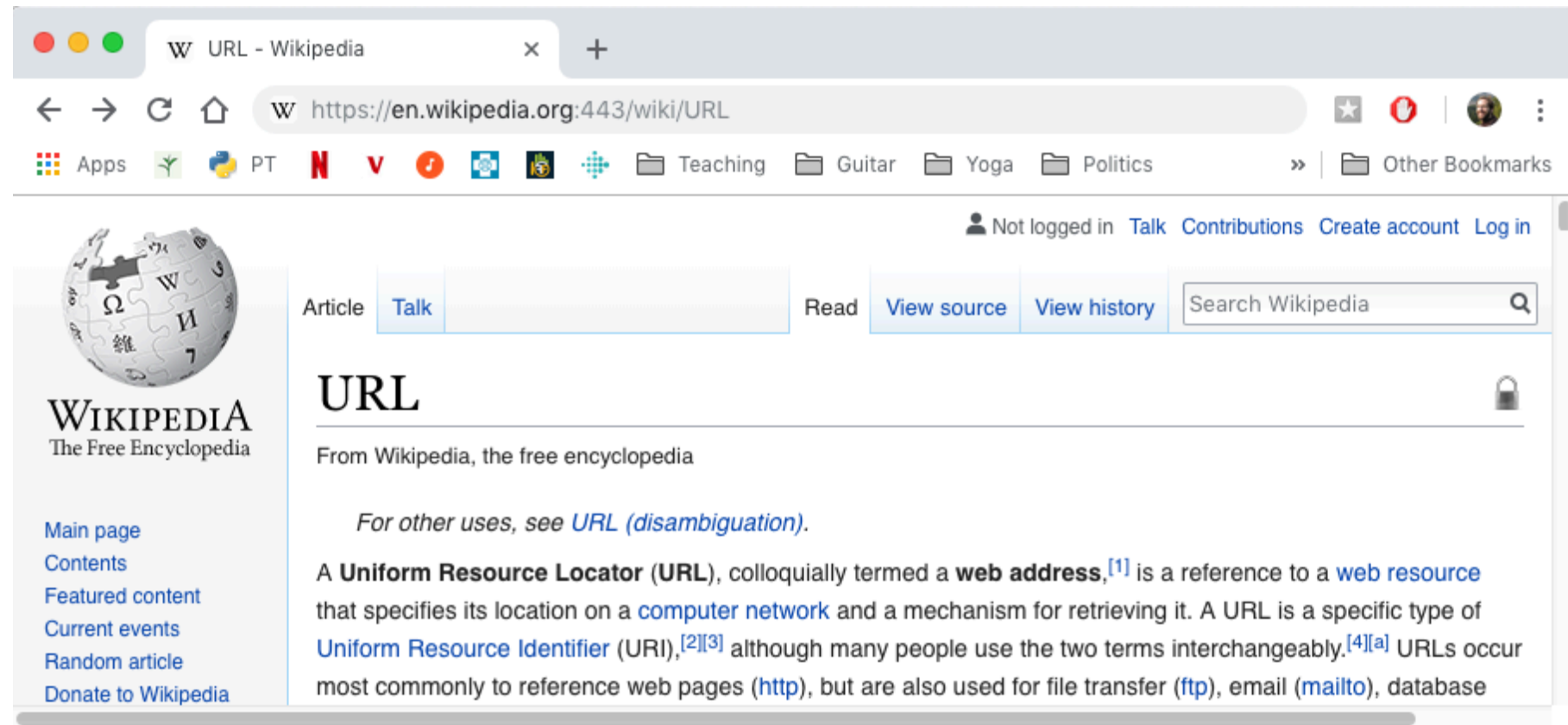
Protocol for communicating web data

- downloading a specific webpage, image, etc



# URLs

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



**URL**

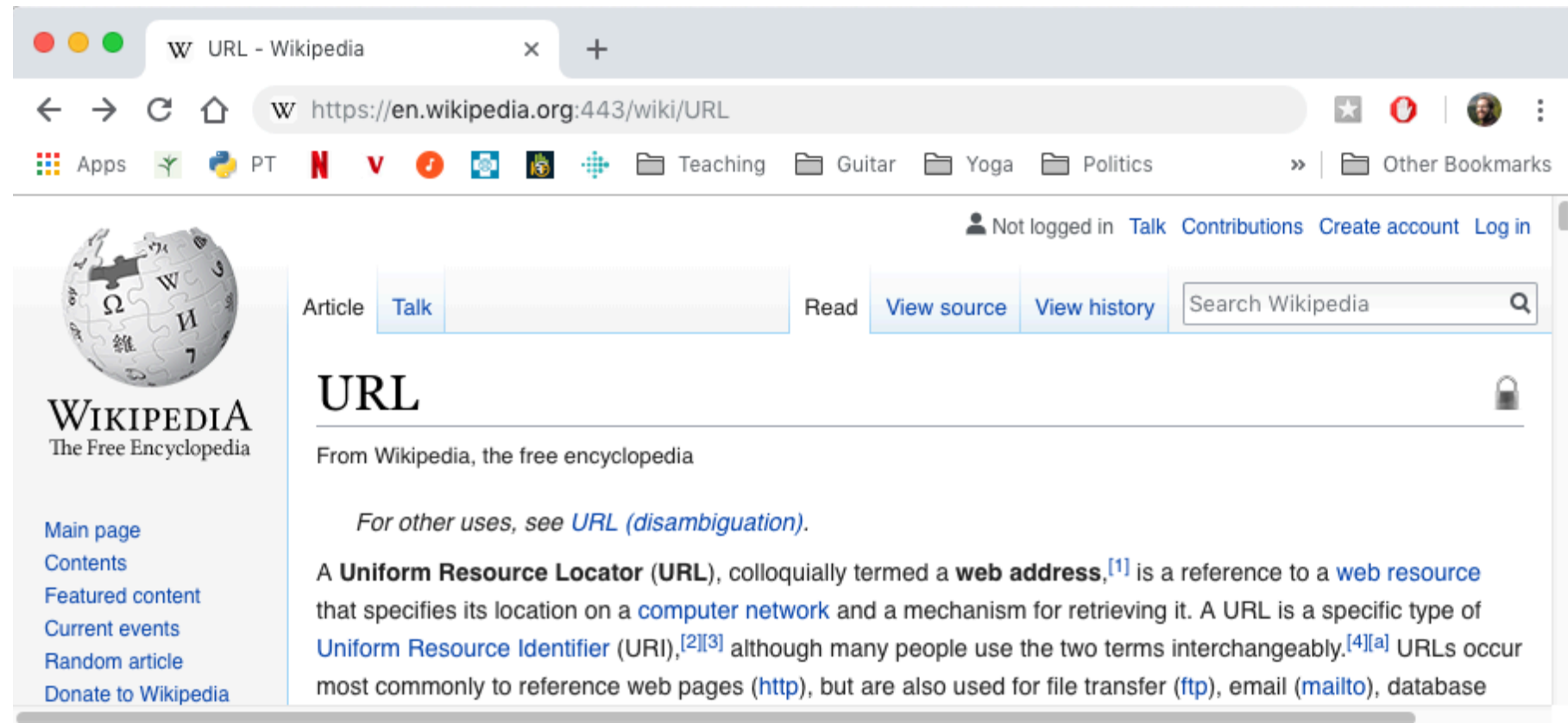
Note we need three things:

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

# URLs

domain name

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



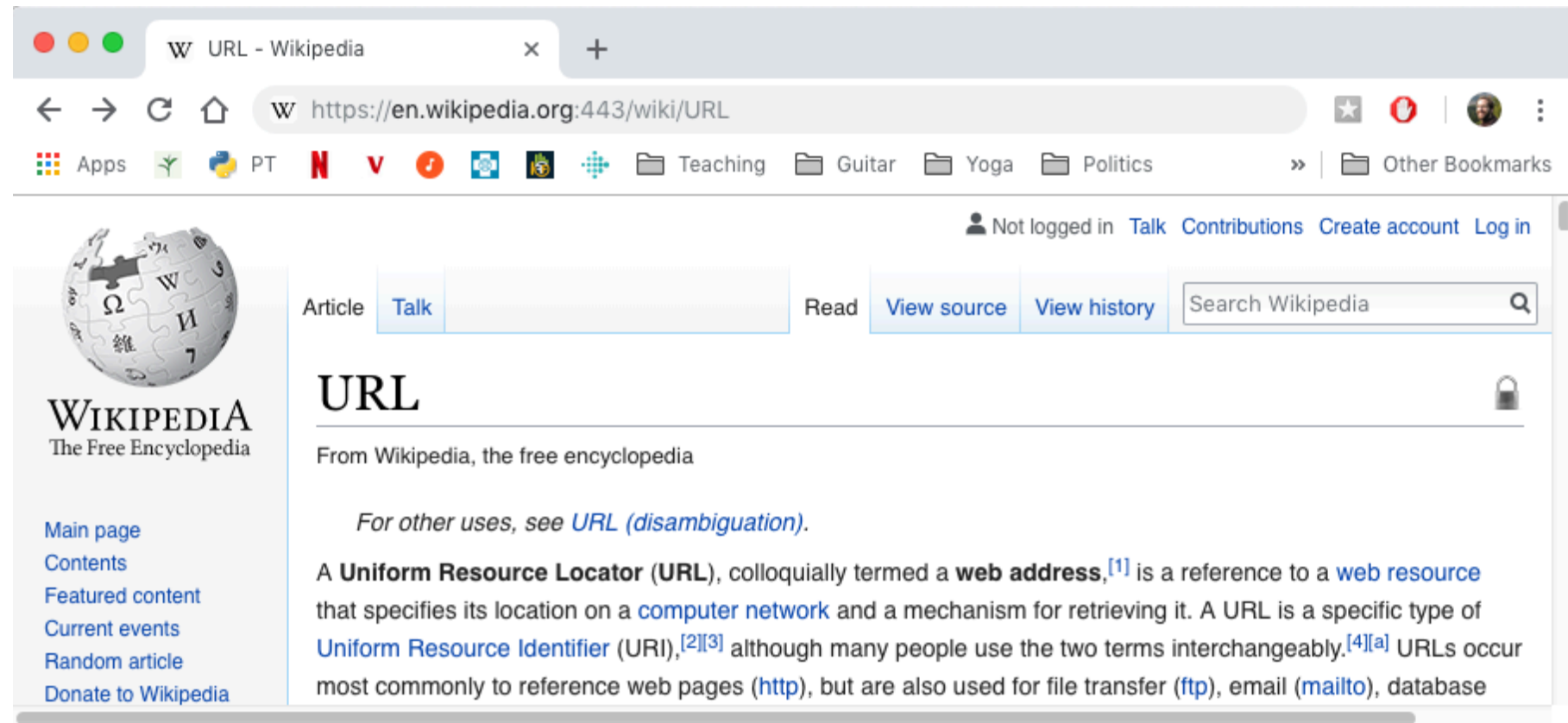
URL

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



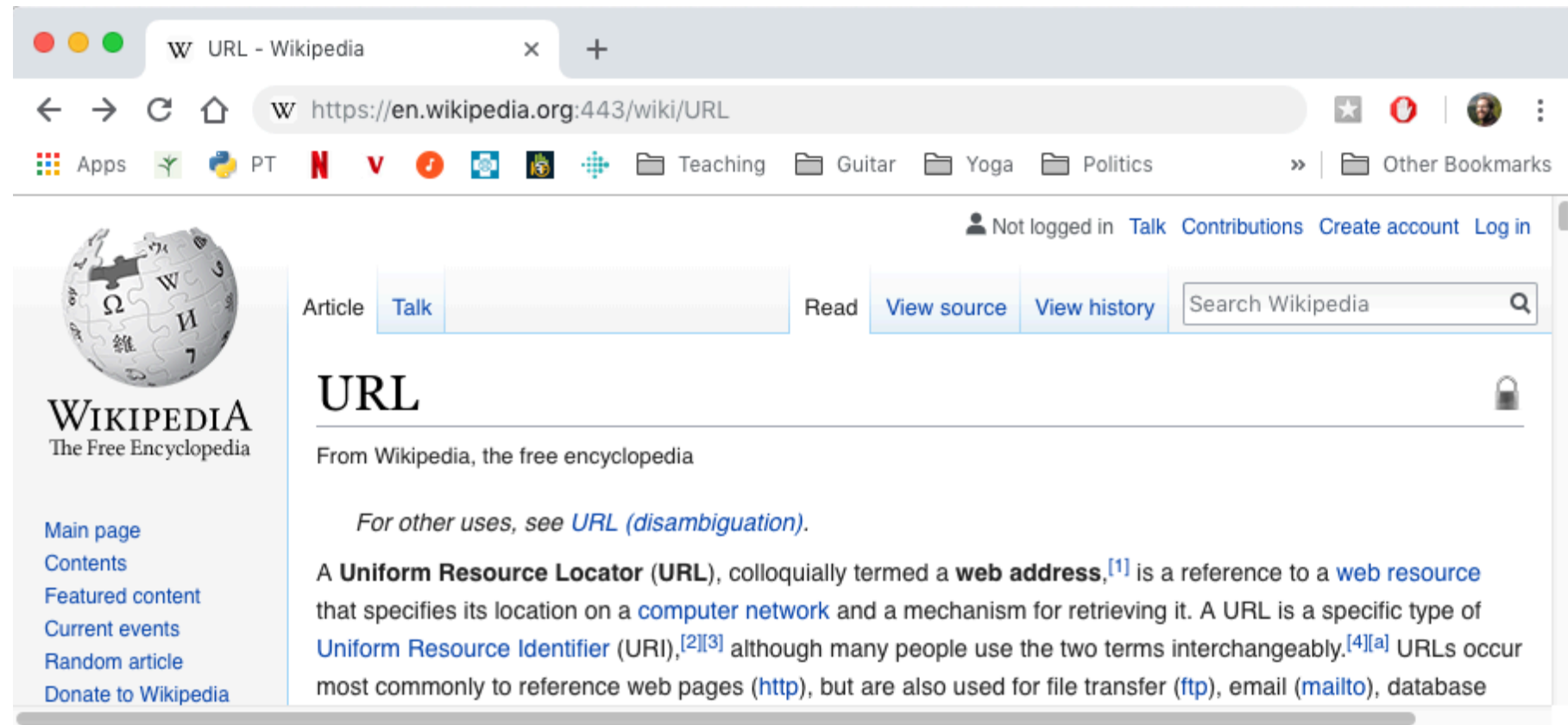
URL

Note we need three things:

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

# URLs

https://<sup>domain name</sup>en.wikipedia.org:<sup>port</sup>443/<sup>resource</sup>wiki/URL



URL

Note we need three things:

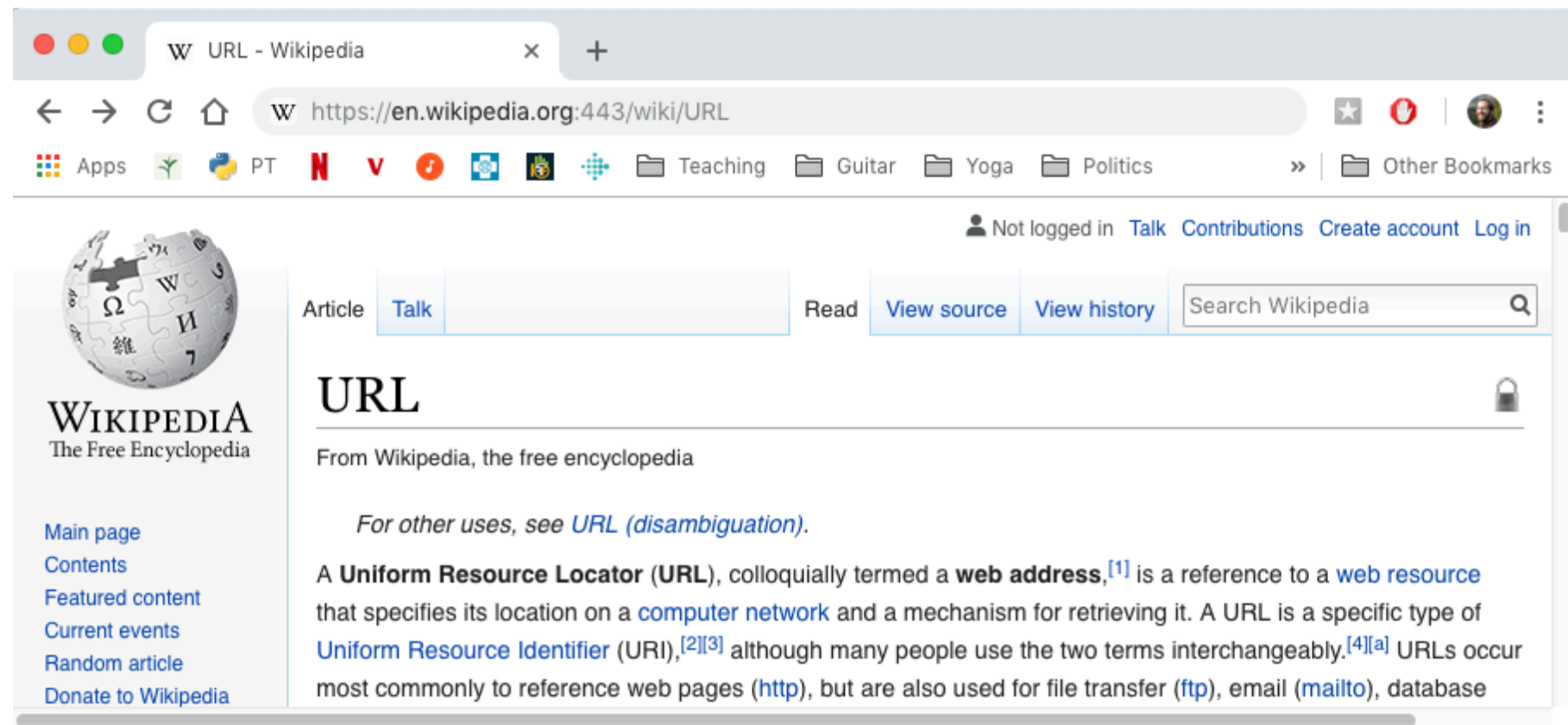
1. domain name
2. port number
3. resource (file name)

# URLs

domain name      resource

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

port would have defaulted to 443 if not specified



**URL**

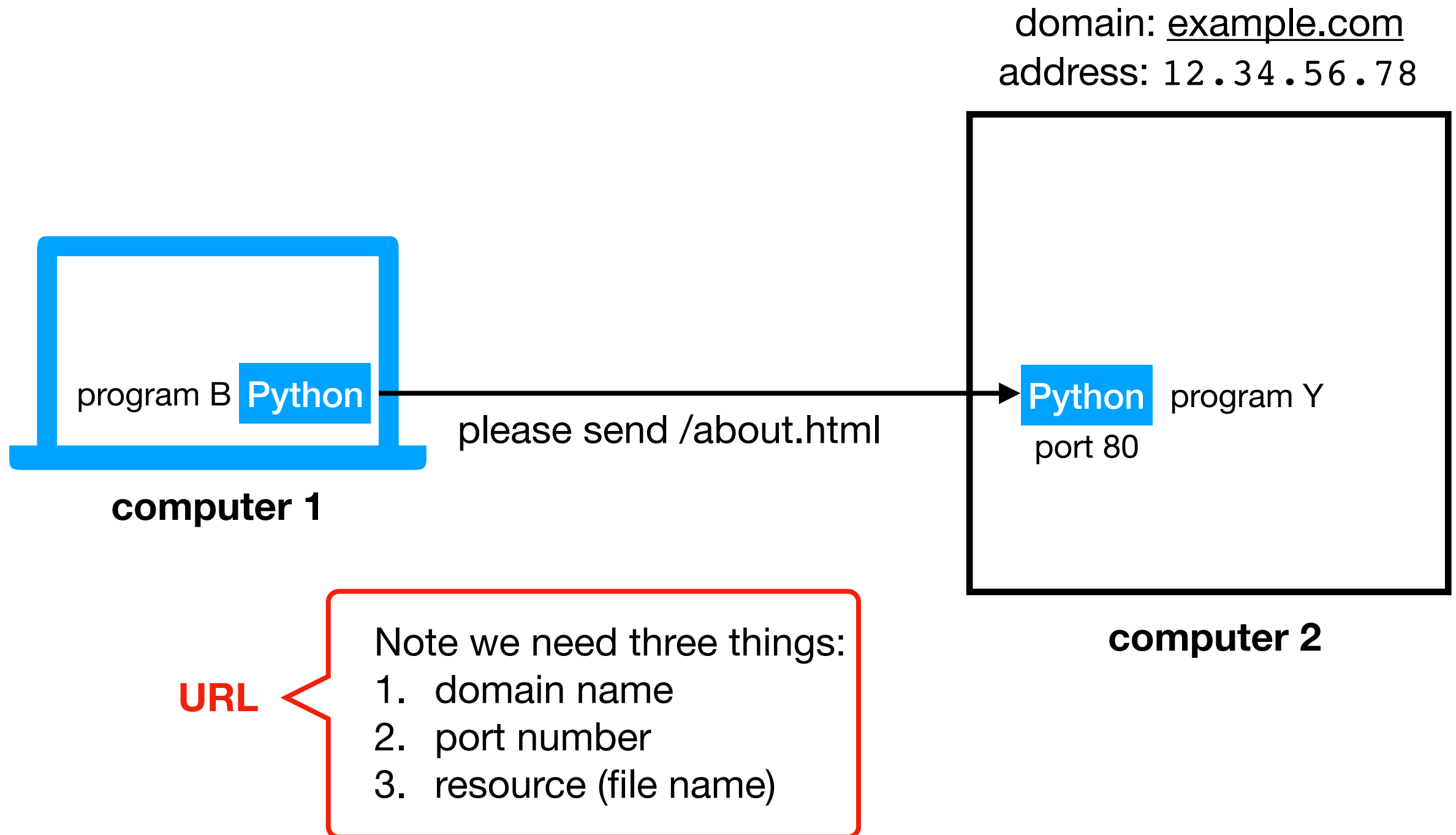
Note we need three things:

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

# HTTP

Protocol for communicating web data

- downloading a specific webpage, image, etc

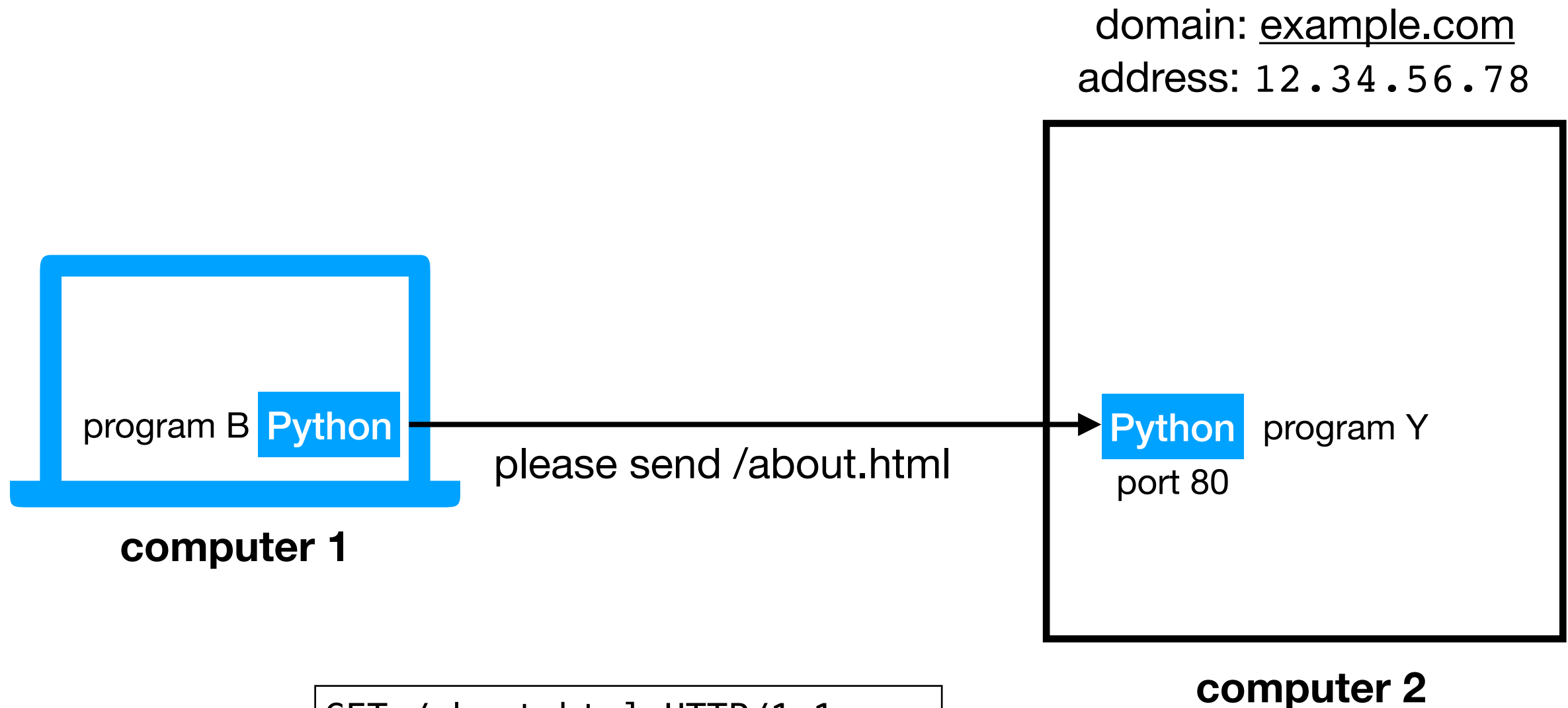




# HTTP

Protocol for communicating web data

- downloading a specific webpage, image, etc



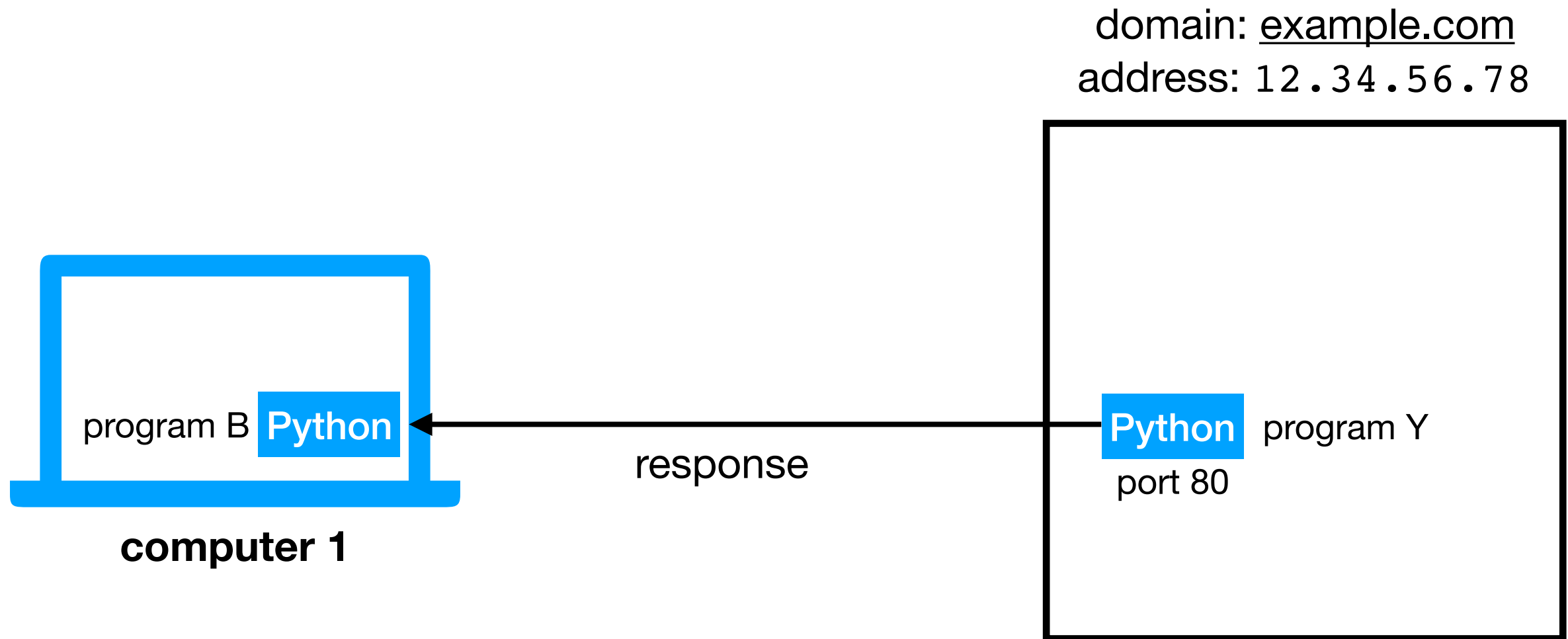
**HTTP Request:**

```
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

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
```

data in about.html

all the contents

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

# Request and Response

## HTTP Request:

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

we want the about.html page

**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

data in about.html

all the contents

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  
wont 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
```

data in about.html

all the contents

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

# Demo 1: State Populations

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

## Input:

- 50 CSV files

## Output:

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

# 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