INFO5002: Intro to Python for Info Sys

Week 8

Slides created by: Zachary Doucet

Northeastern University

Week 8

I. API

II. Regular Expressions

III. Internet

IV. HTML

Recap

Unit tests

- Test individual smallest functioning with hopefully maximal coverage.
- assertEqual(a, b): see if a == b
- assertRaises(ExceptionType)
- Create a class inheriting from unittest.TestCase with every unit test as a funciton starting with **test_**

I/O

- file = open(filename, modes)
- modes: r, w, x, a, b, + (suffix)
- for line in file:
- write(data): write data
- Read CSV with csv.reader or csv.DictReader
- Read and write Json with json.load and json.dump

Web Scraping

- Sometimes need to scrape from web.
- Use requests to fetch webpage by passing url and BeautifulSoup4 to parse web page by searching the text through tags, classes, and ids.

API Integration

PCC 355-368

To cover

- HackerNews API 368-371

Interface between codebases

- We have seen loading data through files, csv, json, and html => All different formats.
- A general format of sharing information is termed an API.
- API is **user defined** and format shared w/ others to use.

Hacker News API

1) Read the docs.

2) Discover what to call to get the info you want.

3) Learn how to call such an API.

4) Learn what is the output of the API.

Items

Stories, comments, jobs, Ask HNs and even polls are just items. They're identified by their ids, wh
unique integers, and live under `/v0/item/<id>.json`.

All items have some of the following properties, with required properties in bold:

Field	Description
id	The item's unique id.
deleted	`true` if the item is deleted.
type	The type of item. One of "job", "story", "comment", "poll", or "pollopt".
by	The username of the item's author.
time	Creation date of the item, in Unix Time.
text	The comment, story or poll text. HTML.
dead	`true` if the item is dead.
parent	The comment's parent: either another comment or the relevant story.
poll	The pollopt's associated poll.
kids	The ids of the item's comments, in ranked display order.
url	The URL of the story.
score	The story's score, or the votes for a pollopt.
title	The title of the story, poll or job. HTML.
parts	A list of related pollopts, in display order.
descendants	In the case of stories or polls, the total comment count.

For example, a story: https://hacker-news.firebaseio.com/v0/item/8863.json?print=pretty

```python
import requests

url = "https://hacker-news.firebaseio.com/v0/item/8863.json"

response = requests.get(url)
print(f"Response code: {response.status_code}")

response_dict = response.json()
print(f"Response: {response_dict}")

title = response_dict["title"]
print(title)
```

Let’s practice

Hacker News also has an API endpoint “https://hacker-news.firebaseio.com/v0/topstories.json” which returns the top news stories id’s as an array. Using this and the previous “https://hacker-news.firebaseio.com/v0/item/{ID}.json” endpoint find all the top news titles and by which author.

Regular Expressions

https://docs.python.org/3/howto/regex.html

A mini language in Python

- Regular expressions (regex) is a language applied to **strings** that allow for the generation of rules.
- Regex makes it easy to verify if user input is of a certain type. e.g. an email address should have an “@” and a “.*{something}*”.

Patterns

- Characters match themselves except for a few key characters which are called metacharacters.
  “test” matches any string that is exactly “test”, “ test” not good.
- You can define an option (character class) with brackets.
  Metacharacters don’t mean anything inside brackets.
  “[ab][de]” matches strings “ad”, “ae”, “bd”, “be”
- Can complement the character class with ^ at start.
  “angel[^a]” matches everything that begins with “angel” except for “angela”

- and \

- You can create a range with **-**.

  ”[a-z]”

- \ is an escape character and allows you to match any of the escape characters by prefixing it.

  ”\[” “\\”

Common escapes

- \d: any decimal digit: [0-9]
- \D: any non digit: [^0-9]
- \s: any whitespace character: [\t\n\r\f\v]
- \S: any non-whitespace character: [^ \t\n\r\f\v]
- \w: any alphanumeric character: [a-zA-Z0-9_]
- \W: any non-alphanumeric character: [^a-zA-Z0-9_]

Repeating

- *: zero or more times.
- +: one or more times.
- ?: zero or one times.
- {m,n}: at least m and at most n.

  ”a/{1,3}b” matches “a/b”, “a//b”, “a///b” but not “a////b”
- {m}: exactly m times

In Python

Raw string notation

Pattern object

```
import re

p = re.compile(r"\w*[^z]")
print(p.match("hellotherez"))
```

```
<re.Match object; span=(0, 10), match='hellothere'>
```

Matching

- match(): determine if exists a match from start of string.
- search(): find any starting location that matches.
- findall(): find all substrings that match and return as list.
- finditer(): find all substrings that match and return as an iterator.

Match Objects

- match() and search() return either None or a Match object.
- Match objects have following methods:
  - group(): return the matched string
  - start(): return starting pos of match.
  - end(): return ending pos of match.
  - span(): tuple of (start, end)

Metacharacters+

- |: OR: “A|B” => “A”, “B”
- ^: beginning of the line (start of string): “^Hello” => “Hello”
- $: end of the line (end of string): “Bye$” => “Bye”

Let’s practice

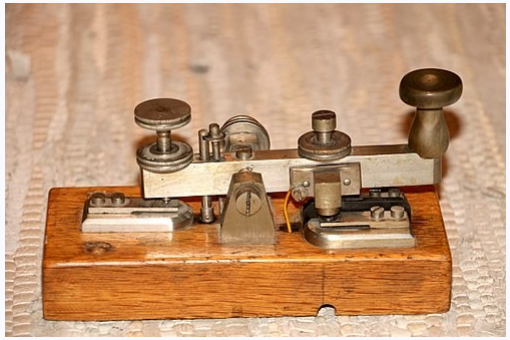
- Create a function **validate_email** which takes in an email as a string and uses regular expression to validate if it is a properly formatted email.
- Create a function **validate_phone_number** which takes in a phone number as a string and makes sure that it follows the Canadian convention of (XXX) XXX-XXXX.

Internet

Before the internet, there was...

Source: Wikimedia

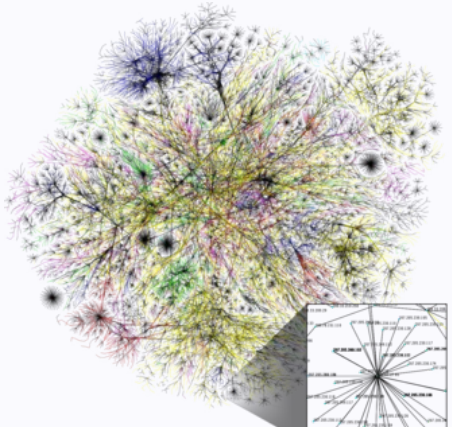
Source: Wikimedia



Source: Wikimedia

Source: Wikimedia

Source: Wikimedia

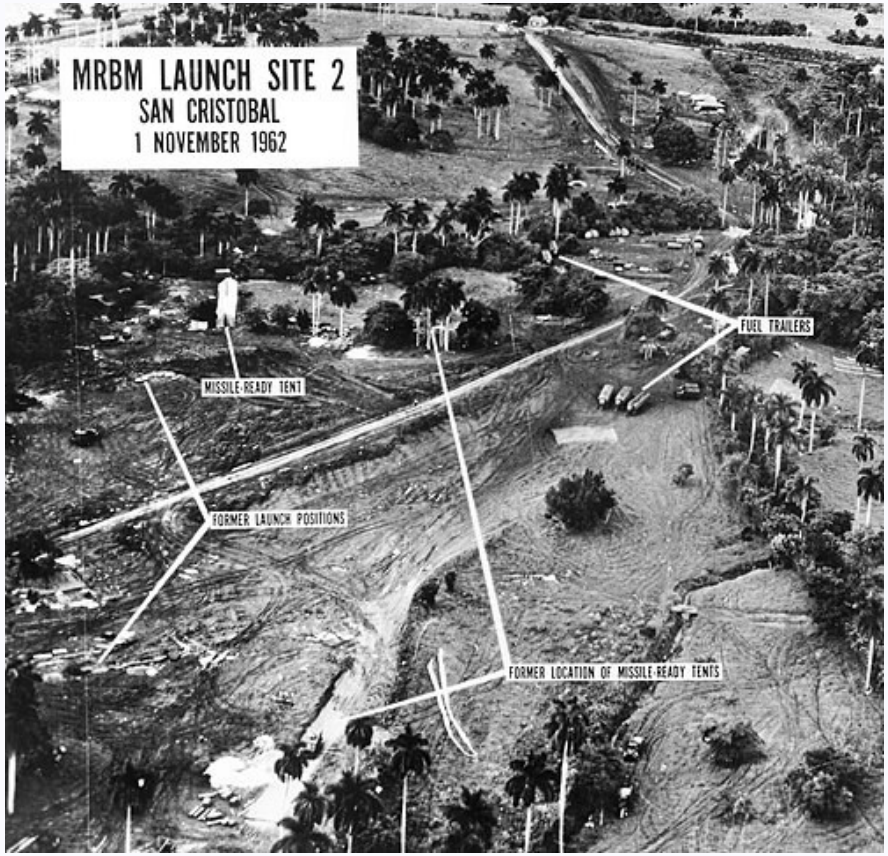


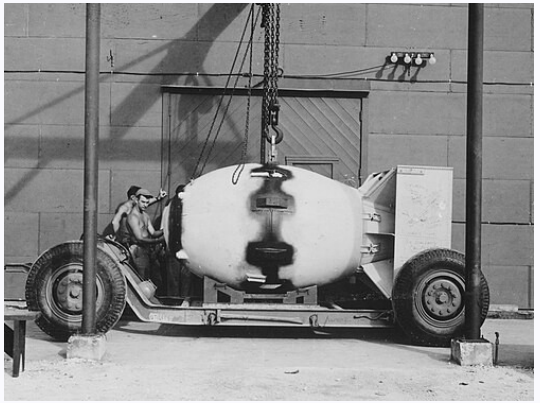
Source: Wikimedia

Tensions were high

Source: Wikimedia

Too high

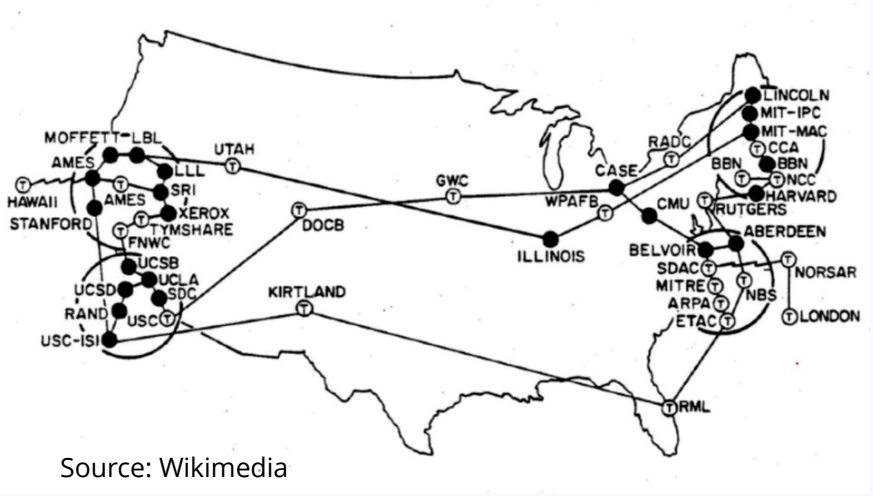




Source: Wikimedia

For resilience

- The United States military needed to ensure the continuation of government in case of a nuclear detonation over Washington DC.
- They developed ARPANET.



Source: Wikimedia

HTML

MDN

The layout

- **HTML** describes the layout of a webpage. i.e. What is the content? How is that content organized?
- The webpage is divided into tags which instructs the browser how to render the content.
- Tags are defined with angled brackets <tag> while a closing tag is defined with angled brackets with a slash </tag>.

Basic structure

Tells the browser the information it will received next is HTML data.

```
<!DOCTYPE html>
<html>
  <head></head>
  <body></body>
</html>
```

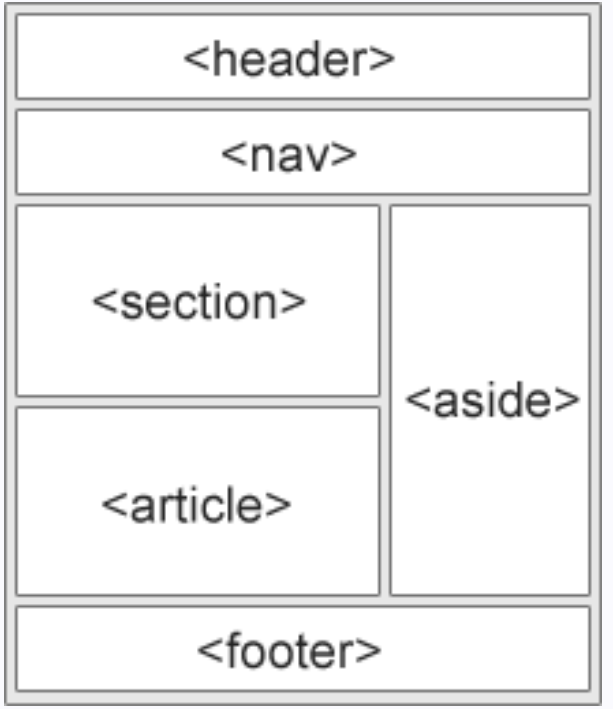
Start of html tag

Start of header section (metadata)

Start of body section (what is visualised on screen)

HTML5 semantic elements

- header: very top of page.
- nav: below header. The navbar.
- main: the main part.
- article: independent self-contained data.
- section: defines a section.
- aside: define content aside some other content.
- footer: very bottom of page.



Source: W3Schools

Text content

- Headers: h1-h6: represent sectional heading.

```
<h1>h1</h1>
```

- Paragraphs: represent regular text.

```
<p>Hello world!</p>
```

Hello World

```
<div>
  <p>This is a paragraph</p>
  <p>This is another</p>
</div>
```

- Group with div (not displayed but makes it cleaner).

Create lists

```
<h1>My favourite fruits</h1>
<ul>
    <li>Apples</li>
    <li>Bananas</li>
    <li>Organges</li>
</ul>
```

Unordered list


```
<h1>Make my famous pie</h1>
<ol>
    <li>Preheat at 500F</li>
    <li>Mix wet</li>
    <li>Mix dry</li>
    <li>Slowly blend wet into dry</li>
    <li>Knead</li>
    <li>Place in oven for 25 mins</li>
</ol>
```

Ordered list

Connecting webpages

```
<a href="https://www.google.ca">Google</a>
```

Connect to another page on the web

```
<a href="home.html">Home</a>
```

Connect to a home.html page on the current website
e.g. If I am on domain.com/hello.html and click on this link it will take me to domain.com/home.html

```
<h1>Search Engines</h1>
<ul>
    <li><a
href="https://www.google.ca">Google</a></li>
    <li><a href="https://www.bing.com">Bing</a></li>
    <li><a
href="https://www.duckduckgo.com">DuckDuckGo</a></li
>
</ul>
```

Images

Where: local or remote

Display if not found

```
<img
  src="scherzer.jpg"
  alt="Max Scherzer in complete shock"
  width="236"
  height="206"
/>
```

Width in px

Height in px


```
<img
   src="thisimagenotexist.jpg"
   alt="A beautiful dog jumping through a hoop!"
   width="1920"
   height="1080"
/>
```

A beautiful dog jumping through a hoop!

Tables

- <table></table>: creates a table.
- <tr></tr>: table row.
- <td></td>: cell.
- <th></th>: used instead of td for header.

```
<table>
  <tr>
    <th>Team</th>
    <th>W</th>
    <th>L</th>
  </tr>
  <tr>
    <td>Toronto Blue Jays</td>
    <td>94</td>
    <td>68</td>
  </tr>
  <tr>
    <td>New York Yankees</td>
    <td>94</td>
    <td>68</td>
  </tr>
  <tr>
    <td>Boston Red Sox</td>
    <td>89</td>
    <td>73</td>
  </tr>
</table>
```

Team	W	L
Toronto Blue Jays	94	68
New York Yankees	94	68
Boston Red Sox	89	73

Forms: collect user data

- <form></form>: create a form
- <input type=”” />
  - type=”text”: single-line input
  - type=”radio”: selecting one of many choices
  - type=”checkbox”: one or more selections
  - type=”submit”: submit button for the form
  - type=”button”: display clickable button

- <label for=””></label>: creating an input for input id=for
- <select><option></option></select>: dropdown menu
- <textarea rows=”” cols=””></textarea>: multi-line resizable input field.
- <button type=””></button>: define a clickable button
  - type=”button”: clickable
  - type=”submit”: submits form
  - type=”reset”: re-initialises form

All input types

- button
- checkbox
- color
- date
- datetime-local
- email
- file
- hidden
- image
- month
- number
- password
- radio
- range
- reset
- search
- submit
- tel
- text
- time
- url
- week

Form submissions

- Form has an action attribute where you give the local or remote path of what to call to give the data to. Data is passed as a dictionary with the keys as the inputs' name attribute.
- Form's target decides where to display result.
  - _blank: new window or tab
  - _self: current window
  - _parent: in the parent's frame
  - _top: full body of window

```
<form action="/scripts/process-form.py" target="_blank">
    <h1>Welcome Form</h1>
    <h2>Part I</h2>
    <label for="fname">First Name</label>
    <input type="text" id="fname" name="fname"/>
    <label for="lname">Last Name</label>
    <input type="text" id="lname" name="lname" />
    <h2>Part II</h2>
    <input type="radio" id="red" name="fav_colour" value="red" />
    <label for="red">Red</label>
    <input type="radio" id="green" name="fav_colour" value="green" />
    <label for="green">Green</label>
    <input type="radio" id="blue" name="fav_colour" value="blue" />
    <label for="blue">Blue</label>
    <h2>Part III</h2>
    <input type="checkbox" id="dog" name="animal1" value="dog" />
    <label for="dog">I have a dog</label>
    <input type="checkbox" id="cat" name="animal2" value="cat" />
    <label for="cat">I have a cat</label>
    <input type="checkbox" id="hamster" name="animal3" value="hamster" />
    <label for="hamster">I have a hamster</label>
    <h2>Finish</h2>
    <button type="submit">Submit Form</button>
</form>
```

Welcome Form

Part I

First Name Last Name

Part II

Red Green Blue

Part III

I have a dog I have a cat I have a hamster

Finish

Submit Form

Metadata

- Metadata is stored in between the head tags.

```
<head>
 <title>Mortgage Calculator</title>
 <meta charset="UTF-8">
 <meta name="description" content="Calculate mortgage payment">
 <meta name="keywords" content="Mortgage, Interest, Principal">
 <meta name="author" content="John Doe">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

Let’s practice

You are to design a personal or fictional website about yourself or a fictitious character. There will be at least 3 pages: a home page, an about page, and a contact page. The contact page should feature a form that will send the request to “localhost:5000”. Keep it quick and simple. Simply practice using the different tags! Nothing fancy here.