Dictionaries and HTML

In this lecture, we will cover:

- python dictionaries
- World Wide Web + HTML

Your future in CS

I used to include this on my slides, but since these slides have changed - going to just leave it up here for every notebook. I get a lot of questions about more programming courses, the concentrations, and minors in computer science. Here is a brief reminder.

CS 164 – Next Course In Sequence, also consider CS 220 (math and stats especially)

- CO Jobs Report 2021 – 77% of *all* new jobs in Colorado require programming
- 60% of all STEM jobs requires *advanced* (200-300 level)
- 31% of all Bachelor of Arts degree titled jobs also required coding skills
- 2016 Report found on average jobs that require coding skills paid $22,000 more

- Concentrations in CS:
    - Computer science has a number of concentrations.
        - General concentration is the most flexible, and even allows students to double major or minor pretty easily.
        - Software Engineering
        - Computing Systems
        - Human Centered Computing
        - Networks and Security
        - Artificial Intelligence
        - Computer Science Education.
    - Minors:
        - Minor in Computer Science - choose your own adventure minor
        - Minor in Machine Learning - popular with stats/math, and engineering
        - Minor in Bioinformatics - Biology + Computer Science

Python Dictionaries

Let's think about a routing table:

- www.cs.colostate.edu is the name for 129.82.45.48
- www.colostate.edu is the name for 129.82.103.64
- www.google.com is the name for 142.250.72.46

- www.amazon.com is the name for 13.33.253.2
- www.microsoft.com is the name for 23.58.117.164

Wouldn't it be great if we could store this information, so we just search for the website, and the IP address is the answer given?

Flipping the other direction, think about a class roster. Let's say every student has an ename, and then a group of data is stored about the statement?

- apond references information about Amy Pond, class, grades, etc
- rwilliams references information about Rory Williams, class, grades, etc.

Essentially, we are **mapping** data to a key:value pair. This is a python dictionary.

In [ ]:
```
routes = {"www.cs.colostate.edu": "129.82.45.48",
          "www.colostate.edu": "129.82.103.64",
          "www.google.com": "142.250.72.46",
          "www.amazon.com": "13.33.253.2",
          "www.microsoft.com": "23.58.117.164"} #notice format it uses key:value

print(routes["www.cs.colostate.edu"])
print(routes["www.google.com"])

## what if I need to change a value?

routes["www.cs.colostate.edu"] = "129.82.45.50"

print("updated route!", routes["www.cs.colostate.edu"])
```

```
129.82.45.48
142.250.72.46
updated route! 129.82.45.50
```

We can also store any type as the value.

In [ ]:
```
students = {} # creating an empty dictionary, going to add students to it

students["apond"] = ('Amy Pond', 'apond', 123456789, 'apond@the60s.com')  ## this is a
students["rwilliams"] = ['Rory Williams', 'rwilliams', 987654321, 'rwilliams@thefirstc
students["drdonna"] = {"name": "Donna", "id": 135792468, "email": "doctordonna@mixed.c

print(students) # yes printing the whole dictionary like we do lists
print()
print("One item of the dictionary=>", students["drdonna"])
print("One element of the single item in the dictionary=>", students["drdonna"]["name"
```

```
{'apond': ('Amy Pond', 'apond', 123456789, 'apond@the60s.com'), 'rwilliams': ['Rory W
illiams', 'rwilliams', 987654321, 'rwilliams@thefirstcentury.com'], 'drdonna': {'nam
e': 'Donna', 'id': 135792468, 'email': 'doctordonna@mixed.com'}}

One item of the dictionary=> {'name': 'Donna', 'id': 135792468, 'email': 'doctordonna
@mixed.com'}
One element of the single item in the dictionary=> Donna
```

In class activity

- Create a dictionary for the people at your table.
- Create 'key' for each of you, (ename, nickname, etc)
- For each of you, create another dictionary that contains the following information (at a minimum)
  - name, fav_color, fav_fruit, fav_movie
- Print out each

Example:

```
team_mystery["scooby"] = {"name": "Scoobert Doo", "fav_color": "gray",
"fav_fruit": "All Fruits", "fav_movie": "Nothing scary!"}
```

print out the list when done

In [ ]:
```
## student activity here

team_mystery = {}

team_mystery["scooby"] = {"name": "Scoobert Doo", "fav_color": "gray",
                          "fav_fruit": "All Fruits", "fav_movie": "Nothing scary!"}

team_mystery["velma"] = {"name":"Velma Dinkley","fav_color":"Orange", "fav_fruit":"Pir

print(team_mystery)
```

```
{'scooby': {'name': 'Scoobert Doo', 'fav_color': 'gray', 'fav_fruit': 'All Fruits',
'fav_movie': 'Nothing scary!'}, 'velma': {'name': 'Velma Dinkley', 'fav_color': 'Oran
ge', 'fav_fruit': 'Pineapple', 'fav_movie': 'Death on the Nile'}}
```

Dictionaries - What about looping through the items?

- dict.keys() - give list of keys
  - in no particular order!
- dict.values() - give items *without keys*
- dict.items() - gives tuple of (key, value)

In [ ]:
```
authorized = {"Valtaja": ["Cut & Thrust", "Rapier Spear", "Rapier"],
              "Aegeon":  ["Cut & Thrust", "Rapier Spear", "Rapier", "Armored Combat",
              "Felix":   ["Armored Combat", "Great Weapons"]}


for key, value in authorized.items():
    print("{} is authorized in {}".format(key, ", ".join(value)))
```

```
Valtaja is authorized in Cut & Thrust, Rapier Spear, Rapier
Aegeon is authorized in Cut & Thrust, Rapier Spear, Rapier, Armored Combat, Great Wea
pons
Felix is authorized in Armored Combat, Great Weapons
```

Student Activity

Take the list you just generated and print out only the favorite colors or favorite fruits in the format

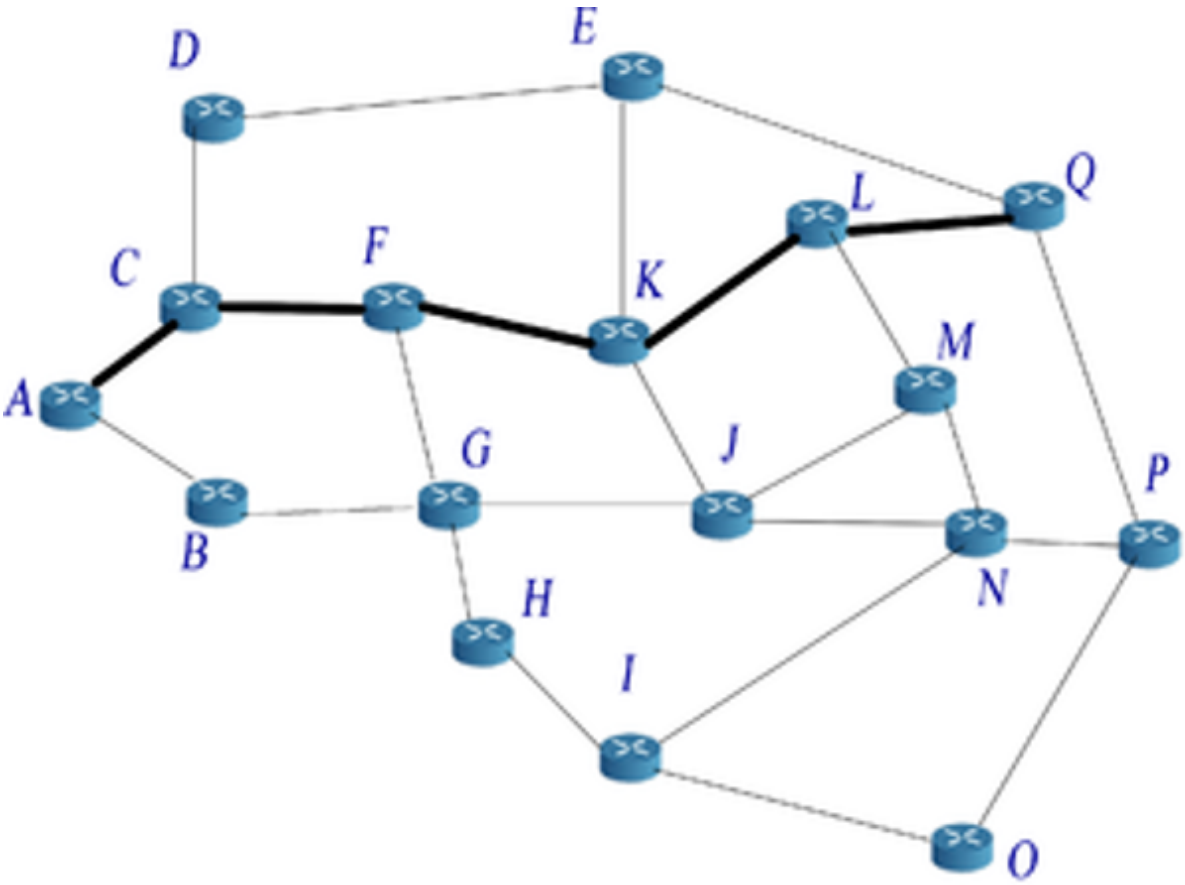
name likes fruit

In [ ]:
```
for value in team_mystery.values(): ## since i only needed the values in this one case
    print(f"{value['name']} likes {value['fav_fruit']}")
```

```
Scoobert Doo likes All Fruits
Velma Dinkley likes Pineapple
```

HTML and the World Wide Web

- Layers
  - Physical Wires transmitting bits
  - IP and DNS - figuring out where to go
  - Routing and TCP helps transfer packets
  - HTTP/S - helps determine files
    - World Wide Web (WWW)
    - Tim Berners Lee (WWW Consortium)
    - Marc Andreesen (Netscape)
    - CERN Laboratories - first Website
    - End of 1993 - 26 websites in existence
    - End of 1994 - Amazon, 1 million browsers
  - By 1998 - 750,000 commercial sites
- All of this is based on the layers - files transferred through multiple computers



Code.org Video on HTML and WWW

Hyper Text Transfer Protocol

- Layer on top of the internet
- Focuses on transferring files
  - Browsers take all the files, and present the webpages!
  - Uses Hypertext Markup Language (HTML)
    - Realistically, just a text file with a set format!

```
<html>
   <head>
     <title>This is the document title</title>
   </head>
   <body>
      <h3>This is a heading - a rank 3 heading</h3>
      <p>Hello this is a paragraph.</p>
      <ul>
         <li>this is a list item</li>
         <li>Another list item</li>
      </ul>
    </body>
</html>
```

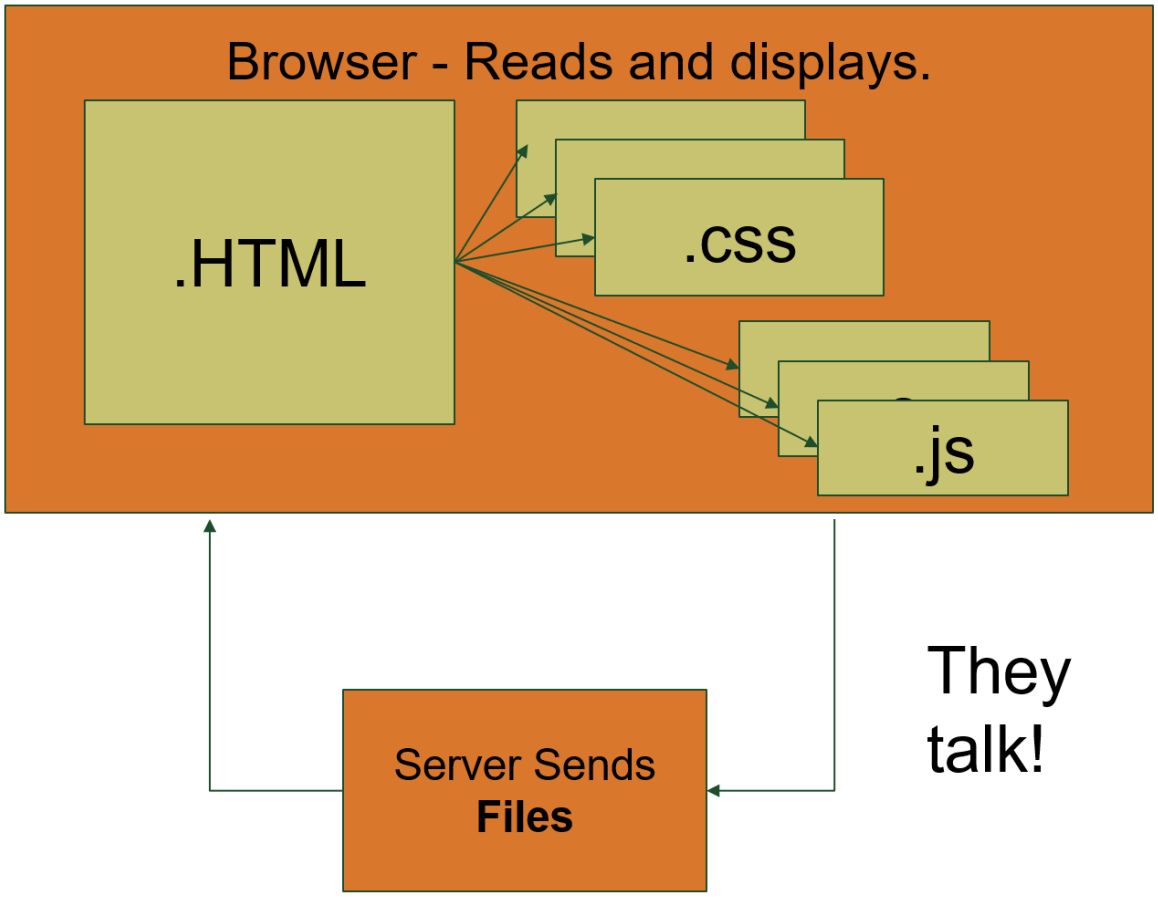
This is a heading - a rank 3 heading

Hello this is a paragraph.

- this is a list item
- Another list item

Adding Style: Cascading Style Sheets and Javascript

```
<div style="background-color:green;color:white;padding:10px 10px 10px
10px;">
    <h3 style="color:yellow">This should be yellow</h3>
    <p>This will be white</p>
</div>
```

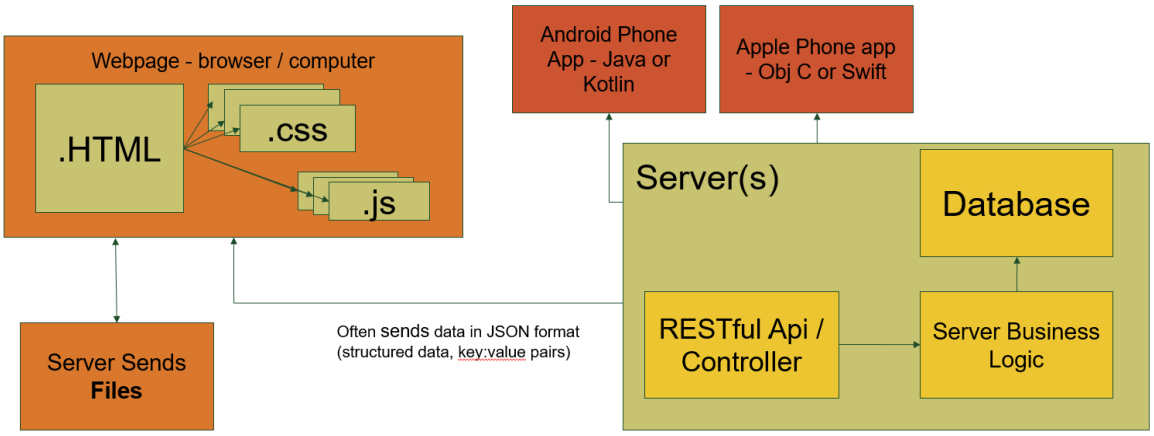


- The internet has evolved
  - HTML pages (Semantic Hints)
  - Cascading Style Sheets (Look and Feel)
    - Color, semantics, and some layout
  - Javascript (Functionality)
    - ECMAScript !
    - Javascript is not Java
  - Your computer talks to another computer
    - Sending files back and forth!
    - Your browser displays the file
    - Similar to how a word doc displays files

Modern Web Applications

- Server applications process data and information
  - Written in multiple languages
    - Python is very popular
    - Java is popular
    - Kotlin is popular
    - Typescript using Node
  - Generate strings to send across the internet
  - Often "dictionary like" strings (JSON)

- Browsers process the javascript
  - Javascript handles interactions
  - The webpage then becomes a way to present the information and events to the client
    - Essentially, a user-interface!
    - Human Centered Computing or Software Engineering are good concentrations for people interested in this!



Overview

- Modern webpages are applications
- Interactions between different components
  - Server, Browser, and the client!
  - This all works because of the **layers** of the internet