Webscraping

Peter Ganong and Maggie Shi

November 6, 2024

Table of contents I

Recap and Outline

Last class, we covered:

- Intro to HTML: tags, attributes, and content
- Using `BeautifulSoup` to parse HTML and extract text

Recap and Outline

Last class, we covered:

- Intro to HTML: tags, attributes, and content
- Using `BeautifulSoup` to parse HTML and extract text

Today, we'll cover:

- Making a web scraper dynamic so that it can "crawl" the web
- Extracting and organizing data from *tables* on webpages
- Webcraping etiquette

Web Crawlers

“Crawling” the Web: Intro

- One powerful way to harness web scraping tools is to use them to ‘crawl’ the web – that is, to automatically navigate from page to page

Roadmap

- Discuss use cases for web crawling
- Walk through an example of developing a Wikipedia web crawler

Use cases for web crawling

- Web crawl when you want to visit multiple URLs
- And the URLs are located in a structured or repetitive way
- This is especially useful if you don't know the exact URLs, but know that they link to each other

Example: scraping from a list of press releases

WHO Africa

News Releases

Scaling up response to curb growing mpox outbreak in African region

15 August 2024

Brazzaville – As the mpox outbreak that has affected the Democratic Republic of the Congo and spread to neighbouring countries continues to grow, World Health Organization (WHO) is intensifying support to countries to scale up measures to curb the virus and save lives.

Read more »

Kenya: Strengthening the health workforce

12 August 2024

Nairobi – Five years ago, Esther Omagwa was one of only two nurses at the Railways Health Centre in Kisumu County, western Kenya. The immense workload often exhausted her, forcing her to turn away clients.

Today, the scene at the community health centre is remarkably different. With the nursing staff now expanded to a team of four, Omagwa and her colleagues are better equipped to deal with the load.

Read more »

African region faces an unprecedent surge in mpox cases

08 August 2024

The African region is experiencing an unprecedented increase in mpox cases since the start of 2024, with more countries previously unaffected by the disease reporting cases in an expanding spread of the virus.

Read more »

Example: navigating from a directory with links

Journal List > Bull World Health Organ

Bulletin of the World Health Organization

See also Bulletin of the World Health Organization Supplement

Bulletin of the World Health Organization Vols. 1 to 102; 1948 to 2024			
Vol. 102 2024	v.102(1): 1–84 2024 Jan 1	v.102(2): 85–148 2024 Feb 1	v.102(3): 149–224 2024 Mar 1
	v.102(4): 225–296 2024 Apr 1	v.102(5): 297–372 2024 May 1	v.102(6): 373–456 2024 Jun 1
	v.102(7): 457–552 2024 Jul 1	v.102(8): 553–620 2024 Aug 1	

Example: Wikipedia Crawler

- Apparently, following the first link in the main text of a Wikipedia article repeatedly will lead to the 'Philosophy' page 97% of the time! (link)
- Let's create a web crawler to test this out, starting from the 'United States' Wikipedia page: https://en.wikipedia.org/wiki/United_States (link)

Example: Wikipedia Crawler

- Apparently, following the first link in the main text of a Wikipedia article repeatedly will lead to the 'Philosophy' page 97% of the time! (link)
- Let's create a web crawler to test this out, starting from the 'United States' Wikipedia page: https://en.wikipedia.org/wiki/United_States (link)
- *(Note: in the future, you will likely not be using web crawling in this way. However, we're going to use this as a simple example to illustrate the mechanics of setting up a web crawler. Wikipedia is a good place to do this because they won't block us if we access it too often.)*

Steps We Want our Crawler to Take

- Step 1: Inspect the HTML of the first Wikipedia page we want to scrape

Steps We Want our Crawler to Take

- Step 1: Inspect the HTML of the first Wikipedia page we want to scrape
- Step 2: Parse HTML from first page and extract the first link

Steps We Want our Crawler to Take

- Step 1: Inspect the HTML of the first Wikipedia page we want to scrape
- Step 2: Parse HTML from first page and extract the first link
- Step 3: Follow that link

Steps We Want our Crawler to Take

- Step 1: Inspect the HTML of the first Wikipedia page we want to scrape
- Step 2: Parse HTML from first page and extract the first link
- Step 3: Follow that link
- Step 4: If not on the Philosophy page, repeat the previous step

Steps We Want our Crawler to Take

- Step 1: Inspect the HTML of the first Wikipedia page we want to scrape
- Step 2: Parse HTML from first page and extract the first link
- Step 3: Follow that link
- Step 4: If not on the Philosophy page, repeat the previous step
- Step 5: build in conditions to stop the loop
  - In addition to checking for Philosophy
  - To prevent an infinite loop, we will also stop if if we go to a link we've been to before
  - Or if we've visited 50 links

Step 1: Inspect website and HTML

- Start at https://en.wikipedia.org/wiki/United_States (link)

United States

Article Talk Read View

From Wikipedia, the free encyclopedia

Several terms redirect here. For other uses, see America (disambiguation), US (disambiguation), USA, United States of America (disambiguation), and United States (disambiguation).

The **United States of America** (**USA** or **U.S.A.**), commonly known as the **United States** (**US** or **U.S.**) or **America**, is a country primarily located in North America. It is a federal union of 50 states, which also includes its federal capital district of Washington, D.C., and 326 Indian reservations.[j] The 48 contiguous states are bordered by Canada to the north and Mexico to the south. The State of Alaska is non-contiguous and lies to the northwest, while the State of Hawaii is an archipelago in the Pacific Ocean. The United States also asserts sovereignty over five major unincorporated island territories and various uninhabited islands.[k] The country has the world's third-largest land area,[d] second-largest exclusive economic zone, and third-largest population, exceeding 334 million.[l]

United

Flag

Motto

Other

Anthem: "T

- The first in-text link is to "North America", so we should check for that

Step 1: Inspect website and HTML

- Right click + Inspect...

```
▼<p>
    "The "
    <b>United States of America</b>
    " ("
    <b>USA</b>
    " or "
    <b>U.S.A.</b>
    "), commonly referred to as the "
    <b>United States</b>
    " ("
    <b>US</b>
    " or "
    <b>U.S.</b>
    ") or "
    <b>America</b>
    ", is a nation primarily situated in "
    <a href="/wiki/North_America" title="North America">North America</a> == $0
    ". It is a "
    <a href="/wiki/Federation" title="Federation">federal union</a>
    " comprising 50 "
    <a href="/wiki/U.S._state" title="U.S. state">states</a>
    ", along with the "
```

- The main text links appear to be nested inside an <a> tag which is in a <p> tag

Step 2: Parse HTML from first page and extract the first link

```
url = "https://en.wikipedia.org/wiki/United_States"
response = requests.get(url)

soup = BeautifulSoup(response.content, 'lxml')
```

Step 2: Parse HTML from first page and extract the first link

- Start by extracting anything with a p tag

```
p_tag = soup.find_all('p')
```

Step 2: Parse HTML from first page and extract the first link

```
my_links = []                                   #1
for p in p_tag:                                 #2
    links = p.find_all('a', href=True)          #3
    for link in links:                          #4
        my_links.append(link.get('href'))       #5
```

1. Create an empty list to store links
2. Then loop through the `p` tags
3. Find the `a` tags with `href` attribute
4. Loop through the `a` tags
5. Append the value of the attribute to `my_links`

Step 2: Parse HTML from first page and extract the first link

```
my_links = []                                   #1
for p in p_tag:                                 #2
    links = p.find_all('a', href=True)          #3
    for link in links:                          #4
        my_links.append(link.get('href'))       #5
```

1. Create an empty list to store links
2. Then loop through the `p` tags
3. Find the `a` tags with `href` attribute
4. Loop through the `a` tags
5. Append the value of the attribute to `my_links`

- Note: we have to work with for loops because `.find_all()` returns a *list*

Step 2: Parse HTML from first page and extract the first link

- Let's preview our saved links

```
print(my_links[0:4])
```

```
['/wiki/North_America', '/wiki/Federation', '/wiki/U.S._state', '/wiki/Was
```

Step 2: Parse HTML from first page and extract the first link

- Let's preview our saved links

```
print(my_links[0:4])
```

```
['/wiki/North_America', '/wiki/Federation', '/wiki/U.S._state', '/wiki/Was
```

- The first link is indeed to the "North America" page!
- But it's a *relative* path: the URL should be `'https://en.wikipedia.org/wiki/North_America'`

Step 2: Parse HTML from first page and extract the first link

- Modify our code: before appending to `my_links`, concatenate the relative path to get the absolute path

```
my_links = []
for p in p_tag:
  links = p.find_all('a', href = True)
  for link in links:
    my_links.append('https://en.wikipedia.org' + link.get('href'))

print(my_links[0:2])
```

```
['https://en.wikipedia.org/wiki/North_America', 'https://en.wikipedia.org/w
```

Step 3: Follow the first link

- Then to follow the link we collected, we simply make it the input to another request

```
url = my_links[0]
print(url)
```

```
https://en.wikipedia.org/wiki/North_America
```

- Our new `url` becomes the input into a new request and call to `BeautifulSoup`

```
# repeating the previous step, but with the new URL
response = requests.get(url)
soup = BeautifulSoup(response.content, 'lxml')
```

Step 4: Follow the first link and repeat

```
p_tag = soup.find_all('p')
my_links = []
for p in p_tag:
  links = p.find_all('a', href = True)
  for link in links:
    my_links.append('https://en.wikipedia.org' + link.get('href'))
```

- Check the first link

```
print(my_links[0])
```

```
https://en.wikipedia.org/wiki/Continent
```

Step 4: Follow the first link and repeat

- We should go back to the 'North America' page to confirm that the first link is to 'Continent'

North America

Article Talk

From Wikipedia, the free encyclopedia

> *"North American" redirects here. For other uses, see North American (disambiguation) or Northern United States.*

North America is a continent[b] in the Northern and Western Hemispheres.[c] North America is bordered to the north by the Arctic Ocean, to the east by the Atlantic Ocean, to the southeast by South America and the Caribbean Sea, and to the west and south by the Pacific Ocean. The region includes the Bahamas, Bermuda, Canada, the Caribbean, Central America, Clipperton Island, Greenland, Mexico, Saint Pierre and Miquelon, Turks and Caicos Islands, and the United States.

Step 5: Build in conditions to stop the loop

Conditions to stop the loop:

1. We've hit the Philosophy page
2. We've been to this link before
3. We've visited 50 links

Step 5: Build in conditions to stop the loop

Conditions to stop the loop:

1. We've hit the Philosophy page
2. We've been to this link before
3. We've visited 50 links

- Let's start with the **third condition**: max out at visiting 50 links

```
for i in range(50):
    # code to crawl here
```

Step 5: Build in conditions to stop the loop

- Then the **first condition**: stop if we've hit the Philosophy page

```
for i in range(50):
    # code to crawl here
    if url == "https://en.wikipedia.org/wiki/Philosophy":
        print('Ended up at Philosophy in ' + str(i) + ' tries!')
        break
```

- `break` exits the `for` loop as soon as the condition is triggered

Step 5: Build in conditions to stop the loop

- Then we can tackle the **second condition**: exit if we've visited this link before

```
visited_urls = []
for i in range(50):
    if url in visited_urls:
        print('Stopped because ended up in a loop.')
        break
    # code to crawl here
    if url == "https://en.wikipedia.org/wiki/Philosophy":
        print('Ended up at Philosophy in ' + str(i) + ' tries!')
        break
    visited_urls.append(url)
```

- Initialize an empty list: `visited_urls[]`
- `break` if the url we extract has already been visited
- If none of the conditions to `break` are true, then add this url to `visited_urls`

Web Crawler: Summary

- We can go beyond scraping from individual URLs by crawling
- Crawler pulls in URLs stored in a tags
- Crawling requires using loops, so we have to design the crawler carefully to avoid infinite loops

Scraping Data from Tables

Extracting Data from Tables: Intro and Roadmap

- One important use case for web scraping is to **automatically extract and save data from tables**
- When should I scrape a table?
- Walk through example: national exam results from Tanzania

When to Scrape?

- If your goal was just to scrape this particular table one time, copy-and-pasting is fine.
- You should webscrape if:
  - Your tables are spread across multiple pages: example with Tanzania (link)
  - Data tables are using "lazy-loading", so you have to scroll over and over again to "Show More": example with ESPN (link)
- Webscraping should be a tool of last resort!
  - If there is an API, use it
  - If there is a portal to download data, use it

Example: A Scrapable Table

- Example: Tanzania National Examination outcomes (link)

maktaba.tetea.org/exam-resu ×

maktaba.tetea.org/exam-results/FTNA2015/S0101.htm

THE NATIONAL EXAMINATIONS COUNCIL OF TANZANIA
FORM TWO NATIONAL ASSESSMENT (FTNA) 2015 RESULTS
CENTRE: S0101 - AZANIA SECONDARY SCHOOL

School Overall Grade Summary

	A	B+	B	C	D	E	F
M	554	531	423	470	459	511	424
TOT	554	531	423	470	459	511	424

CNO	REPEATER	NAME OF CANDIDATE	SEX	CIVICS	HISTORY	GEOGRAPHY	B/KNOWL	E/D/KISLAMU	FINE ARTS	PHY EDU	KISWAHILI	ENGLISH	FRENCH	PHYSICS	CHEMISTRY	BIOLOGY	INFO & COMP STUD	B/MATH	COMERCE	B/KEEPING	GPA	CLASS
0001		ABDALLAH KOMBO ABDALLAH	M	C	C	C				C	C	B+		E	D	C		E	E	D	2.3	CREDIT
0002		ABDALLAH MWALIMU MWINYIKONDO	M	C	C	E					D	B+		E	F	D	F	F	E	D	1.6	CREDIT
0003		ABDUL KARIM AZIZ	M	B	D	D					D	A		F	F	C		F	E	E	1.9	CREDIT

- As of Sep 2024, ChatGPT could not write code to scrape this page (but maybe by the time we give this lecture maybe it will be able to...)

Example: A Scrapable Table

- Example: Tanzania National Examination outcomes (link)

maktaba.tetea.org/exam-resu

maktaba.tetea.org/exam-results/FTNA2015/S0101.htm

THE NATIONAL EXAMINATIONS COUNCIL OF TANZANIA
FORM TWO NATIONAL ASSESSMENT (FTNA) 2015 RESULTS
CENTRE: S0101 - AZANIA SECONDARY SCHOOL

School Overall Grade Summary

	A	B+	B	C	D	E	F
M	554	531	423	470	459	511	424
TOT	554	531	423	470	459	511	424

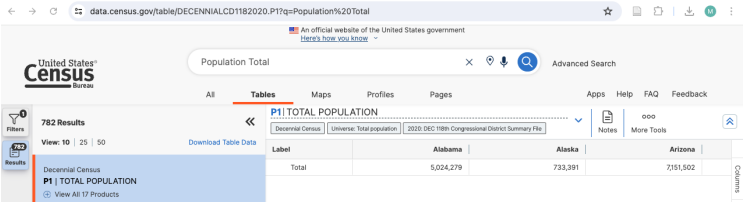
CNO	REPEATER	NAME OF CANDIDATE	SEX	CIVICS	HISTORY	GEOGRAPHY	B/KNOWL	E/D/KISLAMU	FINE ARTS	EDU	PHY	KISWAHILI	ENGLISH	FRENCH	PHYSICS	CHEMISTRY	BIOLOGY	INFO & COMP STUD	B/MATH	COMERCE	B/KEEPING	GPA	CLASS
0001		ABDALLAH KOMBO ABDALLAH	M	C	C	C			C			C	B+		E	D	C		E	E	D	2.3	CREDIT
0002		ABDALLAH MWALIMU MWINYIKONDO	M	C	C	E						D	B+		E	F	D	F	F	E	D	1.6	CREDIT
0003		ABDUL KARIM AZIZ	M	B	D	D						D	A		F	F	C		F	E	E	1.9	CREDIT

- This table is stable and looks like it could be manually copy and pasted
- This makes it a good candidate for scraping

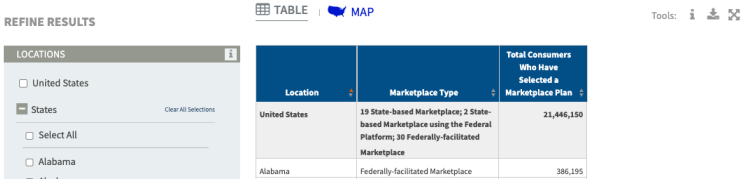
Counterexample: Data Viewers

- In contrast, would not recommend scraping dynamic tables/"data viewers"

Census "Explore Census Data" (link)



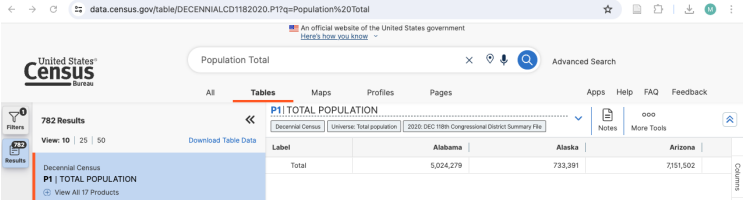
Kaiser Family Foundation "State Health Facts" (link)



Example: Data Viewers

- In contrast, would not recommend scraping dynamic tables/"data viewers"
- These are often rendered dynamically in JavaScript, rather than static HTML
- *(To scrape these more advanced websites, you could use the Selenium package, which allows you to control a web browser through Python)*

Census "Explore Census Data" (link)



Example: Scraping a Table

- Let's try to scrape the table on this webpage

maktaba.tetea.org/exam-resu

maktaba.tetea.org/exam-results/FTNA2015/S0101.htm

THE NATIONAL EXAMINATIONS COUNCIL OF TANZANIA
FORM TWO NATIONAL ASSESSMENT (FTNA) 2015 RESULTS
CENTRE: S0101 - AZANIA SECONDARY SCHOOL

School Overall Grade Summary

	A	B+	B	C	D	E	F
M	554	531	423	470	459	511	424
TOT	554	531	423	470	459	511	424

CNO	REPEATER	NAME OF CANDIDATE	SEX	CIVICS	HISTORY	GEOGRAPHY	B/KNOWL	E/D/KISLAMU	FINE ARTS	PHY EDU	KISWAHILI	ENGLISH	FRENCH	PHYSICS	CHEMISTRY	BIOLOGY	INFO & COMP STUD	B/MATH	COMMERCE	B/KEEPING	GPA	CLASS
0001		ABDALLAH KOMBO ABDALLAH	M	C	C	C			C		C	B+		E	D	C		E	E	D	2.3	CREDIT
0002		ABDALLAH MWALIMU MWINYIKONDO	M	C	C	E					D	B+		E	F	D	F	F	E	D	1.6	CREDIT
0003		ABDUL KARIM AZIZ	M	B	D	D					D	A		F	F	C		F	E	E	1.9	CREDIT

Example: Scraping a Table

Recall that the steps of building a webscraper are:

0. *Manual*: inspect website and HTML to see how the info we want to extract is structured
1. *Code*: download and save HTML code associated with a URL
2. *Code*: parse through HTML code to extract information based on what we learned in Step 0 + refine
3. *Code*: organize and clean extracted information outside of scraper (Pandas, string cleaning, etc)

Exercise

1. Go to the Tanzania National Examination outcomes (link)
2. Familiarize with the web page and table – **what is the unit of observation**?
3. From looking at the table, **name a detail** about the structure and layout of the table that you anticipate could make scraping difficult

Exercise

1. Go to the Tanzania National Examination outcomes (link)
2. Familiarize with the web page and table – **what is the unit of observation**?
3. From looking at the table, **name a detail** about the structure and layout of the table that you anticipate could make scraping difficult
4. Right click over one of the observations and "Inspect" it: **what tag(s) does the data appear to be stored in**?

Step 0. Inspect website and HTML

- Given the issue with the header, we're going to start with the first row with a name
- Search for the name in the first row: 'ABDALLAH'
- Appears to be in a nested tag: `<table>` → `<tbody>` → `<tr>` → `<td>`
- *Note: `<tbody>` is not in the actual HTML file, but is instead added by the browser you use to inspect*

```
▼<table>
  ▼<tbody>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr> == $0
    ▼<tr>
       <td align="center">0001</td>
       <td class="mark"> </td>
       <td>ABDALLAH KOMBO ABDALLAH</td>
       <td class="mark">M</td>
       <td class="mark">C</td>
```

Step 1. Download and save HTML

- Start with the usual: making a request to website and parsing the response with `BeautifulSoup`

```
url = 'https://maktaba.tetea.org/exam-results/FTNA2015/S0101.htm'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'lxml')
```

Step 2. Extract + refine

- Let's use `.find` to find the *first* instance of `table` tag
- Then apply `.find_all` to see what's inside

```
table = soup.find('table')
tr = table.find_all('tr') #tr is table row
tr[0:5]
```

```
[<tr><th class="head" colspan="25">THE NATIONAL EXAMINATIONS COUNCIL OF TA
 <tr><th class="head" colspan="25">FORM TWO NATIONAL ASSESSMENT (FTNA) 201
 <tr><th class="head" colspan="25"></th></tr>,
 <tr><th class="head" colspan="25">CENTRE: S0101 - AZANIA SECONDARY SCHOOL
 <tr><th colspan="25" style="border-left-color:#ADD8E6;border-right-color:
 <table class="top_sum">
 <tr style="font-weight:bold;"><td> </td><td>A</td><td>B+</td><td>B</td><t
```

Step 2. Extract + refine

- The output from looking for the `tr` tag seems to align with the HTML code

```
▼<table>
  ▼<tbody>
    ▼<tr>
       <th class="head" colspan="25">THE NATIONAL EXAMINATIONS COUNCIL OF TANZANIA</th>
     </tr>
    ▼<tr>
       <th class="head" colspan="25">FORM TWO NATIONAL ASSESSMENT (FTNA) 2015 RESULTS</th>
     </tr>
    ▼<tr> == $0
       <th class="head" colspan="25"></th>
     </tr>
```

- But we don't want to scrape the first lines or the header
- At this point, we should loop through the items in `tr`

Step 2. Extract + refine

- After a bit of searching through the `tr` object, we will find that the first row starts in index 24

```
tr[24]
```

```
<tr><td align="center">0001</td><td class="mark"> </td><td>ABDALLAH KOMBO A
```

- We're almost at the data!
- But need to drill down one more level to each cell, which are denoted by `td`

Step 2. Extract + refine

- Apply `.find_all()` again to look for `td`

```
td = tr[24].find_all('td')
td[0:3]
```

```
[<td align="center">0001</td>,
 <td class="mark"> </td>,
 <td>ABDALLAH KOMBO ABDALLAH</td>]
```

Step 2. Extract + refine

- Apply `.find_all()` again to look for `td`

```
td = tr[24].find_all('td')
td[0:3]
```

```
[<td align="center">0001</td>,
 <td class="mark"> </td>,
 <td>ABDALLAH KOMBO ABDALLAH</td>]
```

```
td[0].text
```

```
'0001'
```

```
td[2].text
```

```
'ABDALLAH KOMBO ABDALLAH'
```

Step 3. Organize and save extracted information

- To extract information, we need *two* nested for loops

```
data_rows = []

for row in table.find_all('tr')[24:]:                #1
    td_tags = row.find_all('td')                     #2
    data_rows.append([val.text for val in td_tags]) #3
```

Step 3. Organize and save extracted information

- To extract information, we need *two* nested for loops

```
data_rows = []

for row in table.find_all('tr')[24:]:                #1
    td_tags = row.find_all('td')                     #2
    data_rows.append([val.text for val in td_tags]) #3
```

1. Loops through all `tr` tags in `table` (after the 24th row)
2. Within each `tr` tag, finds all the `td` tags
3. Loops through `td_tags` and appends text within to `data_rows`

Step 3. Organize and save extracted information

- data_rows is a list, so we need to convert it into a dataframe

```
data_rows = pd.DataFrame(data_rows)
print(data_rows.head())
```

```
     0  1                            2  3  4  5   6  7  8  9   ...  13 14
0  0001          ABDALLAH KOMBO ABDALLAH  M  C  C   C        C  ...      E
1  0002     ABDALLAH MWALIMU MWINYIKONDO  M  C  C   E           ...      E
2  0003                 ABDUL KARIM AZIZ  M  B  D   D           ...      F
3  0004                ABDUL SEIF MBONDE  M  A  A   A           ...  B+  A
4  0005       ABDULATIFU HARUNI MAGIMILA  M  C  C  B+        C  ...      D

   16 17 18  19 20   21           22
0   C     E   E  D  2.3       CREDIT
1   D  F  F   E  D  1.6       CREDIT
2   C     F   E  E  1.9       CREDIT
3   A     A  B+  A  5.0  DISTINCTION
4  B+     C   D  C  2.1       CREDIT
```

Step 3. Organize and save extracted information

- We forgot the header!
- Can we scrape the header from the table?

THE NATIONAL EXAMINATIONS COUNCIL OF TANZANIA
FORM TWO NATIONAL ASSESSMENT (FTNA) 2015 RESULTS
CENTRE: S0101 - AZANIA SECONDARY SCHOOL

School Overall Grade Summary

	A	B+	B	C	D	E	F
M	554	531	423	470	459	511	424
TOT	554	531	423	470	459	511	424

CNO	REPEATER	NAME OF CANDIDATE	SEX	CIVICS	HISTORY	GEOGRAPHY	B/KNOWL	ED/KISLAMU	FINE ARTS	PHY EDU	KISWAHILI	ENGLISH	FRENCH	PHYSICS	CHEMISTRY	BIOLOGY	INFO & COMP STUD	B/MATH	COMMERCE	B/KEEPING	GPA	CLASS
0001		ABDALLAH KOMBO ABDALLAH	M	C	C	C			C		C	B+		E	D	C		E	E	D	2.3	CREDIT
0002		ABDALLAH MWALIMU MWINYIKONDO	M	C	C	E					D	B+		E	F	D	F	F	E	D	1.6	CREDIT

Step 3. Organize and save extracted information

- We forgot the header!
- Can we scrape the header from the table?

THE NATIONAL EXAMINATIONS COUNCIL OF TANZANIA
FORM TWO NATIONAL ASSESSMENT (FTNA) 2015 RESULTS
CENTRE: S0101 - AZANIA SECONDARY SCHOOL

School Overall Grade Summary

	A	B+	B	C	D	E	F
M	554	531	423	470	459	511	424
TOT	554	531	423	470	459	511	424

CNO	REPEATER	NAME OF CANDIDATE	SEX	CIVICS	HISTORY	GEOGRAPHY	B/KNOWL	ED/KISLAMU	FINE ARTS	PHY EDU	KISWAHILI	ENGLISH	FRENCH	PHYSICS	CHEMISTRY	BIOLOGY	INFO & COMP STUD	B/MATH	COMMERCE	B/KEEPING	GPA	CLASS
0001		ABDALLAH KOMBO ABDALLAH	M	C	C	C			C		C	B+		E	D	C		E	E	D	2.3	CREDIT
0002		ABDALLAH MWALIMU MWINYIKONDO	M	C	C	E					D	B+		E	F	D	F	F	E	D	1.6	CREDIT

- Given that the header is spread out across several rows and is sometimes vertical, scraping it would be a hassle

Fix Header

- Instead of scraping it, we'll just hard-code it: define the header manually

```
my_cols = ['CNO', 'Repeater', 'Name', 'Sex', 'Civics', 'History',
 ↪  'Geography', 'Knowledge', 'EDK', 'Fine_Arts', 'Phys_Ed',
 'Kiswahili', 'English', 'French', 'Physics', 'Chemistry', 'Biology',
  ↪  'Info_Computer', 'Math', 'Commerice', 'Keeping', 'GPA', 'Class']

data_rows.columns = my_cols
```

- There's a tradeoff with webscraping: if you are doing something *just once*, probably not worth it to do it via scraping

Step 3. Organize and save extracted information

```
print(data_rows.head())
```

```
    CNO Repeater                          Name Sex Civics History Geography
0  0001                ABDALLAH KOMBO ABDALLAH   M      C       C         C
1  0002           ABDALLAH MWALIMU MWINYIKONDO   M      C       C         B
2  0003                       ABDUL KARIM AZIZ   M      B       D         B
3  0004                      ABDUL SEIF MBONDE   M      A       A         A
4  0005             ABDULATIFU HARUNI MAGIMILA   M      C       C        B-

  Knowledge EDK Fine_Arts  ... French Physics Chemistry Biology Info_Compu
0                       C  ...              E         D       C
1                          ...              E         F       D
2                          ...              F         F       C
3                          ...     B+       A         A       A
4                       C  ...              D         B      B+

  Math Commerice Keeping  GPA        Class
```

Check Original Table Again

CNO	REPEATER	NAME OF CANDIDATE	SEX	CIVICS	HISTORY	GEOGRAPHY	B/KNOWL	E/D/KISLAMU	FINE ARTS	PHY EDU	KISWAHILI	ENGLISH	FRENCH	PHYSICS	CHEMISTRY	BIOLOGY	INFO & COMP STUD	B/MATH	COMMERCE	B/KEEPING	GPA	CLASS
0001		ABDALLAH KOMBO ABDALLAH	M	C	C	C			C		C	B+		E	D	C		E	E	D	2.3	CREDIT
0002		ABDALLAH MWALIMU MWINYIKONDO	M	C	C	E					D	B+		E	F	D	F	F	E	D	1.6	CREDIT
0003		ABDUL KARIM AZIZ	M	B	D	D					D	A		F	F	C		F	E	E	1.9	CREDIT
0004		ABDUL SEIF MBONDE	M	A	A	A					B+	A	B+	A	A	A		A	B+	A	5.0	DISTINCTION
0005		ABDULATIFU HARUNI MAGIMILA	M	C	C	B+			C		C	A		D	B	B+		C	D	C	3.1	MERIT
0006		ABDULAZIZ HEBERT NCHIRA	M	B	C	B		C			B	B+		D	E	B		D	D	E	2.9	MERIT
0007		ABDULLATIF KHAMIS SAID	M	B+	D	B					C	B+	E	E	D	C		F	E	D	2.4	CREDIT
0008		ABDULATIFU FADHIL TAMBA	M	B+	C	A		E			C	A		B+	B	B		D	B	B	3.9	DISTINCTION
0009		ABDULMUTWALIB HAMIS TEMBO	M	A	B+	F					B	A	E	B	C	A		B	B+	B+	4.3	DISTINCTION
0010		ABUBAKARI EBRAHIM ABDALLAH	M	D	E	D					E	A	F	E	E	C		F	E	E	1.3	PASS
0011		ABUBAKARI ISSA KADEGE	M	B	D	B					C	B+		D	B	B	E	D	D	B+	3.1	MERIT
0012	REPEATER	ABUU ALLY JUMA	M	Absent																		ABS

- Looking back at the original table, it looks like it should say `'Absent'` for each grade and `'ABS'` for GPA and class
- The rest of the data cleaning can be done by cleaning dataframes in `pandas`

Check Subset of Students Who were Absent

```
absent_subset = data_rows[data_rows['Repeater'] == "REPEATER"]
print(absent_subset.head())
```

```
      CNO  Repeater                     Name Sex  Civics History Geography
11   0012  REPEATER          ABUU ALLY JUMA   M  Absent     ABS      None
44   0045  REPEATER       BOHAZ GOD MANGULA   M       D       C         B
71   0072  REPEATER         ELTON A MARANDU   M  Absent     ABS      None
107  0108  REPEATER    HAFIDHI HEMED HOUMUD   M       D       C         E
178  0179  REPEATER  KHALIFA MASSOUD MRISHA   M       F       E         F

    Knowledge   EDK Fine_Arts  ... French Physics Chemistry Biology  \
11       None  None      None  ...   None    None      None    None
44                             ...              E         F       E
71       None  None      None  ...   None    None      None    None
107         F                  ...              E         F       F
178                            ...              E         F       F
```

Replace Grades of Absent Students

```
data_rows.loc[data_rows['Repeater'] == "REPEATER",
 ↪  data_rows.columns[4:-2]] = "Absent"

data_rows.loc[data_rows['Repeater'] == "REPEATER",
 ↪  data_rows.columns[-2:]] = "ABS"
```

Check Subset of Students Who were Absent Again

```
absent_subset = data_rows[data_rows['Repeater'] == "REPEATER"]
print(absent_subset.head())
```

```
      CNO  Repeater                     Name Sex  Civics History Geography
11   0012  REPEATER          ABUU ALLY JUMA   M  Absent  Absent    Absent
44   0045  REPEATER       BOHAZ GOD MANGULA   M  Absent  Absent    Absent
71   0072  REPEATER         ELTON A MARANDU   M  Absent  Absent    Absent
107  0108  REPEATER    HAFIDHI HEMED HOUMUD   M  Absent  Absent    Absent
178  0179  REPEATER  KHALIFA MASSOUD MRISHA   M  Absent  Absent    Absent

    Knowledge     EDK Fine_Arts  ...  French Physics Chemistry Biology  \
11     Absent  Absent    Absent  ...  Absent  Absent    Absent  Absent
44     Absent  Absent    Absent  ...  Absent  Absent    Absent  Absent
71     Absent  Absent    Absent  ...  Absent  Absent    Absent  Absent
107    Absent  Absent    Absent  ...  Absent  Absent    Absent  Absent
178    Absent  Absent    Absent  ...  Absent  Absent    Absent  Absent
```

Extracting Data from Tables: Summary

- BeautifulSoup can scrape HTML tables (but not dynamic JavaScript ones)
- HTML tables should be easy to spot via `table` tag
- Most of the work will be in
  - step 2: scraping what you want from the table
  - step 3: cleaning up the scrapers mistakes in Pandas
- Tradeoff with scraping: it is costly to code up a scraper, so if you're only doing something once, don't necessarily have to scrape it

Can I Scrape It?

Can I Scrape It?: Roadmap

- Discuss cases where websites block scraping + one solution
- How to determine if a site is scrapable ahead of time

Websites Can Block Scraping: Examples

- Many modern websites have built in mechanisms to prevent scraping

```
url = "https://www.amazon.com/"
response = requests.get(url)
soup = BeautifulSoup(response.content, 'lxml')
soup.text[0:100]
```

```
'\n\n\n\n\nSorry! Something went wrong!\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\
```

Websites Can Block Scraping: Examples

- Many modern websites have built in mechanisms to prevent scraping

```python
url = "https://www.amazon.com/"
response = requests.get(url)
soup = BeautifulSoup(response.content, 'lxml')
soup.text[0:100]
```

```
'\n\n\n\n\nSorry! Something went wrong!\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\
```

```python
url =
 ↪  "https://www.nytimes.com/2024/08/15/us/politics/us-to-announce-prices-
response = requests.get(url)
soup = BeautifulSoup(response.content, 'lxml')
soup.text[0:100]
```

```
'nytimes.comPlease enable JS and disable any ad blocker'
```

- In these cases, check if the data is accessible via an API

Websites Can Block Crawlers: Rate-Limiting

- Websites can also block *crawling* by 'rate-limiting': track how often a bot accesses the site and block it if it access it too often
- In this case, your request will return a `429 Too Many Requests` error
- Can avoid this by adding a delay with `time.sleep()` in between iterations

```
import time

for url in urls:
    response = requests.get(url)
    time.sleep(2)  # Add a 2-second delay
```

Can I Scrape It?

1. Check Terms of Service

User Agreement | LinkedIn

linkedin.com/legal/user-agreement#dos

LinkedIn

User Agreement | Privacy Policy | Professional Community Policies | Cookie Policy | Copyright Policy

You agree that you will *not*:

1. Create a false identity on LinkedIn, misrepresent your identity, create a Member profile for anyone other than yourself (a real person), or use or attempt to use another's account;

2. Develop, support or use software, devices, scripts, robots or any other means or processes (including crawlers, browser plugins and add-ons or any other technology) to scrape the Services or otherwise copy profiles and other data from the Services;

Can I Scrape It?

2. Check `robots.txt`

linkedin.com/robots.txt

linkedin.com/robots.txt

```
# Notice: The use of robots or other automated means to access LinkedIn without
# the express permission of LinkedIn is strictly prohibited.
# See https://www.linkedin.com/legal/user-agreement.
# LinkedIn may, in its discretion, permit certain automated access to certain LinkedIn pages,
# for the limited purpose of including content in approved publicly available search engines.
# If you would like to apply for permission to crawl LinkedIn, please email whitelist-crawl@linkedin.com.
# Any and all permitted crawling of LinkedIn is subject to LinkedIn's Crawling Terms and Conditions.
# See http://www.linkedin.com/legal/crawling-terms.

User-agent: LinkedInBot
Allow: /
```

Can I Scrape It?

2. Check `robots.txt`

en.wikipedia.org/robots.txt

en.wikipedia.org/robots.txt

```
#
# Friendly, low-speed bots are welcome viewing article pages, but not
# dynamically-generated pages please.
#
# Inktomi's "Slurp" can read a minimum delay between hits; if your
# bot supports such a thing using the 'Crawl-delay' or another
# instruction, please let us know.
#
# There is a special exception for API mobileview to allow dynamic
# mobile web & app views to load section content.
# These views aren't HTTP-cached but use parser cache aggressively
# and don't expose special: pages etc.
#
# Another exception is for REST API documentation, located at
# /api/rest_v1/?doc.
#
```

Can I Scrape It?: Summary

- Some sites block all scraping, others just rate-limit
- Check Terms of Service or `robots.txt` to see if a website allows scraping

Webscraping: Summary

- Webscraping downloads, parses, and extracts information from the HTML of webpage
- Most of the work is in familiarizing yourself with the unique structure of the website you want to scrape and cleaning up afterwards
- Webscraping tools can be adapted to 'crawl' the web by recursively extracting URLs