ECON446 :Applications of Cloud Computing and Big Data

Week 2, Lecture 2 | Web Crawling and RegEx

Sam Borghese

Wednesday, April 10th, 2024

1. Web crawling with Requests
2. User Agents and Security
3. Scraping tables from Websites
4. Regular Expressions

Web Crawling with Requests

What is Web Crawling?

AKA Spider, Crawler or SpiderBot

- A computer program that systematically moves through the urls on a given page and extracts data

- A computer program that systematically moves through the urls on a given page and extracts data

What is Web Crawling?

AKA Spider, Crawler or SpiderBot

- A computer program that systematically moves through the urls on a given page and extracts data

  - Can have a more generalized approach to data gathering than the specific
  - Not site-specific
  - Not need to have prior knowledge of a web page before crawling
  - Just has to be given a base URL

Web Crawling - Using Requests

Step 1 : Scrape all the links from the seed url

- Recall from last class that all links are contained in an anchor tag

```
▶ <a class="_yb_h5ha8 rapid-noclick-resp" href="https://finance.yahoo.com/" data-ylk="cpos:
2;slk:Finance;elm:navcat;sec:ybar;subsec:navrail;pkgt:mid;itc:0;" id="root_3" data-rapid_p=
"29" data-v9y="1">…</a>
```

- Locate all anchor tags with hrefs

```
soup.find_all('a', href=True)
```

Web Crawling - Using Requests

Step 1 : Scrape all the links from the seed url

Checks if there is text present
Helps filter out dead links

```
links_with_text = []
for a in soup.find_all('a', href=True):
    if a.text:
        links_with_text.append(a['href'])
```

```
links_with_text
```

```
5910&wpFormIdentifier=titleform',
 'https://www.wikidata.org/wiki/Special:EntityPag
 '/w/index.php?title=Special:DownloadAsPdf&page=A
how-download-screen',
 '/w/index.php?title=Anything&printable=yes',
 'https://es.wikipedia.org/wiki/Anything',
 'https://it.wikipedia.org/wiki/Anything',
 'https://nl.wikipedia.org/wiki/Anything',
 'https://pl.wikipedia.org/wiki/Anything',
```

<=>

```
links_with_text = [a['href']
                   for a in soup.find_all('a', href=True)
                   if a.text]
```

Web Crawling - Using Requests

Step 2: Filter links

- Need to decide which links you want to store and continue with

Relative Links

```
'/w/index.php?title=Anything&printable=yes',
```

Starts with a “/“

Absolute Links

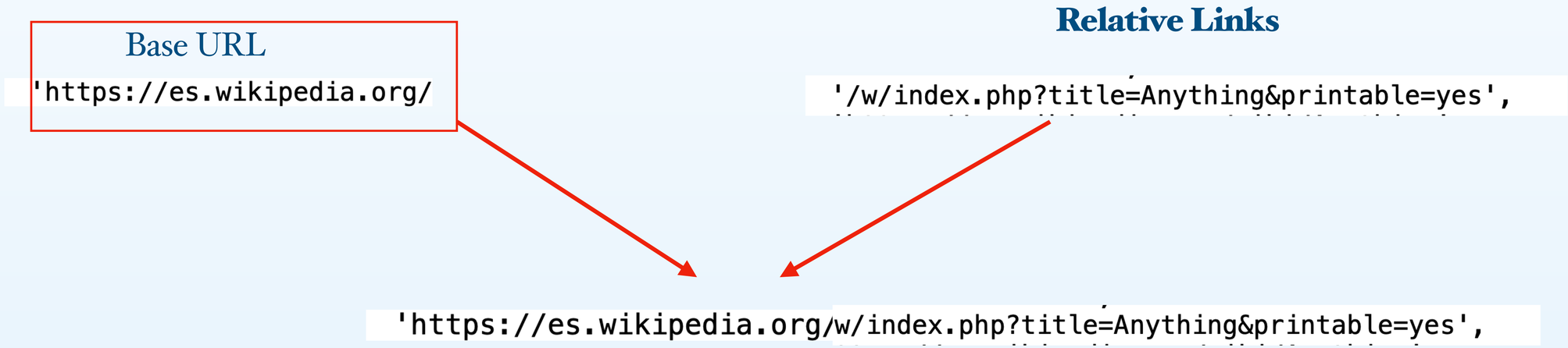
```
'https://es.wikipedia.org/wiki/Anything',
```

Starts with a “http“

Web Crawling - Using Requests

Step 2: Filter links

- Relative links need to be combined with the base url



Base urls : Side Note

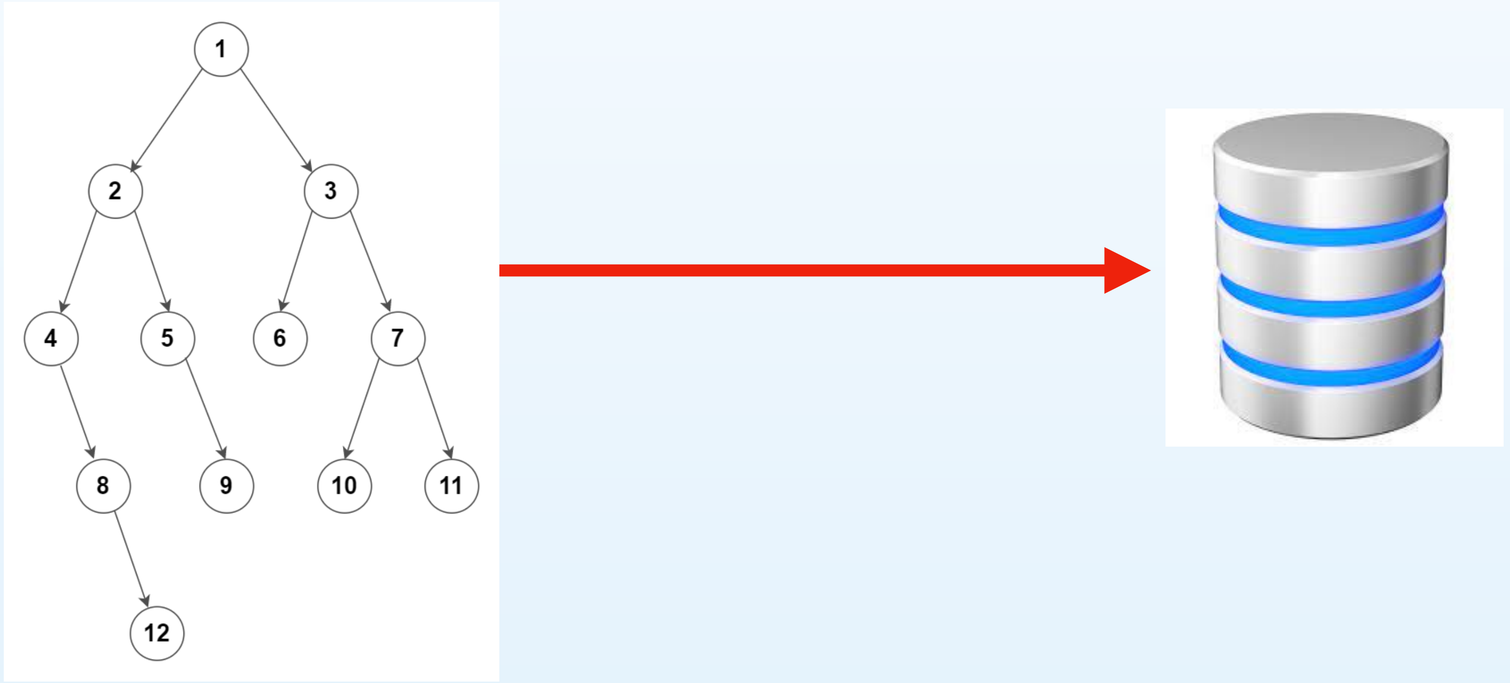
Relative links always attache to the base url

Base URL ends after the domain registrar (.com, .org, .gov)

Web Crawling - Using Requests

Step 3: Gather Data from All other pages

- Go deeper down the link tree depending on what you want to achieve from your spider



Web Crawling Example

Absolute Reference Crawling

Modify your code form last class to scrape apartment data from the first 10 pages of Trulia

Relative Reference Crawling

Scrape all Economic Excel data CSVs from the Dallas Fed

User Agents and Security

Legality of Web Scraping/Crawling

Very Legal

- NOT ILLEGAL
- Scrapers love it because it is cheap and powerful way to gather data

- People owning the websites frown upon it because they may take valuable data and cause your servers to be requested an excessive amount of times

- The only cases won against Web Scraping have been copyright lawsuits

Blocking Web Scraping

Certain Websites attempt to close access to Web Scraping

- When you go to your normal browser and submit a request for a page, your user-agent communicates with the server

- User-Agent is an identifier for the system querying a website

Blocking Web Scraping

Certain Websites attempt to close access to Web Scraping

```
{'User-Agent': 'python-requests
```

- When using python requests the User-Agent is python-requests

- Certain websites WILL NOT send user-agents like this their HTML code

Typically Error code 999 if blocking your request

Stopping Blocking Web Scraping

Can create your own User-Agent using a String in the request query

Method 1 : Wrote out Whole User Agent String

```
headers = {'User-Agent' :
           'Mozilla/5.0 (Windows NT 6.2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1464.0 Safari/537.36'}
```

Method 2 : Use User Agent Module

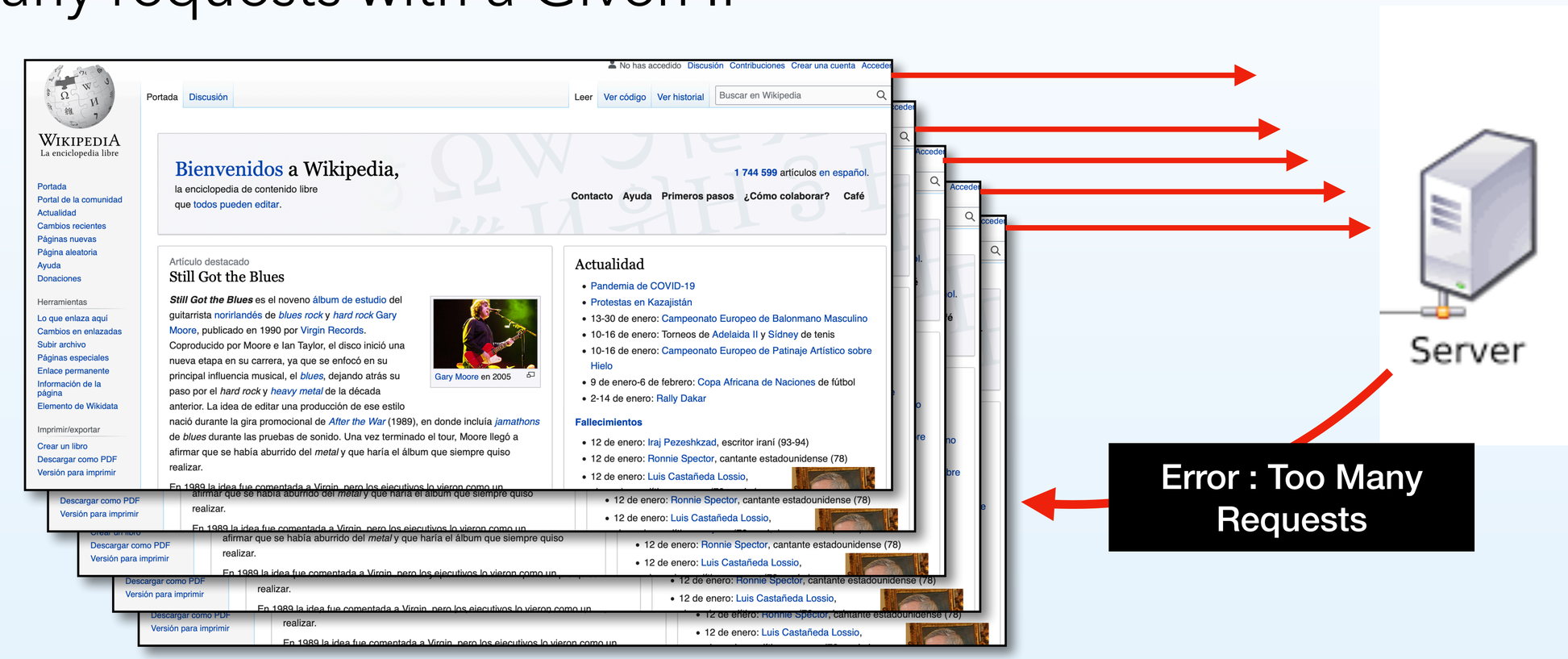
```
from fake_useragent import UserAgent
```

```
ua = UserAgent()
```

```
r = get(URL, headers={'User-Agent': ua.chrome})
```

Blocking WS because too many requests

Even when using a User Agent, Websites may block because of too many requests with a Given IP



Stopping Blocking WS

Use Multiple IP addresses

- Get a list of User-Agents online and loop through them in a list

1. **Windows 10/ Edge browser:** Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.135 Safari/537.36 Edge/12.246
2. **Windows 7/ Chrome browser:** Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.111 Safari/537.36
3. **Mac OS X10/Safari browser:** Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/601.3.9 (KHTML, like Gecko) Version/9.0.2 Safari/601.3.9
4. **Linux PC/Firefox browser:** Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:15.0) Gecko/20100101 Firefox/15.0.1
5. **Chrome OS/Chrome browser:** Mozilla/5.0 (X11; CrOS x86_64 8172.45.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.64 Safari/537.36

```
user_agent_list = ["Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebK
,"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like (
,"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/601.3.9
,"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:15.0) Gecko/20100101 Firef
,"Mozilla/5.0 (X11; CrOS x86_64 8172.45.0) AppleWebKit/537.36 (KHTML,
```

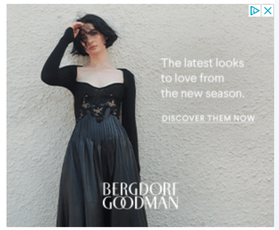
```
headers = {"User-Agent" : user_agent_list[i]}
```

Scraping Tables from Websites

Scraping Tables

Utilizes Two Packages

Date(s)	Sport	Event	Location
Jan 13–31	Handball	World Championships (men)	Egypt
Jan 29-31	Extreme Sports	Winter X Games 25	Aspen, Colorado, USA
Feb 4-11	Football (Soccer)	2020 FIFA Club World Cup	Doha, Qatar
~~Feb 6–12~~ (postponed until 2022)	Multi-sports	Special Olympics World Winter Games	~~Jämtland County, Sweden~~
Feb 6 - Mar 20	Rugby	Six Nations	UK, Ireland, France & Italy
Feb 7	Gridiron/Football	Super Bowl	Raymond James Stadium, Tampa, Florida
Feb 8-21	Tennis	Australia Open	Melbourne, Australia
Feb 8-21	Skiing	World Alpine Ski Championships	Cortina d'Ampezzo, Italy
Feb 23-Mar 7	Skiing (Nordic)	Nordic World Ski Championships	Oberstdorf, Germany



Events Calendar Menu

Upcoming Events

2022, 2023, 2024, 2025, 2026

Specific Calendars

- Football (soccer)
- American Football
- Rugby league
- Rugby union

Date(s)	Sport	Event
Jan 13–31	Handball	World Championships (men)
Jan 29-31	Extreme Sports	Winter X Games 25
Feb 4-11	Football (Soccer)	2020 FIFA Club World Cup
ed until 2022)	Multi-sports	Special Olympics World Winter Games
eb 6 - Mar 20	Rugby	Six Nations
...	...	...
Nov 23-29	Table Tennis	World Championship

Scraping Tables

Works for Majority of Tables

```
import requests
import pandas as pd
url = "https://www.topendsports.com/events/calendar-2021.htm"
html = requests.get(url).content
df_list = pd.read_html(html)
df = df_list[-1]
```

If there are multiple tables all you need to do is select the one you want from the list

Example

Example 1 :

Scrape the table of all publicly traded company info from NYSE

Example 2 :

Get the table of all game dates from TopEndSports

Regular Expressions

What are Regular Expressions?

Teaching Python to Read

```
import re
```

- “String Searching Algorithm
- Sequence of characters defining a pattern of search
- “Regex”

Regular Expressions

Search for a Fixed String

Defining Search Pattern
Searches for any portion of the string that has an “a”

```
pattern = re.compile(r'a')
```

Sample string

```
text = "I want to learn Regular Expressions"
```

Regular Expressions

Search For a Fixed String

```
pattern = re.compile(r'a')
```

```
matches = pattern.finditer(text)
```

- .finditer() loops through each character of a string and tests if the pattern begins

NO NO NO Yes

```
text = "I want to learn Regular Expressions"
```

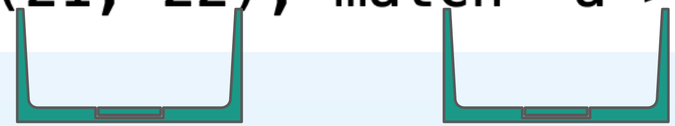
Regular Expressions

Search for a Fixed String

```
text = "I want to learn Regular Expressions"
```

```
[match for match in matches]
```

```
[<re.Match object; span=(3, 4), match='a'>,
 <re.Match object; span=(12, 13), match='a'>,
 <re.Match object; span=(21, 22), match='a'>]
```



What place in the string
That the match occurs

What the
match was

Regular Expressions

Search for a Fixed String

```
text = "I want to learn Regular Expressions"
```

```
pattern = re.compile(r'ar')
```

```
matches = pattern.finditer(text)
```

```
[match for match in matches]
```

```
[<re.Match object; span=(12, 14), match='ar'>,
 <re.Match object; span=(21, 23), match='ar'>]
```

Regular Expressions

Character Sets

- Character Sets are contained in Brackets

```
r'[ar]'
```

- Ex. Either an “a” OR “r” appears in place of the character set

```
[match for match in matches]
```

```
[<re.Match object; span=(3, 4), match='a'>,
 <re.Match object; span=(12, 13), match='a'>,
 <re.Match object; span=(13, 14), match='r'>,
 <re.Match object; span=(21, 22), match='a'>,
 <re.Match object; span=(22, 23), match='r'>,
 <re.Match object; span=(27, 28), match='r'>]
```

Regular Expressions

Character Sets

```
r'[a-z]'
```

Any Lowercase letter

```
r'[A-Z]'
```

Any Uppercase letter

```
r'[0-9]'
```

Any Number

```
r'[.]'
```

Anything

```
r'a[.]'
```

- “a” followed by any symbol
- Ex.

“ad”
“a3”
“a:”

Regular Expressions

Character Sets Example

Checks if there is any Upper or Lower case letter

```
r'[A-Za-z]'
```

Checks for an upper case letter followed by a lower case letter

```
r'[A-Z][a-z]'
```

```
[<re.Match object; span=(16, 18), match='Re'>,
 <re.Match object; span=(24, 26), match='Ex'>]
```

Regular Expressions

Meta Characters

- These are used to specify specific characteristics in RegEx

```
( ) [ ] \ | { } . ^ $ * + ?
```

Regular Expressions

If you want to find a meta character

- If a Question Mark is in the Text

```
r'\?'
```

- If a period is present

```
r'\.'
```

Regular Expressions

"+" Recursively tests a pattern until it fails

```
r'[a-z]+'
```

```
[<re.Match object; span=(2, 6), match='want'>,
 <re.Match object; span=(7, 9), match='to'>,
 <re.Match object; span=(10, 15), match='learn'>,
 <re.Match object; span=(17, 23), match='egular'>,
 <re.Match object; span=(25, 35), match='xpressions'>]
```

```
r'[A-Z]+ [0-9]+'
```

```
[match for match in matches]
```

```
[<re.Match object; span=(36, 44), match='HELP 911'>]
```

Regular Expressions

Short Hand Expressions

```
r'[\d]' : Digits
r'[\w]' : Alpha numerics
r'[\D]' : Not Digits
r'[\W]' : Symbols
r'[|]'  : Or command
r'[A?]' : A could be there or not be there
r'[A*]' : A Could be there multiple times or not be
r'[A+]' : A could be there multiple times or just 1
```

```
re.compile(r'[A-Z]+ [\d]+')
```

Regular Expressions

Use Regular Expressions to Identify Classes referenced in the Text

```
text = """
446A is my favorite class except for
443A which is my real favorite class
441B is fun too
"""
```

```
pattern = re.compile(r'\d\d\d[A-Z]')
pattern = re.compile(r'\d+[A-Z]')
matches = pattern.finditer(text)
[match for match in matches]
```

```
[<re.Match object; span=(1, 5), match='446A'>,
 <re.Match object; span=(38, 42), match='443A'>,
 <re.Match object; span=(75, 79), match='441B'>]
```

Regular Expressions

Need to think through many possibilities

```
text = """
446A is my favorite class except for
443A which is my real favorite class
441B is fun too
I got 100Percent in all
"""
```

```
pattern = re.compile(r'\d\d\d[A-Z]')
pattern = re.compile(r'\d+[A-Z]')
matches = pattern.finditer(text)
[match for match in matches]
```

```
<re.Match object; span=(99, 103), match='100P'>]
```

Regular Expressions

Qualifiers allow for a specific string type to occur a specific number of times

Have exactly three digits appear

```
r'\d{3}[A-Z]'
```

Have two to five digits appear inclusive

```
r'\d{2,5}[A-Z]'
```

Regular Expressions

Using Spaces to locate words

```
text = """
446A is my favorite class except for
443A which is my real favorite class
441B is fun too
I got 100Percent in all
"""
```

```
pattern = re.compile(r' \d{2,5}[A-Z] ')
matches = pattern.finditer(text)
[match for match in matches]
```

```
[]
```

Why does it return the empty set?

Regular Expressions

In HTML and RegEx a newline is defined as \n

```
r'[ |\n]\d{2,5}[A-Z][ |\n]'
```

A space or line break. 3 numbers. A letter. A space or line break.

Read RegEx Documentation for more : https://docs.python.org/3/library/re.html