The Internet

Thanks to Mehran Sahami and Chris Piech for today’s slides!

Housekeeping

- Assignment #6 has been released!

- Remaining lecture schedule
  - Friday: last content lecture!
  - Tuesday: Life after 106A + AMA
  - Wednesday: Final review
  - Thursday: no lecture, OH during lecture time

- Acknowledging end-of-quarter stress
  - Take care of yourselves and each other
  - Be proud of what you have accomplished!

Today

- Review classes
  - Very briefly

- Let's talk about the internet!
  - How the internet works at a high level
  - Learn how we can use classes to implement a key part of it (make a server!)

<review>

Classes Review

Dog.py

```
class Dog:
    def __init__(self):
        self.times_barked = 0

    def bark():
        print('woof')
        self.times_barked += 1
```

life.py

```
def main():
    simba = Dog()
    juno = Dog()

    simba.bark()
    juno.bark()
    simba.bark()

    print(simba.__dict__)
    print(juno.__dict__)
```

Classes Review

Dog.py

```
class Dog:
    def __init__(self):
        self.times_barked = 0
    def bark():
        print('woof')
        self.times_barked += 1
```

life.py

```
def main():
    simba = Dog()
    juno = Dog()
    simba.bark()
    juno.bark()
    simba.bark()
    print(simba.__dict__)
    print(juno.__dict__)
```

1. What happens when you make a new one?

Classes Review

Dog.py

```
class Dog:
    def __init__(self):
        self.times_barked = 0
    def bark():
        print('woof')
        self.times_barked += 1
```

life.py

```
def main():
    simba = Dog()
    juno = Dog()
    simba.bark()
    juno.bark()
    simba.bark()
    print(simba.__dict__)
    print(juno.__dict__)
```

2. What **variables** does each instance store?

Classes Review

Dog.py

```
class Dog:
    def __init__(self):
        self.times_barked = 0
    def bark():
        print('woof')
        self.times_barked += 1
```

life.py

```
def main():
    simba = Dog()
    juno = Dog()
    simba.bark()
    juno.bark()
    simba.bark()
    print(simba.__dict__)
    print(juno.__dict__)
```

2. What **methods** can you call on an instance?

Classes Review

Dog.py

```
class Dog:
    def __init__(self):
        self.times_barked = 0
    def bark():
        print('woof')
        self.times_barked += 1
```

life.py

```
def main():
    simba = Dog()
    juno = Dog()
    simba.bark()
    juno.bark()
    simba.bark()
    print(simba.__dict__)
    print(juno.__dict__)
```

Terminal:

{ times_barked : 2 }

{ times_barked : 1 }

Did I mention that a class is like a fancy dictionary?

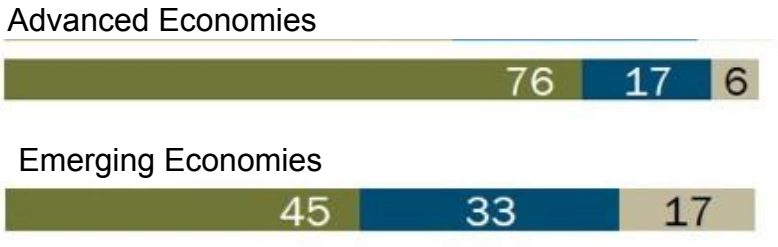
Classes define new variable *types*

Classes decompose your program across files

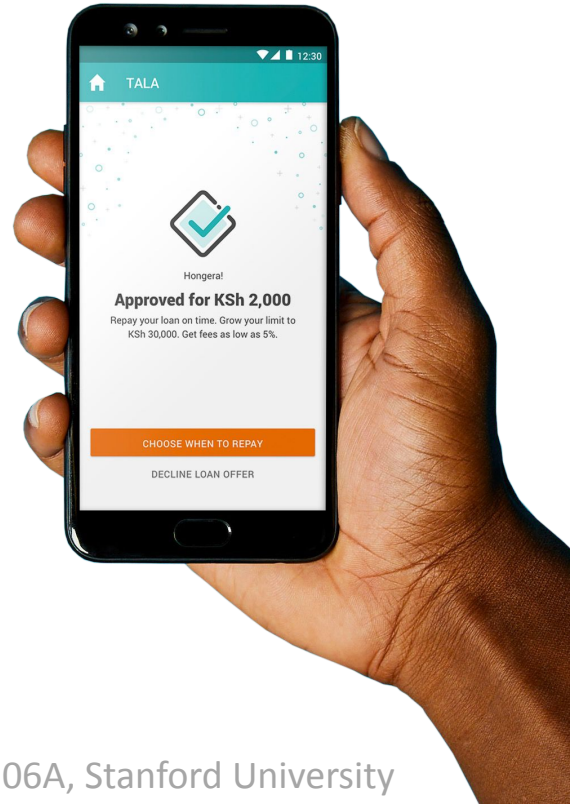
</ review>

One reason programming is fun is because of the internet...

Smart Phone Access

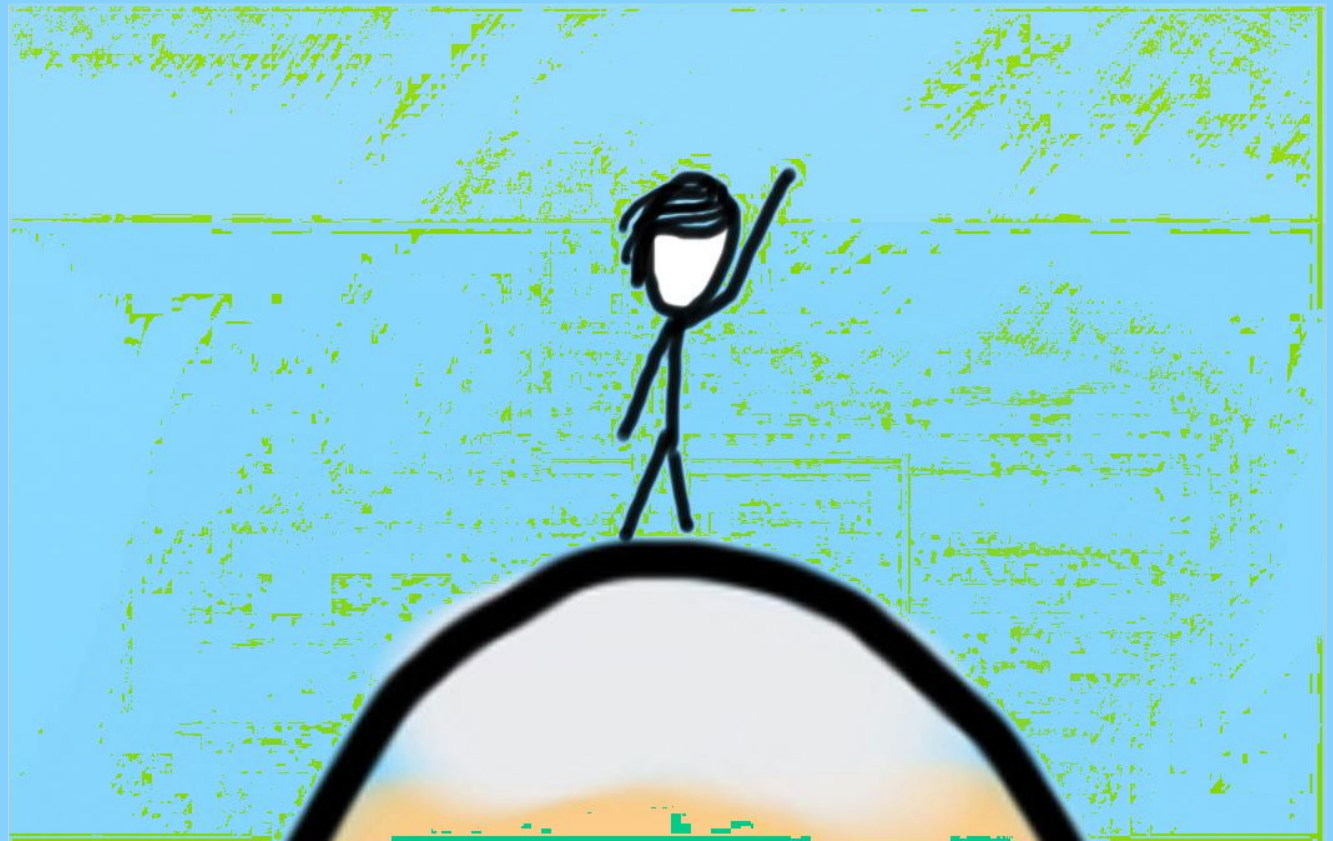


Smartphone
Mobile
No phone



Learning Goals

1. Write a program that can respond to internet requests



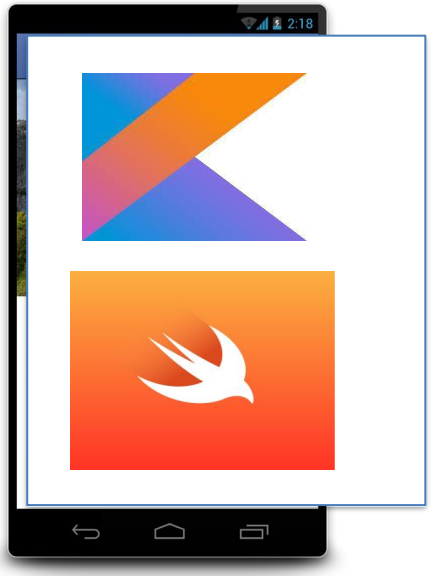
How does your phone communicate with Facebook?

The program on your **phone**
talks to the program at
Facebook

JavaScript with HTML are the languages of websites

Kotlin is the language of Android phones

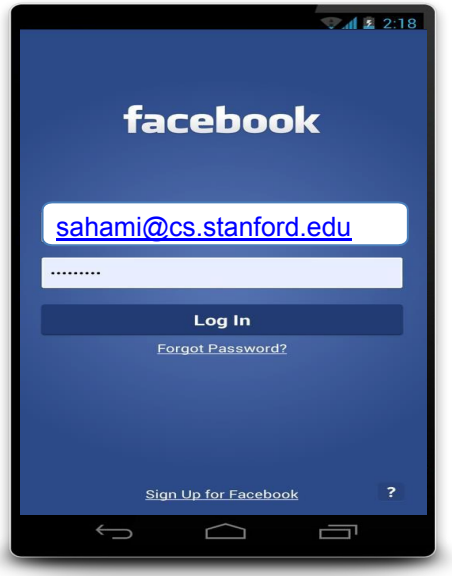
Swift is the language of Apple phones



Facebook Server

Facebook Server

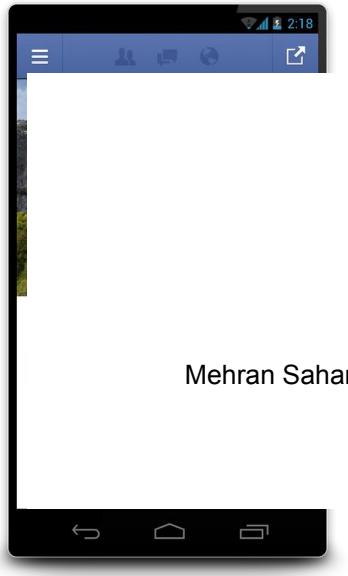
Is this
authenticated login?



sahami@cs.stanford.edu
is now logged in

Facebook Server

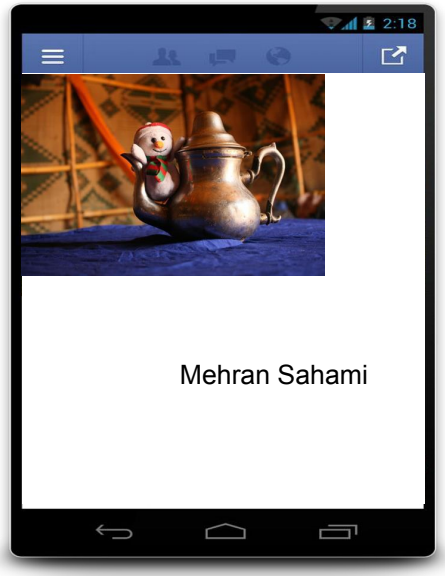
Send me the full name for sahami@cs.stanford.edu



"Mehran Sahami"

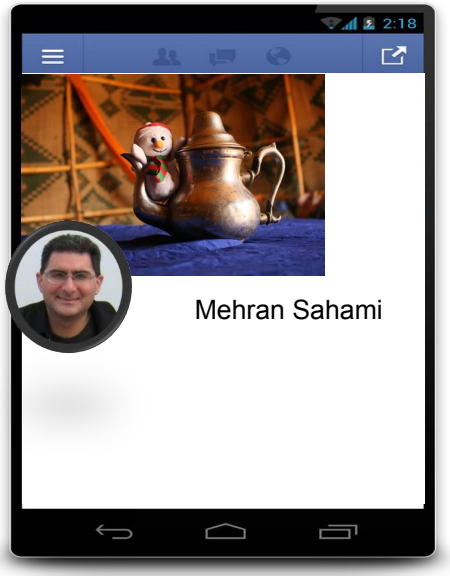
Send me the cover photo for sahami@cs.stanford.edu

Facebook Server



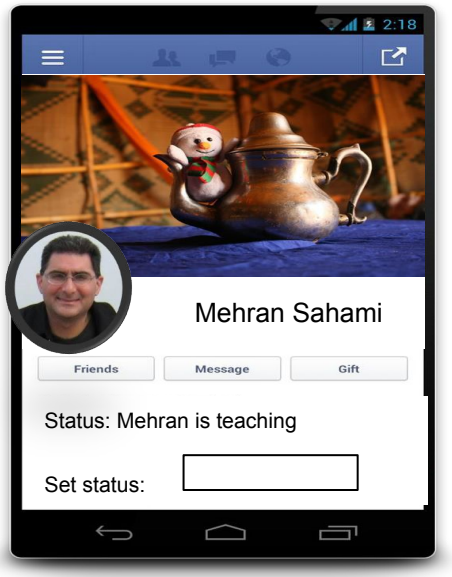
Facebook Server

Send the profile photo for sahami@cs.stanford.edu



Facebook Server

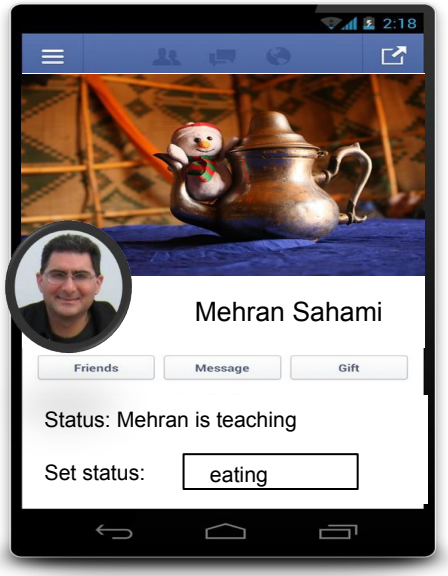
Send the status for sahami@cs.stanford.edu



"teaching"

Set the status for sahami@cs.stanford.edu to be "eating"

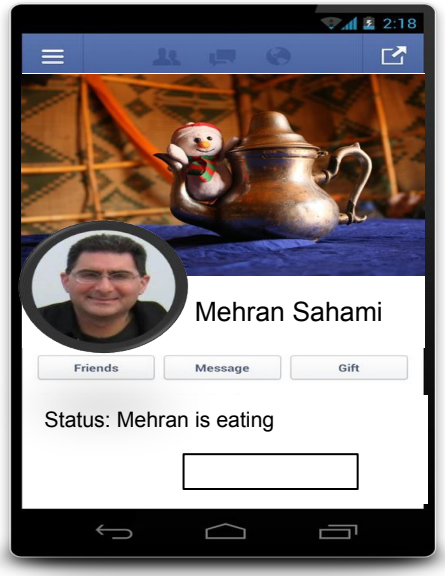
Facebook Server



"success"

Facebook Server

Send me the status for sahami@cs.stanford.edu



"eating"

Background: The Internet

The internet is just many programs sending messages (as *Strings*)

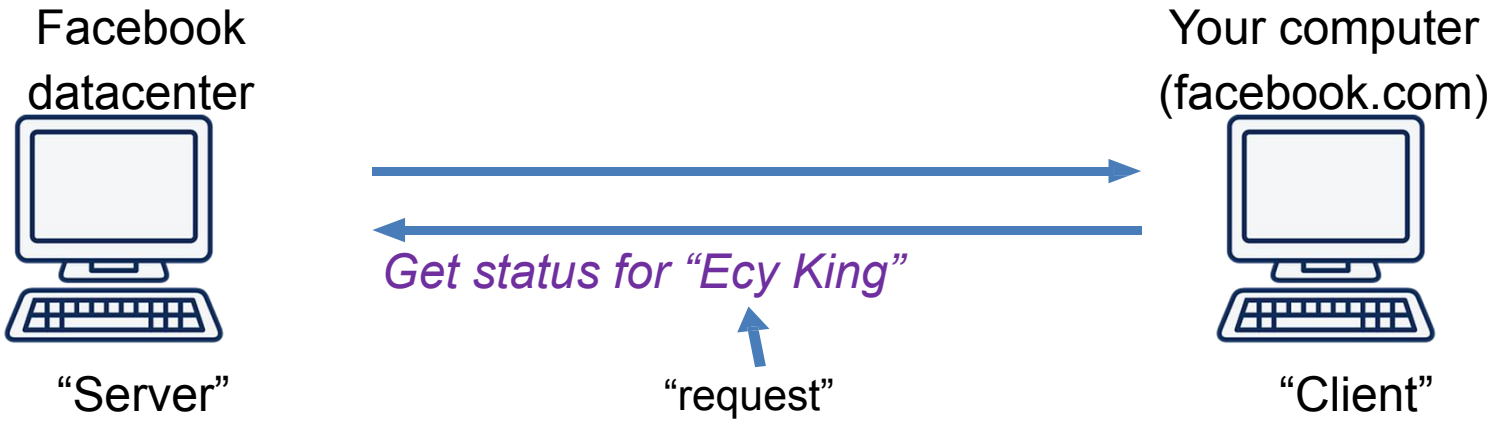
Background: The Internet

The internet is just many programs sending messages (as *Strings*)

Background: The Internet

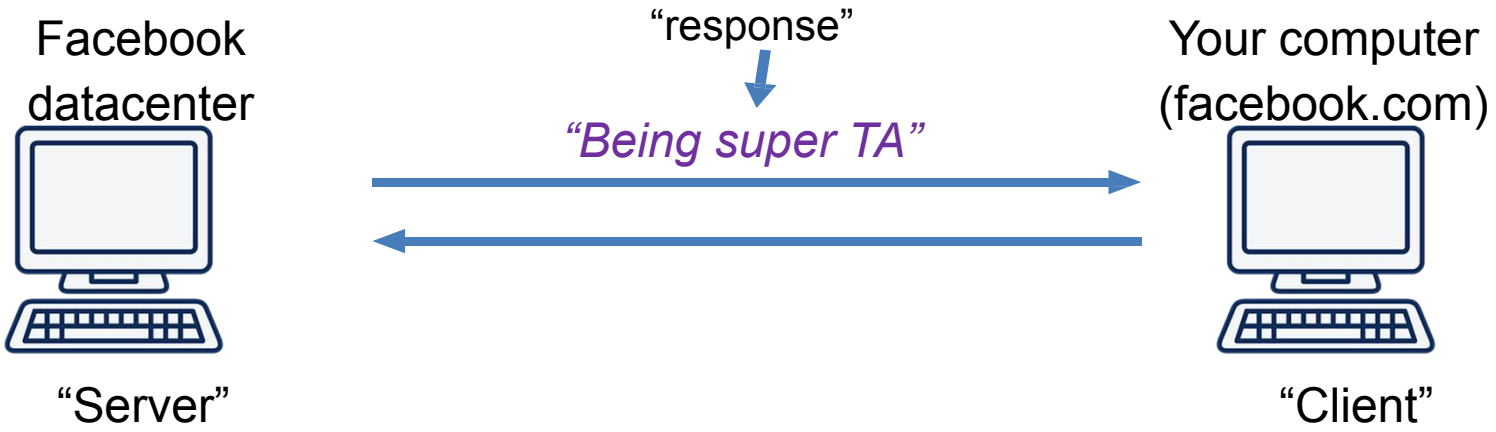
The internet is just many programs sending messages (as *Strings*)

Background: The Internet



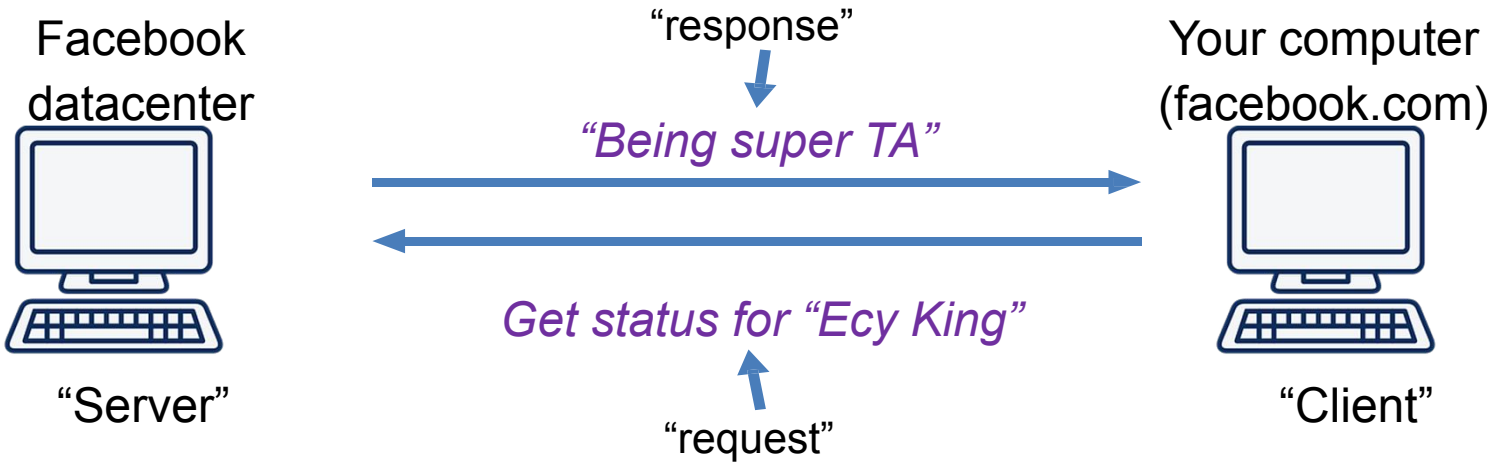
The internet is just many programs sending messages (as *Strings*)

Background: The Internet



The internet is just many programs sending messages (as *Strings*)

Background: The Internet

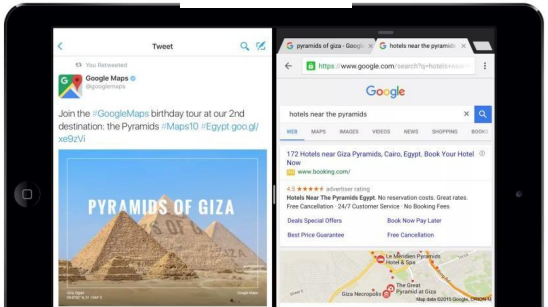


The internet is just many programs sending messages (as *Strings*)

There are (generally) two types of internet programs: Servers and Clients

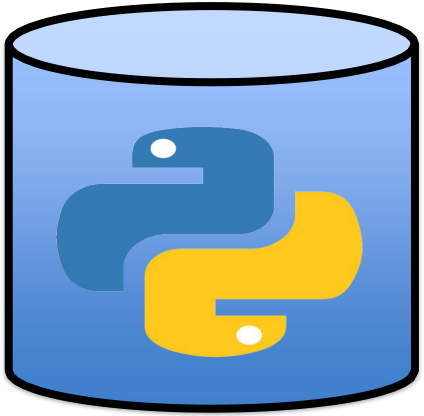
Internet 101

Computers on the internet

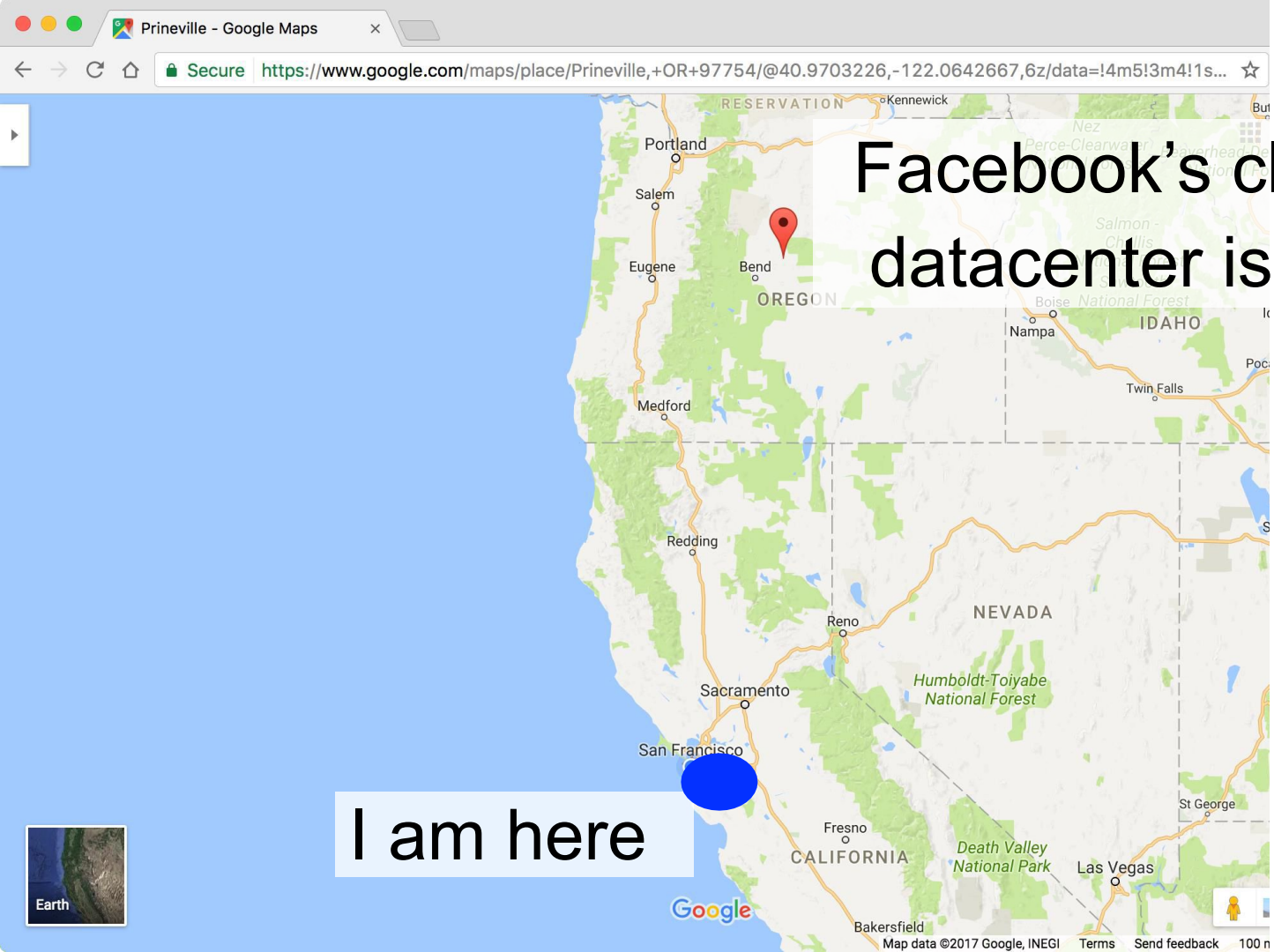


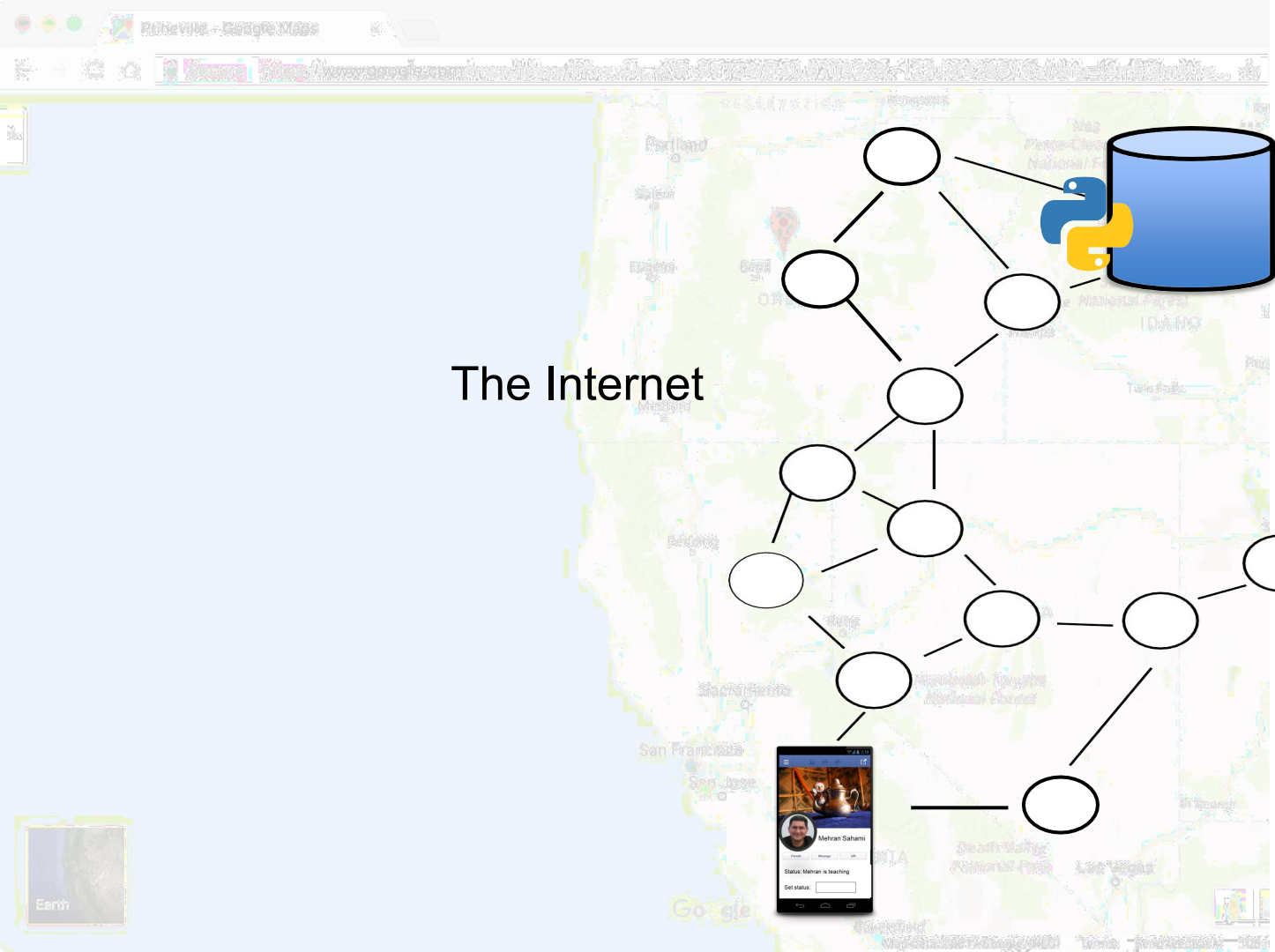
Servers are computers (running code)

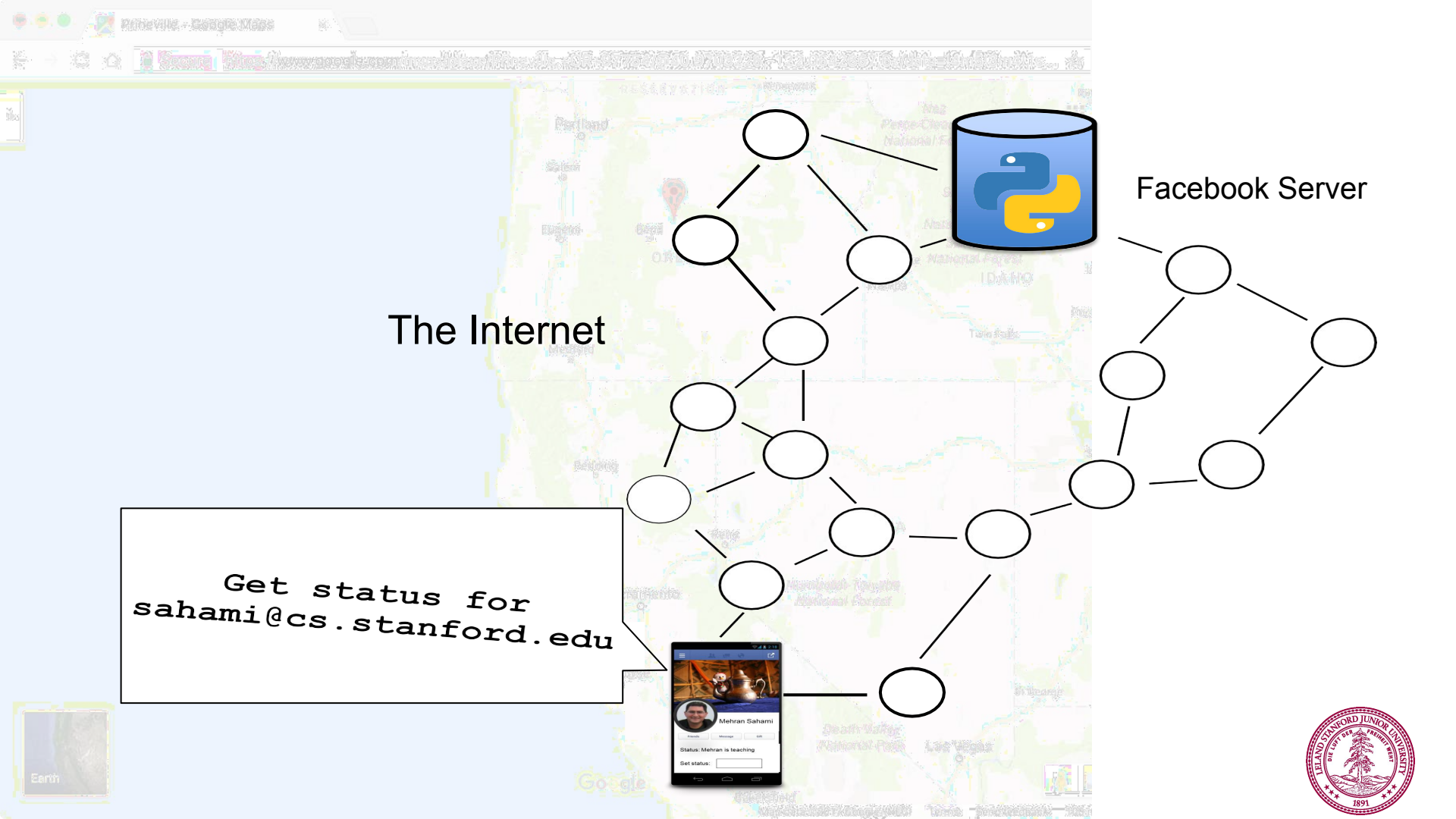
Facebook Server

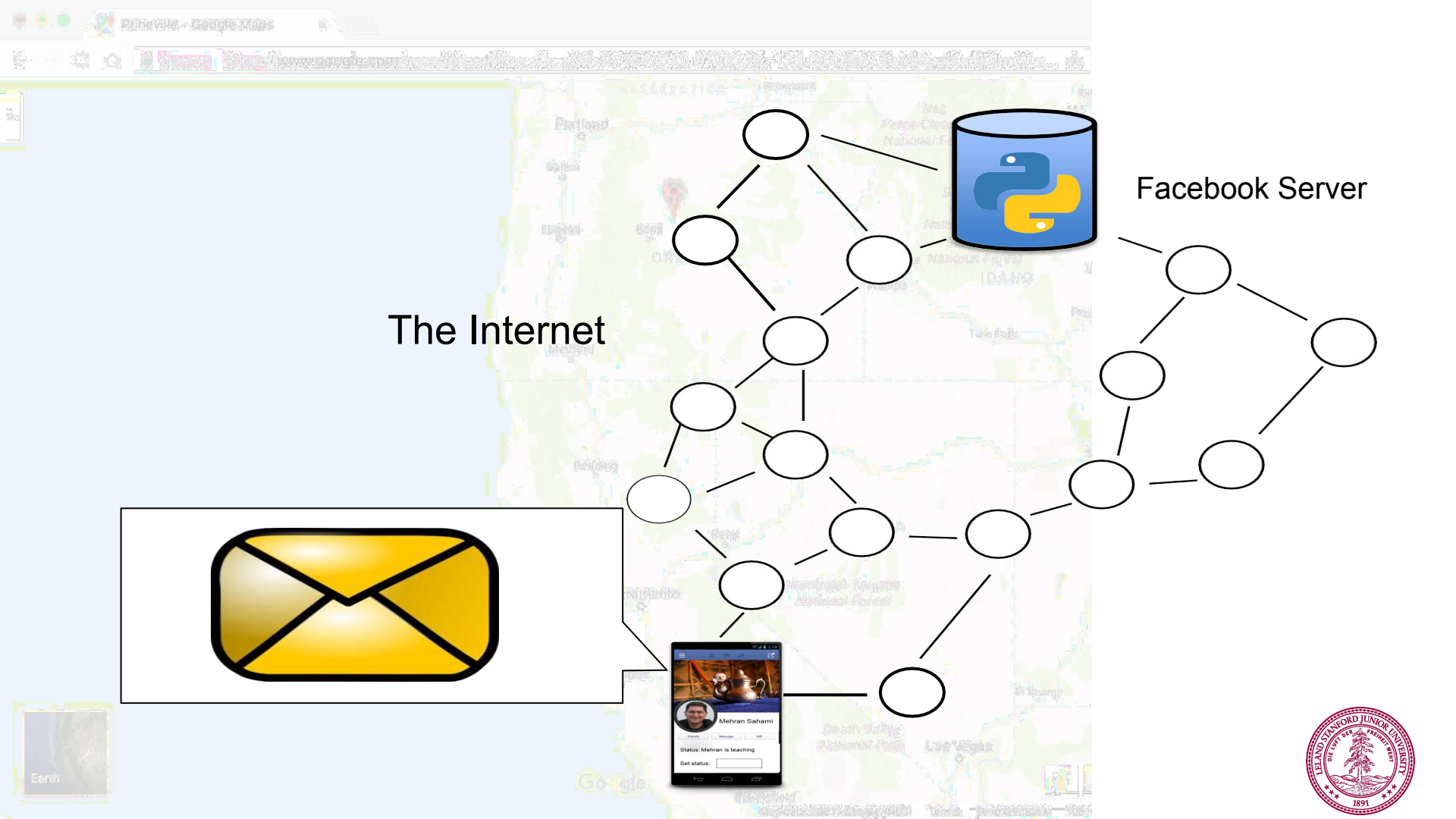


=

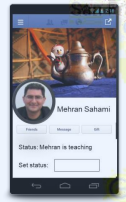

Prineville - Google Maps
Secure
https://www.google.com/maps/place/Prineville,+OR+97754/@40.9703226,-122.0642667,6z/data=!4m5!3m4!1s...
Facebook's closest datacenter is here
I am here
Earth
RESERVATION
Kennewick
Portland
Salem
Eugene
Bend
OREGON
Boise National Forest
IDAHO
Nampa
Twin Falls
Medford
Redding
NEVADA
Reno
Humboldt-Toiyabe National Forest
Sacramento
San Francisco
Fresno
CALIFORNIA
Death Valley National Park
Las Vegas
St George
Bakersfield
Google
Map data ©2017 Google, INEGI
Terms
Send feedback

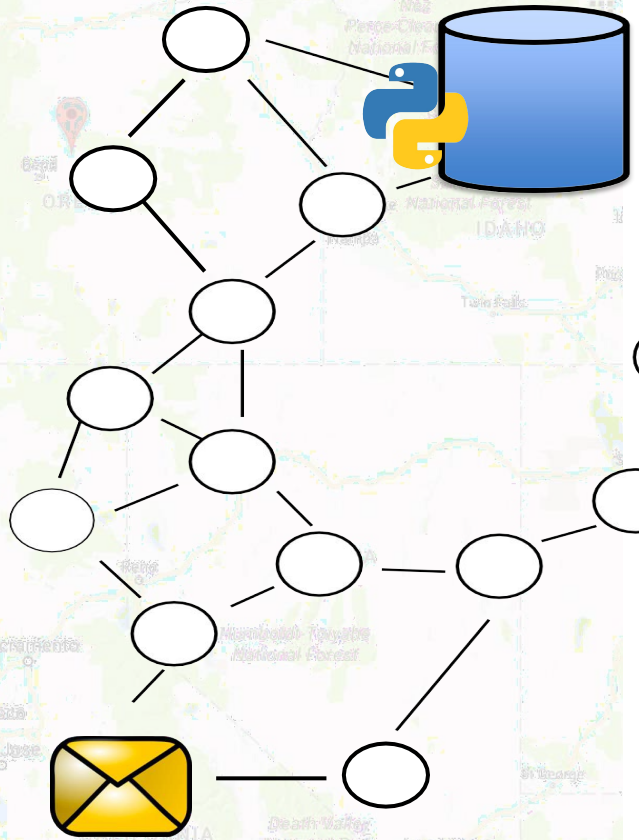

The Internet
Facebook Server
Mehran Sahami
Status: Mehran is teaching
Set status:


Facebook Server
The Internet
Get status for sahami@cs.stanford.edu
Mehran Sahami
Status: Mehran is teaching
Set status:

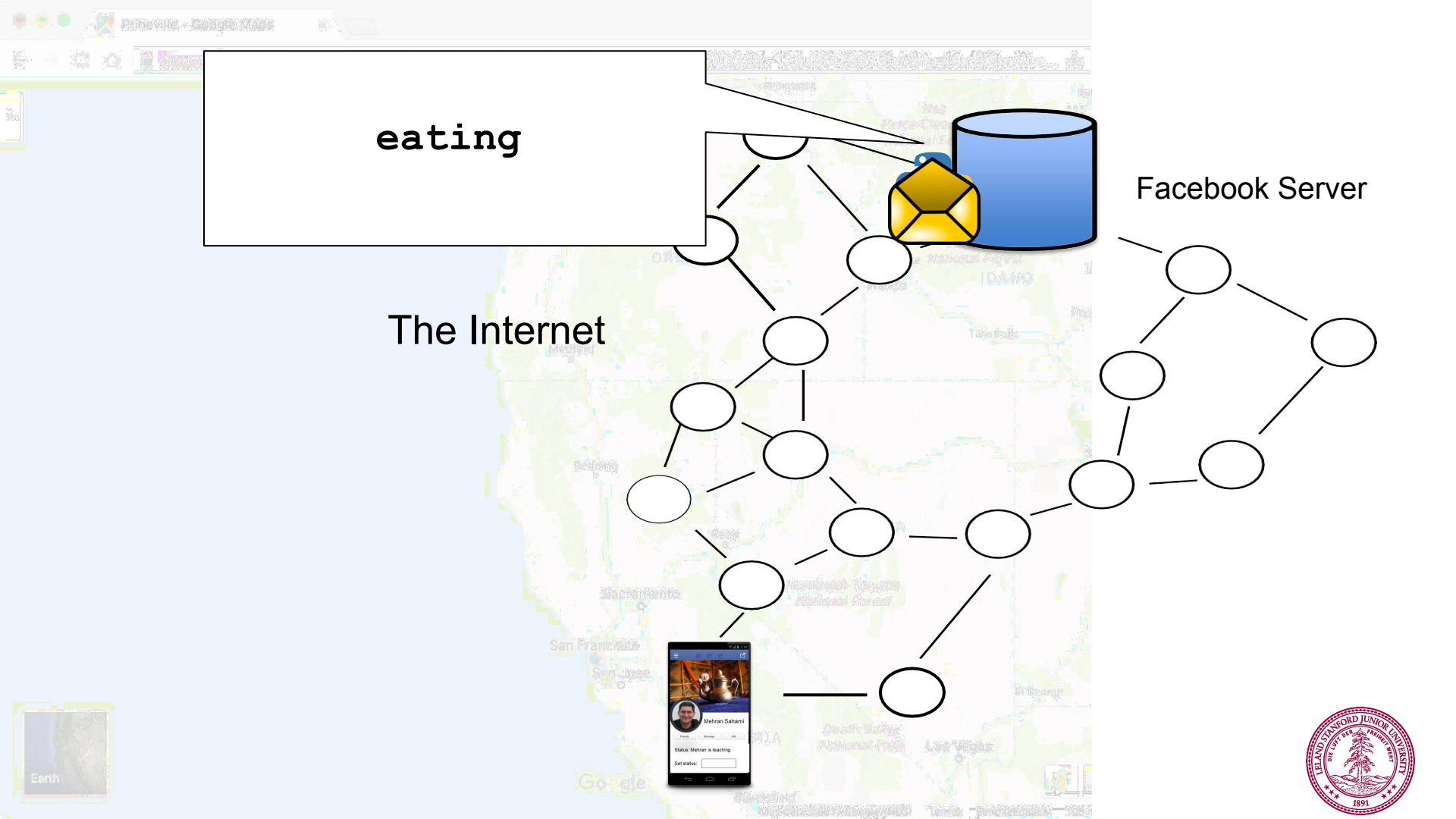

Facebook Server
The Internet
Mehran Sahami
Status: Mehran is teaching
Set status:

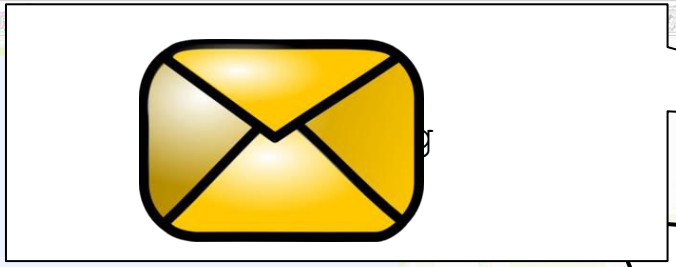
The Internet

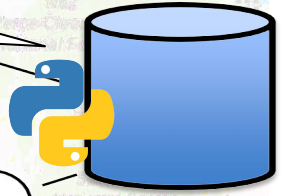




Facebook Server


eating
Facebook Server
The Internet
Mehran Sahami
Status: Mehran is teaching
Set status:

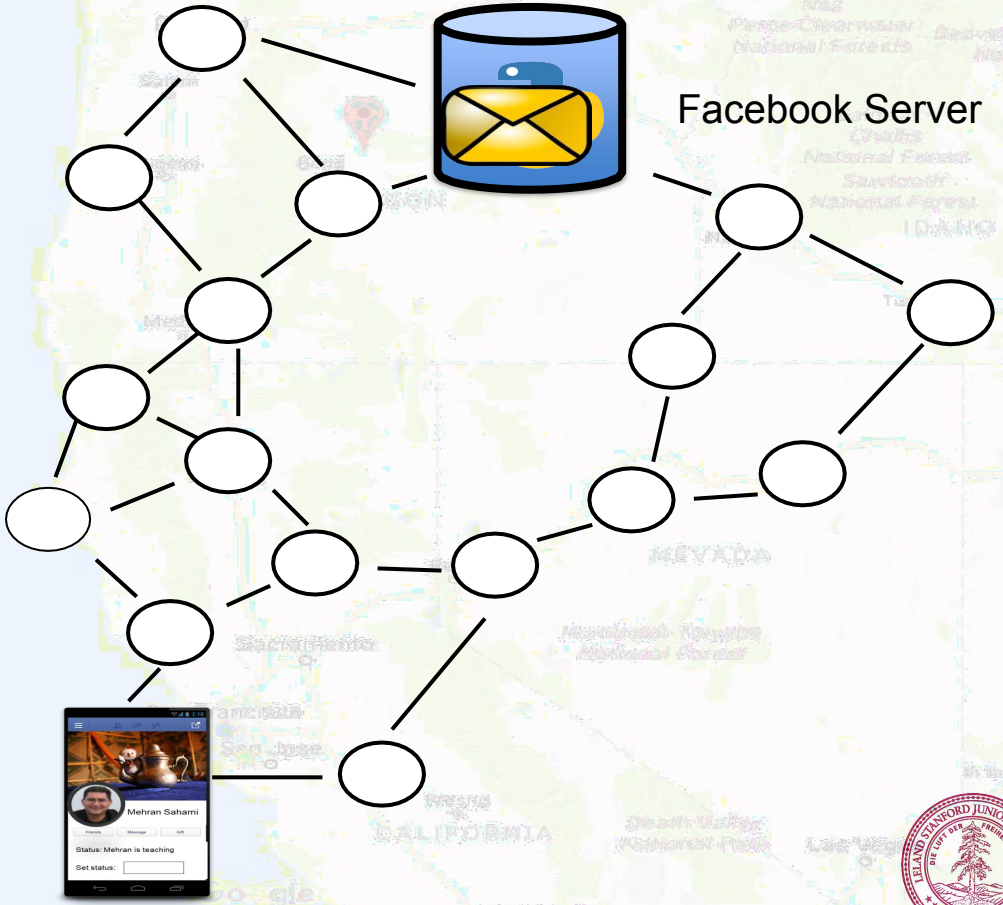




Facebook Server

The Internet

The Internet

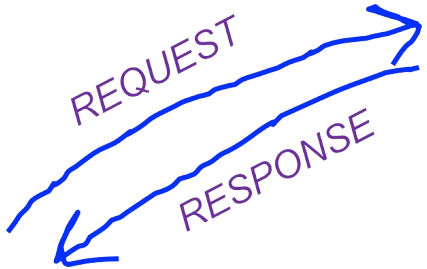


Many computers can connect
to the same server

The Internet

Facebook datacenter

"Server"



Your computer (facebook.com)

"Client"

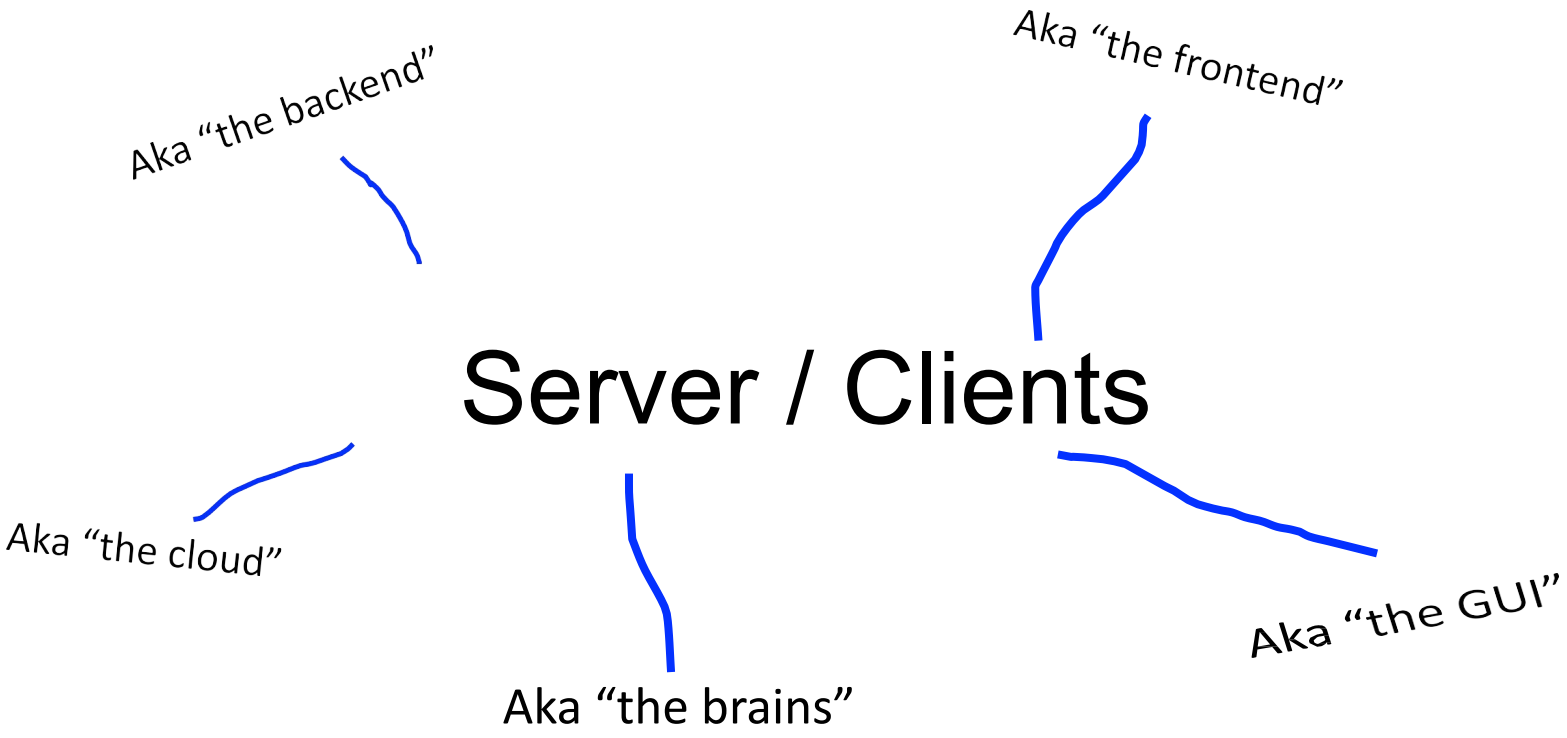
Mehran's phone (facebook app)

"Client"

Your mom's computer (linux shell)

"Client"

Most of the Internet



Today, the server

A server’s main job is to respond to requests

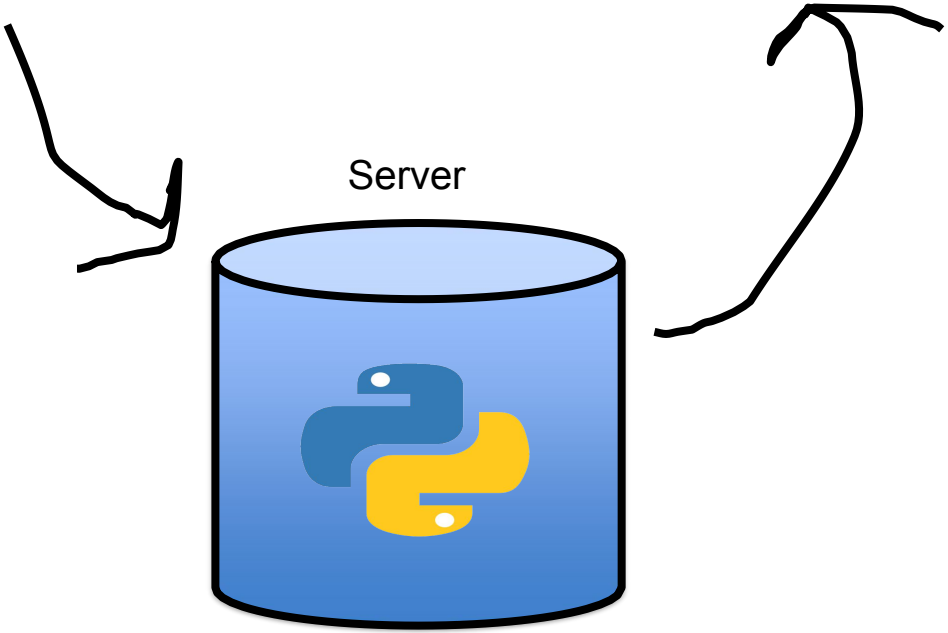
A Server's Simple Purpose

Request

From a client

Response

To the client



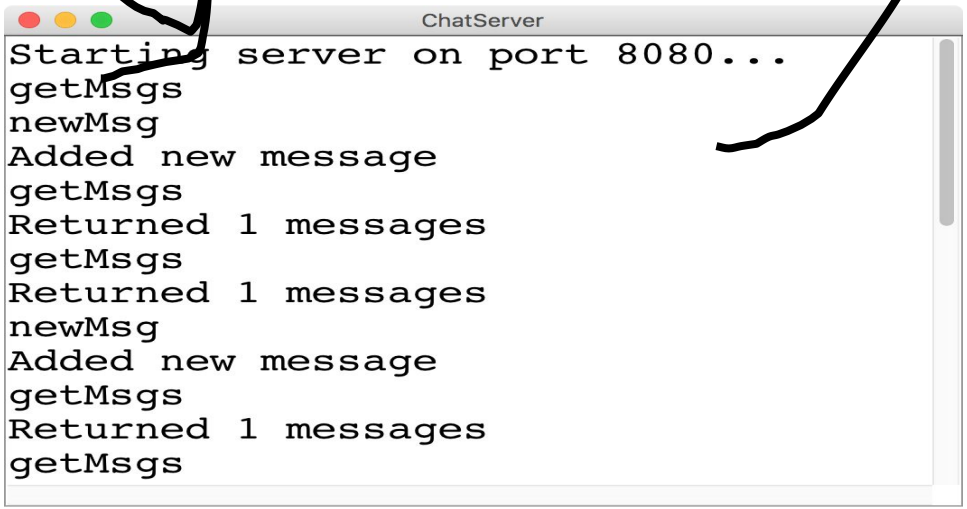
A Server's Simple Purpose

Request

`someRequest`

String

`serverResponse`



Servers on one slide

1

```
# handle server requests (must be in a
class)
def handle_request(self, request): #
    return a string response!
```

2

```
# turn on the server
def main():
    # make an instance of your server class
    handler = MyServer()
    # start the server!
    SimpleServer.run_server(handler, 8000)
    # enjoy
```

3

Servers on one slide

1

```
# handle server requests (must be in a
class)
def handle_request(self, request):
    # return a string response!
```

2

```
# turn on the server
def main():
    # make an instance of your server class
    handler = MyServer()
    # start the server!
    SimpleServer.run_server(handler, 8000)
    # enjoy
```

3

Servers on one slide

1

```
# handle server requests (must be in a
class)
def handle_request(self, request):
    # return a string response!
```

2

```
# turn on the server
def main():
    # make an instance of your server class
    handler = MyServer() # has handleRequest method
```

3

```
    # start the server!
    SimpleServer.run_server(handler, 8000)
    # enjoy
```

Servers on one slide

1

```
# handle server requests (must be in a
class)
def handle_request(self, request):
    # return a string response!
```

2

```
# turn on the server
def main():
    # make an instance of your server class
    handler = MyServer()
```

3

```
    # start the server!
    SimpleServer.run_server(handler, 8000)
    # enjoy
```

Servers on one slide

1

```
# handle server requests (must be in a
class)
def handle_request(self, request):
    # return a string response!
```

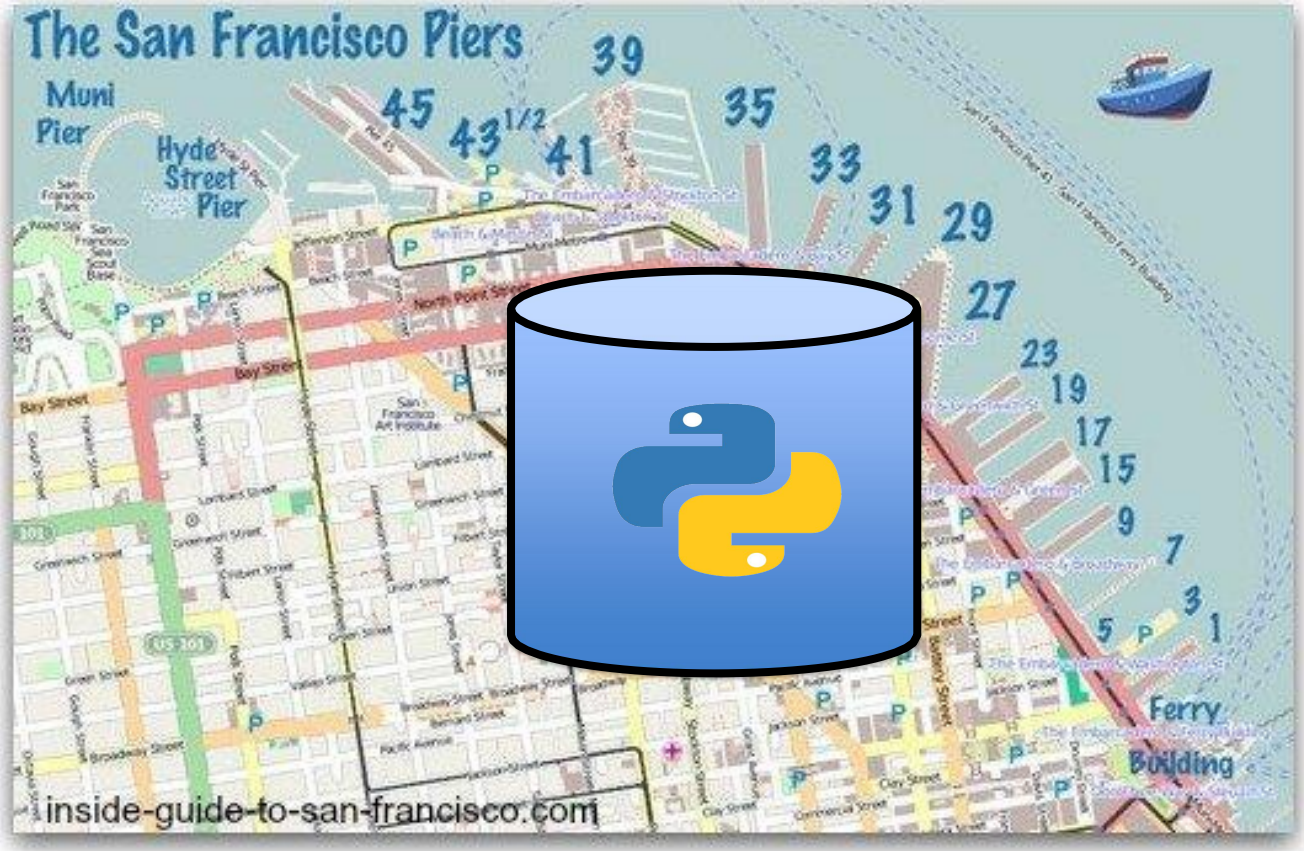
2

```
# turn on the server
def main():
    # make an instance of your server class
    handler = MyServer()
```

3

```
    # start the server!
    SimpleServer.run_server(handler, 8000)
    # enjoy
```

What is a Port?



Servers on one slide

1

```
# handle server requests (must be in a
class)
def handle_request(self, request):
    # return a string response!
```

2

```
# turn on the server
def main():
    # make an instance of your server class
    handler = MyServer()
```

3

```
    # start the server!
    SimpleServer.run_server(handler, 8000)
    # enjoy
```

What is a Request?

```
/* Request has a command */
command (type is string)

/* Request has parameters */
params (type is dict)
```

```
// methods that the server calls on requests
request.command
request.params
```

Servers on one slide

1

```
# handle server requests (must be in a
class)
def handle_request(self, request):
    # return a string response!
```

2

```
# turn on the server
def main():
    # make an instance of your server class
    handler = MyServer()
    # start the server!
    SimpleServer.run_server(handler, 8000)
    # enjoy
```

3

```
class Request:
    """
    Request class packages the key information from an internet request.
    """
    def __init__(self, request_command, request_params):
        # Every request has a command (string)
        self.command = request_command
        # Every request has params (dictionary). Can be empty: {}.
        self.params = request_params

    def get_params(self):
        # A 'getter' method to get the params return self.params

    def get_command(self):
        # A 'getter' method to get the command return self.command

    def __str__(self):
        # A special method which allows you to 'print' a request as a string.
        return str(self.  dict  )
```

First Server Example!

```
import SimpleServer

# We define a class to handle server requests class
class MyFirstServer:
    def __init__(self):
        pass

    # This is the server request callback function.
    def handle_request(self, request):
        print(request)
        return 'Happy Thursday, wonderful cs106a!!!'

def main():
    # Make the server handler
    handler = MyFirstServer()
    # Start the server to handle internet requests at specified port
    SimpleServer.run_server(handler, 8000)
```

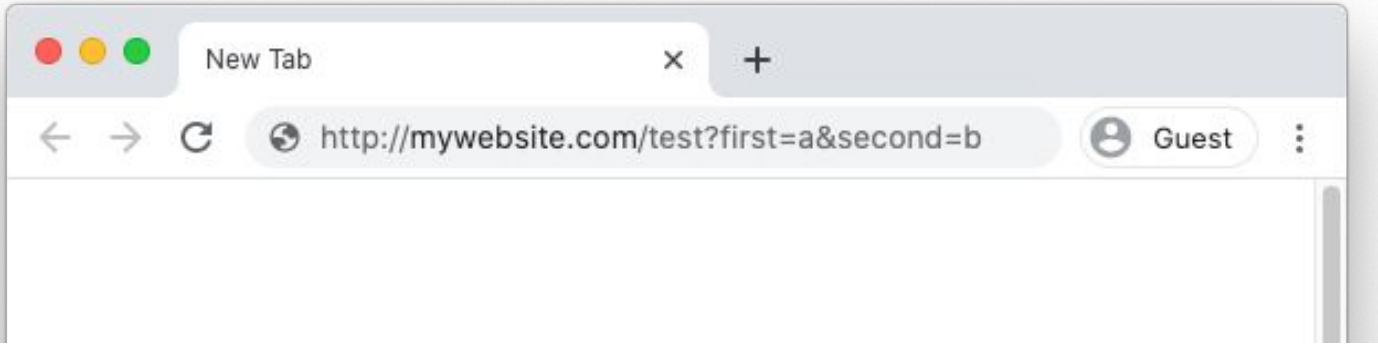
Who makes requests?

Who makes requests?

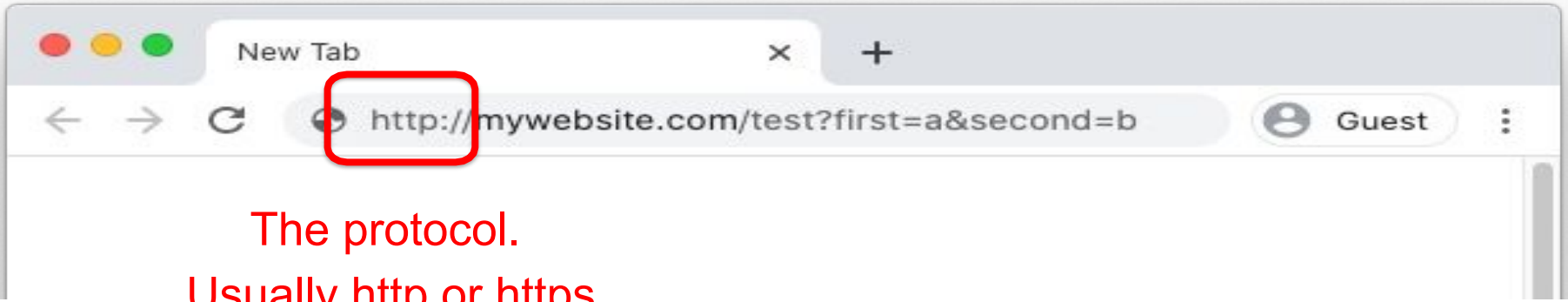
Other programs can send requests!

Web browsers can send requests!

Anatomy of a Browser Request



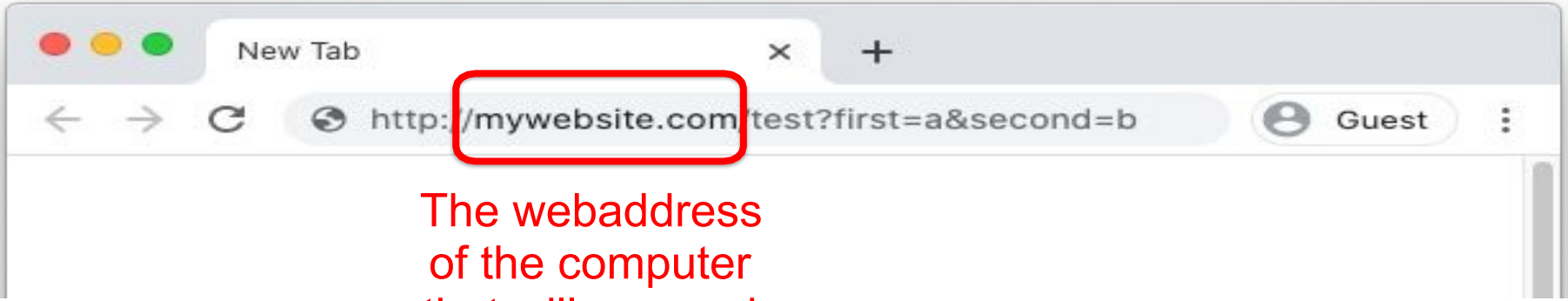
Anatomy of a Browser Request



The protocol.

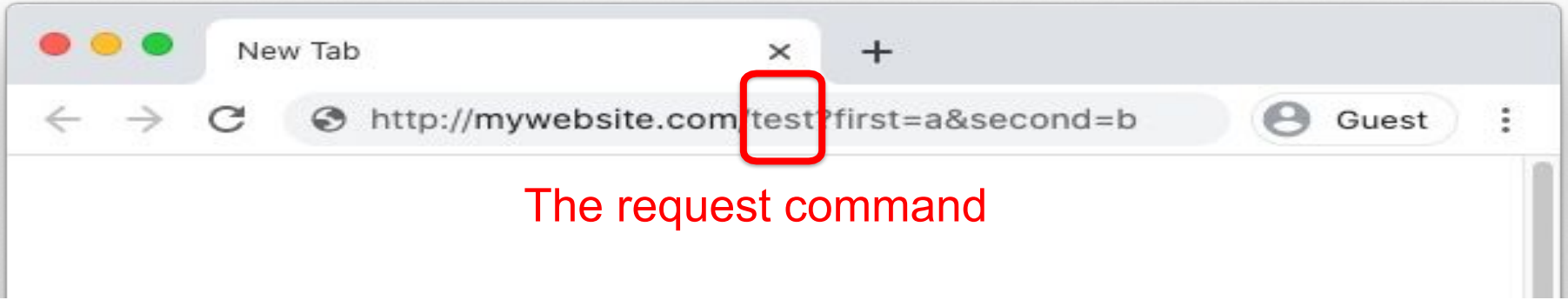
Usually http or https

Anatomy of a Browser Request

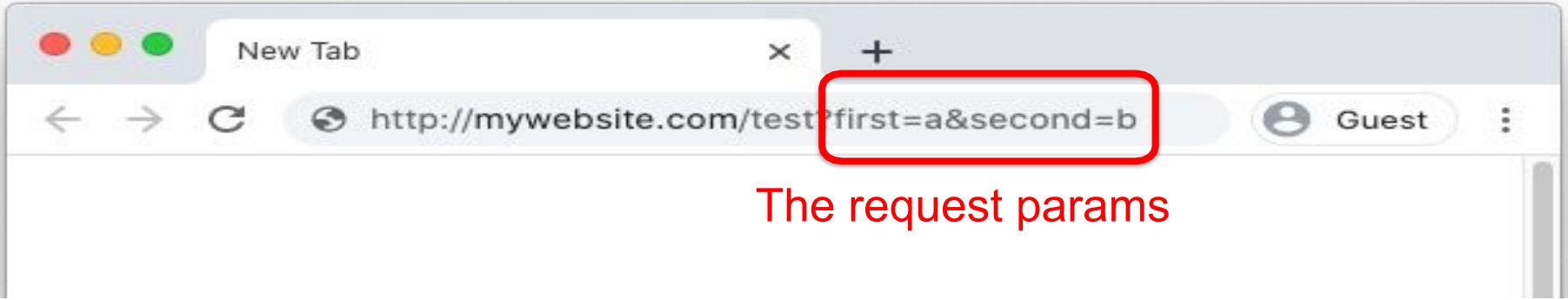


The webaddress of the computer that will respond to the request

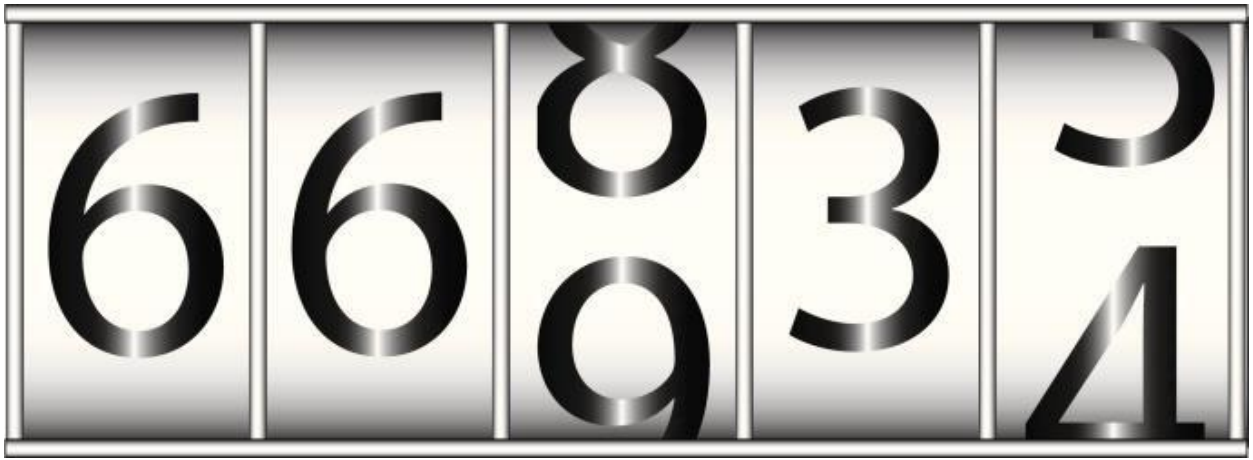
Anatomy of a Browser Request



Anatomy of a Browser Request



Hit Counter



Recall Requests

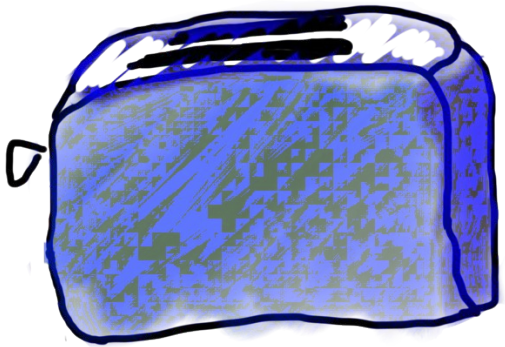
```
/* Request has a command */
command (string)

/* Request has parameters */
params (dict)
```

```
// methods that the server calls on requests
request.command
request.params
```

Requests are like Remote Method Calls

Server has a bunch of discrete things it can do

Server



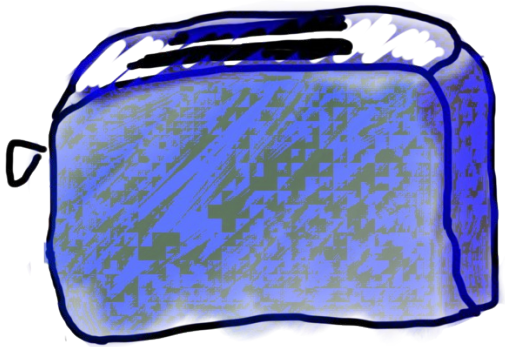
make_toast

blend

Requests are like Remote Method Calls

Server has a bunch of discrete things it can do

Server

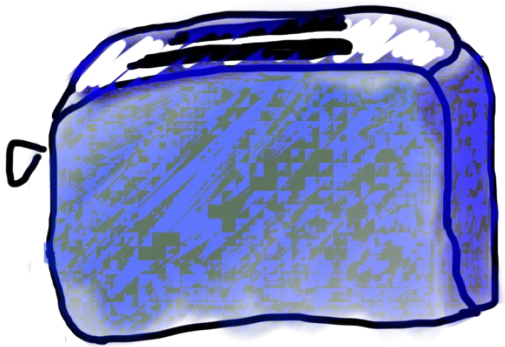


get_status

add_user

Requests are like Remote Method Calls

Server

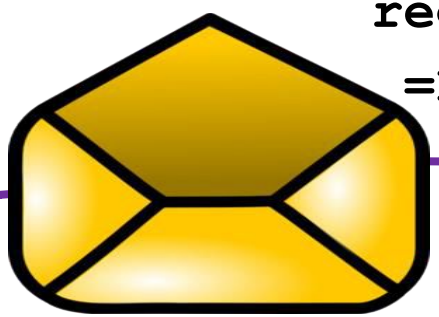


get_status

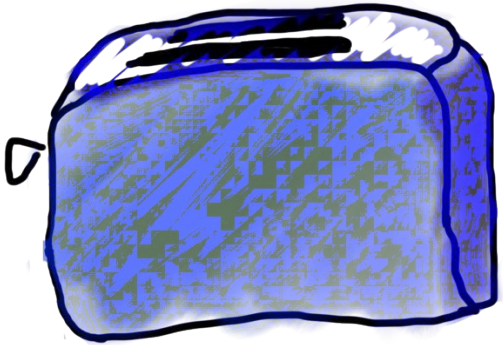
add_user

Requests are like Remote Method Calls

```
request.get_command()
 => "get_status"
```



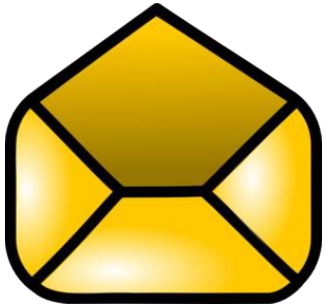
Server



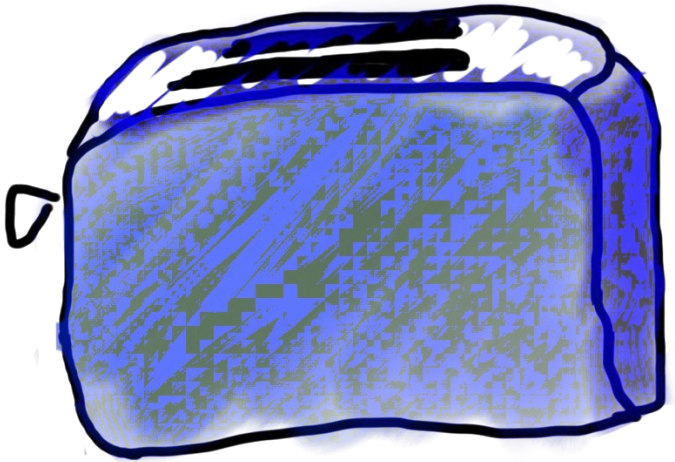
get_status

add_user

Requests are like Remote Method Calls

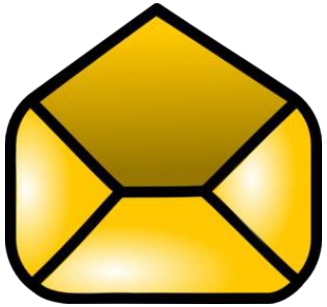


To make toast, I need a parameter



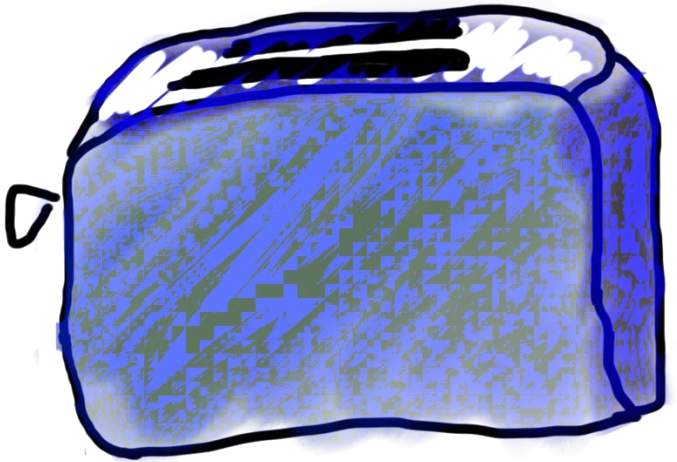
get_status

Requests are like Remote Method Calls



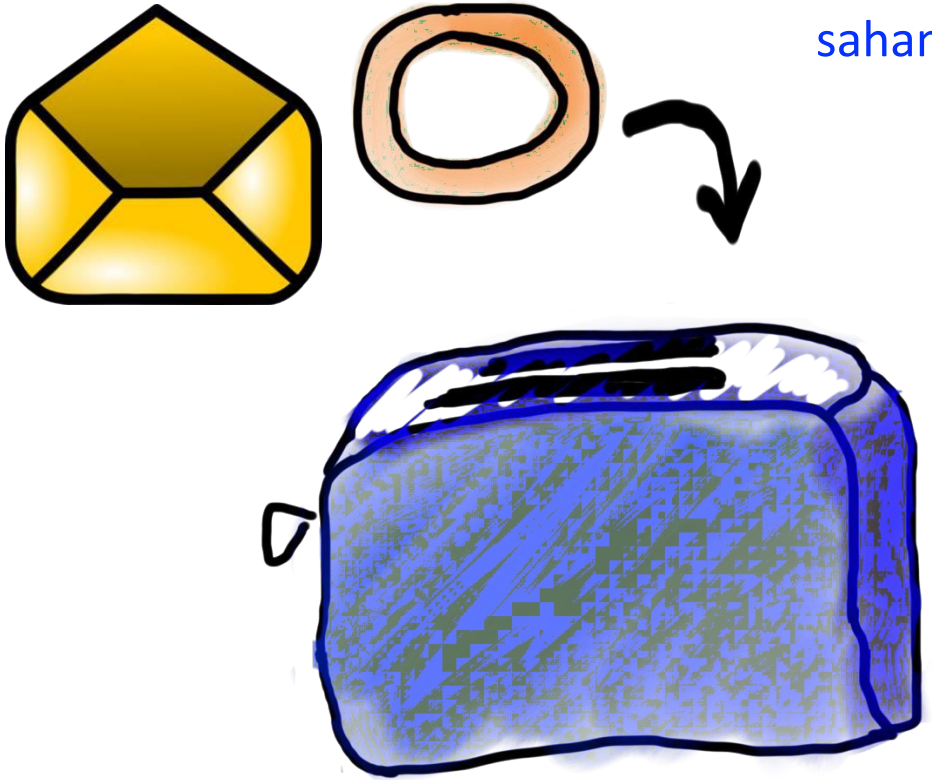
I was given a parameter!

`request.params["userName"]`

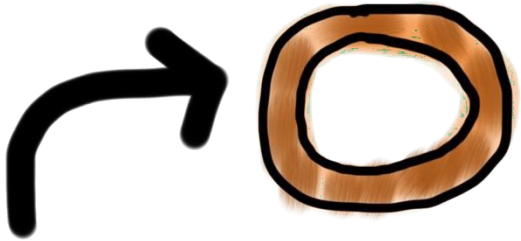


get_status

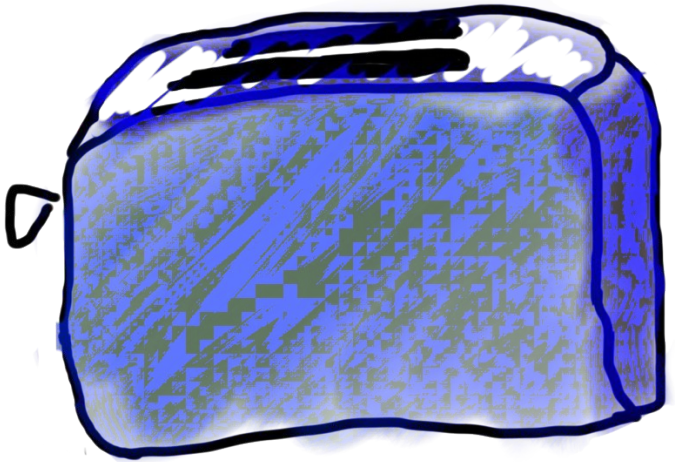
Requests are like Remote Method Calls



Requests are like Remote Method Calls



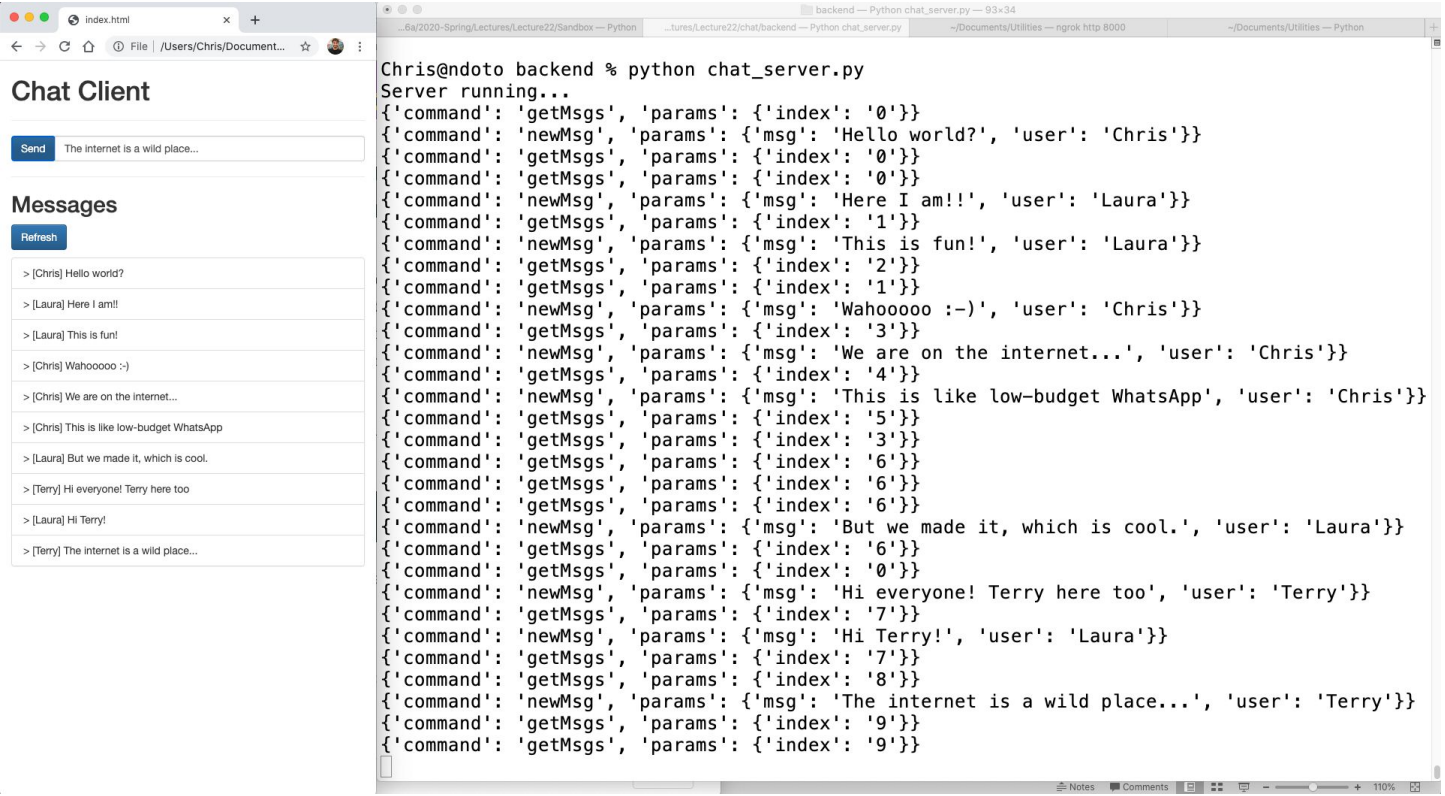
teaching

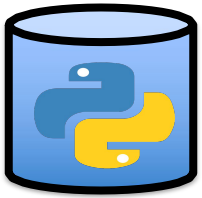


```
def handle_request(self, request):
    cmd = request.command
    if cmd == 'get_status':
        user = request.params['userName']
        status = self.get_status(user)
        return status

    if cmd == 'add_user':
        user = request.params['userName']
        status = self.add_user(user)
```

Time for a little chat

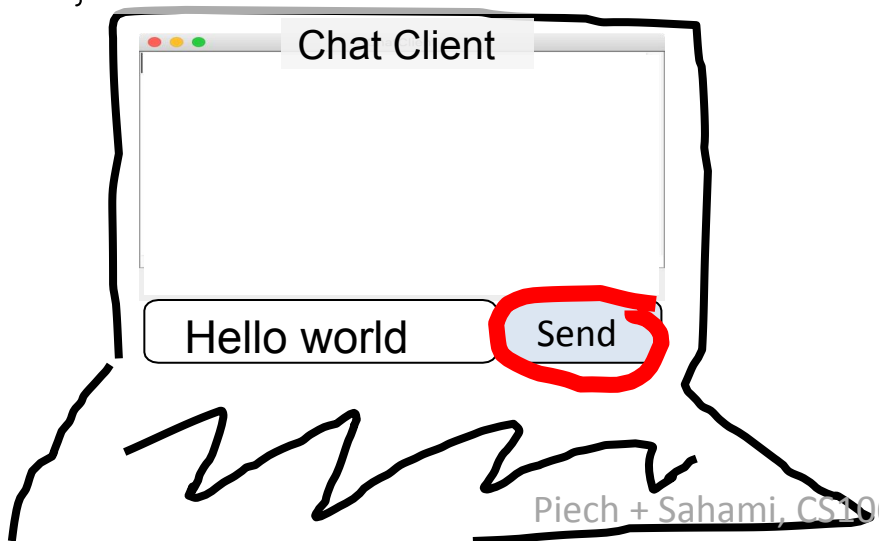
Chat Server and Client

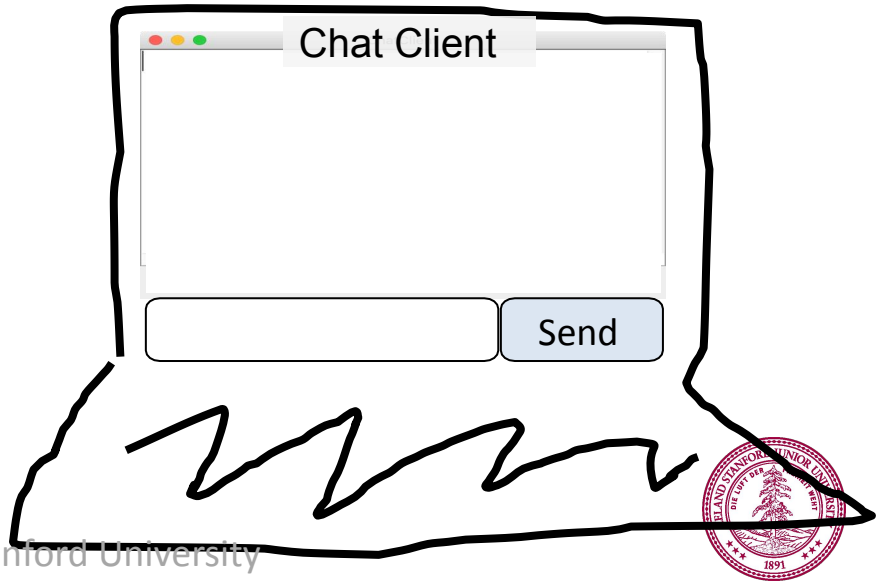


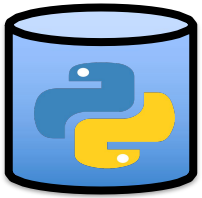


```
history = []
```

```
addMsg
{
    'msg' : Hello world,
    'user' : 'C'
}
```

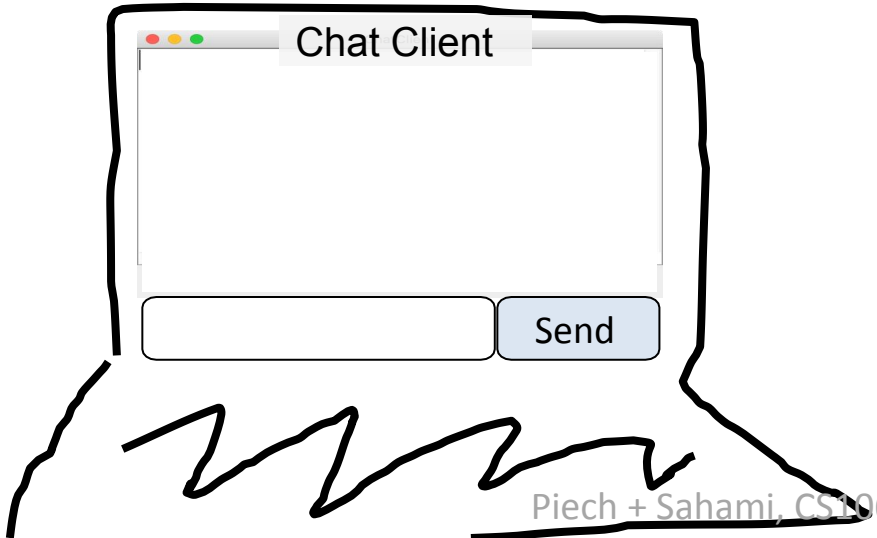


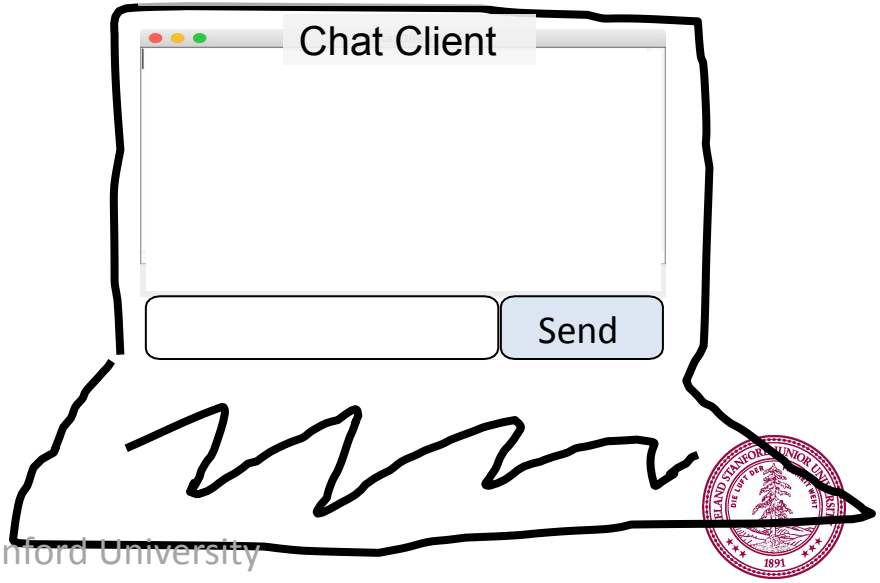


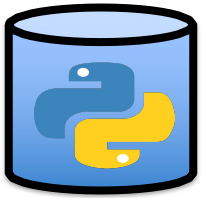


```
history = [
    '[C] Hello World'
]
```

```
getMsgs
{
    'index' : 0
}
```

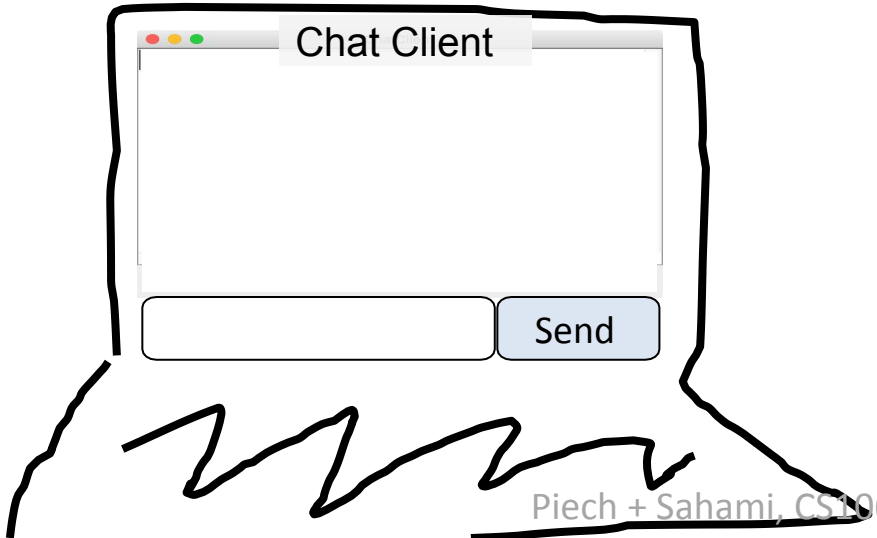


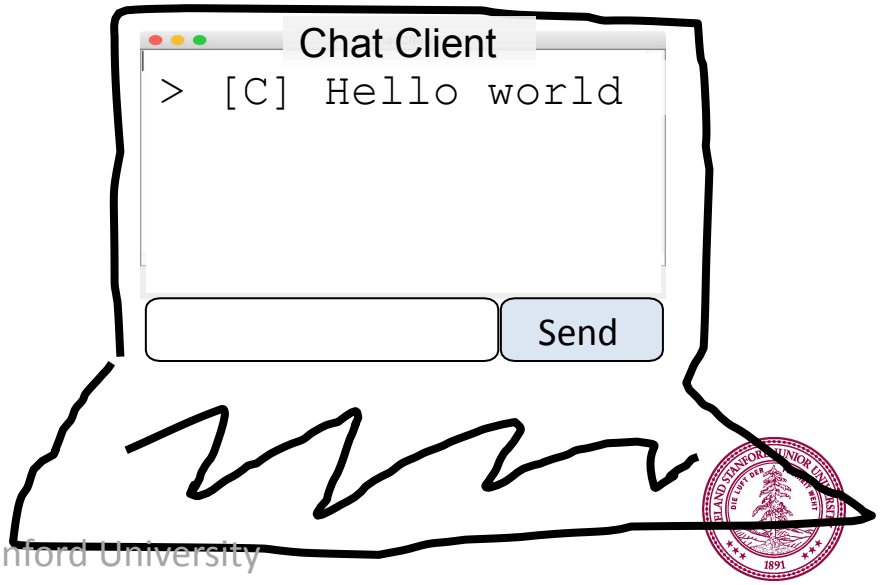




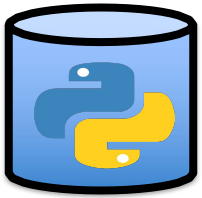
```
history = [
    '[C] Hello world'
]
```

'["[C] Hello world"]'

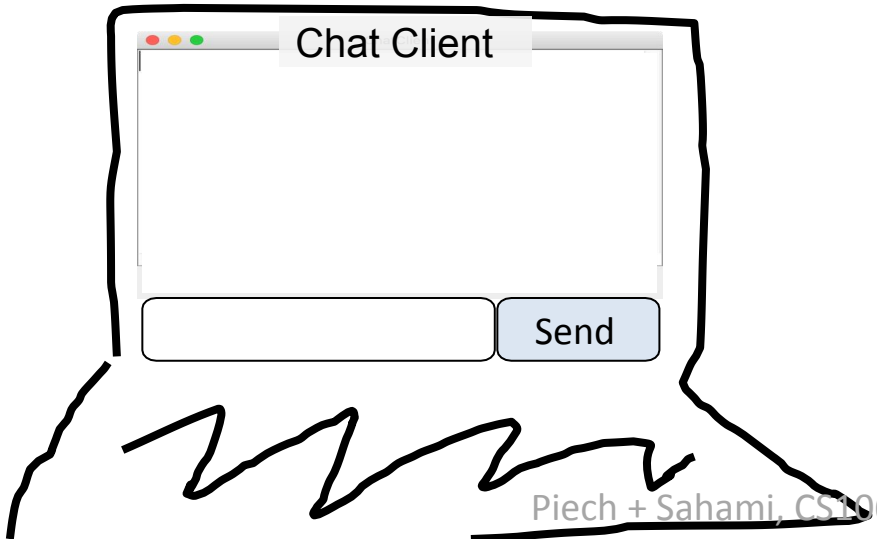


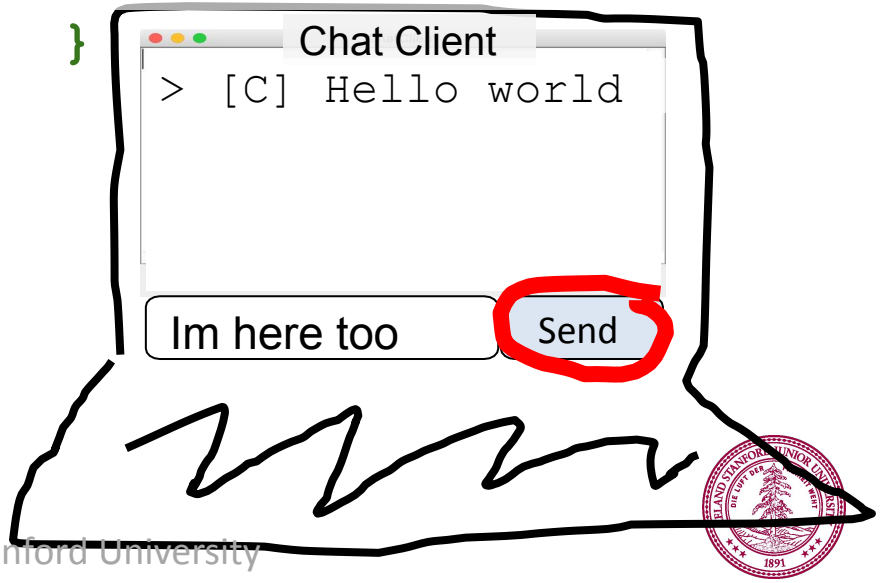


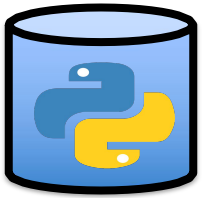
```
history = ['[C] Hello World']
```



```
addMsg
{
    'msg' : 'Im here too'
    'user' : 'B'
}
```

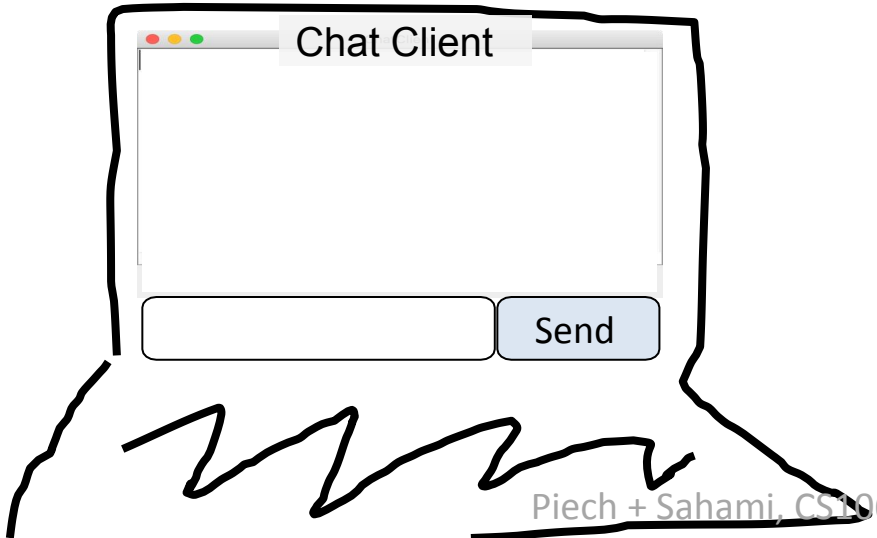


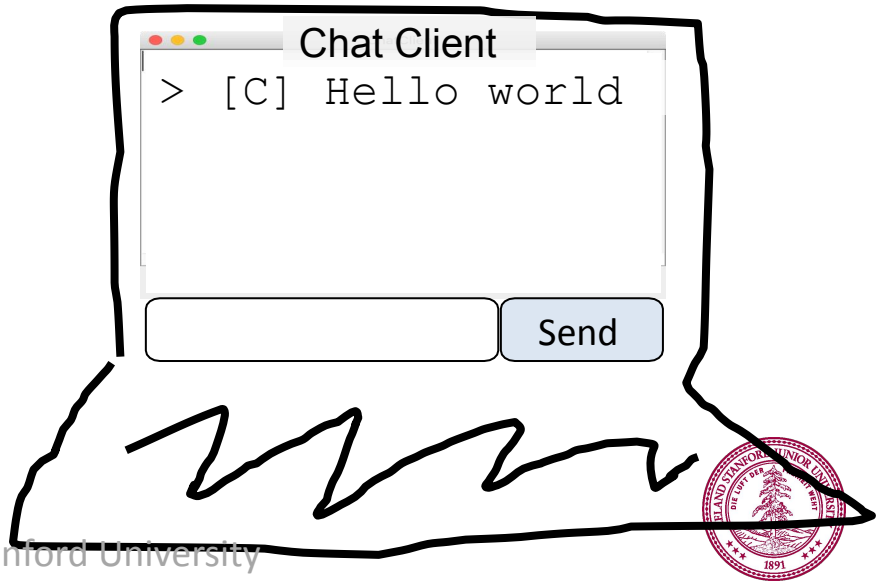


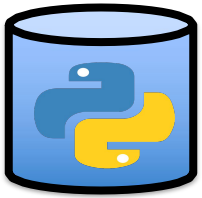


```
history = [
    '[C] Hello world',
    '[B] Im here too'
]
```

```
'Got it'
```

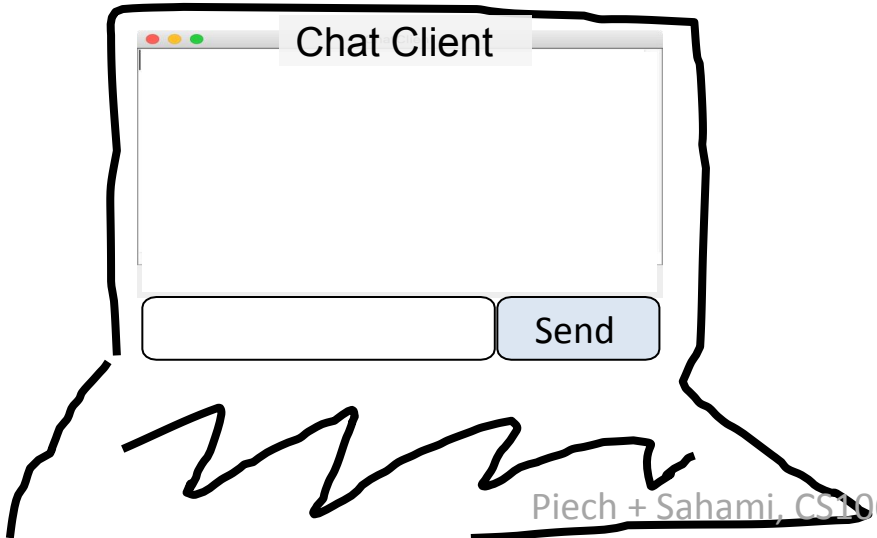


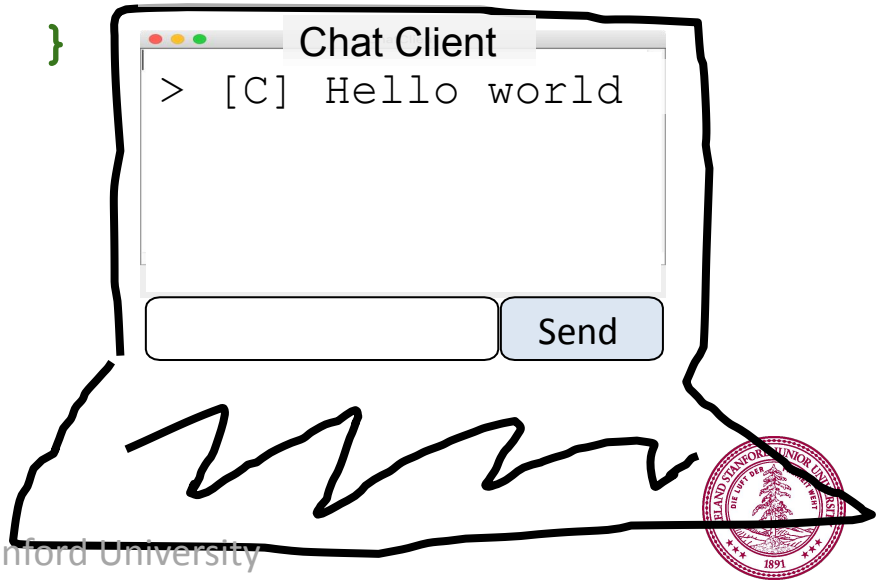


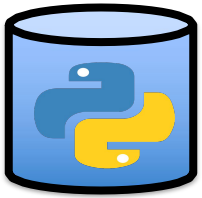


```
history = [
    '[C] Hello world',
    '[B] Im here too'
]
```

```
getMsgs
{
    'index' : 1
}
```

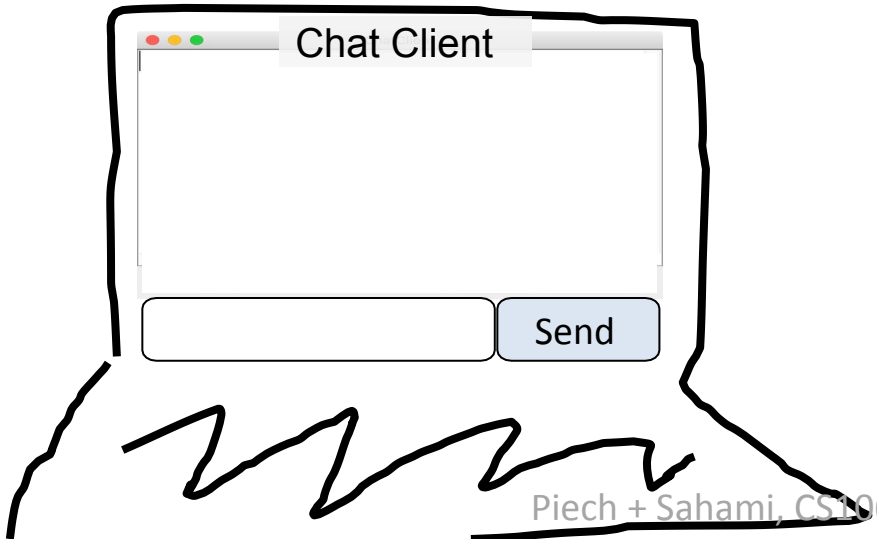


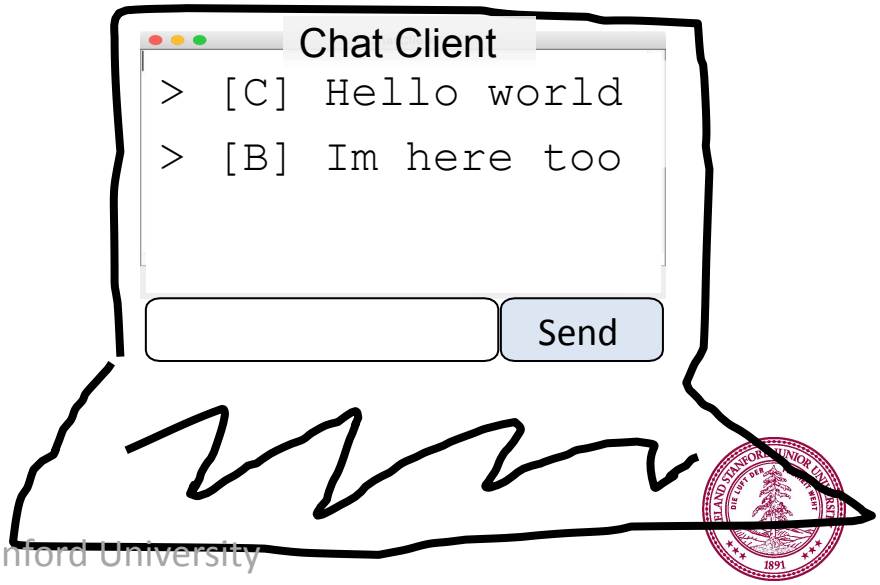


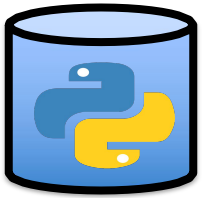


```
history = [
    '[C] Hello world',
    '[B] Im here too'
]
```

```
'["[B] Im here too"]'
```

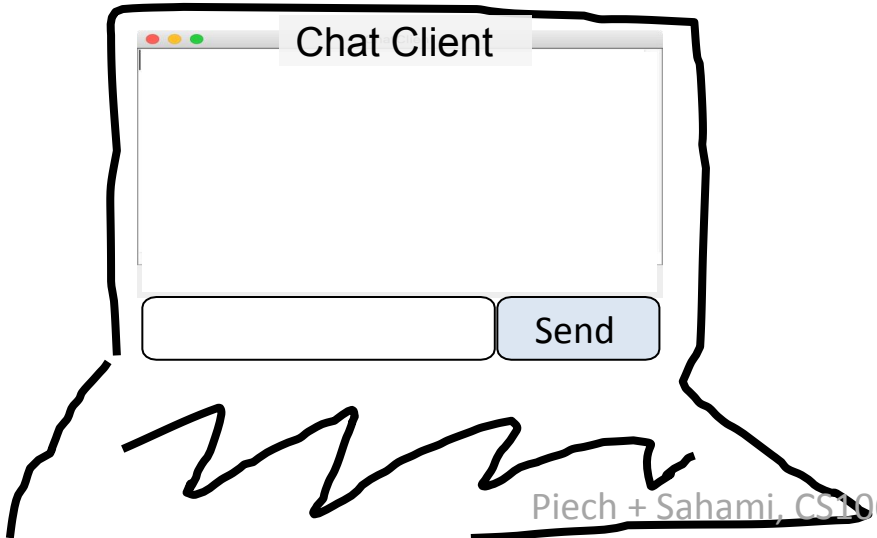


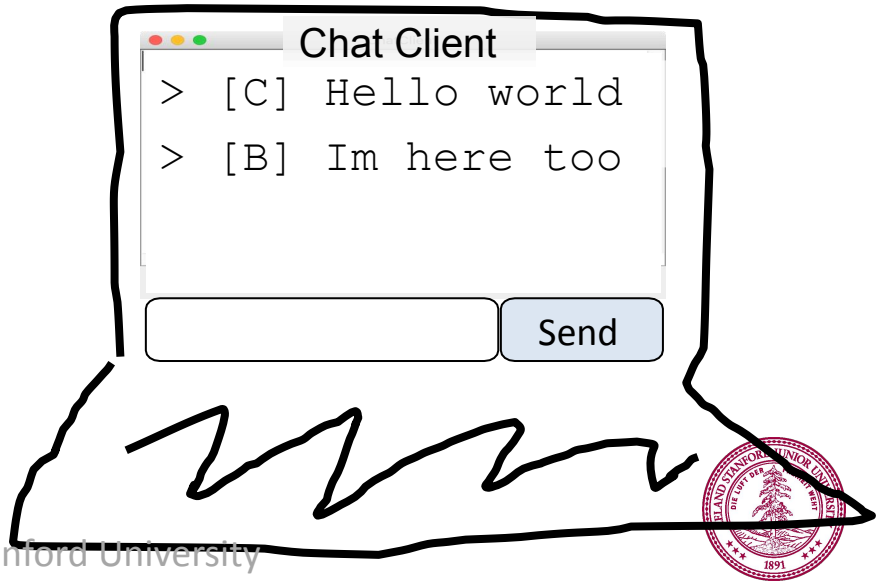


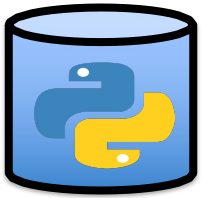


```
history = [
    '[C] Hello world',
    '[B] Im here too'
]
```

```
getMsgs
{
    'index' : 0
}
```

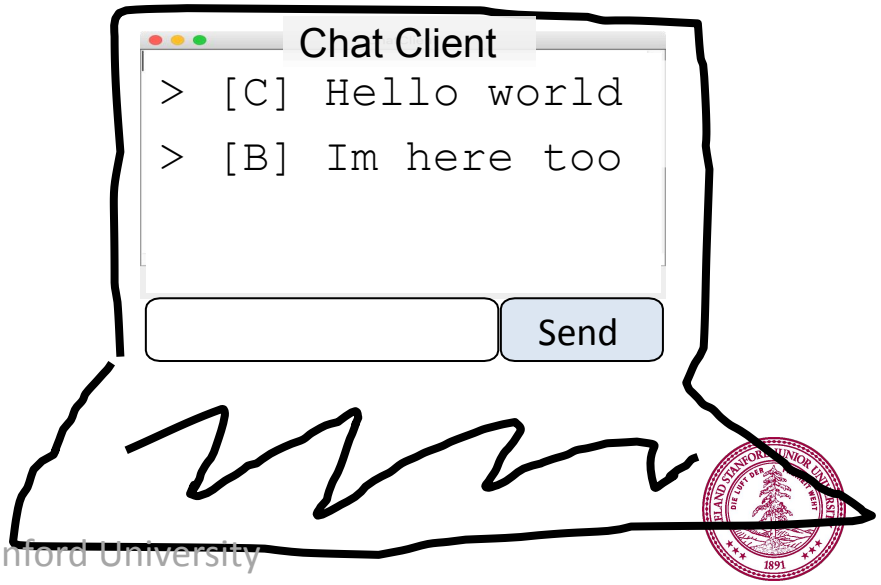






```
history = [
    '[C] Hello world',
]   '[B] Im here too'
```

```
'["[C] Hello world",
  "[B] Im here too"]'
```



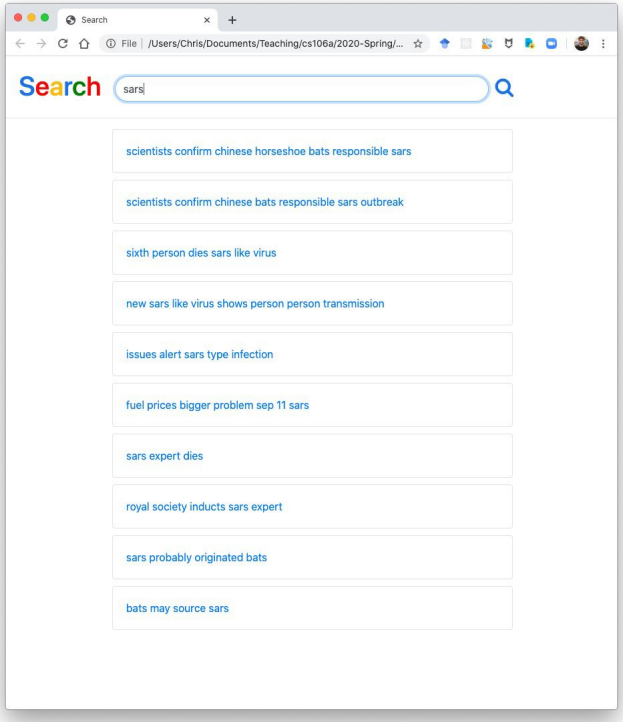
Chat Server

Chat Server

addMsg
msg = *text*
user = *user*

getMsgs index = start_*index*

Bajillion Extension



Search Engine

Recap

- The internet is just a bunch of connected computers
- Each computer is either a server or a client
- We can use classes in Python to make servers!
- A server's one job is to handle requests

Bonus

Internet sends data as strings…

How do you send a list or a dictionary?

Requests responses are strings, often encoded using JSON

JSON Crash Course

ages.json

```
{
    "Chris":32,
    "Gary":70,
    "Mehran":50,
    "Brahm":23,
    "Rihanna":32,
    "Adele":32
}
```

```
import json

# load data
data = json.load(open('ages.json'))
# save data
json.dump(data, open('ages.json'))
```

JSON Crash Course

ages.json

```
{
    "Chris":32,
    "Gary":70,
    "Mehran":50,
    "Brahm":23,
    "Rihanna":32,
    "Adele":32
}
```

```
import json

# load data
data = json.load(open('ages.json'))
# save data
json.dump(data, open('ages.json'))
```

JSON Crash Course

ages.json

```
{
    "Chris":32,
    "Gary":70,
    "Mehran":50,
    "Brahm":23,
    "Rihanna":32,
    "Adele":32
}
```

```
import json

# load data
data = json.load(open('ages.json'))
# save data
json.dump(data, open('ages.json'))
# write a variable to a string
data_str = json.dumps(data)
```