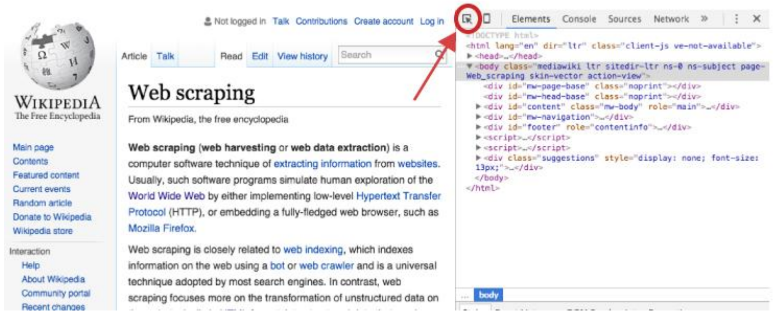
Applied Web Scraping II

THE KNOWLEDGE HOUSE

Celebrating 10 Years of Diversifying Tech

Agenda - Schedule

1. Warm Up
2. Web Scraping Review
3. List Methods
4. Regular Expressions
5. Break
6. Web Scraping Lab II



Web scraping

From Wikipedia, the free encyclopedia

Web scraping (**web harvesting** or **web data extraction**) is a computer software technique of extracting information from websites. Usually, such software programs simulate human exploration of the World Wide Web by either implementing low-level Hypertext Transfer Protocol (HTTP), or embedding a fully-fledged web browser, such as Mozilla Firefox.

Web scraping is closely related to web indexing, which indexes information on the web using a bot or web crawler and is a universal technique adopted by most search engines. In contrast, web scraping focuses more on the transformation of unstructured data on

```
<!DOCTYPE html>
<html lang="en" dir="ltr" class="client-js ve-not-available">
▶<head>…</head>
▼<body class="mediawiki ltr sitedir-ltr ns-0 ns-subject page-
Web_scraping skin-vector action-view">
    <div id="mw-page-base" class="noprint"></div>
    <div id="mw-head-base" class="noprint"></div>
  ▶<div id="content" class="mw-body" role="main">…</div>
  ▶<div id="mw-navigation">…</div>
  ▶<div id="footer" role="contentinfo">…</div>
  ▶<script>…</script>
  ▶<script>…</script>
  ▶<div class="suggestions" style="display: none; font-size:
  13px;">…</div>
  </body>
</html>
```

Web scraping software may directly access the World Wide Web using the Hypertext Transfer Protocol or a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a bot or web crawler.
https://en.wikipedia.org/wiki/Web_scraping

Agenda - Announcements

- Week 5 Pre-Class Quiz due 4/8
- TLAB #2 due 4/21
- Add music to your respective Cohort Link!
    - We will use this for the music recommendation algorithm in Phase 2!

Agenda - Goals

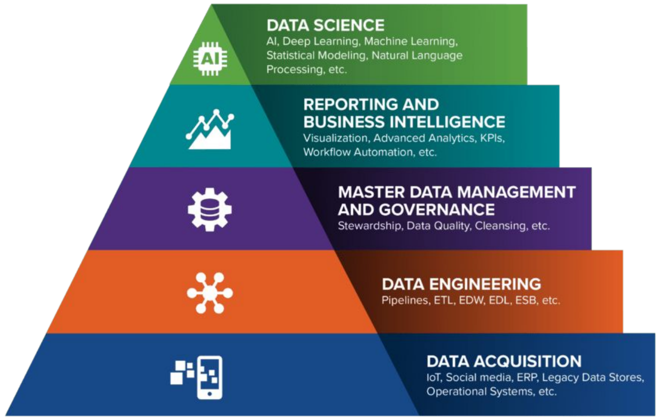
- Review web-scraping methods
- Learn about “advanced” web-scraping techniques
- Review string methods for TLAB #2 success
- Learn about the concept of regular expressions

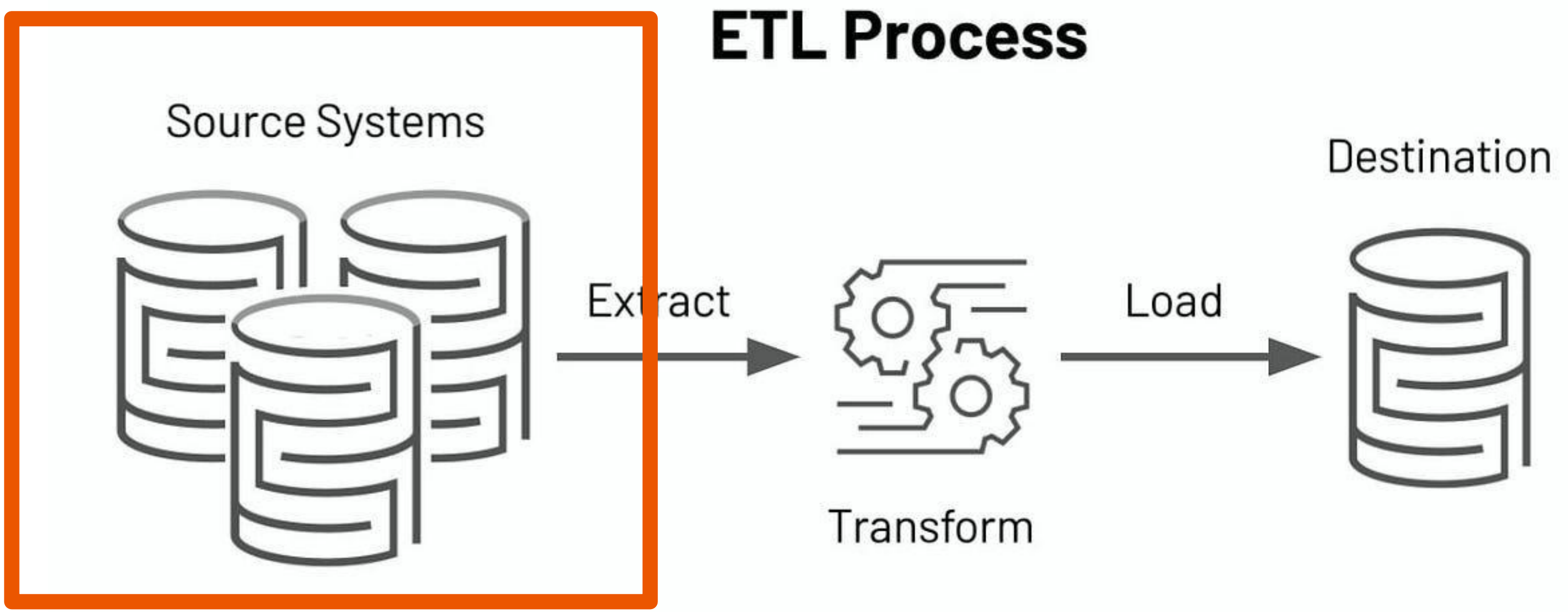
Web Scraping Review

Web Scraping Review

Let’s take a second to review where web-scraping falls on the data science hierarchy of needs.

Like we established, we must first capture data before we can apply any sort of **analysis** or **machine learning**.


DATA SCIENCE
AI, Deep Learning, Machine Learning, Statistical Modeling, Natural Language Processing, etc.
REPORTING AND BUSINESS INTELLIGENCE
Visualization, Advanced Analytics, KPIs, Workflow Automation, etc.
MASTER DATA MANAGEMENT AND GOVERNANCE
Stewardship, Data Quality, Cleansing, etc.
DATA ENGINEERING
Pipelines, ETL, EDW, EDL, ESB, etc.
DATA ACQUISITION
IoT, Social media, ERP, Legacy Data Stores, Operational Systems, etc.



Just like we learned about with WebAPI requests, we can **integrate** web-scraping **into our ETL pipeline** to eventually load data into a database!

WebScraping Review

To recap web-scraping, we first begin parsing our HTML by creating a **BeautifulSoup** object. From there we can access our data hierarchically through 2 simple methods:

- **find**(*name, attributes*) : find one element
- **find_all**(*name, attributes*) : find all elements (loaded into the list)

Remember, you must start web scraping by **looking** at your HTML! There's no way to tell which data you need **unless you see it**.

```
URL = "https://realpython.github.io/fake-jobs/"

page = requests.get(URL )

soup = BeautifulSoup(page.content, "html.parser")

results = soup.find(...., class_="...")
```

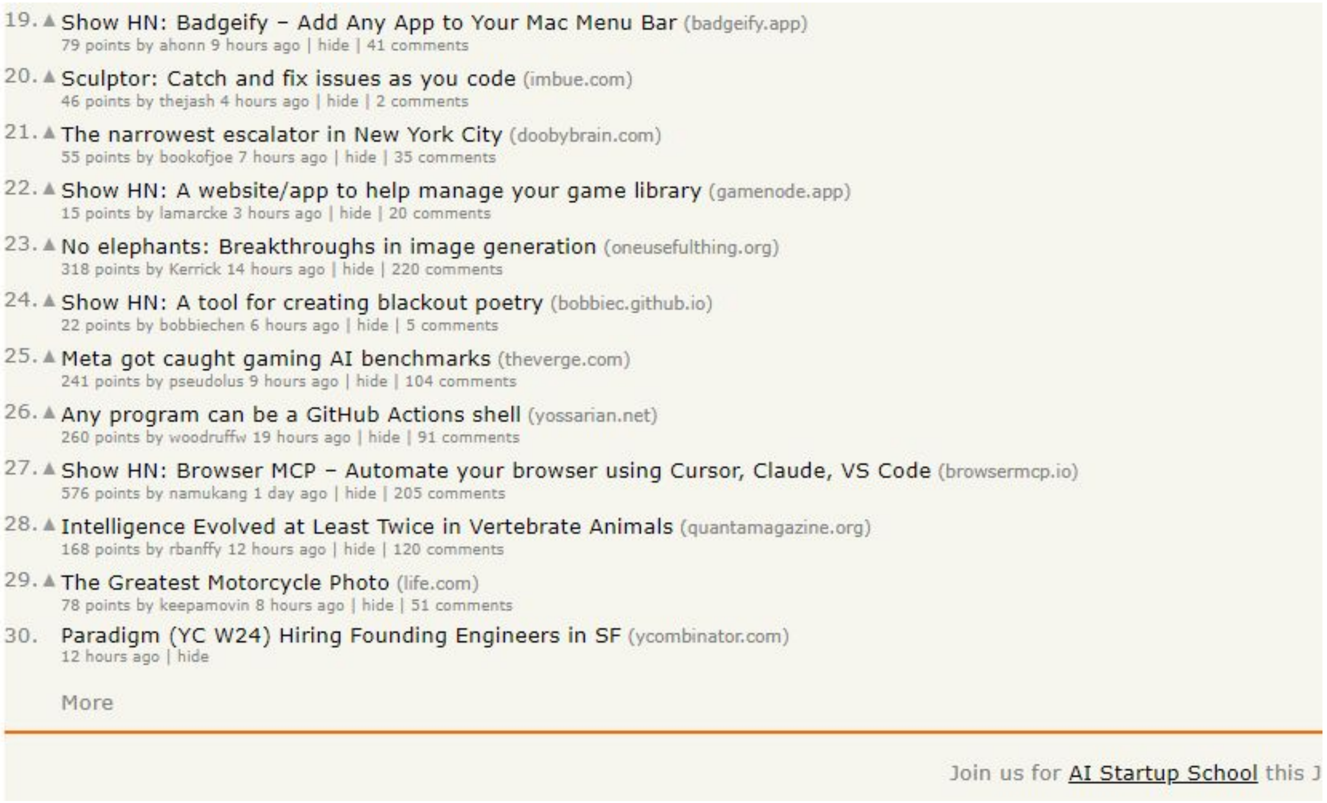
Handling Pagination

One last concept we want to introduce you to is pagination.

Like we established, we won’t always have the **luxury** of infinite scroll.

Instead, a website will be broken down into multiple pages.

In order to implement this functionality, we can make use the `requests` package and simply keep requesting data until we have no more pages!


19. ▲ Show HN: Badgeify – Add Any App to Your Mac Menu Bar (badgeify.app)
79 points by ahonn 9 hours ago | hide | 41 comments

20. ▲ Sculptor: Catch and fix issues as you code (imbue.com)
46 points by thejash 4 hours ago | hide | 2 comments

21. ▲ The narrowest escalator in New York City (doobybrain.com)
55 points by bookofjoe 7 hours ago | hide | 35 comments

22. ▲ Show HN: A website/app to help manage your game library (gamenode.app)
15 points by lamarcke 3 hours ago | hide | 20 comments

23. ▲ No elephants: Breakthroughs in image generation (oneusefulthing.org)
318 points by Kerrick 14 hours ago | hide | 220 comments

24. ▲ Show HN: A tool for creating blackout poetry (bobbiec.github.io)
22 points by bobbiechen 6 hours ago | hide | 5 comments

25. ▲ Meta got caught gaming AI benchmarks (theverge.com)
241 points by pseudolus 9 hours ago | hide | 104 comments

26. ▲ Any program can be a GitHub Actions shell (yossarian.net)
260 points by woodruffw 19 hours ago | hide | 91 comments

27. ▲ Show HN: Browser MCP – Automate your browser using Cursor, Claude, VS Code (browsermcp.io)
576 points by namukang 1 day ago | hide | 205 comments

28. ▲ Intelligence Evolved at Least Twice in Vertebrate Animals (quantamagazine.org)
168 points by rbanffy 12 hours ago | hide | 120 comments

29. ▲ The Greatest Motorcycle Photo (life.com)
78 points by keepamovin 8 hours ago | hide | 51 comments

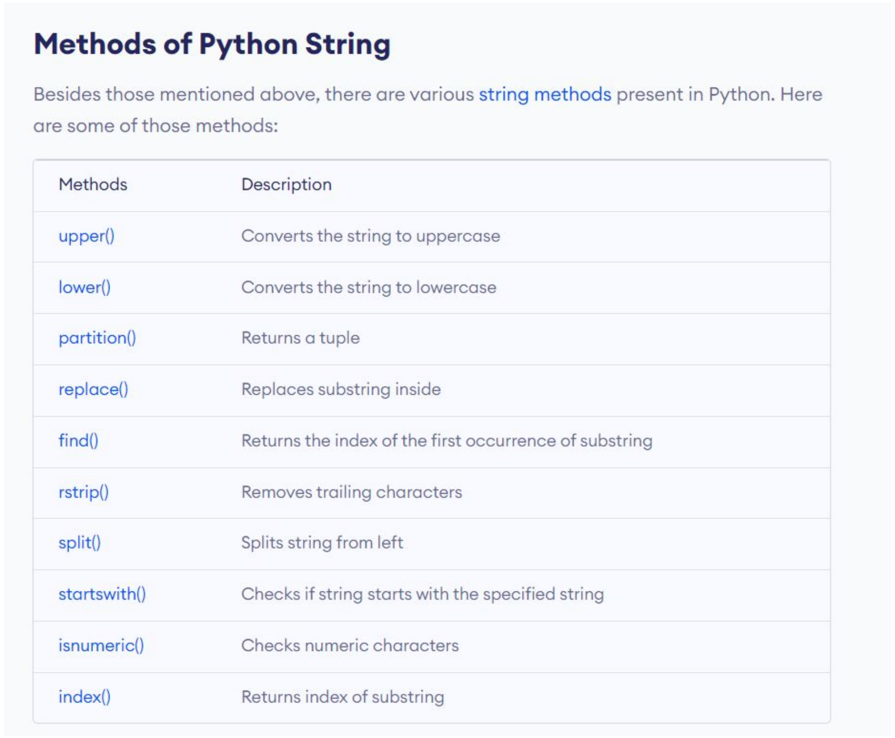
30. Paradigm (YC W24) Hiring Founding Engineers in SF (ycombinator.com)
12 hours ago | hide

More

Join us for AI Startup School this J

For this example, we will work through the “hacker news” website in order to pull all articles from this site.

String Methods



Methods of Python String

Besides those mentioned above, there are various string methods present in Python. Here are some of those methods:

Methods	Description
upper()	Converts the string to uppercase
lower()	Converts the string to lowercase
partition()	Returns a tuple
replace()	Replaces substring inside
find()	Returns the index of the first occurrence of substring
rstrip()	Removes trailing characters
split()	Splits string from left
startswith()	Checks if string starts with the specified string
isnumeric()	Checks numeric characters
index()	Returns index of substring

For your TLAB #2 success, we will go over a few string methods that are imperative for your success.

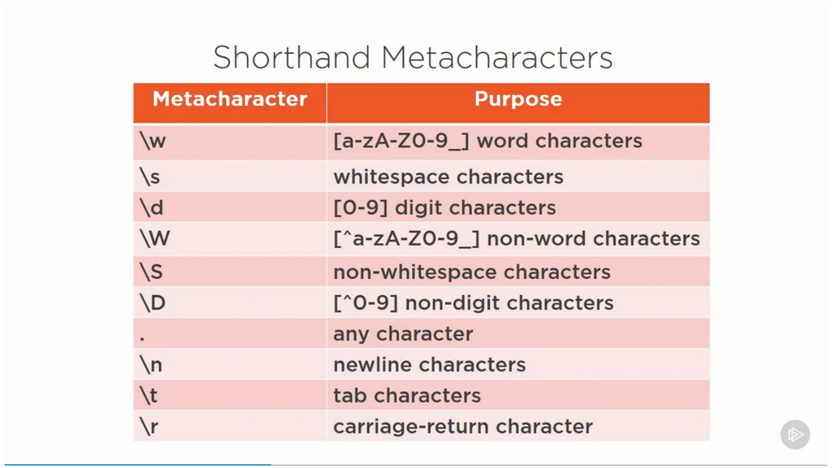
Regex

Regular Expressions (Regex)

One tool you will find yourself using for text-parsing is regular expressions (**regex**)

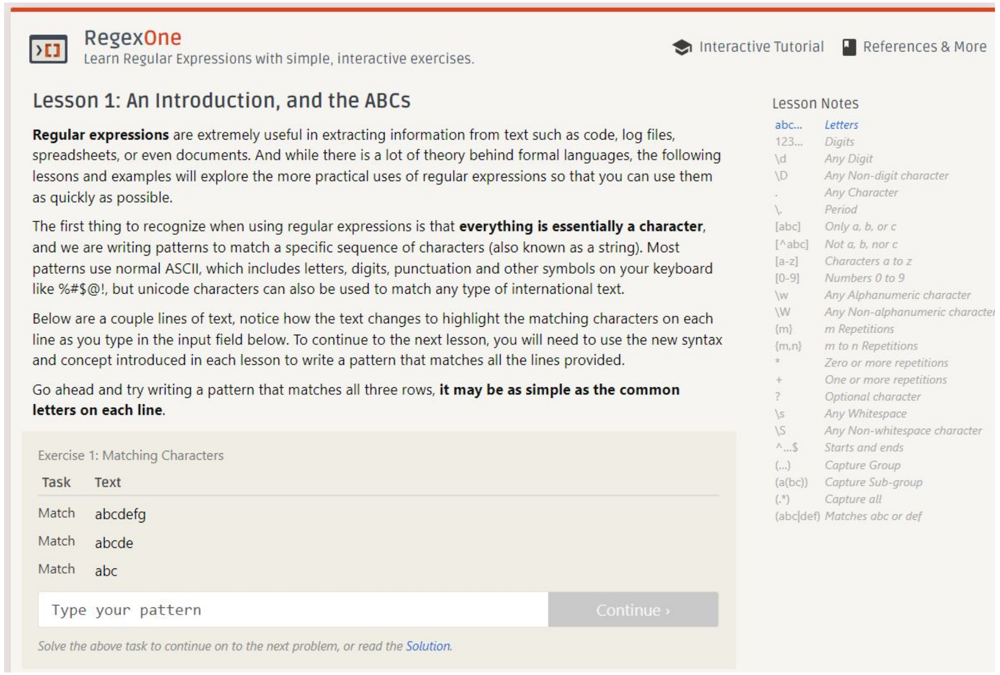
Regex is a powerful concept taken from **linguistics** that allows us to quickly search for text in a **text corpus**.

text corpus: collection of words



Shorthand Metacharacters

Metacharacter	Purpose
\w	[a-zA-Z0-9_] word characters
\s	whitespace characters
\d	[0-9] digit characters
\W	[^a-zA-Z0-9_] non-word characters
\S	non-whitespace characters
\D	[^0-9] non-digit characters
.	any character
\n	newline characters
\t	tab characters
\r	carriage-return character



RegexOne
Learn Regular Expressions with simple, interactive exercises.

Interactive Tutorial | References & More

Lesson 1: An Introduction, and the ABCs

Regular expressions are extremely useful in extracting information from text such as code, log files, spreadsheets, or even documents. And while there is a lot of theory behind formal languages, the following lessons and examples will explore the more practical uses of regular expressions so that you can use them as quickly as possible.

The first thing to recognize when using regular expressions is that **everything is essentially a character**, and we are writing patterns to match a specific sequence of characters (also known as a string). Most patterns use normal ASCII, which includes letters, digits, punctuation and other symbols on your keyboard like %#$@!, but unicode characters can also be used to match any type of international text.

Below are a couple lines of text, notice how the text changes to highlight the matching characters on each line as you type in the input field below. To continue to the next lesson, you will need to use the new syntax and concept introduced in each lesson to write a pattern that matches all the lines provided.

Go ahead and try writing a pattern that matches all three rows, **it may be as simple as the common letters on each line**.

Exercise 1: Matching Characters

Task	Text
Match	abcdefg
Match	abcde
Match	abc

Type your pattern | Continue ›

Solve the above task to continue on to the next problem, or read the Solution.

Lesson Notes

abc…	*Letters*
123…	*Digits*
\d	*Any Digit*
\D	*Any Non-digit character*
.	*Any Character*
\.	*Period*
[abc]	*Only a, b, or c*
[^abc]	*Not a, b, nor c*
[a-z]	*Characters a to z*
[0-9]	*Numbers 0 to 9*
\w	*Any Alphanumeric character*
\W	*Any Non-alphanumeric character*
{m}	*m Repetitions*
{m,n}	*m to n Repetitions*
*	*Zero or more repetitions*
+	*One or more repetitions*
?	*Optional character*
\s	*Any Whitespace*
\S	*Any Non-whitespace character*
^…$	*Starts and ends*
(…)	*Capture Group*
(a(bc))	*Capture Sub-group*
(.*)	*Capture all*
(abc|def)	*Matches abc or def*

We'll go over a *few* regex patterns. However this will not be an exhaustive lesson, the best resource to learn regex is arguably: https://regexone.com/lesson/introduction_abcs

Regex Pattern

review
Farukh is great
Well, Farukh is ok
Farrrrukh is terrible
I like Python27
@@@19586

To search for text using regex, you simply describe a **pattern of text to search for**. Regex then searches your selected string to check if it satisfy’s this **pattern**. Let’s assume we’re iterating through a list of strings.

Regex Pattern

Farukh

review
Farukh is great
Well, Farukh is ok
Farrrrukh is terrible
I like Python27
@@@19586

For example, by just using the regex string “Farukh”, we will look for all rows that contain the string “Farukh.” Which rows will be matched?

Regex Pattern

Farukh

review
Farukh is great
Well, Farukh is ok
Farrrrukh is terrible
I like Python27
@@@19586

Elements 0 & 1 get matched. Why doesn’t row 2 get matched?

Regex Pattern

Farukh

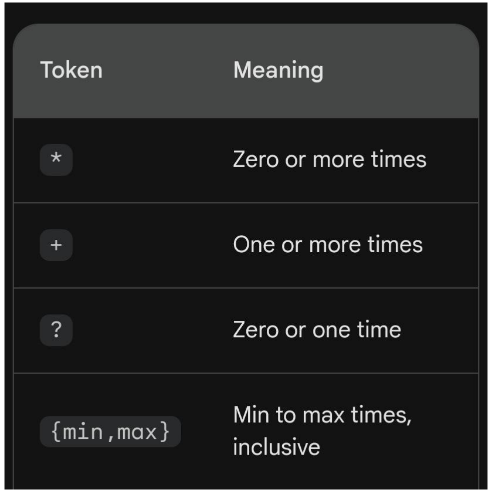
review
Farukh is great
Well, Farukh is ok
Farrrrukh is terrible
I like Python27
@@@19586

Remember, a computer does not understand intent. It will only do exactly what you want it to do

Farrrrukh is not the same as *Farukh*

Regex Pattern

Farukh



Token	Meaning
`*`	Zero or more times
`+`	One or more times
`?`	Zero or one time
`{min,max}`	Min to max times, inclusive

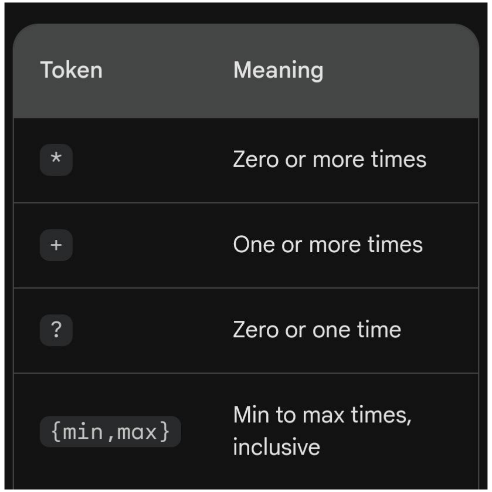
review
Farukh is great
Well, Farukh is ok
Farrrrukh is terrible
I like Python27
@@@19586

However, we can use special regex characters to indicate when we want to match words that contain **repeating characters**.

We place the * **after the letter we want to match multiple times.** To match the 2nd row (along with the 0th and 1st), where should we place our asterisk?

Regex Pattern

*Far*ukh*



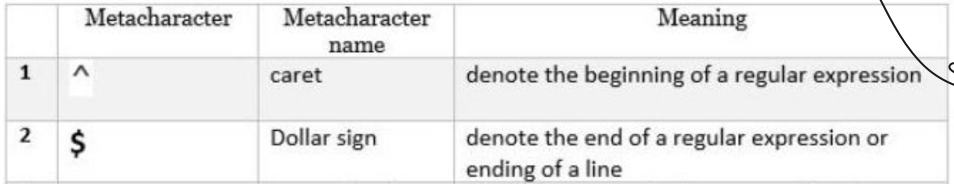
Token	Meaning
*	Zero or more times
+	One or more times
?	Zero or one time
{min,max}	Min to max times, inclusive

review
Farukh is great
Well, Farukh is ok
Farrrrukh is terrible
I like Python27
@@@19586

We place it after the "r." Now we will match all misspellings of "Farukh" that contain duplicate r's

Regex Pattern

*^Far*ukh*

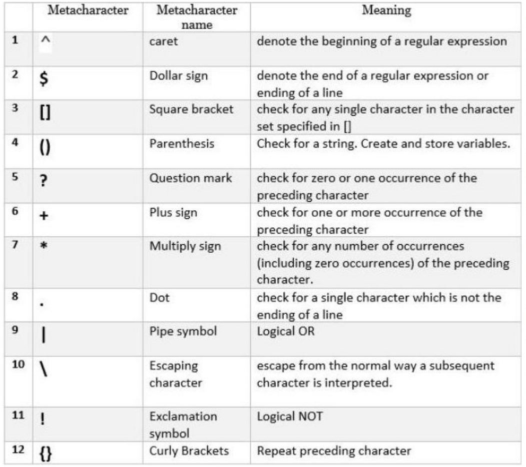


	Metacharacter	Metacharacter name	Meaning
1	^	caret	denote the beginning of a regular expression
2	$	Dollar sign	denote the end of a regular expression or ending of a line

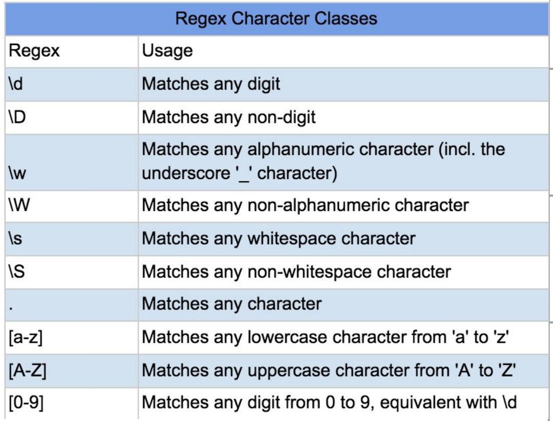
review
Farukh is great
Well, Farukh is ok
Farrrrukh is terrible
I like Python27
@@@19586

By placing a caret at the front, we only find reviews that that begin with the word "Farukh" with an arbitrary number of r's/

Regex Pattern



	Metacharacter	Metacharacter name	Meaning
1	^	caret	denote the beginning of a regular expression
2	$	Dollar sign	denote the end of a regular expression or ending of a line
3	[]	Square bracket	check for any single character in the character set specified in []
4	()	Parenthesis	Check for a string. Create and store variables.
5	?	Question mark	check for zero or one occurrence of the preceding character
6	+	Plus sign	check for one or more occurrence of the preceding character
7	*	Multiply sign	check for any number of occurrences (including zero occurrences) of the preceding character.
8	.	Dot	check for a single character which is not the ending of a line
9	|	Pipe symbol	Logical OR
10	\	Escaping character	escape from the normal way a subsequent character is interpreted.
11	!	Exclamation symbol	Logical NOT
12	{}	Curly Brackets	Repeat preceding character



Regex Character Classes

Regex	Usage
\d	Matches any digit
\D	Matches any non-digit
\w	Matches any alphanumeric character (incl. the underscore '_' character)
\W	Matches any non-alphanumeric character
\s	Matches any whitespace character
\S	Matches any non-whitespace character
.	Matches any character
[a-z]	Matches any lowercase character from 'a' to 'z'
[A-Z]	Matches any uppercase character from 'A' to 'Z'
[0-9]	Matches any digit from 0 to 9, equivalent with \d

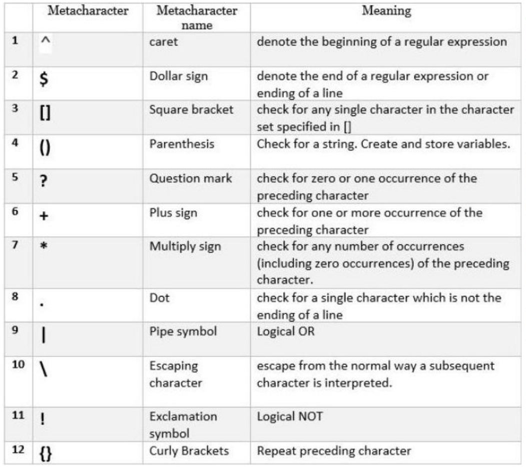
review
Farukh is great
Well, Farukh is ok
Farrrrukh is terrible
I like Python27
@@@19586

There are many more regex patterns, and the only way to figure out which one to use is via practice. So as long as you understand the **general idea of regex**, you should be able to make your own pattern.

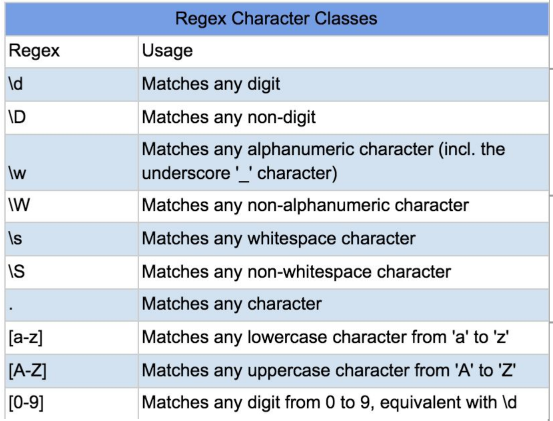
Using these tables, what do we write to match rows that **end with a number**?

Regex Pattern

\d$



	Metacharacter	Metacharacter name	Meaning
1	^	caret	denote the beginning of a regular expression
2	$	Dollar sign	denote the end of a regular expression or ending of a line
3	[]	Square bracket	check for any single character in the character set specified in []
4	()	Parenthesis	Check for a string. Create and store variables.
5	?	Question mark	check for zero or one occurrence of the preceding character
6	+	Plus sign	check for one or more occurrence of the preceding character
7	*	Multiply sign	check for any number of occurrences (including zero occurrences) of the preceding character.
8	.	Dot	check for a single character which is not the ending of a line
9	|	Pipe symbol	Logical OR
10	\	Escaping character	escape from the normal way a subsequent character is interpreted.
11	!	Exclamation symbol	Logical NOT
12	{}	Curly Brackets	Repeat preceding character

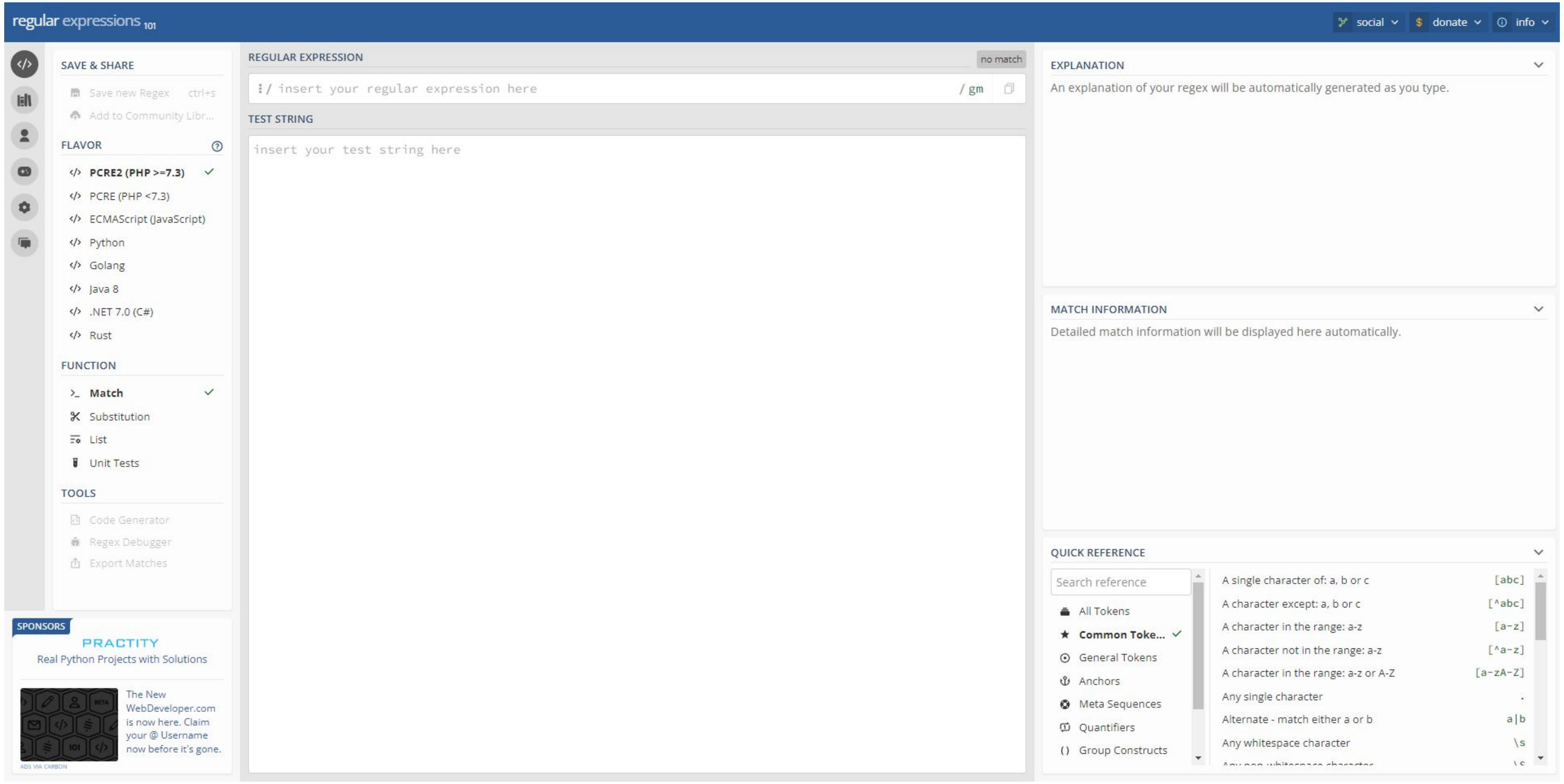


Regex Character Classes

Regex	Usage
\d	Matches any digit
\D	Matches any non-digit
\w	Matches any alphanumeric character (incl. the underscore '_' character)
\W	Matches any non-alphanumeric character
\s	Matches any whitespace character
\S	Matches any non-whitespace character
.	Matches any character
[a-z]	Matches any lowercase character from 'a' to 'z'
[A-Z]	Matches any uppercase character from 'A' to 'Z'
[0-9]	Matches any digit from 0 to 9, equivalent with \d

review
Farukh is great
Well, Farukh is ok
Farrrrukh is terrible
I like Python27
@@@19586

Knowing regex will save you **hours of work.**



FLAVOR

- PCRE2 (PHP >=7.3) ✓
- PCRE (PHP <7.3)
- ECMAScript (JavaScript)
- Python
- Golang
- Java 8
- .NET 7.0 (C#)
- Rust

FUNCTION

- Match ✓
- Substitution
- List
- Unit Tests

REGULAR EXPRESSION — no match

```
/ insert your regular expression here / gm
```

TEST STRING

```
insert your test string here
```

EXPLANATION

An explanation of your regex will be automatically generated as you type.

MATCH INFORMATION

Detailed match information will be displayed here automatically.

QUICK REFERENCE

A single character of: a, b or c	`[abc]`
A character except: a, b or c	`[^abc]`
A character in the range: a-z	`[a-z]`
A character not in the range: a-z	`[^a-z]`
A character in the range: a-z or A-Z	`[a-zA-Z]`
Any single character	`.`
Alternate - match either a or b	`a|b`
Any whitespace character	`\s`

To practice more regex, check out regex101

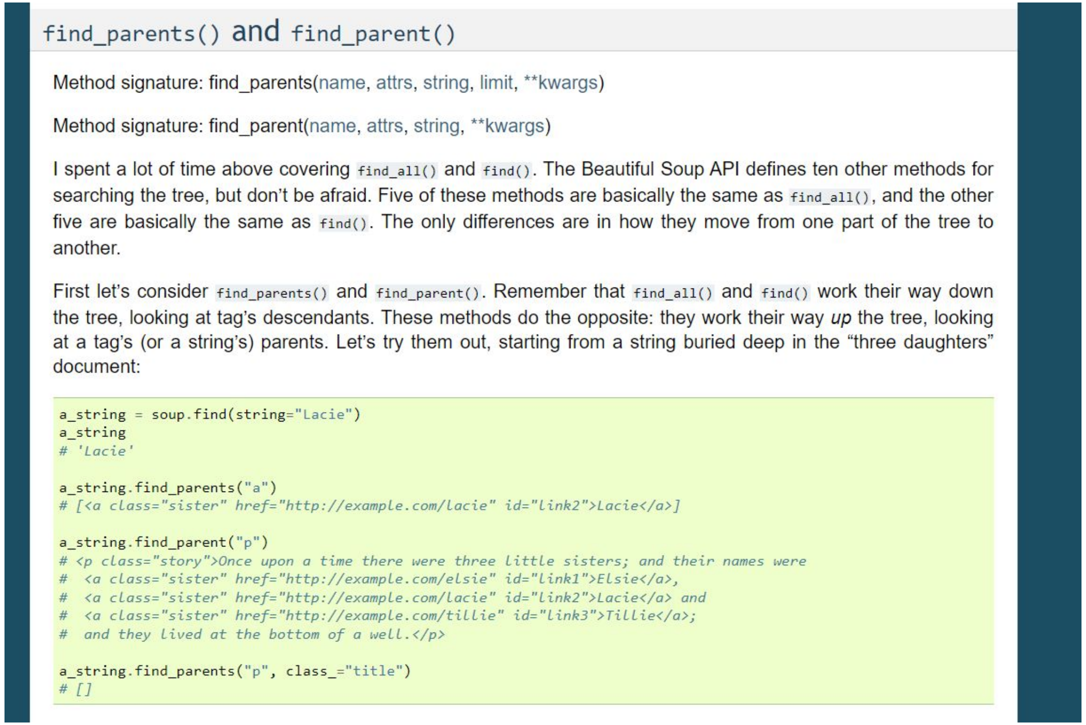
Parsing HTML Using BeautifulSoup4

With this new tool in our belt, we can effectively search for strings that contain the “Python” keyword.

This, however, is the most simple example of a regex string that you will be constructing.

Just as we’ve seen in the previous slides, web-data (especially when it’s posted on a **social network**) is rarely this clean.

```
import re

job_elements = results.find_all(“div”, class_=”card”)

for job in job_elements:
        title = job.find(“h2”, class_=”title”)

        match = re.search(r"Python", title .text)
        if match:
                print(title.text.strip())
```



find_parents() and find_parent()

Method signature: find_parents(name, attrs, string, limit, **kwargs)

Method signature: find_parent(name, attrs, string, **kwargs)

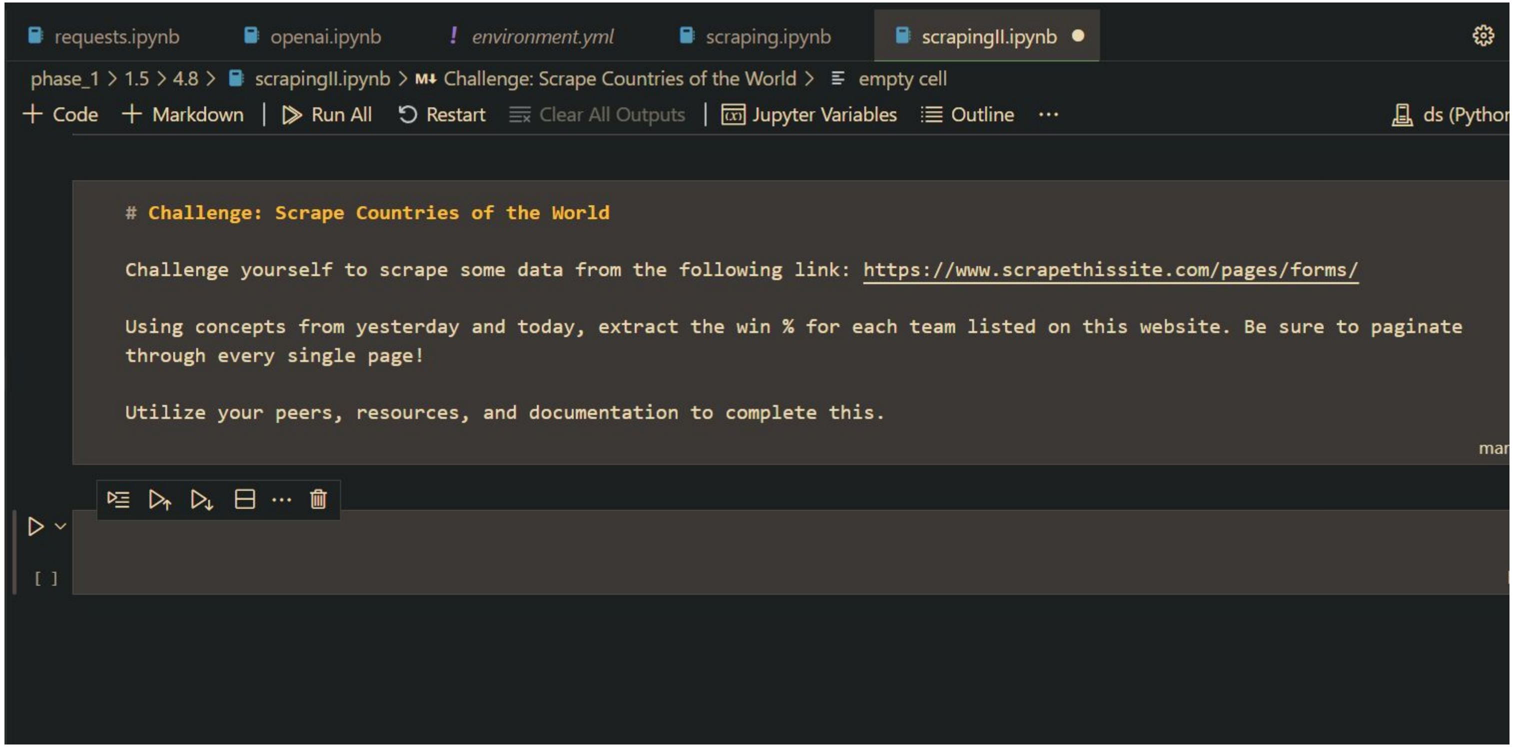
I spent a lot of time above covering `find_all()` and `find()`. The Beautiful Soup API defines ten other methods for searching the tree, but don't be afraid. Five of these methods are basically the same as `find_all()`, and the other five are basically the same as `find()`. The only differences are in how they move from one part of the tree to another.

First let's consider `find_parents()` and `find_parent()`. Remember that `find_all()` and `find()` work their way down the tree, looking at tag's descendants. These methods do the opposite: they work their way *up* the tree, looking at a tag's (or a string's) parents. Let's try them out, starting from a string buried deep in the "three daughters" document:

```
a_string = soup.find(string="Lacie")
a_string
# 'Lacie'

a_string.find_parents("a")
# [<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>]

a_string.find_parent("p")
# <p class="story">Once upon a time there were three little sisters; and their names were
#  <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>;
#  and they lived at the bottom of a well.</p>

a_string.find_parents("p", class_="title")
# []
```

The fact is, we will not be going over the entire API for any package. And truthfully this isn't knowledge that you necessarily should be carrying around anyways. **Always use the documentation.**

Web Scraping Lab II



```
# Challenge: Scrape Countries of the World

Challenge yourself to scrape some data from the following link: https://www.scrapethissite.com/pages/forms/

Using concepts from yesterday and today, extract the win % for each team listed on this website. Be sure to paginate
through every single page!

Utilize your peers, resources, and documentation to complete this.
```

Using this syntax, get started with your web-scraping lab!

Wrap-Up

Lab (Due 04/21)

Vancouver, Canada

You are a growth analyst at a Vancouver-based consulting firm called Monica Group. Your manager is spearheading the completion of a a new analytical tool which will automatically label if a review is positive, neutral, negative, or irrelevant.

You will be kicking off completion of this milestone by independently implementing a minimal-viable-product. **This will be a Python pipeline that ingests a text-file of review data and interfaces with the Open AI API in order to automatically label each review.**

We will release API keys on 4/1

Wednesday

Wednesday will entail:

- Hypothesis testing
- Applied hypothesis testing in data analysis

Jupyter: scratchpad of the data scientist

If you understand what you're doing, you're not learning anything. - Anonymous