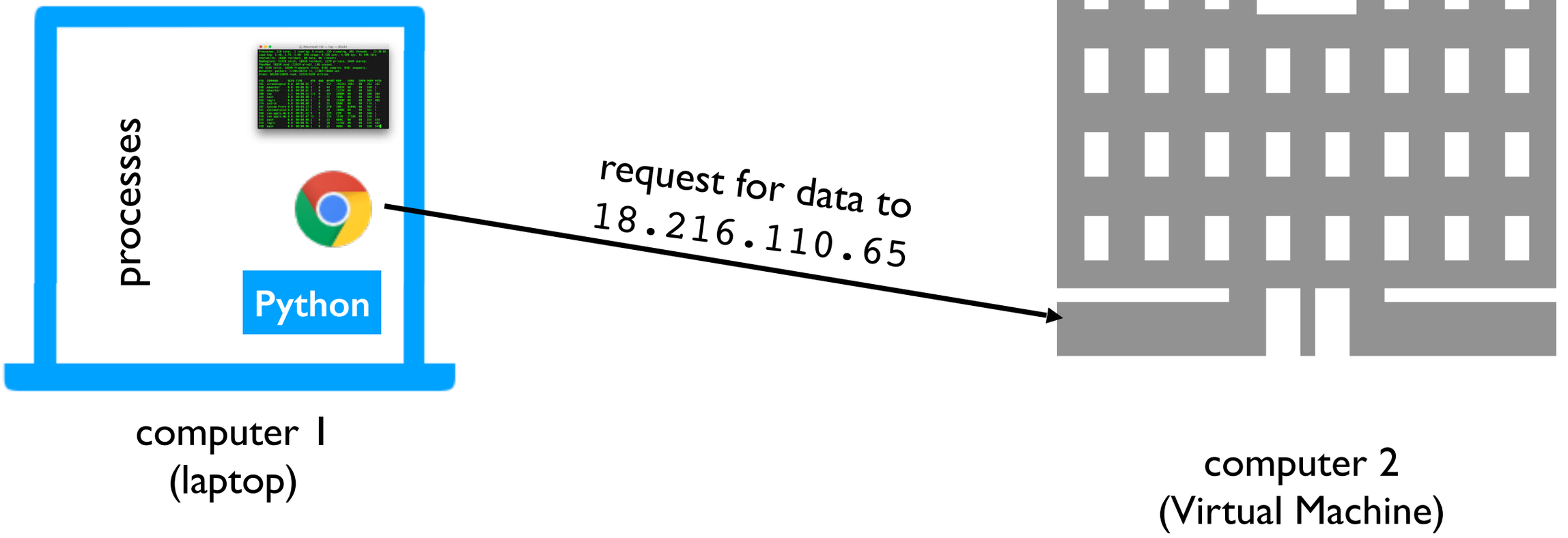
# [320] Web 2: Flask

Department of Computer Sciences
University of Wisconsin-Madison

# Getting Requests Through

[like street address]

IP address: `18.216.110.65`

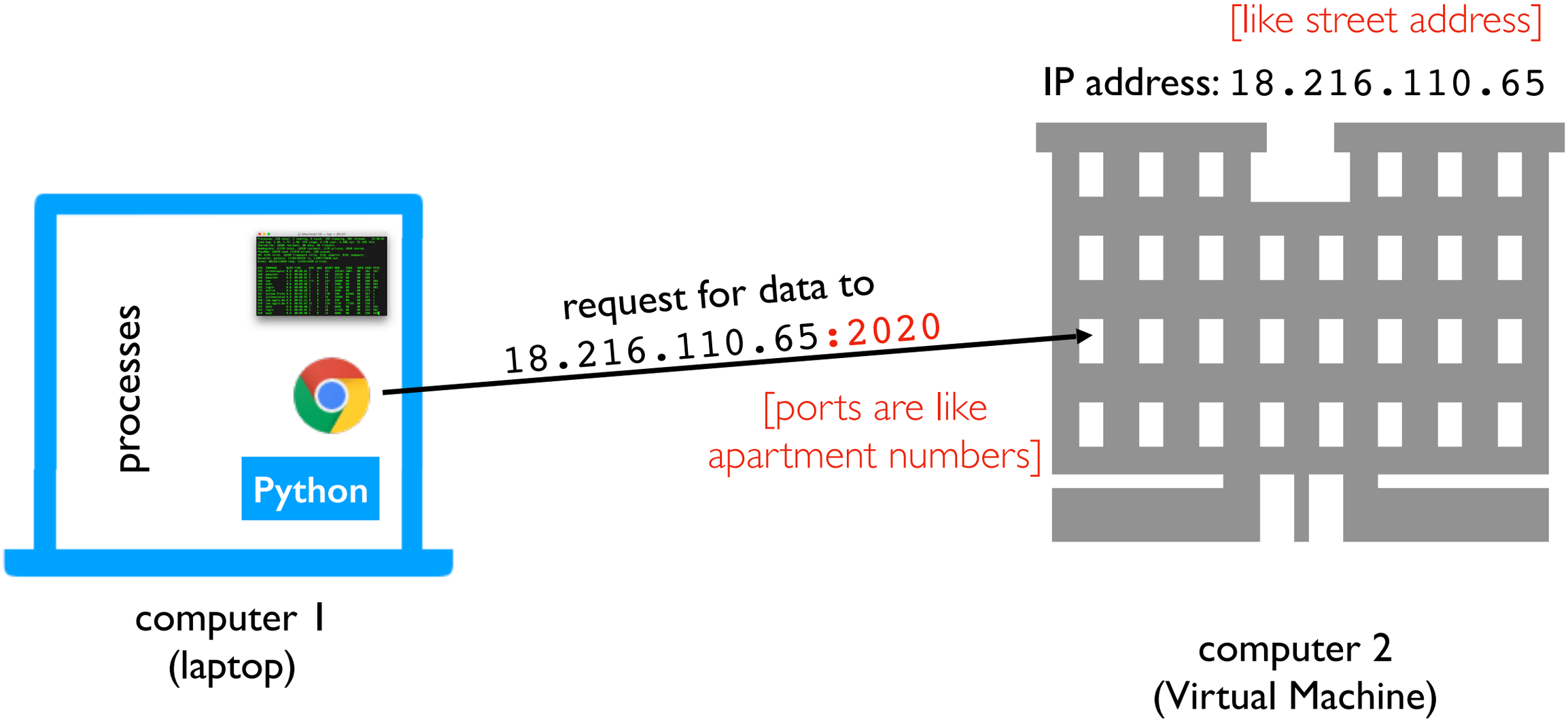



computer 1
(laptop)

computer 2
(Virtual Machine)

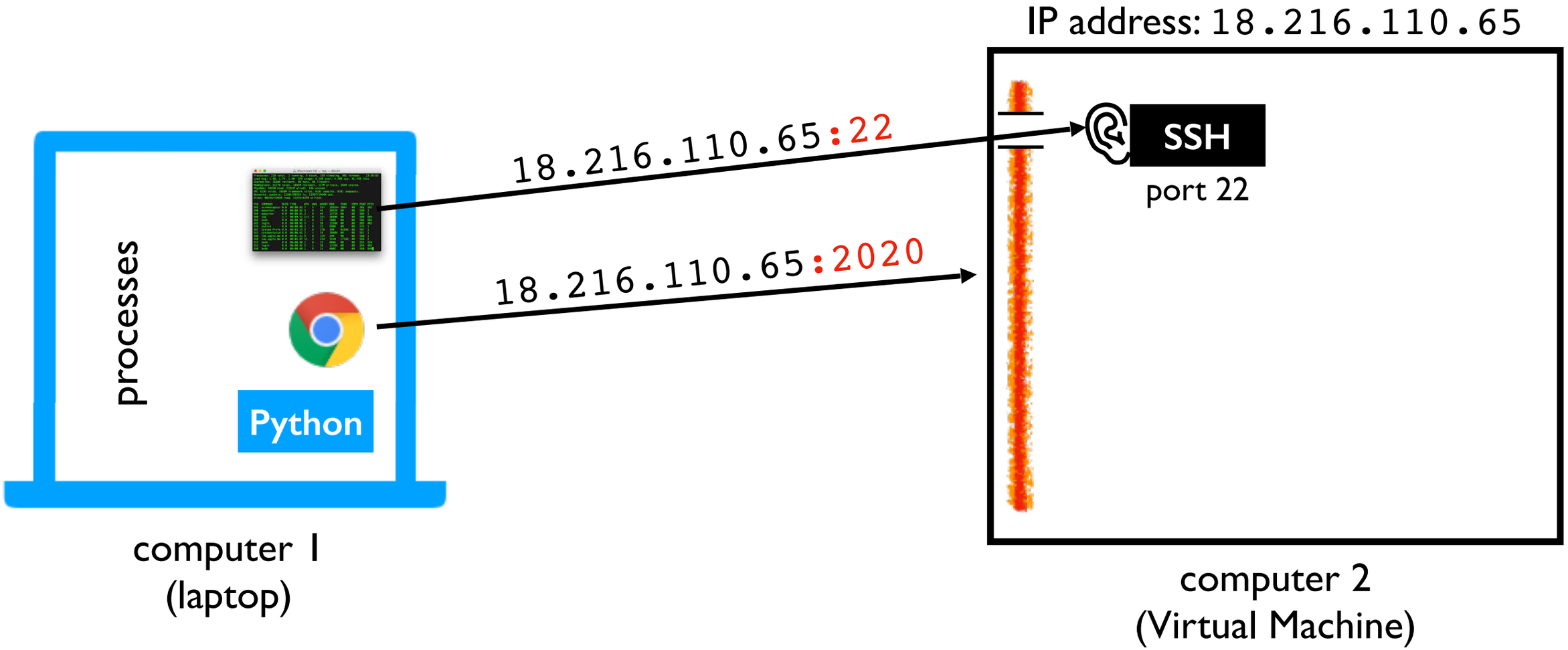
**Scenario:** we want to access Jupyter on our virtual machine from our laptop

# Getting Requests Through



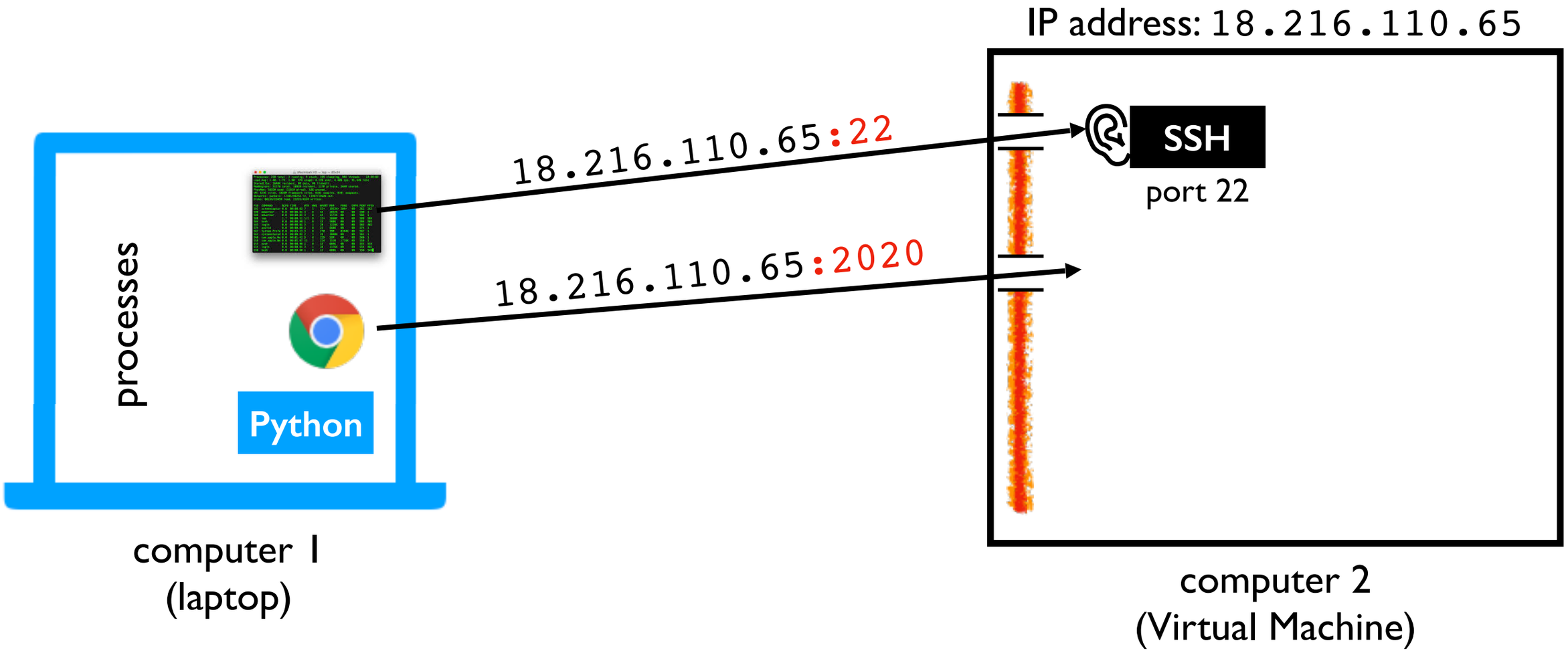

**Scenario:** we want to access Jupyter on our virtual machine from our laptop

# Getting Requests Through




**Issue 1:** firewall may be blocking some ports (we disabled this in lab)

# Getting Requests Through

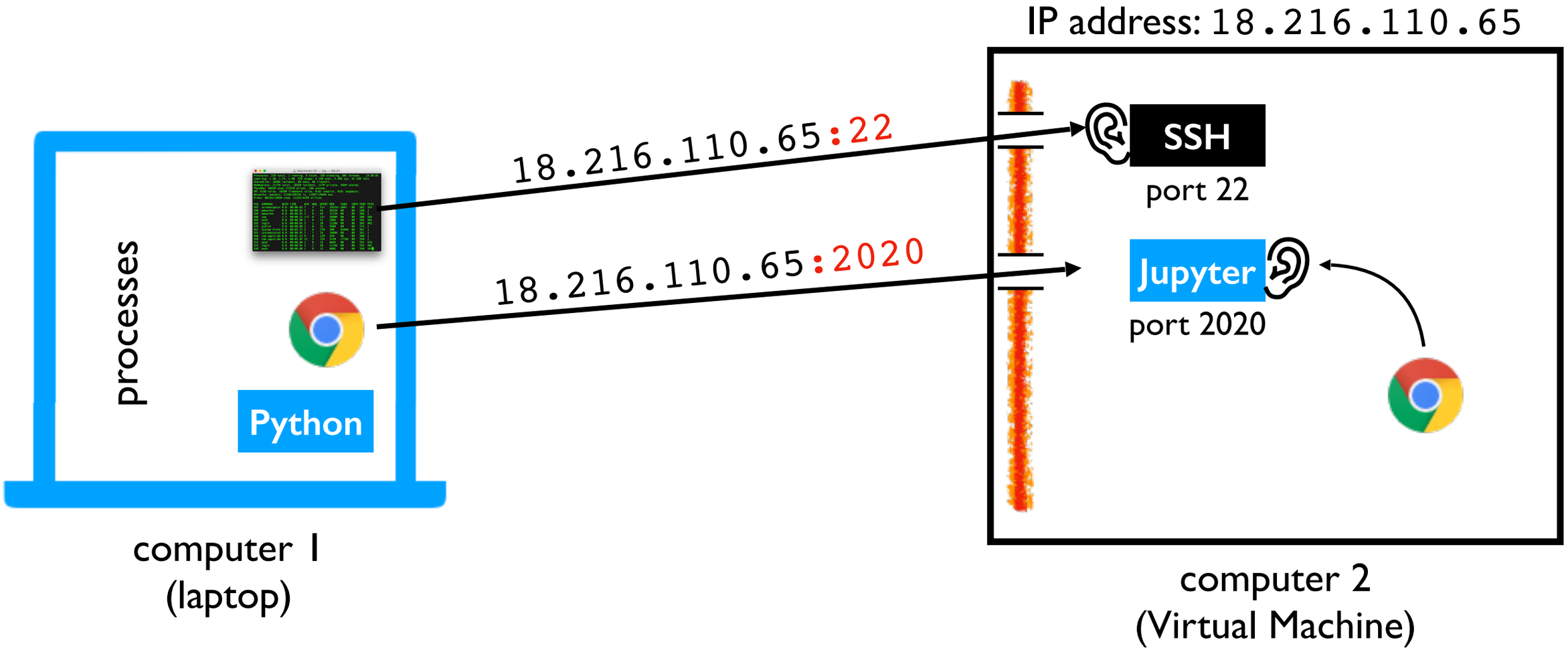



**Issue 2:** there might not be any process listening on port 2020

# Getting Requests Through

[127.0.0.1 means "localhost", the default]

**Start command:** python3 -m notebook --no-browser **--ip=127.0.0.1** --port=2020

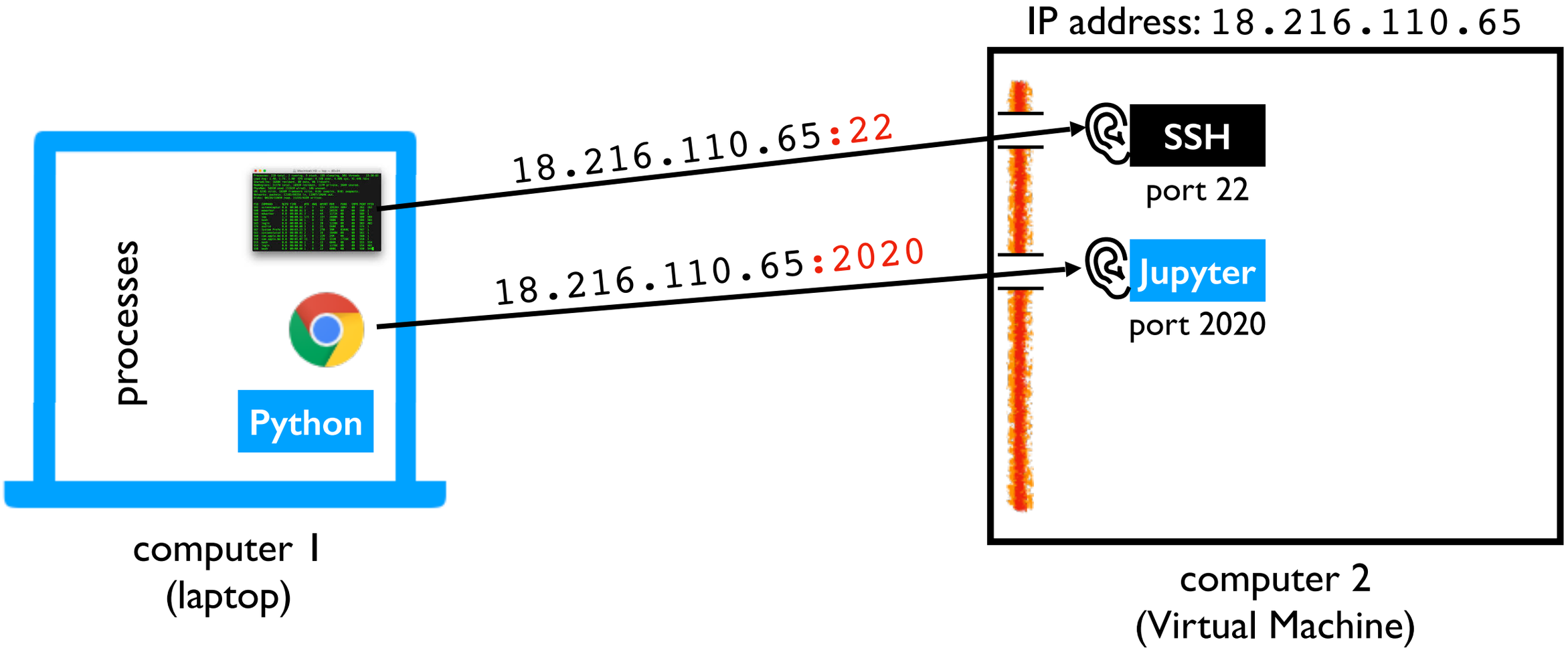



**Issue 3:** the process may only be listening for local (not external) requests

# Getting Requests Through

[0.0.0.0 means all IP addresses]

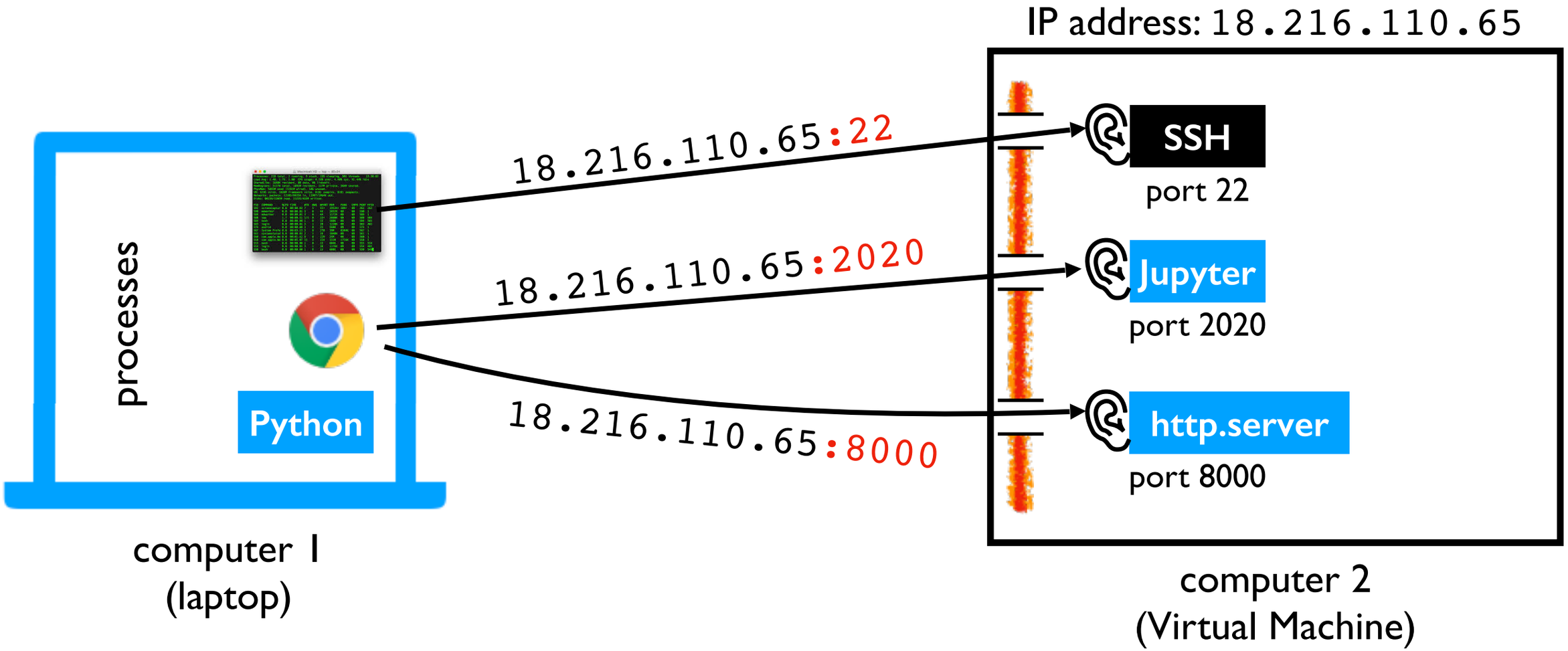
**Start command:** python3 -m notebook --no-browser **--ip=0.0.0.0** --port=2020




**Success:** Jupyter is listening for all 2020 requests, and the firewall isn't blocking them!

# Getting Requests Through

**Start command:** python3 -m notebook --no-browser **--ip=0.0.0.0** --port=2020

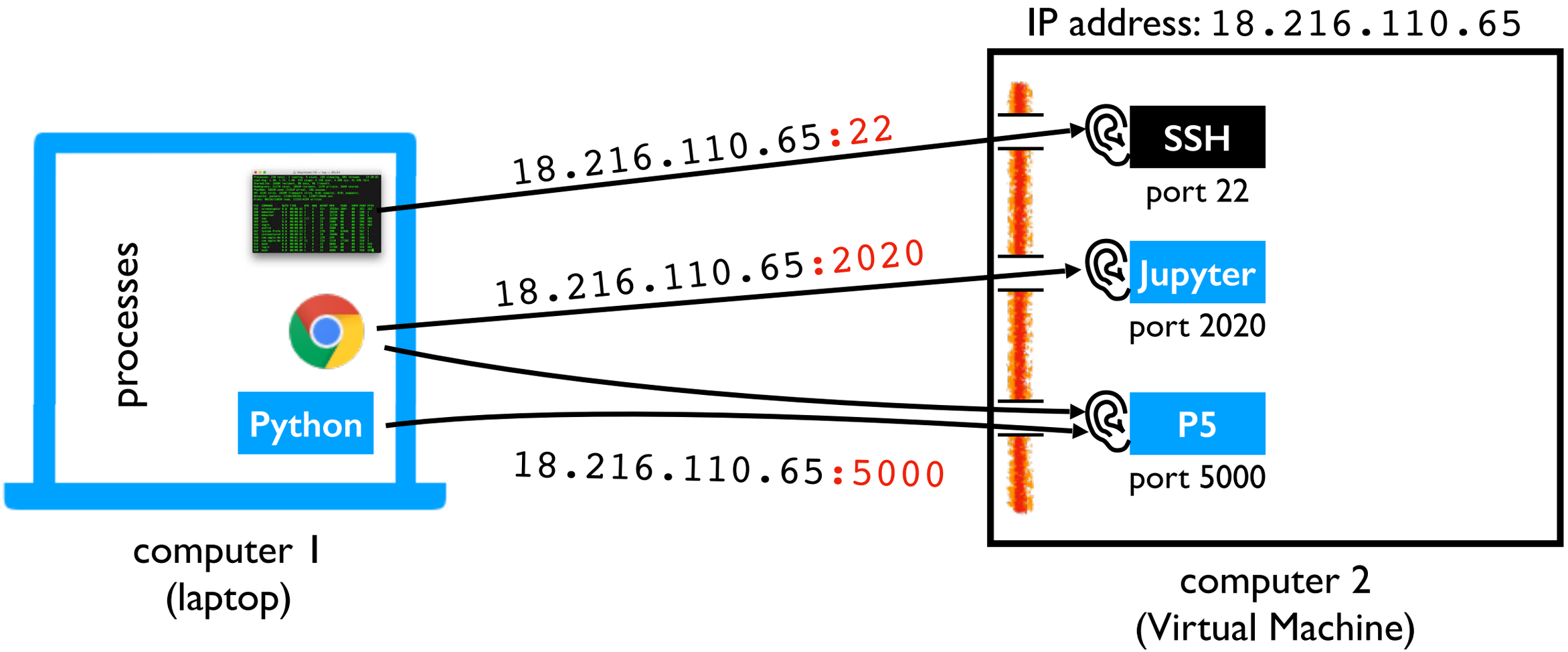



**Demo:** start web server with http.server

```
mkdir -p demo
cd demo
echo "<b>Hello</b> world!" > index.html
sudo python3 -m http.server --bind=0.0.0.0 8000
```

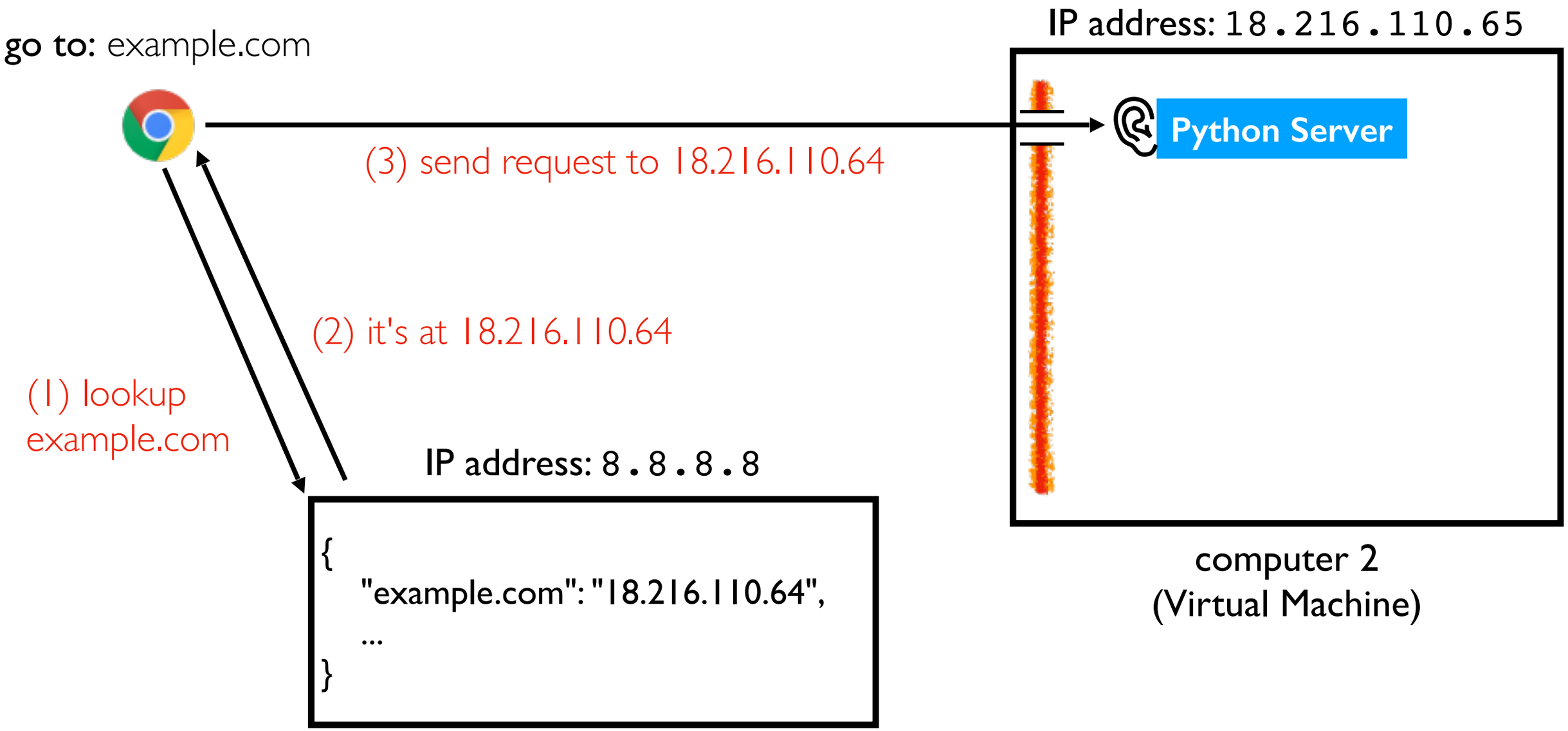
# Getting Requests Through

**Start command:** python3 -m notebook --no-browser **--ip=0.0.0.0** --port=2020



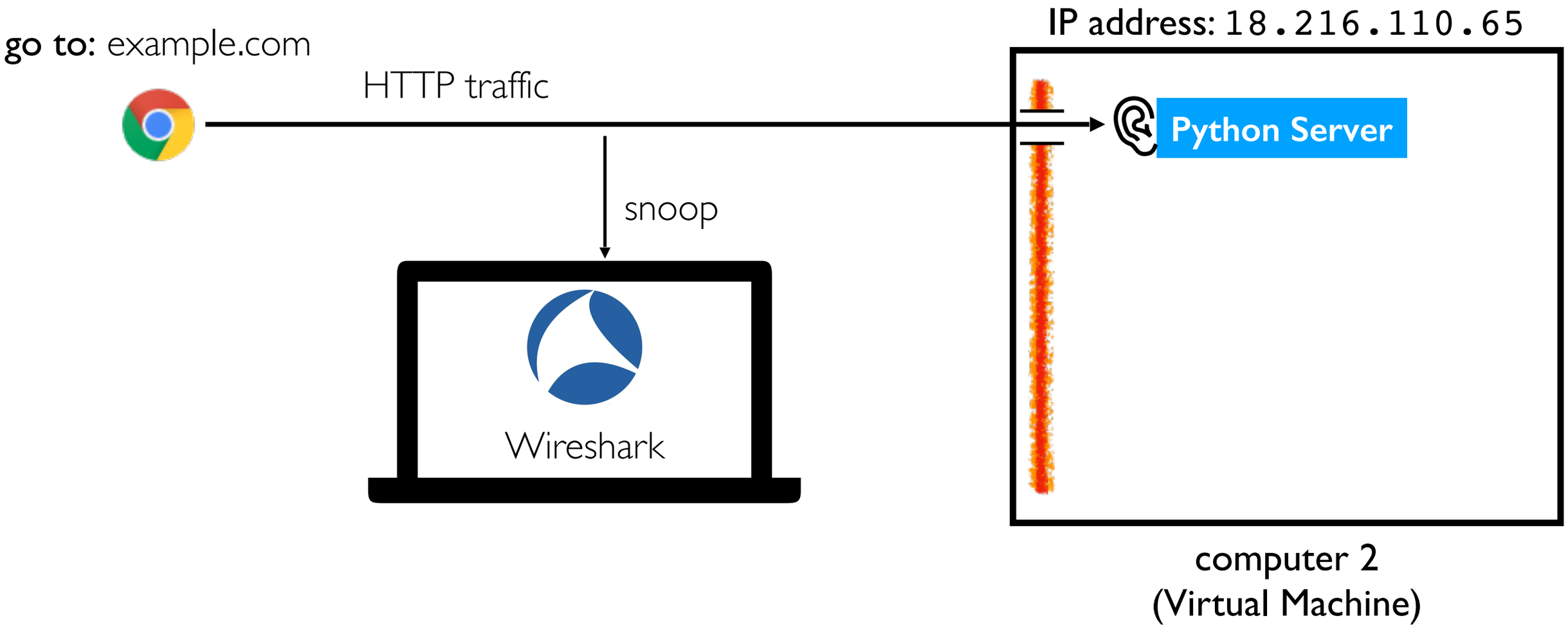

**Your Goal:** build a web application for P5

# DNS (Domain Name Service)

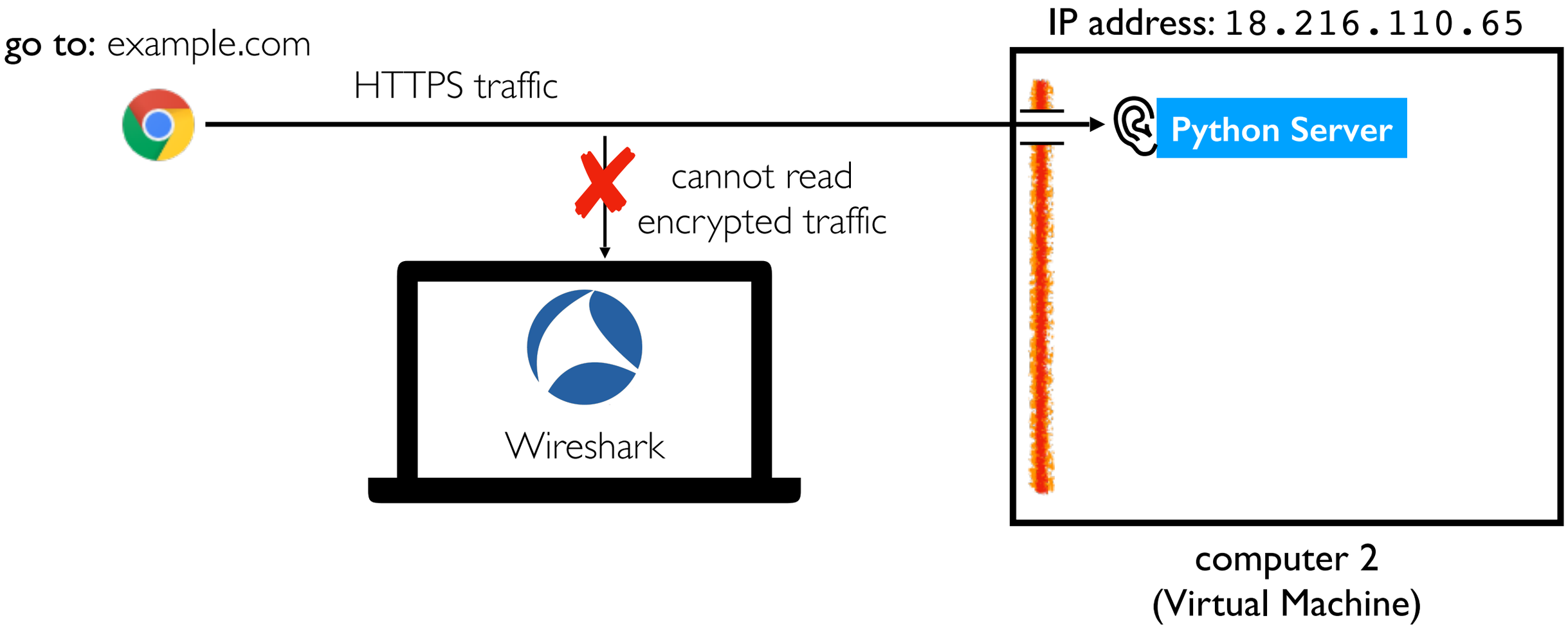



paying to register domain name is ~$10-15 / year

# HTTPS: Hypertext Transfer Protocol Secure



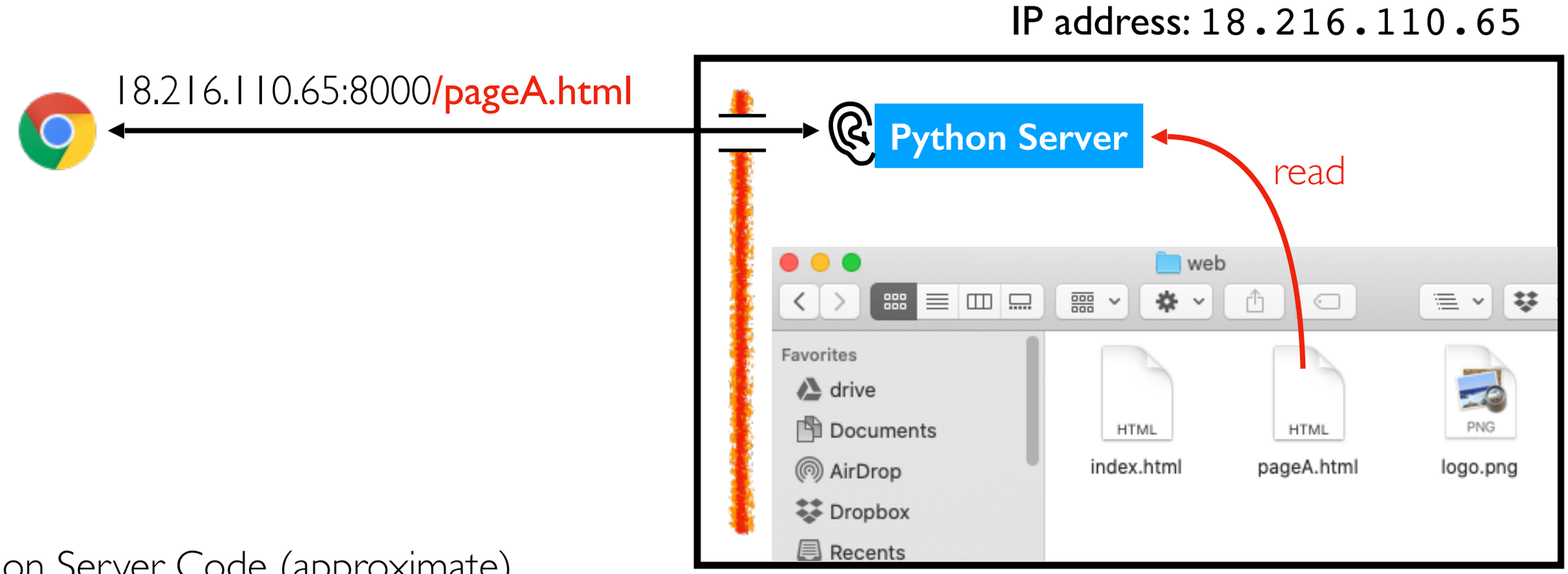
# HTTPS: Hypertext Transfer Protocol Secure




paying to register SSL certificate for encryption name is ~$5-10 / year
(or free: https://letsencrypt.org/)

# Pages vs. Files

# Static Pages Correspond to Files

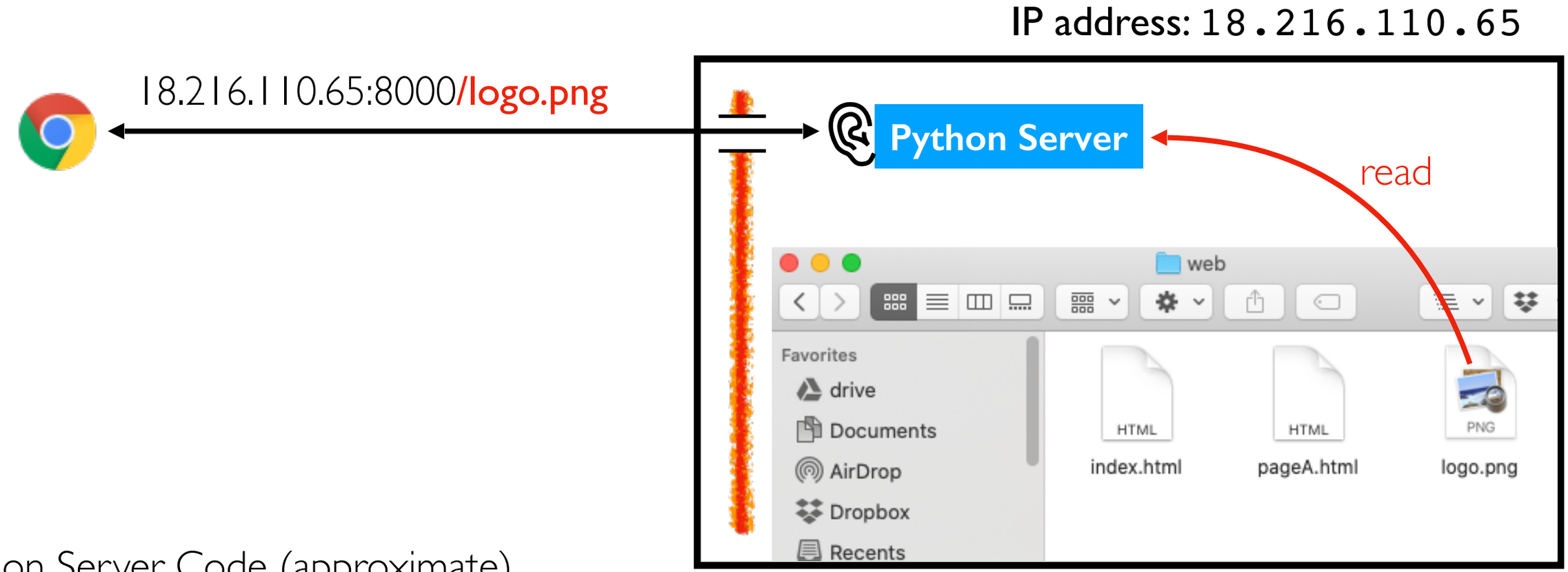



computer 2
(Virtual Machine)

Python Server Code (approximate)

```
def get_page(resource):
    with open(resource, "rb") as f:
        return f.read()
```

# Static Pages Correspond to Files

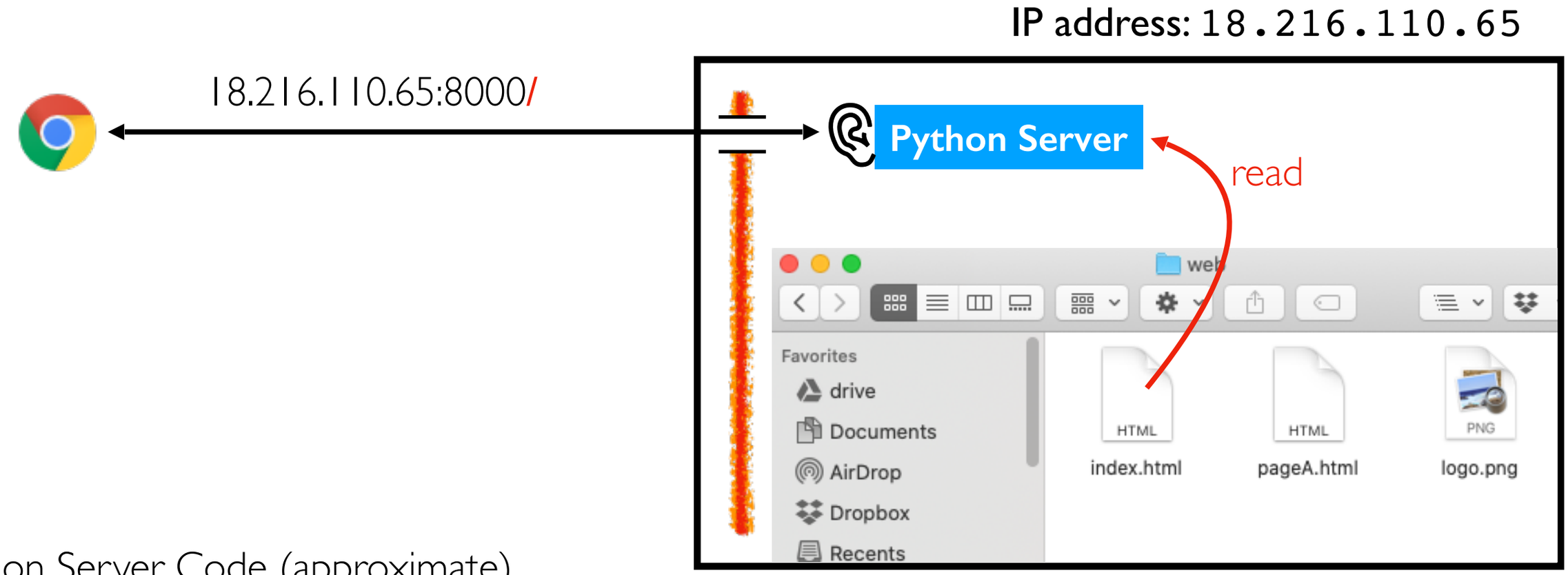



computer 2
(Virtual Machine)

Python Server Code (approximate)

```
def get_page(resource):
    with open(resource, "rb") as f:
        return f.read()
```

# Static Pages Correspond to Files



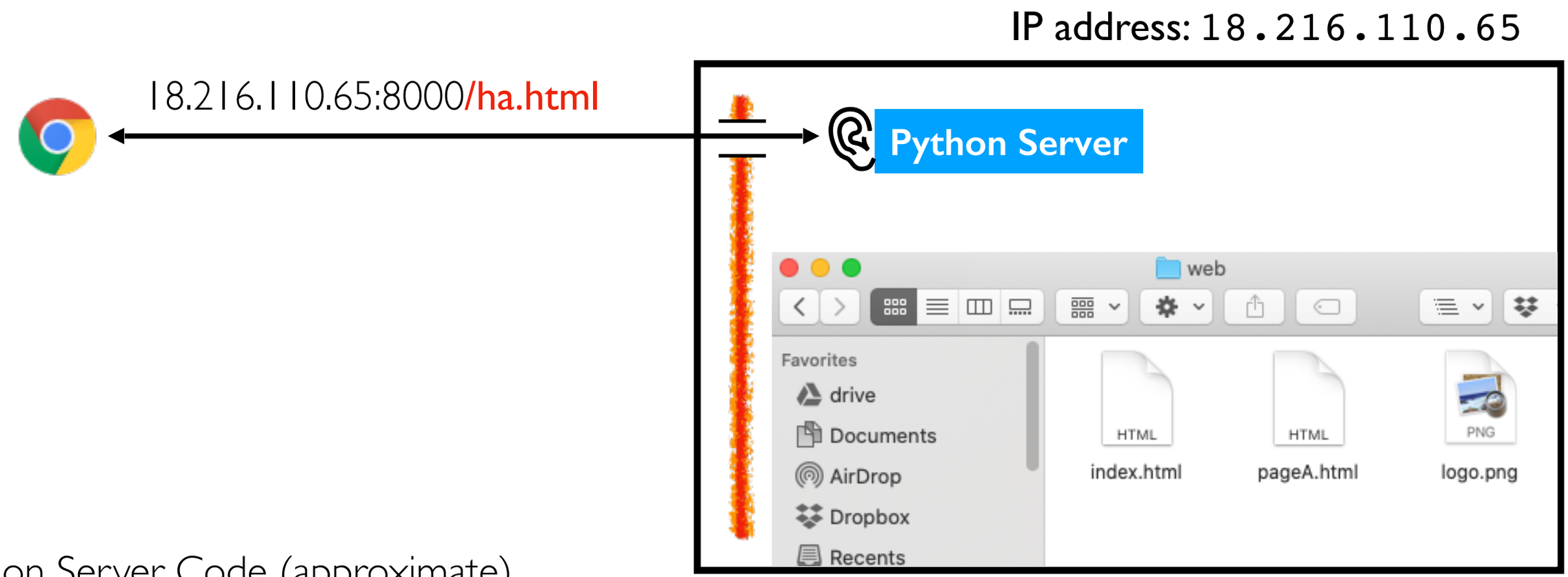

computer 2
(Virtual Machine)

Python Server Code (approximate)

```
def get_page(resource):
    if resource == "/":
        resource = "index.html"

    with open(resource, "rb") as f:
        return f.read()
```

# Dynamic Pages Generated by Code




computer 2
(Virtual Machine)

Python Server Code (approximate)

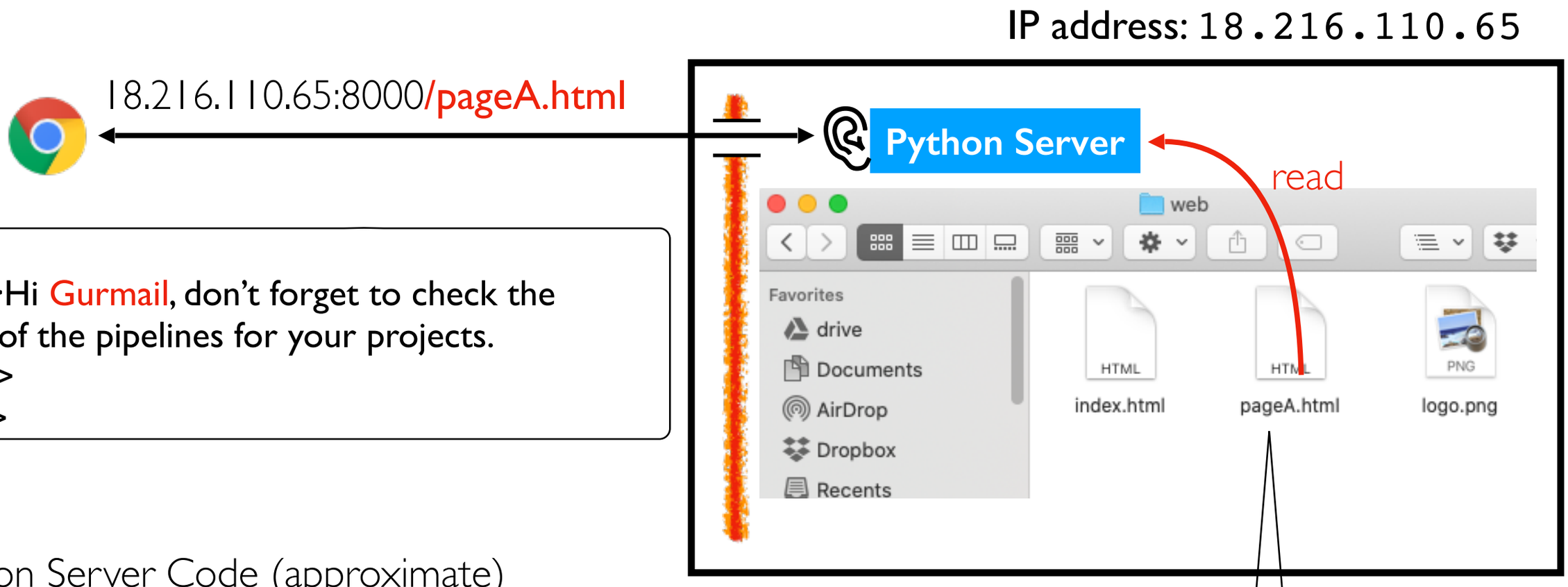
```
def get_page(resource):
    if resource == "/ha.html":
        return "<b>Ha!</b>" * 100

    with open(resource, "rb") as f:
        return f.read()
```

ha.html is dynamic

others are static

# Templating: Add Dynamic Content to File




```
<html>
<body>Hi Gurmail, don't forget to check the
output of the pipelines for your projects.
</body>
</html>
```

computer 2
(Virtual Machine)

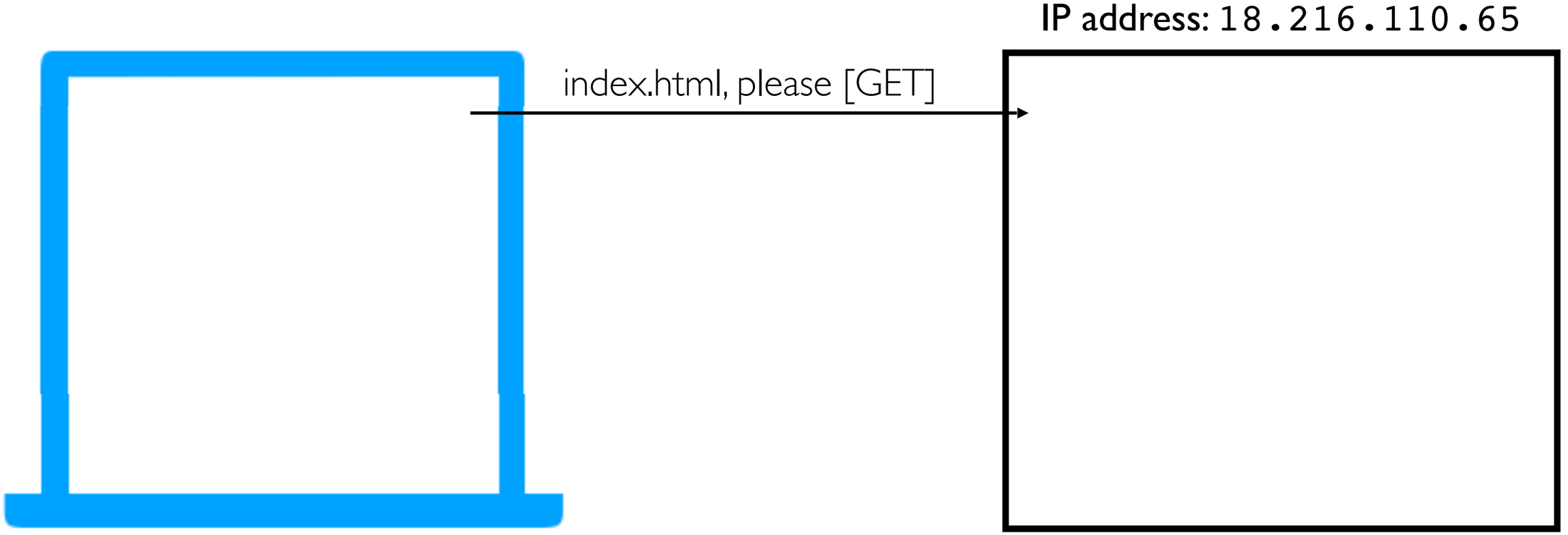
Python Server Code (approximate)

```
def get_page(resource):
    with open(resource, "rb") as f:
        s = f.read()
        if resource = "/pageA.html":
            s = s.format(user.first_name)
        return s
```

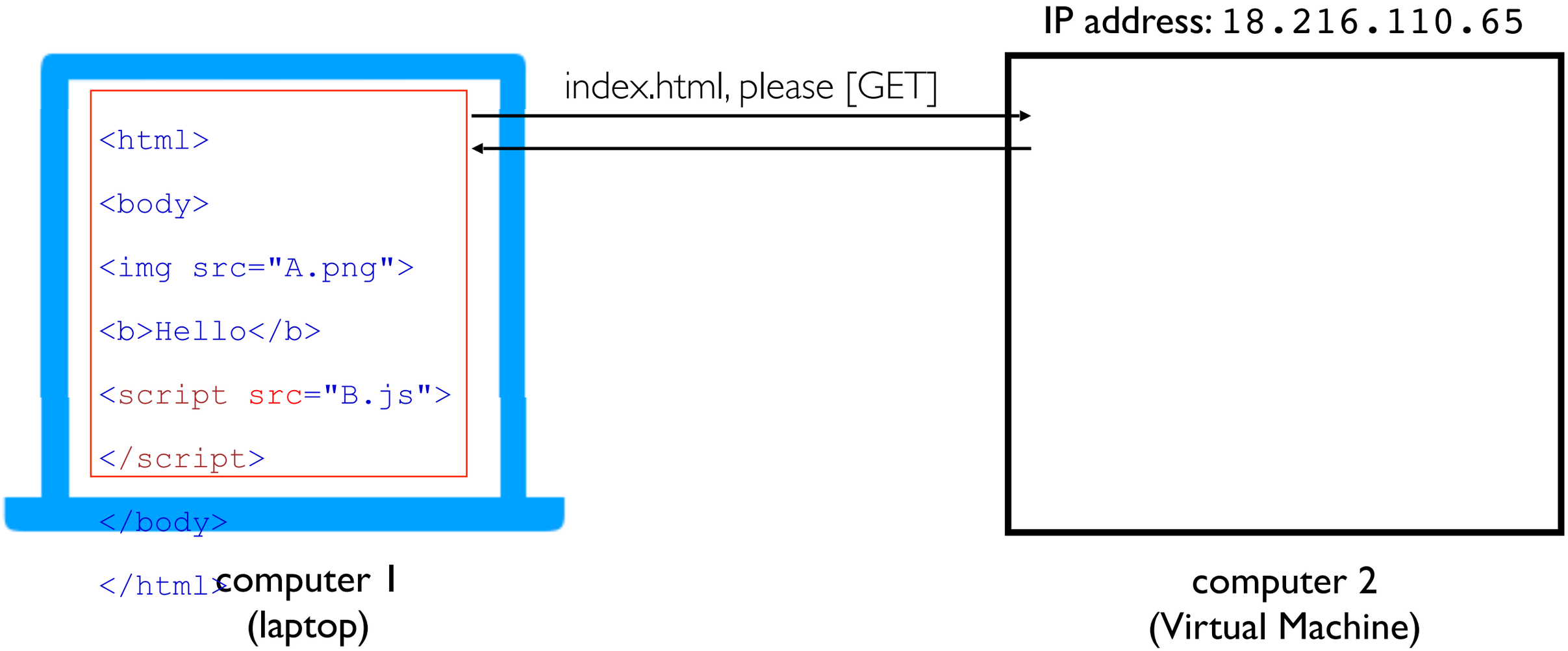
```
<html>
<body>Hi {}, don't forget to check the output of the
pipelines for your projects.
</body>
</html>
```

# Multi-File Pages

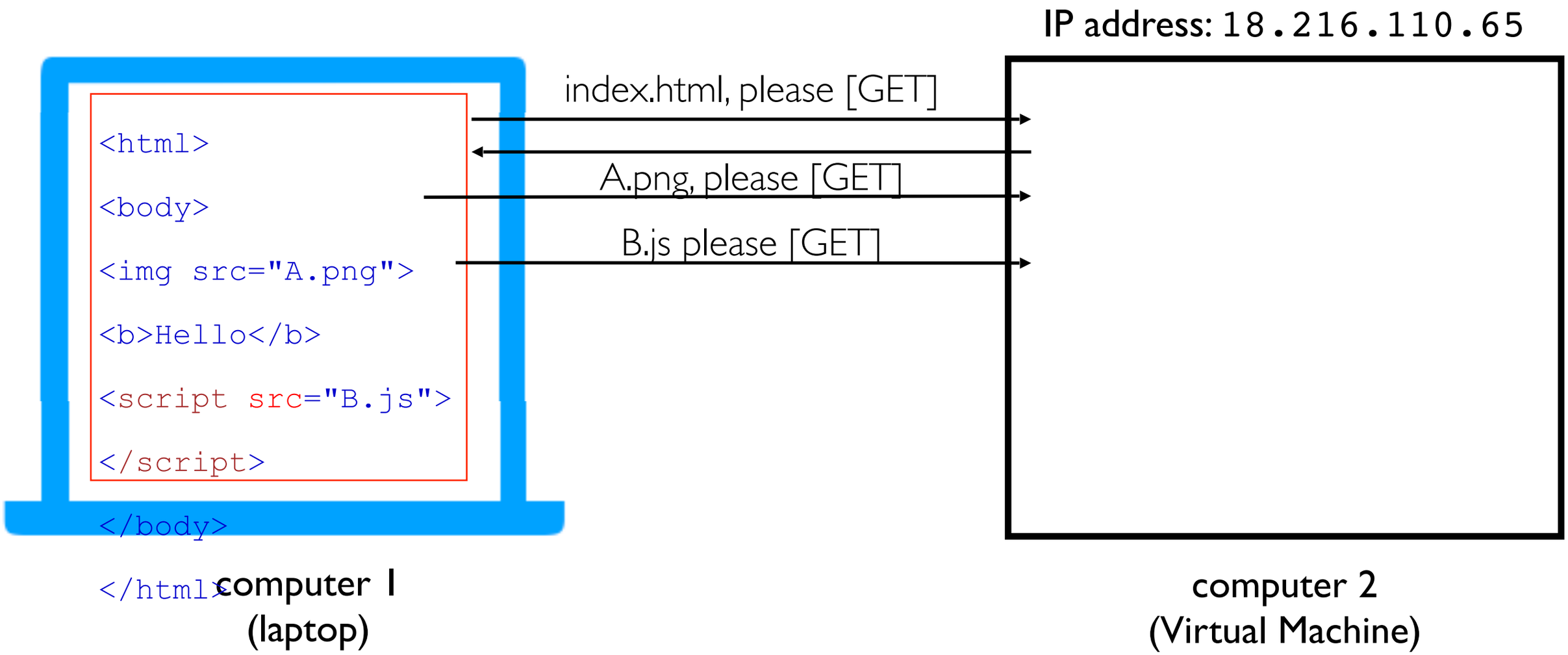
# Page Load, the Big Picture



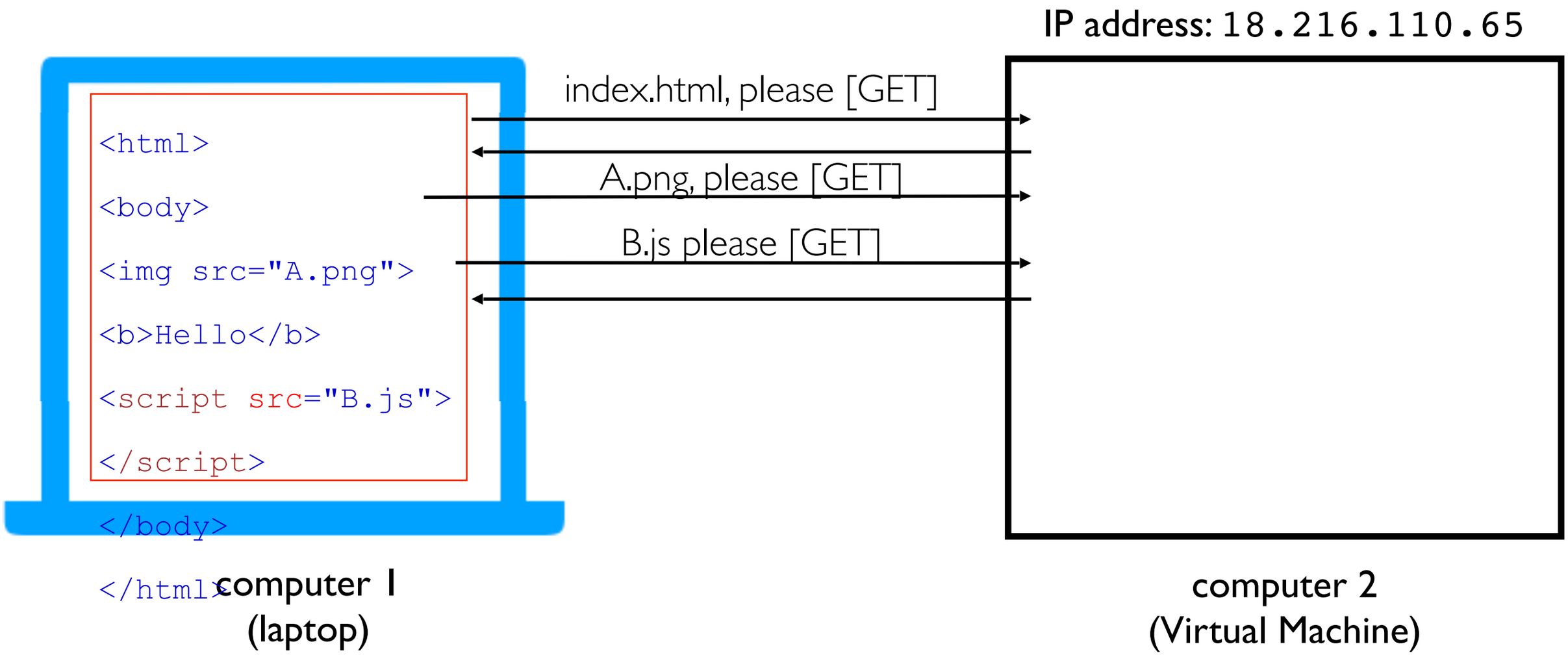
# Page Load, the Big Picture



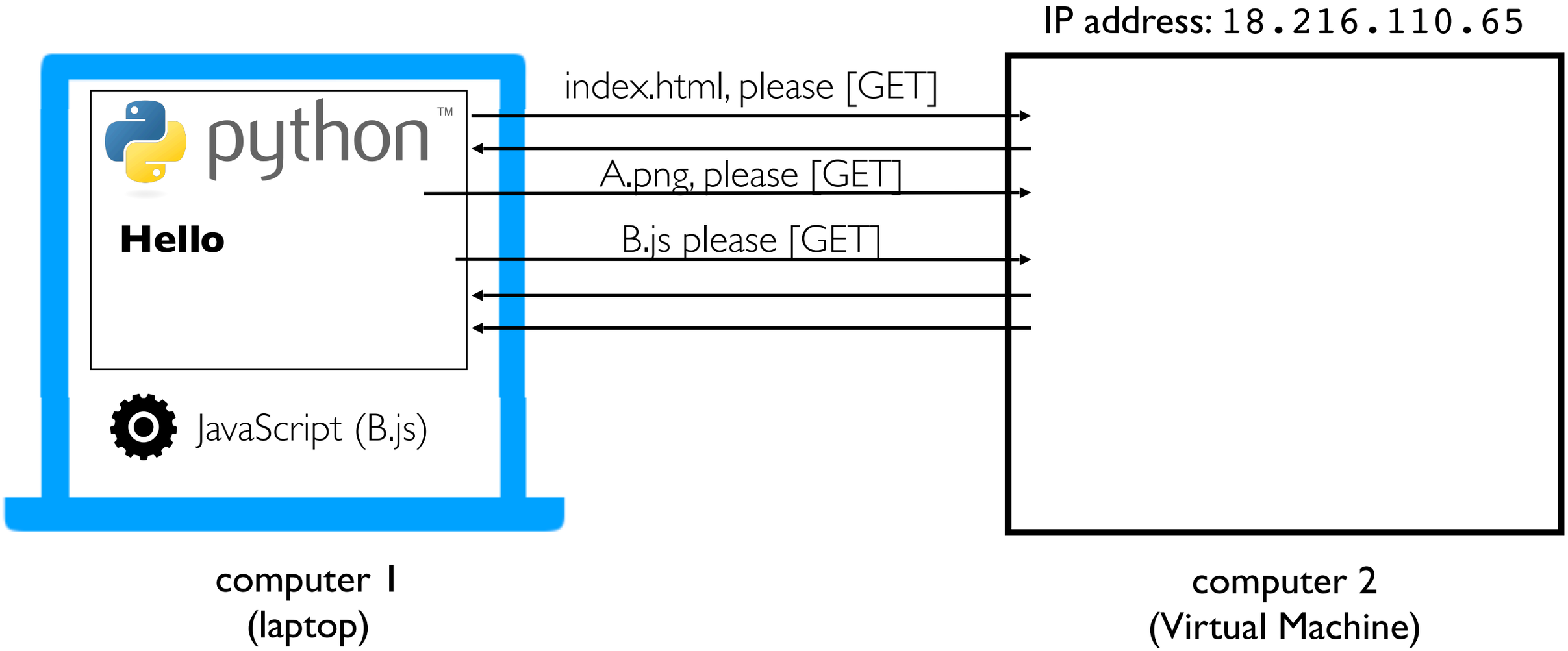
# Page Load, the Big Picture



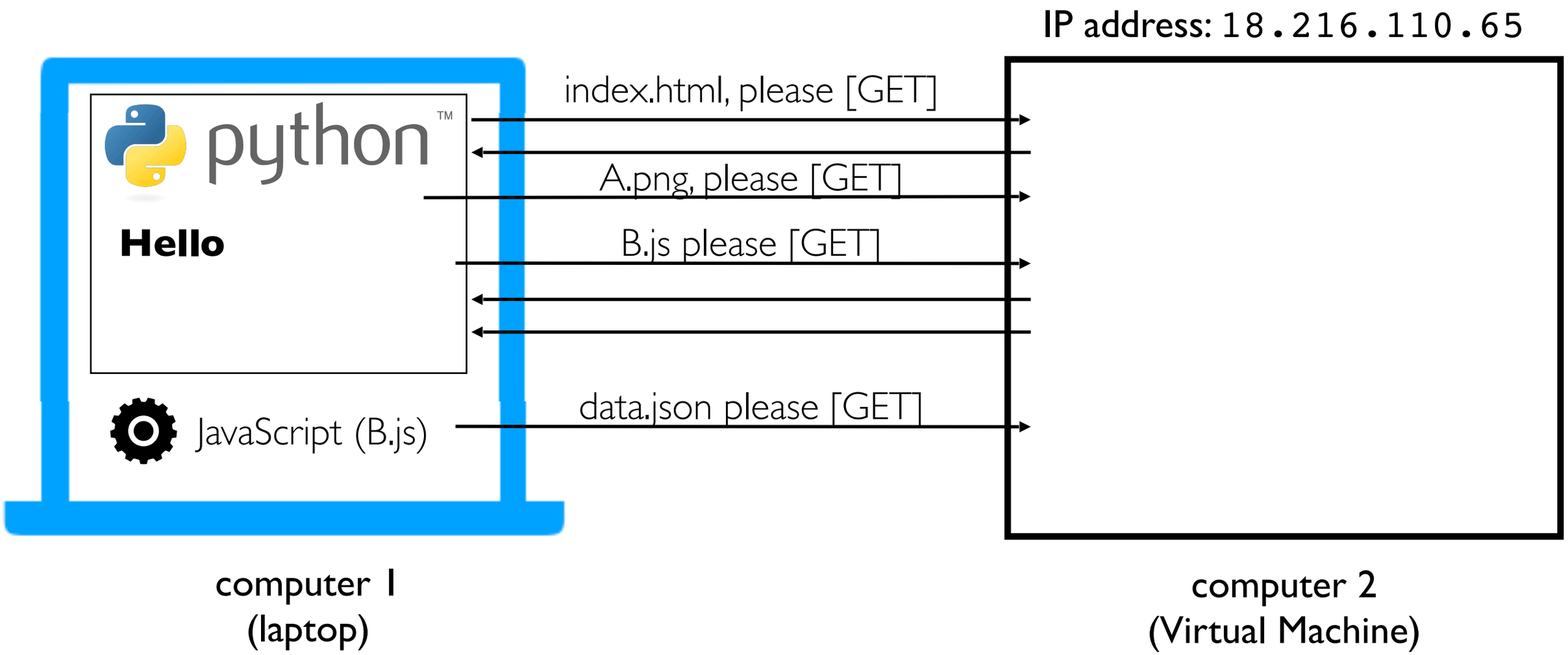
# Page Load, the Big Picture



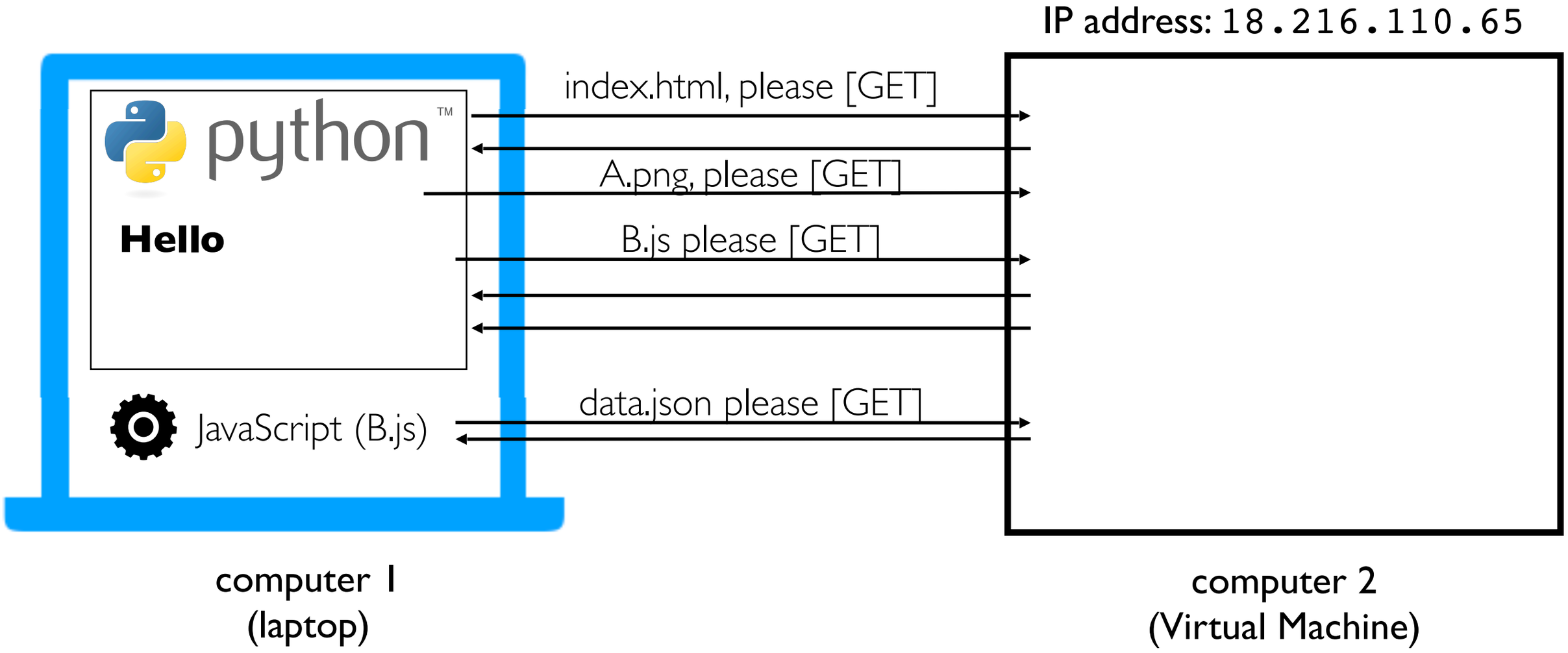
# Page Load, the Big Picture



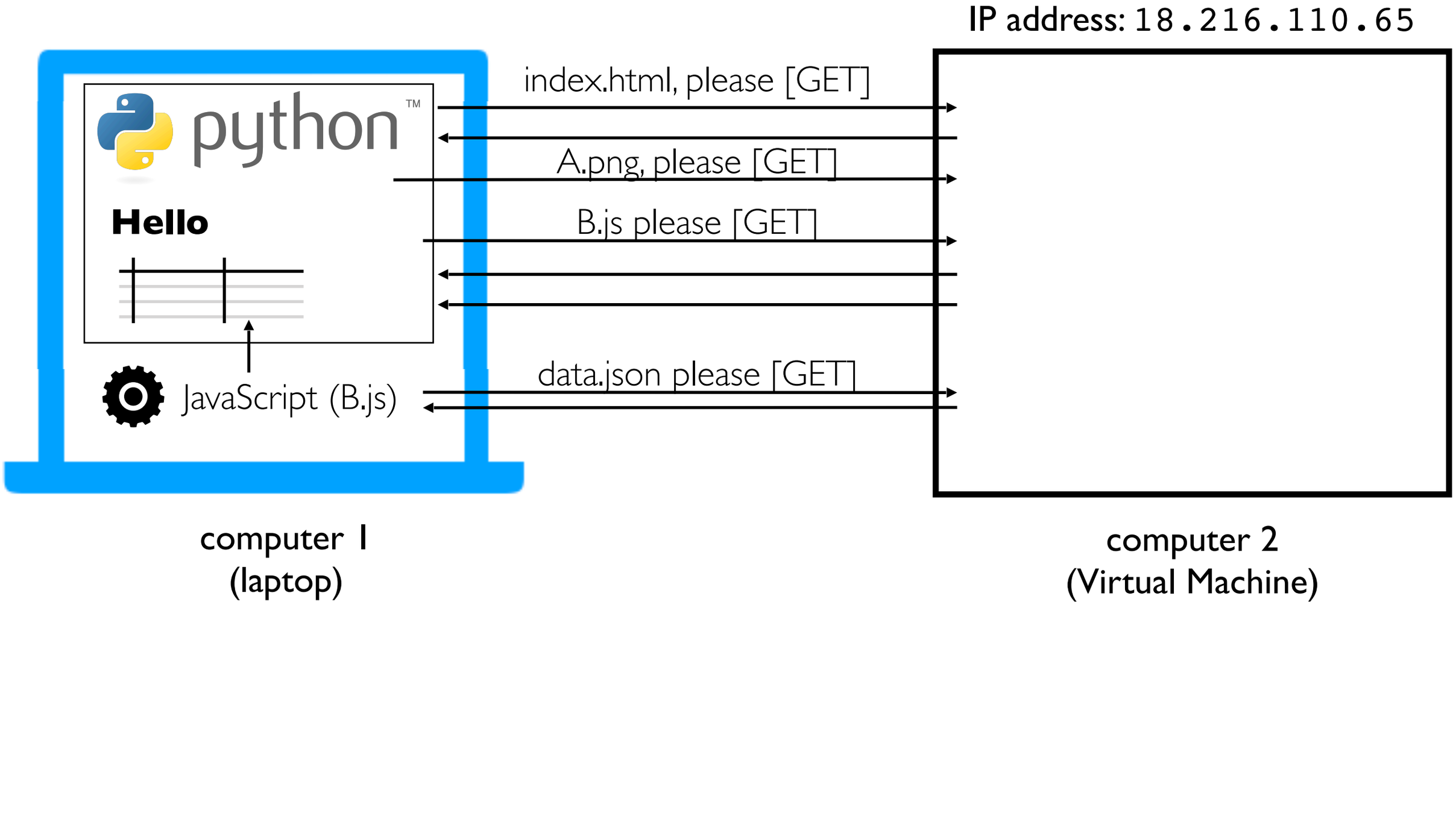
# Page Load, the Big Picture



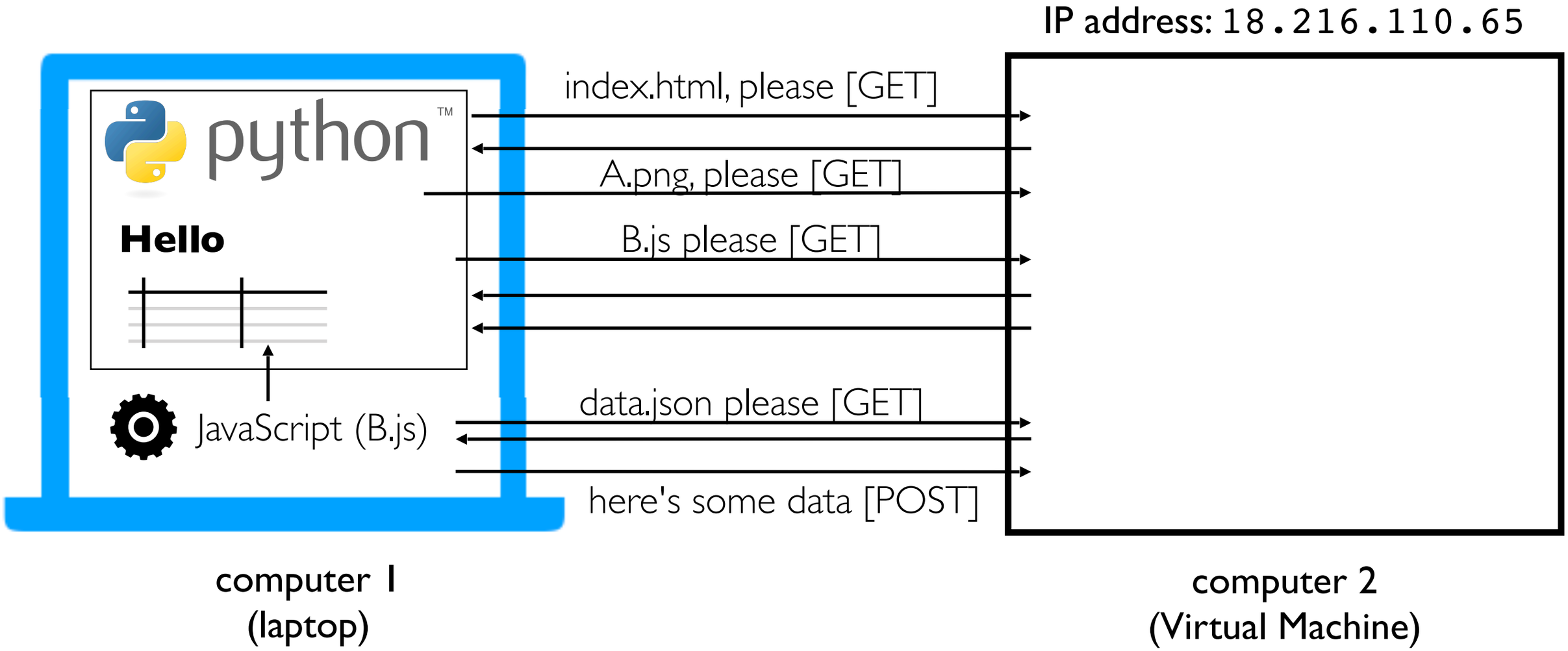
# Page Load, the Big Picture



# Page Load, the Big Picture



# Page Load, the Big Picture




Building a dashboard or similar app will involve many routes/requests

# Summary: Key Web Concepts

IP address: identifier for a computer (or network card on computer)

port number: identifier used to route to specific process on computer

firewall: software to block certain requests, often for certain ports

listening: process is ready to receive requests from an IP/port

DNS: service for converting domains to IP addresses

HTTPS: encrypted HTTP traffic so others can't watch traffic on WIFI, etc

static pages: pages that correspond to files on the server

dynamic pages: pages generated on-the-fly by some Python code

templating: insert dynamic content into certain places in a file

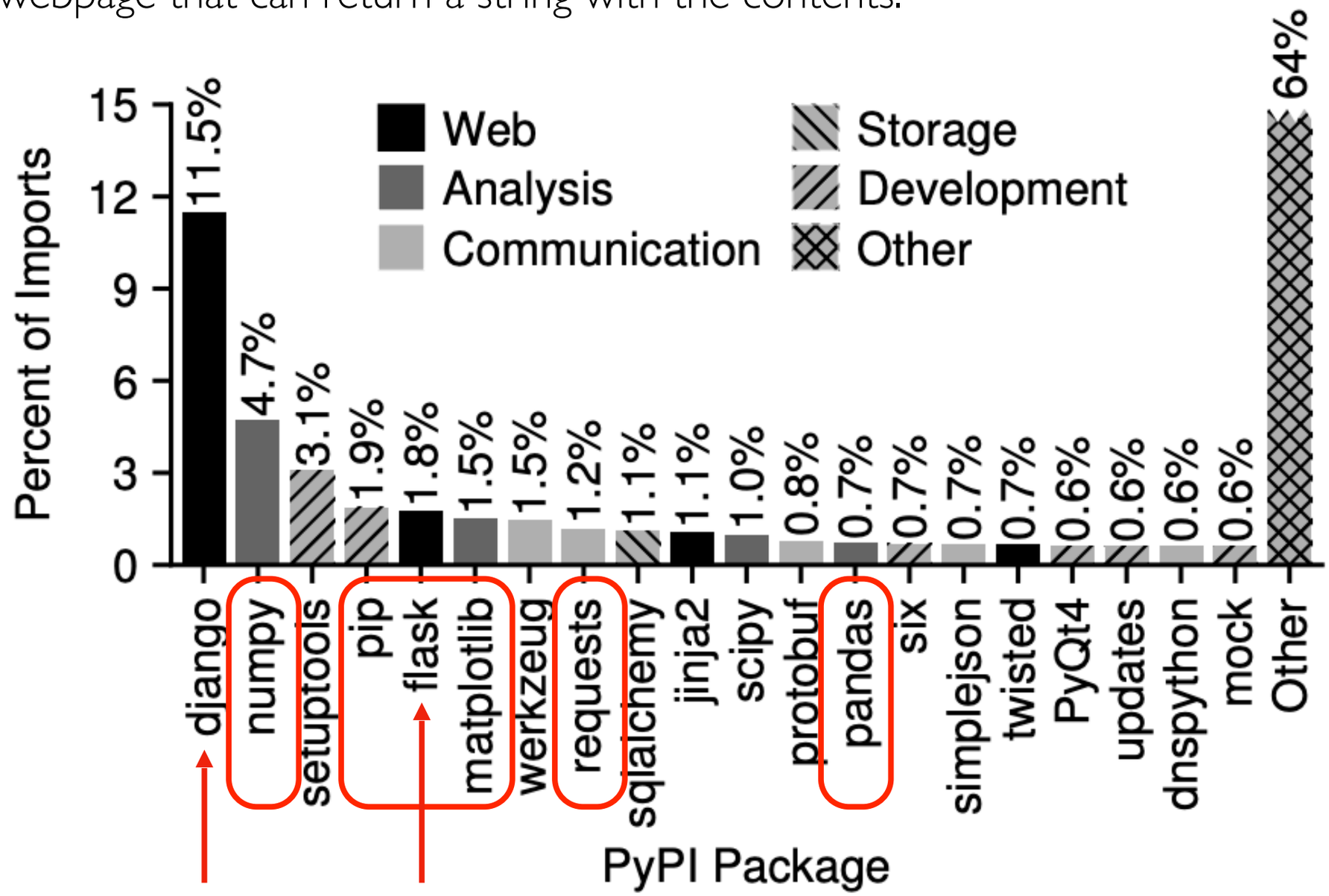
HTTP GET: request to download data

HTTP POST: request to upload data

# Web Frameworks

# Python Web Frameworks (and other packages)

Python web frameworks like Flask and Django make it easy to write functions for each webpage that can return a string with the contents.




we'll use **Flask**
for CS 320 because
it is simpler than **Django**

https://www.usenix.org/system/files/conference/atc18/atc18-oakes.pdf

# Flask Example

Start from lecture snippets