The Internet (Wrap-Up) and Life After CS106A
CS106A, Stanford University
Assignment #6

<table>
<thead>
<tr>
<th>Time Range</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-2 hours</td>
<td>7%</td>
</tr>
<tr>
<td>2-4 hours</td>
<td>13%</td>
</tr>
<tr>
<td>4-6 hours</td>
<td>16%</td>
</tr>
<tr>
<td>6-8 hours</td>
<td>18%</td>
</tr>
<tr>
<td>8-10 hours</td>
<td>11%</td>
</tr>
<tr>
<td>10-12 hours</td>
<td>11%</td>
</tr>
<tr>
<td>12-14 hours</td>
<td>9%</td>
</tr>
<tr>
<td>14-16 hours</td>
<td>7%</td>
</tr>
<tr>
<td>16-18 hours</td>
<td>9%</td>
</tr>
<tr>
<td>18+ hours</td>
<td>9%</td>
</tr>
</tbody>
</table>
Learning Goals

1. Write a chat program that can respond to internet requests
Recall, Creating Servers

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

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

3. # enjoy
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 responses are strings, often encoded using JSON
Reading a dictionary from a file

ages.json

```json
{
    "Chris": 34,
    "Gary": 70,
    "Mehran": 52,
    "Brahm": 25,
    "Rihanna": 34
}
```
JSON: JavaScript Object Notation

It's a format for storing a data structure as human (and machine) readable text

ages.json

{  
"Chris": 34,  
"Gary": 70,  
"Mehran": 52,  
"Brahm": 25,  
"Rihanna": 34  
}
import json

def main():
    file = open('ages.json')
    data = json.load(file)
    for name in data:
        age = data[name]
        print(name, age)

print_ages.py

{  
    "Chris": 34,
    "Gary": 70,
    "Mehran": 52,
    "Brahm": 25,
    "Rihanna": 34
}
**JSON: handling data structures**

---

**ages.json**

```json
{
  "Chris": 34,
  "Gary": 70,
  "Mehran": 52,
  "Brahm": 25,
  "Rihanna": 34
}
```

**print_ages.py**

```python
import json

def main():
    file = open('ages.json')
    data = json.load(file)
    for name in data:
        age = data[name]
        print(name, age)
```

---
import json

def main():
    file = open('ages.json')
    data = json.load(file)
    for name in data:
        age = data[name]
        print(name, age)
import json

def main():
    file = open('ages.json')
    data = json.load(file)
    for name in data:
        age = data[name]
        print(name, age)

print_ages.py

ages.json

{
    "Chris": 34,
    "Gary": 70,
    "Mehran": 52,
    "Brahm": 25,
    "Rihanna": 34
}
JSON: handling data structures

**ages.json**

```json
{
    "Chris": 34,
    "Gary": 70,
    "Mehran": 52,
    "Brahm": 25,
    "Rihanna": 34
}
```

**print_ages.py**

```python
import json

def main():
    file = open('ages.json')
    data = json.load(file)
    for name in data:
        age = data[name]
        print(name, age)
```

**data**

```json
{
    "Chris": 34,
    "Gary": 70,
    "Mehran": 52,
    "Brahm": 25,
    "Rihanna": 34
}
```
import json

def main():
    file = open('ages.json')
    data = json.load(file)
    for name in data:
        age = data[name]
        print(name, age)

print_ages.py

ages.json
{
  "Chris": 34,
  "Gary": 70,
  "Mehran": 52,
  "Brahm": 25,
  "Rihanna": 34
}

data
{
  "Chris": 34
  "Gary": 70,
  "Mehran": 52,
  "Brahm": 25,
  "Rihanna": 34
}

Console
Chris 34
Gary 70
Mehran 52
Brahm 25
Rihanna 34
import json

# load data from a JSON file
data = json.load(open('ages.json'))

# write a data structure to a JSON file
json.dump(data, open('ages.json', 'w'))
import json

# load data from a JSON file
data = json.load(open('ages.json'))

# write a data structure to a JSON file
json.dump(data, open('ages.json', 'w'))

ages.json

```json
{
    "Chris": 34,
    "Gary": 70,
    "Mehran": 52,
    "Brahm": 25,
    "Rihanna": 34
}
```

data

```json
{ 
    "Chris": 34 
    "Gary": 70,
    "Mehran": 52,
    "Brahm": 25,
    "Rihanna": 34 
}
```
import json

# load data from a JSON file
data = json.load(open('ages.json'))

# write a data structure to a JSON file
json.dump(data, open('new.json', 'w'))
JSON can also create strings!

```python
import json

# load data from a JSON file
data = json.load(open('ages.json'))

# write a data structure to a JSON file
json.dump(data, open('ages.json', 'w'))

# write a variable to a string
data_str = json.dumps(data)
```

```
ages.json

```

```json
{
    "Chris": 34,
    "Gary": 70,
    "Mehran": 52,
    "Brahm": 25,
    "Rihanna": 34
}
```

```
data

```

```json
{
    "Chris": 34
    "Gary": 70,
    "Mehran": 52,
    "Brahm": 25,
    "Rihanna": 34
}
import json

# load data from a JSON file
data = json.load(open('ages.json'))

# write a data structure to a JSON file
json.dump(data, open('ages.json', 'w'))

# write a variable to a string
data_str = json.dumps(data)

ages.json

{  
  "Chris": 34,  
  "Gary": 70,  
  "Mehran": 52,  
  "Brahm": 25,  
  "Rihanna": 34  
}

data

{  
  "Chris": 34  
  "Gary": 70,  
  "Mehran": 52,  
  "Brahm": 25,  
  "Rihanna": 34  
}

data_str

'{"Chris":34, "Gary":70, "Mehran":52, "Brahm":25, "Rihanna":34}''
Time for a little chat
Chat Server and Client

Chat Client

Server running...

['command': 'getMsgs', 'params': {'index': '0'}]

['command': 'newMsg', 'params': {'msg': 'Hello world?', 'user': 'Chris'}]

['command': 'getMsgs', 'params': {'index': '0'}]

['command': 'newMsg', 'params': {'msg': 'I am here!', 'user': 'Laura'}]

['command': 'getMsgs', 'params': {'index': '1'}]

['command': 'newMsg', 'params': {'msg': 'This is fun!', 'user': 'Laura'}]

['command': 'getMsgs', 'params': {'index': '2'}]

['command': 'newMsg', 'params': {'msg': 'Wahooee :)', 'user': 'Chris'}]

['command': 'getMsgs', 'params': {'index': '3'}]

['command': 'newMsg', 'params': {'msg': 'We are on the internet...', 'user': 'Chris'}]

['command': 'getMsgs', 'params': {'index': '4'}]

['command': 'newMsg', 'params': {'msg': 'This is like low-budget WhatsApp', 'user': 'Chris'}]

['command': 'getMsgs', 'params': {'index': '5'}]

['command': 'newMsg', 'params': {'msg': 'This is a wild place...'}]

['command': 'getMsgs', 'params': {'index': '6'}]

['command': 'newMsg', 'params': {'msg': 'But we made it, which is cool.', 'user': 'Laura'}]

['command': 'getMsgs', 'params': {'index': '7'}]

['command': 'newMsg', 'params': {'msg': 'Hi everyone! Terry here too', 'user': 'Terry'}]

['command': 'getMsgs', 'params': {'index': '8'}]

['command': 'newMsg', 'params': {'msg': 'Hi Terry!', 'user': 'Laura'}]

['command': 'getMsgs', 'params': {'index': '9'}]

['command': 'newMsg', 'params': {'msg': 'The internet is a wild place...', 'user': 'Terry'}]

['command': 'getMsgs', 'params': {'index': '9'}]
```json
newMsg
{
    'msg': 'Hello world',
    'user': 'C'
}
```
newMsg
{
    'msg' : 'Hello world'
    'user' : 'C'
}

def sendChatClient(newMsg):
    history = [ '[C] Hello world' ]
history = ['[C] Hello world']

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

getMsgs
{
    'index' : 0
}
history = ['[C] Hello world']

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

newMsg = {
    'msg': 'Im here too',
    'user': 'B'
}
```

Chat Client

```
> [C] Hello world

Im here too
```

Send
history = [
  '[C] Hello world',
  '[B] I'm here too'
]
'success'
history = [    '[C] Hello world',    '[B] I'm here too'  ]

getMsgs
{
  'index' : 1
}

Chat Client
> [C] Hello world
history = [
    '[C] Hello world',
    '[B] Im here too'
]

'"["[B] Im here too"]'
history = [
    '[C] Hello world',
    '[B] I m here too'
]

def getMsgs():
    return {'index': 0}

Chat Client

Chat Client

> [C] Hello world
> [B] I m here too
history = ['[C] Hello world', '[B] Im here too']

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

newMsg
msg = text
user = user

getMsgs
index = start_index
Let's take it out for a spin:

```
chat_server.py
```
Optional web server for Assign #7
Learning Goals

1. Write a chat program that can respond to internet requests
Life After CS106A!

Is there any?
“Any sufficiently advanced technology is indistinguishable from magic.”

—Arthur C. Clarke
Test Run Video

“It’s a no-brainer that 50 to 60 years from now, cars will drive themselves”

—Sebastian Thrun

Faculty director, “Junior” autonomous car project quoted in Forbes, May 11, 2011
“Nevada has become the first state to issue an ‘autonomous’ license for a driverless car”

—USA Today, May 8, 2012
Google Self-Driving Car on El Camino Real
August 2015
Autonomous Waymo minivan in Los Altos
November 2017
Computing as a Growth Accelerant
“Google Data Center” Circa 1997

Image courtesy of Google
Creating awareness of "CS in the large"

Computing is increasingly needed for work in other fields

Providing context for computing
  - Programming is a *means*, not an *end*

It’s about empowerment!
Editor’s Note: Two-dimensional projection clearly does not capture the relative importance or organizational nuances of the field. Some topics may be closer to you than they appear on this slide.
CS Major Allows Exploration

- Computer Vision
- Systems
- Algorithms
- AI
- Data mining
- Robotics
- Databases
- Computer Vision
- Machine Learning
- Natural Language
- Comp. Bio.
- Comp. Economics
- Theory
- Geometric Comp.
- Networking
- Security
- Graphics
- Hardware
- Distributed Systems
- HCI
...in a Diverse Set of Areas
The "Big Tent" of Computer Science
CS Core Course Sequence

You are here (almost)

CS106A
Programming Methodology

CS106B/X
Programming Abstractions

CS107
Computer Organization and Systems

CS110
Principles of Computer Systems

CS103
Math. Foundations of Computing

CS109
Probability for Computer Scientists

CS161
Data Structures and Algorithms
Track Areas

- Artificial Intelligence
- Theory
- Systems
- Computer Engineering
- Human-Computer Interaction
- Graphics
- Information
- Biocomputation
  - Incorporates many pre-medical school requirements
- Unspecialized
- Individually Designed
Sample of CS Research Areas

- Artificial Intelligence
  - Robotics, machine learning, computer vision, ...

- Computational Biology
  - Bioinformatics, genomics, drug design, ...

- Graphics
  - Animation, modeling, motion capture, architecture, ...

- Databases and information systems
  - Web search, transaction management, data integrity, ...

- Security
  - Cryptography, secure protocols, ...

- Systems
  - Network design, cloud computing, virtualization, ...

- Human-Computer Interaction
  - Interface design, user-centric computing, ...
Sampling of Career Paths

• High-tech industry
  – Research and development
  – Engineering management
  – Product management

• Entrepreneurship (consider Mayfield Fellows Program)
  – Start-ups (over 2,500 companies found by Stanford community)
  – Venture capital

• Graduate and professional schools
  – Graduate school → Academia/research/teaching
  – Law school → Public policy (consider CS181/182)
  – Business school → Management/entrepreneurship

• Teaching (consider CS198)
CS Minor

• Math through Math 51

• Required:
  – CS106B
  – CS107
  – CS103
  – CS109

• Two additional CS elective courses
Math 51 + 2 CS electives (which could be CS110 and CS161, among others)

You are here (almost)

CS106A Programming Methodology

CS106B/X Programming Abstractions

CS107 Computer Organization and Systems

CS103 Math. Foundations of Computing

CS109 Probability for Computer Scientists

CS110 Principles of Computer Systems

CS161 Data Structures and Algorithms
Related Majors

• Math and Computational Science (Data Science)
  – Math, CS, Statistics, and MS&E, ...
  – Tracks in Biology, Engineering, Statistics

• Electrical Engineering
  – Hardware, information science, analog/physical systems, ...
  – Tracks in: areas above as well as Bio-EE, Green-EE, Music-EE

• Symbolic Systems
  – CS, Philosophy, Linguistics, Psychology. ...
Everyone is Welcome
A Now Some Fun Movies...

- Lighthouse
- Fireball
- Curtain
- Robot