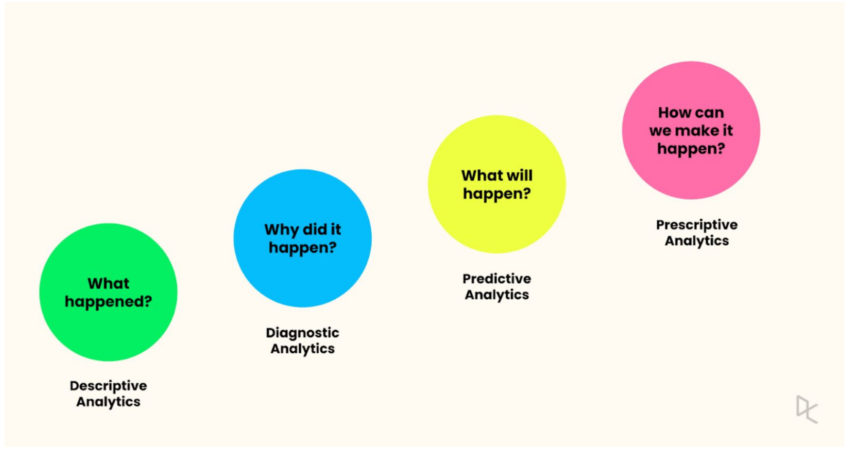
Twitter Dataset Case Study

THE KNOWLEDGE HOUSE

Agenda - Schedule

1. Case Study Introduction
2. Time Series Data
3. Regex
4. Break (30 Mins)
5. Continue Case Study



Agenda - Goals

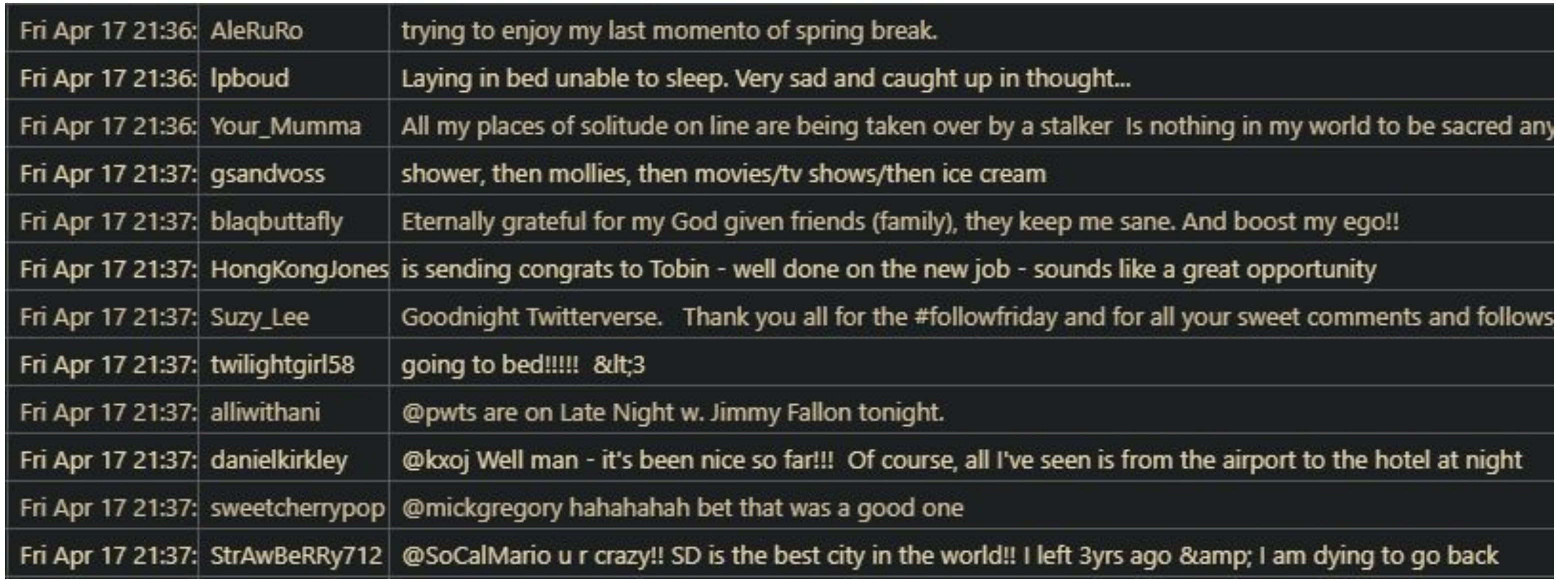
- Perform time series exploration using pandas
- Extract key date-time features like hour and weekday
- Analyze and visualize patterns in tweet volume and sentiment
- Use string matching techniques to filter tweets by content
- Focus on developing insightful conclusions, not just writing code

Announcements

- Week 8 Pre-Class Quiz due 4/29 (2 *attempts*)
- Review Session on 5/1
- TLAB #3 due 5/14

“*be-leaf in yourself*!”

Twitter Dataset Case Study



Fri Apr 17 21:36:	AleRuRo	trying to enjoy my last momento of spring break.
Fri Apr 17 21:36:	lpboud	Laying in bed unable to sleep. Very sad and caught up in thought...
Fri Apr 17 21:36:	Your_Mumma	All my places of solitude on line are being taken over by a stalker Is nothing in my world to be sacred any
Fri Apr 17 21:37:	gsandvoss	shower, then mollies, then movies/tv shows/then ice cream
Fri Apr 17 21:37:	blaqbuttafly	Eternally grateful for my God given friends (family), they keep me sane. And boost my ego!!
Fri Apr 17 21:37:	HongKongJones	is sending congrats to Tobin - well done on the new job - sounds like a great opportunity
Fri Apr 17 21:37:	Suzy_Lee	Goodnight Twitterverse. Thank you all for the #followfriday and for all your sweet comments and follows
Fri Apr 17 21:37:	twilightgirl58	going to bed!!!!! <3
Fri Apr 17 21:37:	alliwithani	@pwts are on Late Night w. Jimmy Fallon tonight.
Fri Apr 17 21:37:	danielkirkley	@kxoj Well man - it's been nice so far!!! Of course, all I've seen is from the airport to the hotel at night
Fri Apr 17 21:37:	sweetcherrypop	@mickgregory hahahahah bet that was a good one
Fri Apr 17 21:37:	StrAwBeRRy712	@SoCalMario u r crazy!! SD is the best city in the world!! I left 3yrs ago &amp; I am dying to go back

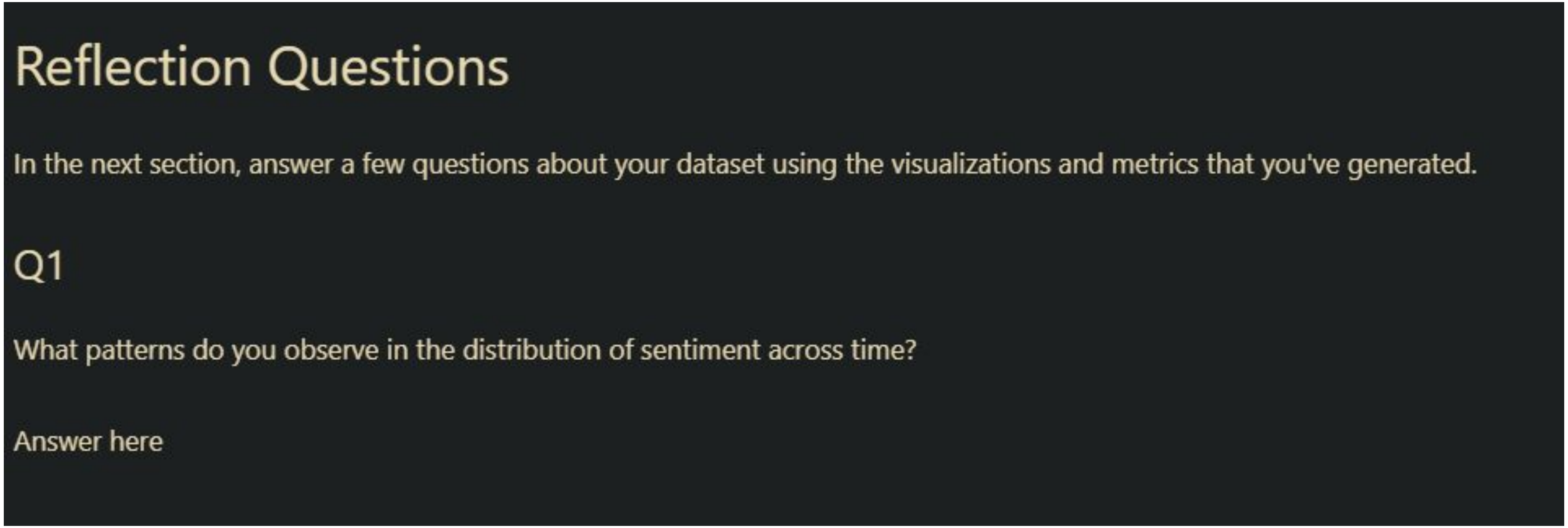
You are a Data Scientist for *SuperEgo* an NYC-based research institute looking to create a language model that closely emulates a twitter-user.

Twitter Dataset Case Study

Like yesterday, we will use the first-half of class to work on this case study together.

After break, we will ask you to complete this case study in your groups.

We will congregate back at 9:20 to discuss results (with the wheels help).



Reflection Questions

In the next section, answer a few questions about your dataset using the visualizations and metrics that you've generated.

Q1

What patterns do you observe in the distribution of sentiment across time?

Answer here

Just like yesterday, we will prepare a report by answering a set of reflection questions.

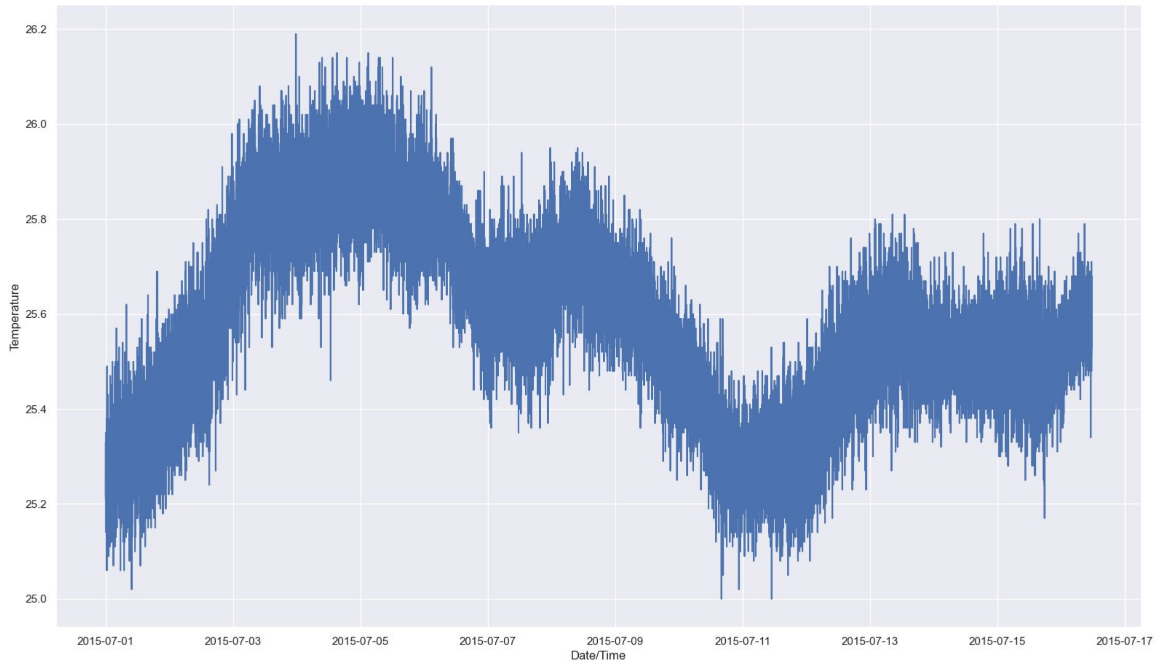
Pandas Time Series Data

Pandas Review - Time Bound Data

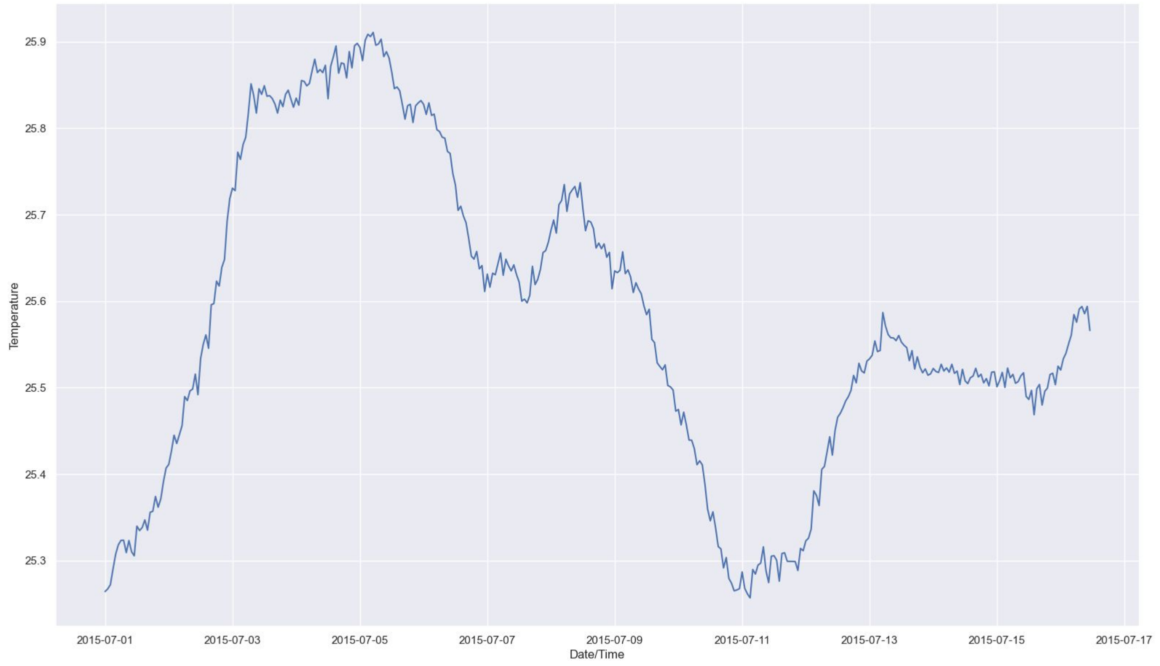
Often times when we work with a dataset that contains some **time-bound component**, we would like to **combine samples** according to days, months, or even years.

This is called **resampling.**

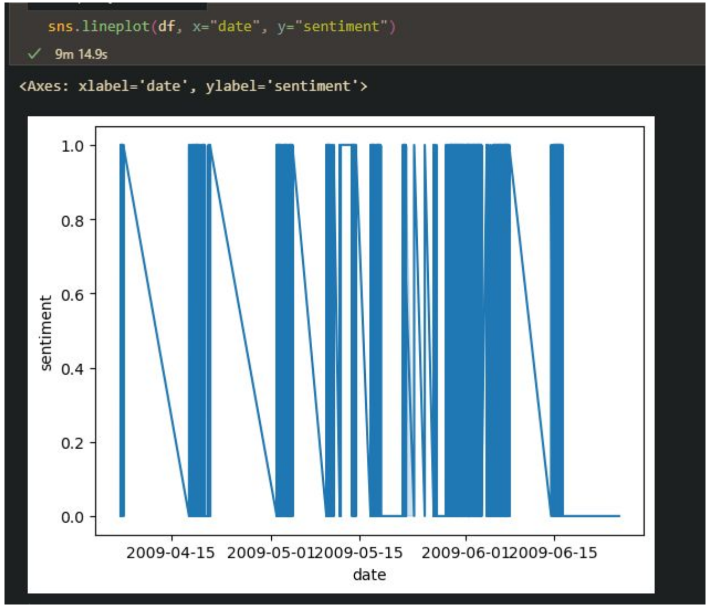
This entails **downsampling** (aggregating) and **upsampling** (filling in null values) our dataset so that we can view our data in different time regularities and view overall trends.



For example, let’s say we have this super fuzzy line-plot which expresses data **recorded every second**. While we can definitely notice the “trend” of this plot, but there is a lot of “noise” expressed as the variance around this trend.



By resampling this dataset by calculating the **average value across 24-hours** of data, we can clean up this visualization. Note that while this does "remove" data (downsample), it also allows **greater clarity of trend.**



```
sns.lineplot(df, x="date", y="sentiment")
✓ 9m 14.9s
<Axes: xlabel='date', ylabel='sentiment'>
```

NOTE: If it takes more than 2 minutes to make your plot (and it comes out looking like nonsense), you've made the **wrong plot.**

Don't make plots just to make plots.

sns.lineplot(df, x="date", y="sentiment")

This is currently what our sentiment line plot looks when observing our raw data. Let's find out how we can **resample** this dataframe so that we could better observe the trend.

ID	Date	Amount
1	Mon Apr 17 20:30 2023	5.24
2	Mon Apr 17 20:31 2023	10.98
3	Mon Apr 17 20:35 2023	1.58
4	Mon Apr 17 23:35 2023	50.60
5	Tues Apr 18 04:20 2023	34.01
6	Tues Apr 18 07:50 2023	25.05
7	Thurs Apr 20 12:50 2023	5.63

```
df[“Date”] = pd.to_datetime(df[“Date”])
```

Before we do ANYTHING! We must ensure that our date column is being treated as a “datetime” column.

We can do this by calling the “**to_datetime**()” method on our date column, and reassigning it back into our original column. Remember, if we don’t save our changes, we lose them!

For demonstration purposes, let’s look at a dataset of **transactions** in an online shopping platform.

This dataset is not in our case study, but it will help us understand the resample method.

ID	Date	Amount
1	Mon Apr 17 20:30 2023	5.24
2	Mon Apr 17 20:31 2023	10.98
3	Mon Apr 17 20:35 2023	1.58
4	Mon Apr 17 23:35 2023	50.60
5	Tues Apr 18 04:20 2023	34.01
6	Tues Apr 18 07:50 2023	25.05
7	Thurs Apr 20 12:50 2023	5.63

df.resample(“1D”, on=”Date”)

Group by “one single day.”

You can specify any list of rules!

“4W” → 4 Weeks
“2M” → 2 Months
“1Y” → 1 Year

Now that we’ve converted our “Date” column to the date-time datatype, we can use this column in our **resample() method** to combine all of our values in a row according to some **window of time.** In this example, we use 1-day resampling windows.

ID	Date	Amount
1	Mon Apr 17 20:30 2023	5.24
2	Mon Apr 17 20:31 2023	10.98
3	Mon Apr 17 20:35 2023	1.58
4	Mon Apr 17 23:35 2023	50.60
5	Tues Apr 18 04:20 2023	34.01
6	Tues Apr 18 07:50 2023	25.05
7	Thurs Apr 20 12:50 2023	5.63

df.resample("1D", on="Date")

$\downarrow$

```
<pandas.core.resample.DatetimeIndexResampler object at 0x000001F6DF15AA90>
```

But keep note that is a "**groupby**"-like operation, so only calling resample gives us a memory address. What should we do to **calculate summary stats on this?**

ID	Date	Amount
1	Mon Apr 17 20:30 2023	5.24
2	Mon Apr 17 20:31 2023	10.98
3	Mon Apr 17 20:35 2023	1.58
4	Mon Apr 17 23:35 2023	50.60
5	Tues Apr 18 04:20 2023	34.01
6	Tues Apr 18 07:50 2023	25.05
7	Thurs Apr 20 12:50 2023	5.63

```
df.resample(“1D”, on=”Date”)[“Amount”].mean()
```

Note that we also get NaN values. These are days that **don’t have data**

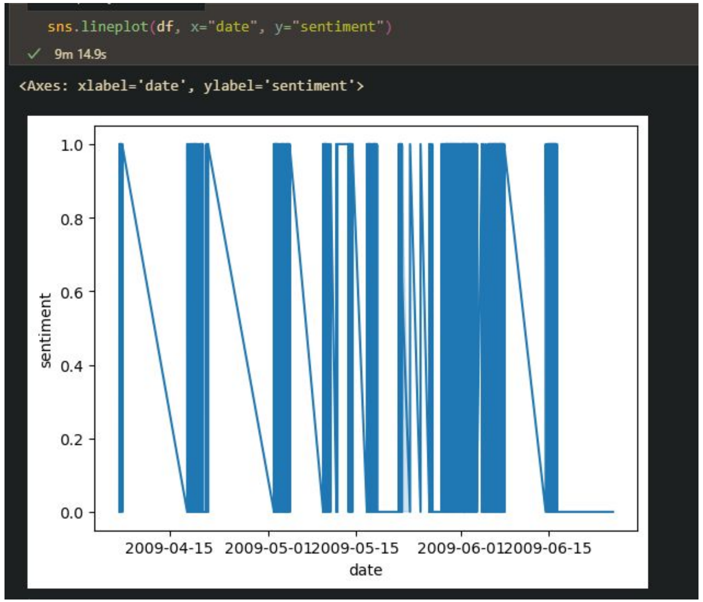
Just like with **groupby** , we can add a method to calculate some **summary statistic**. This will select all rows that fall on **the same day…**

Date	Amount
Mon Apr 17 2023	17.1
Tues Apr 18 2023	29.53
Wednesday Apr 19 2023	NaN
Thurs Apr 20 2023	5.63

```
df.resample(“1D”, on=”Date”)[“Amount”].mean()
```

Note that we also get NaN values. These are days that **don’t have data**

…and calculates the **average transaction** for each day! Note that just like groupby, we must specify a numeric column to calculate our aggregates on.



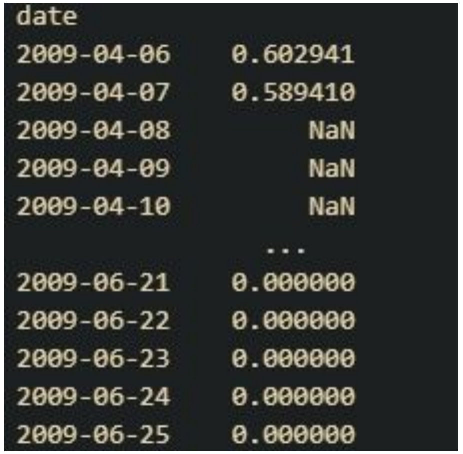
```
sns.lineplot(df, x="date", y="sentiment")
```

```
✓ 9m 14.9s
```

```
<Axes: xlabel='date', ylabel='sentiment'>
```

df["date"] = pd.to_datetime(df["date"])
df.resample(on="date", rule="1D")["sentiment"].mean()

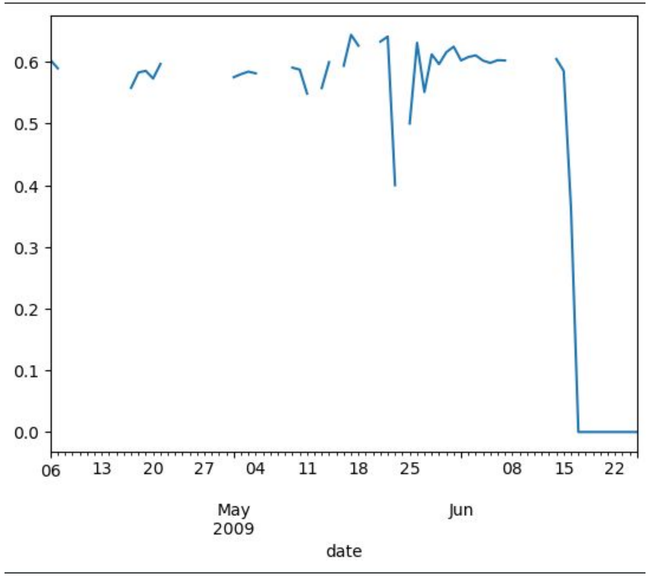
Let's apply these same principles to our Twitter dataset to better visualize our former line plot. First we will convert our "date" column to a date-time type.



```
date
2009-04-06    0.602941
2009-04-07    0.589410
2009-04-08         NaN
2009-04-09         NaN
2009-04-10         NaN
                ...
2009-06-21    0.000000
2009-06-22    0.000000
2009-06-23    0.000000
2009-06-24    0.000000
2009-06-25    0.000000
```

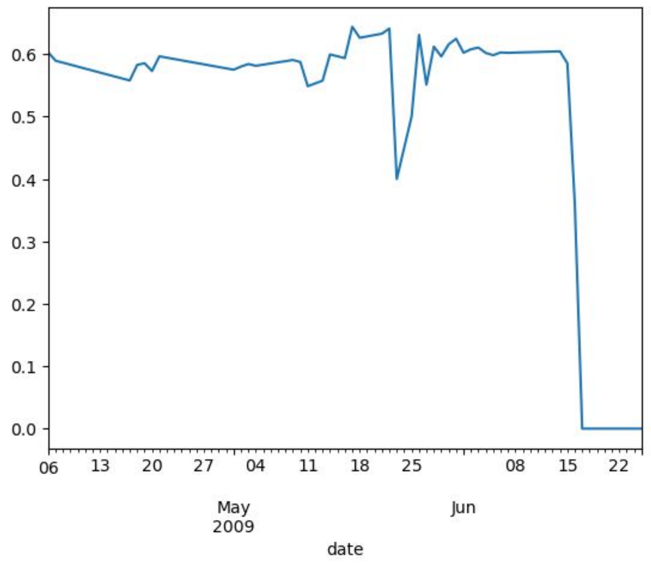
```
df["date"] = pd.to_datetime(df["date"])
df.resample(on="date", rule="1D")["sentiment"].mean()
```

Now we can resample our dataframe to calculate the average sentiments pers day. **While this is a good start**, how can we plot these values using a line plot?



```
df["date"] = pd.to_datetime(df["date"])
df.resample(on="date", rule="1D")["sentiment"].mean().plot.line()
```

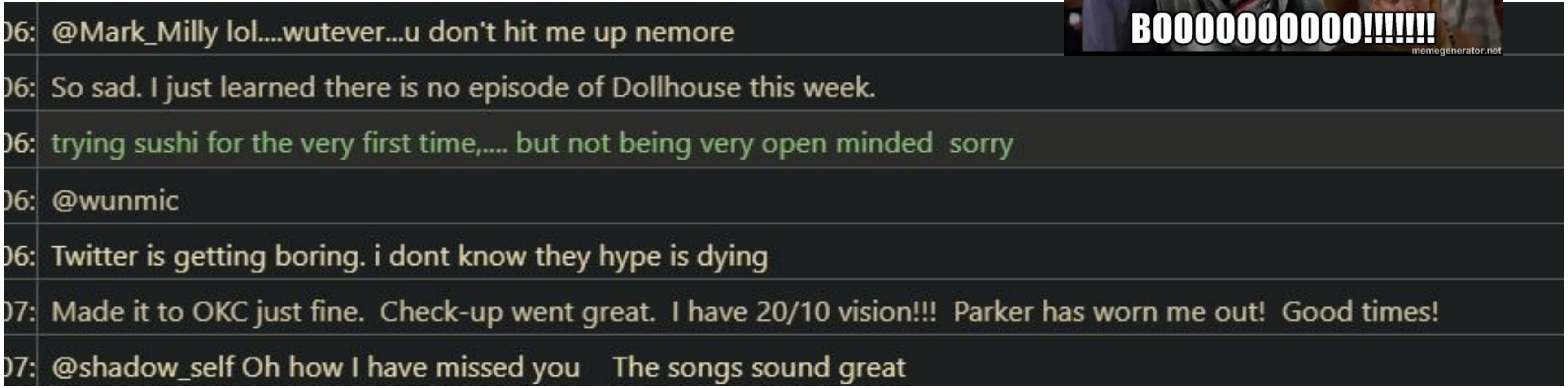
By simply attaching **plot.line()** at the end of our method chain, we can express this as a lineplot. We're closer to a "full" picture, but what do we appear to be missing here?



df.resample(on="date", rule="1D")["sentiment"].mean().interpolate().plot.line()

Notice that we have **null values** across days where we have **no sentiment data**. We can fill in these missing values by including **interpolate()**, which creates a linear connection between the last known pieces of data.

Regex



```
)6: @Mark_Milly lol....wutever...u don't hit me up nemore
)6: So sad. I just learned there is no episode of Dollhouse this week.
)6: trying sushi for the very first time,.... but not being very open minded  sorry
)6: @wunmic
)6: Twitter is getting boring. i dont know they hype is dying
)7: Made it to OKC just fine.  Check-up went great.  I have 20/10 vision!!!  Parker has worn me out!  Good times!
)7: @shadow_self Oh how I have missed you    The songs sound great
```

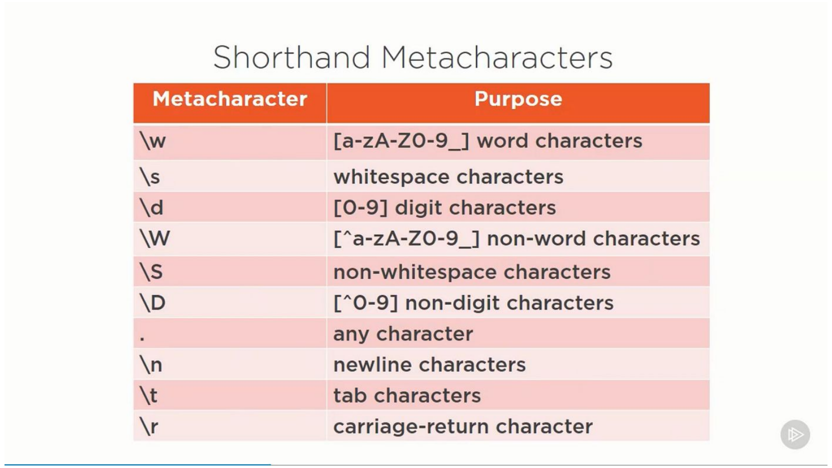
As this is a Twitter dataset, we should probably utilize some sort of text-based analysis. What have we learned about in the past which will help us figure out what people are talking about?

Regular Expressions (Regex)

Let's bring back regex (reg-ecks) to continue interpreting our dataset.

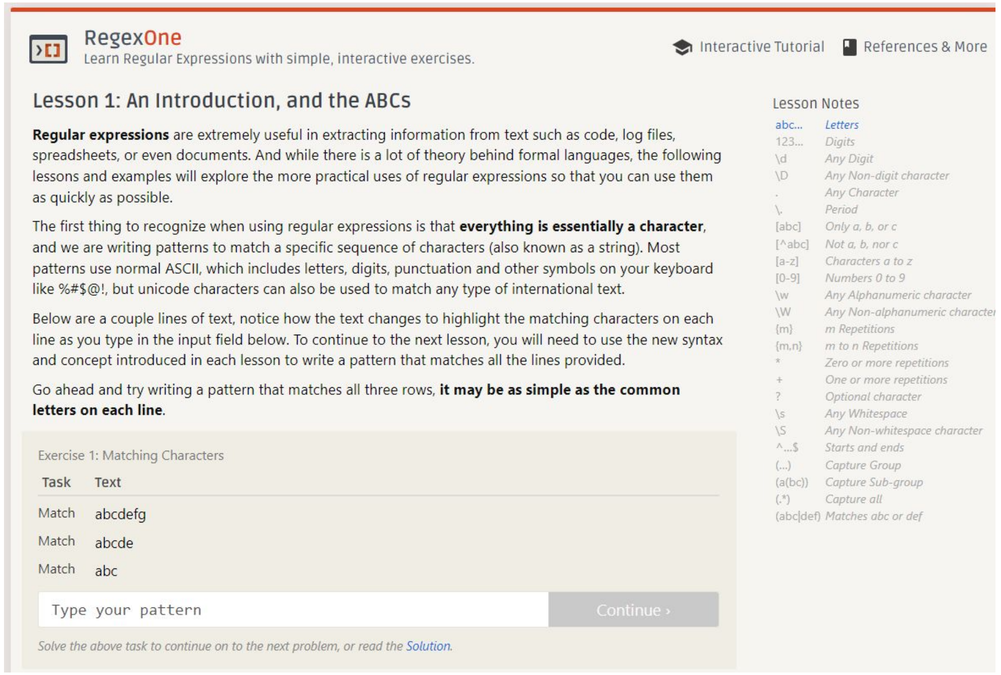
Regular expressions are a powerful concept taken from **linguistics** that allows us to quickly search for text in a **text corpus**.

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 characte*
{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
Where is Farukh?
Farrrrukh is terrible
I like Python27
@@@19586

To search for text using regex, you simply describe a p**attern of text to search for**. Regex then searches all strings (or dataset rows) for strings that satisfy this **pattern**.

Regex Pattern

Farukh

review
Farukh is great
Where is Farukh?
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
Where is Farukh?
Farrrrukh is terrible
I like Python27
@@@19586

Rows 0 & 1 get matched. Why doesn't row 2 get matched?

Regex Pattern

Farukh

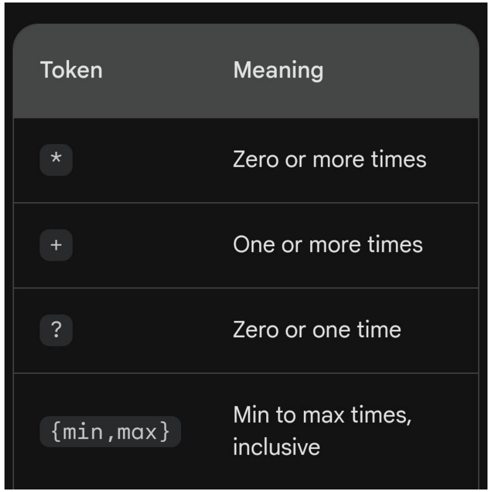
review
Farukh is great
Where is Farukh?
Farrrrukh is terrible
I like Python27
@@@19586

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

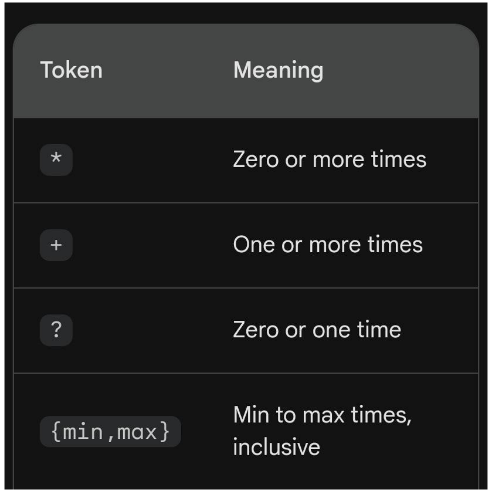
review
Farukh is great
Where is Farukh?
Farrrrukh is terrible
I like Python27
@@@19586

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



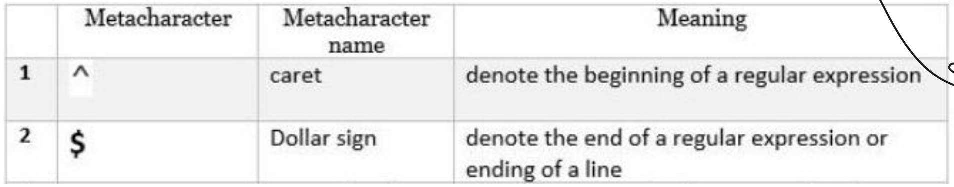
Token	Meaning
`*`	Zero or more times
`+`	One or more times
`?`	Zero or one time
`{min,max}`	Min to max times, inclusive

review
Farukh is great
Where is Farukh?
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*

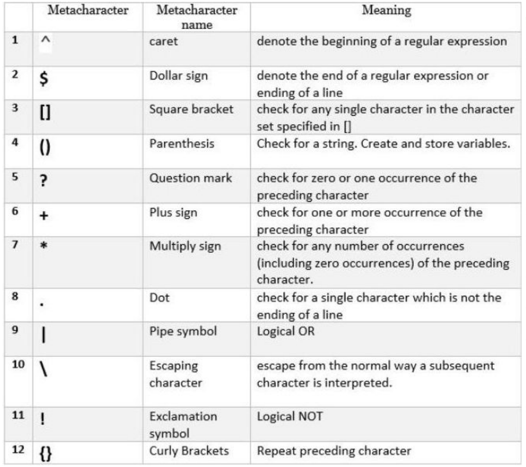


	Metacharacter	Metacharacter name	Meaning
1	^	caret	denote the beginning of a regular expression
2	$	Dollar sign	denote the end of a regular expression or ending of a line

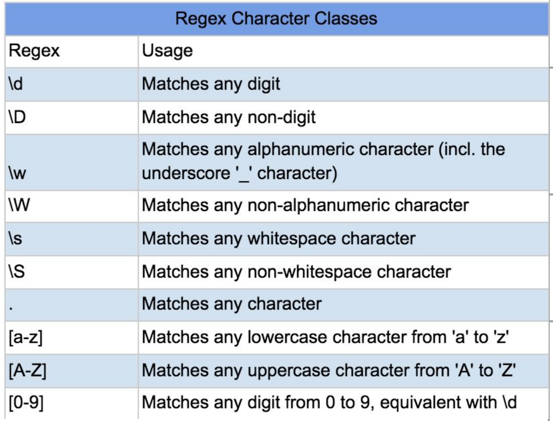
review
Farukh is great
Where is Farukh?
Farrrrukh is terrible
I like Python27
@@@19586

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

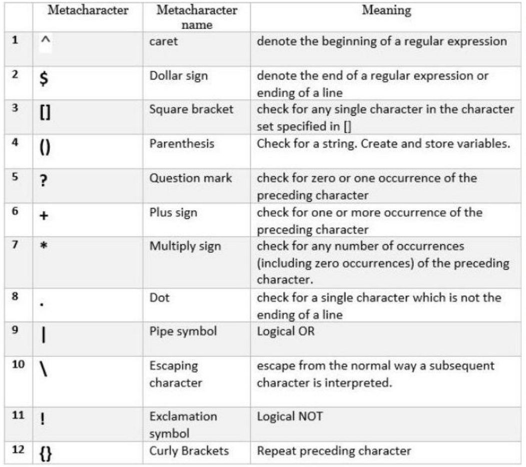
review
Farukh is great
Where is Farukh?
Farrrrukh is terrible
I like Python27
@@@19586

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.

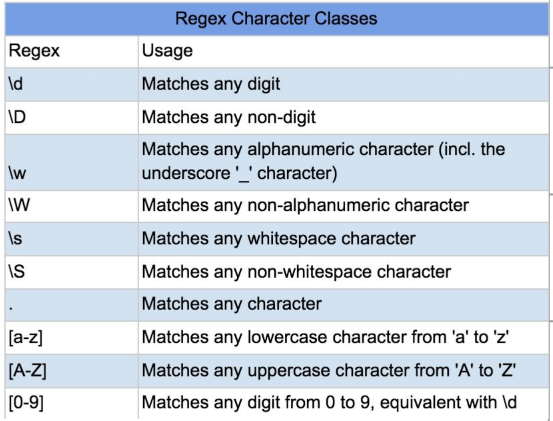
Using these tables, what do we write to match rows that **end with a number**?

Regex Pattern

\d$



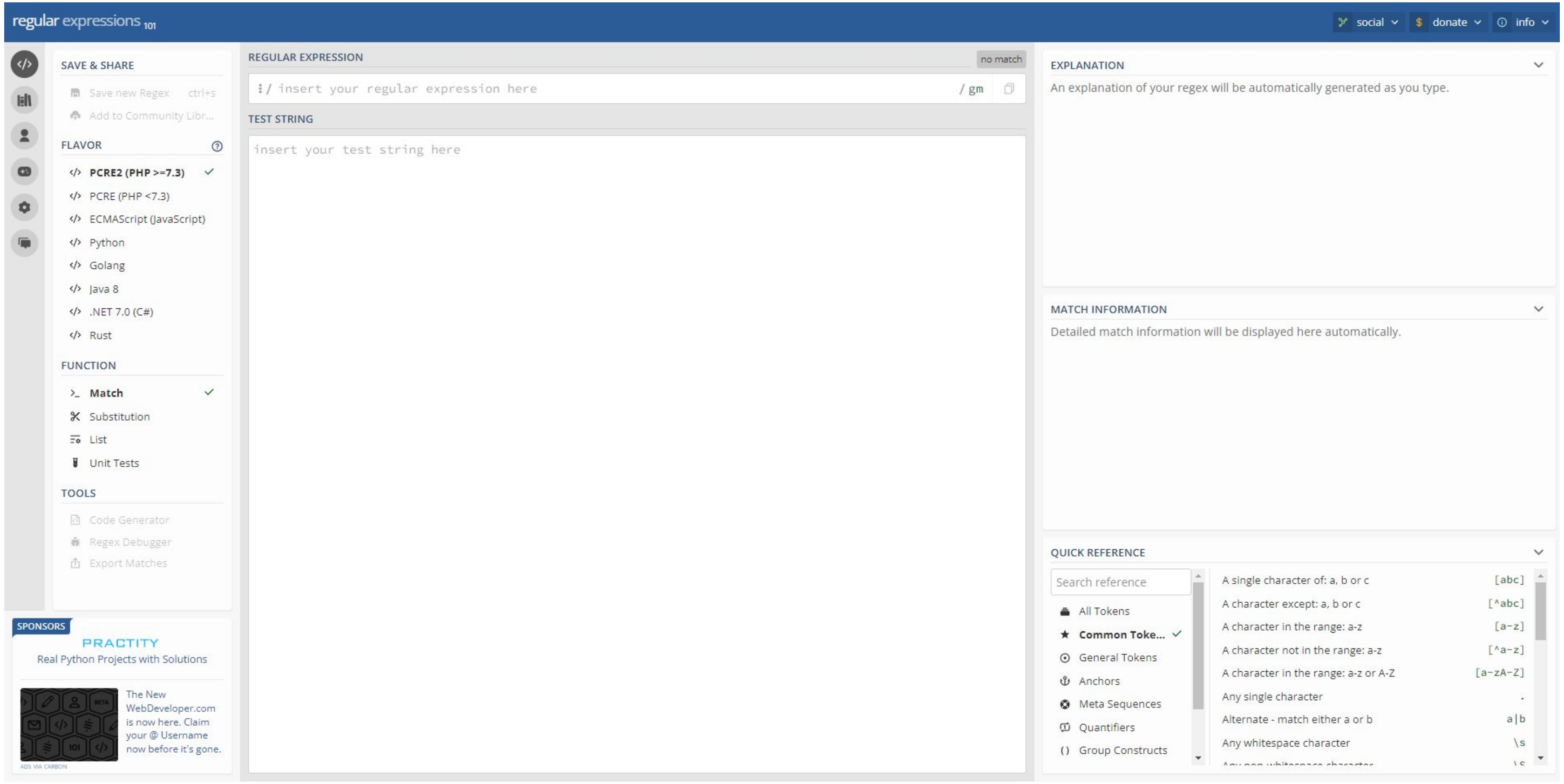
	Metacharacter	Metacharacter name	Meaning
1	^	caret	denote the beginning of a regular expression
2	$	Dollar sign	denote the end of a regular expression or ending of a line
3	[]	Square bracket	check for any single character in the character set specified in []
4	()	Parenthesis	Check for a string. Create 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
Where is Farukh?
Farrrrukh is terrible
I like Python27
@@@19586

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



Flavor: PCRE2 (PHP >=7.3) ✓

Function: Match ✓

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

Detailed match information will be displayed here automatically.

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]()

Regular Expressions (Regex)

There are many ways we can use regex in pandas. **We'll only go over 1.**

- ***Matching rows in pandas***

Remember, you will not learn the full breadth of syntax from 5 slides.

Instead, you'll gain these skills from practice, completing labs, projects, etc.

Also no one memorizes regex, **look up the documentation!**

age	age_spouse	State	income
32	35	New York	30,000
48	47	New Mexico	70,000
21	NA	California	20,0000

Find all participants who live in states starting with “New”

Let’s say we want to only get the rows in our dataset who’s string value matches some specific pattern.

age	age_spouse	State	income
32	35	New York	30,000
48	47	New Mexico	70,000
21	NA	California	20,0000

Find all participants who live in states starting with “New”

```
df[df[“State”].str.contains(“...”)]
```

This could be handled with a combination of **boolean indexing** & the string “**contains**” method. What will be the regex pattern we use in these quotes to get only states that start with “***New***”?

Note that this isn’t limited to pandas! This is base Python.

age	age_spouse	State	income
32	35	New York	30,000
48	47	New Mexico	70,000
21	NA	California	20,0000

Find all participants who live in states starting with “New”

```
df[df.State.str.contains(“^New”)]
```

user	tweet
Daniiej	omg i've an economics test.
sensuoushel p	FOX and the contestants won't go about it right
Taj_Milahi	good luck bro on the test

df[df[“tweet”].str.contains(“...”)]

Let’s say we’re attempting to find all tweets that make mention of the word “test.” We could use regex for this exact use-case! Which regex pattern could we input into our boolean index to match all “tests?”

user	tweet
Daniiej	omg i've an economics test.
sensuoushel p	FOX and the contestants won't go about it right
Taj_Milahi	good luck bro on the test

```
df[df["tweet"].str.contains("test")]
```

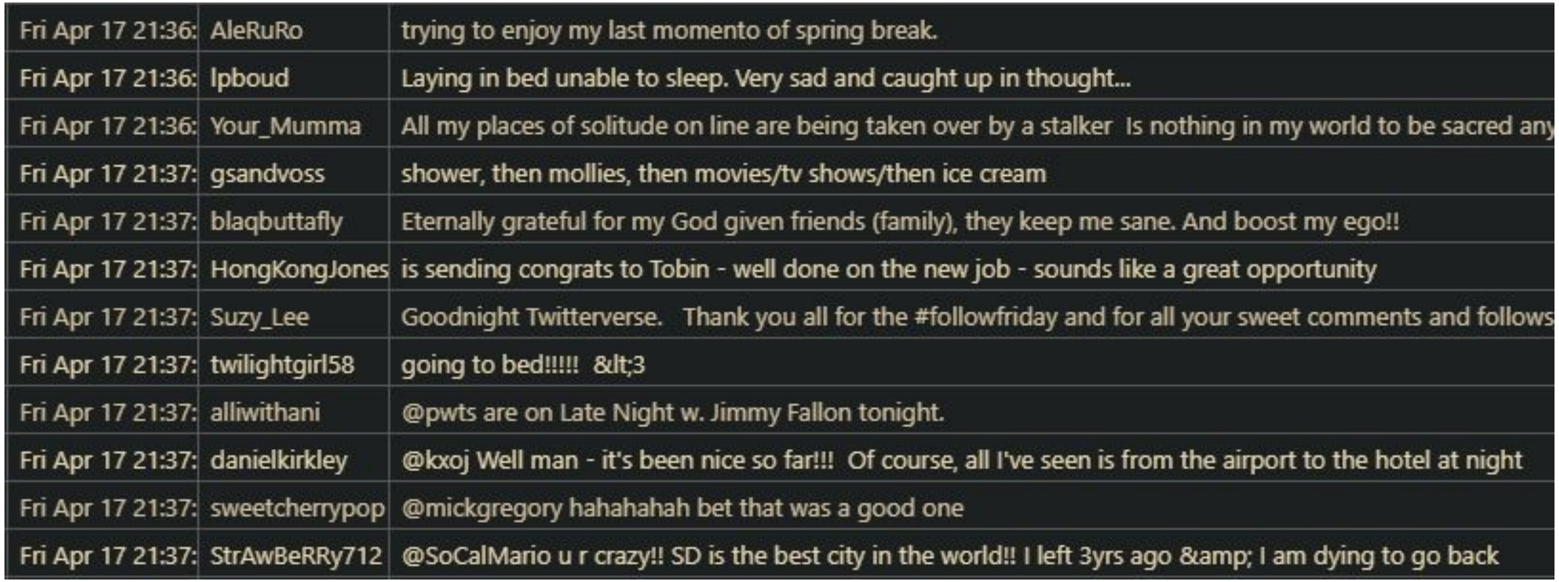
We could possibly use **test** however notice that this matches words that include the substring of "**test**" (such as "**contestant**", "**shortest**", "**Greatest**"). What should we include in our pattern to ensure we only match the word "test" (think, what usually indicates the beginning of a word?)

user	tweet
Daniiej	omg i've an economics test.
sensuoushelp	FOX and the contestants won't go about it right
Taj_Milahi	good luck bro on the test

```
df[df[“tweet”].str.contains(“ test”)]
```

By including a space in the beginning of our regex pattern, we will strictly match rows that contain the pattern “ test.” Notice that this is different from “test” by itself!

Twitter Dataset Case Study



Fri Apr 17 21:36:	AleRuRo	trying to enjoy my last momento of spring break.
Fri Apr 17 21:36:	lpboud	Laying in bed unable to sleep. Very sad and caught up in thought...
Fri Apr 17 21:36:	Your_Mumma	All my places of solitude on line are being taken over by a stalker Is nothing in my world to be sacred any
Fri Apr 17 21:37:	gsandvoss	shower, then mollies, then movies/tv shows/then ice cream
Fri Apr 17 21:37:	blaqbuttafly	Eternally grateful for my God given friends (family), they keep me sane. And boost my ego!!
Fri Apr 17 21:37:	HongKongJones	is sending congrats to Tobin - well done on the new job - sounds like a great opportunity
Fri Apr 17 21:37:	Suzy_Lee	Goodnight Twitterverse. Thank you all for the #followfriday and for all your sweet comments and follows
Fri Apr 17 21:37:	twilightgirl58	going to bed!!!!! <3
Fri Apr 17 21:37:	alliwithani	@pwts are on Late Night w. Jimmy Fallon tonight.
Fri Apr 17 21:37:	danielkirkley	@kxoj Well man - it's been nice so far!!! Of course, all I've seen is from the airport to the hotel at night
Fri Apr 17 21:37:	sweetcherrypop	@mickgregory hahahahah bet that was a good one
Fri Apr 17 21:37:	StrAwBeRRy712	@SoCalMario u r crazy!! SD is the best city in the world!! I left 3yrs ago &amp; I am dying to go back

Complete this analysis and meet back at 9:20 to answer analytical questions via the wheel.

TLAB #3

Lab (Due 5/14)

Please ensure that your EDA is completed **independently**.

Minas Gerais, Brazil

You are a data engineer at a Brazil-based weather prediction startup called Curu-Sight. The goal of this startup is to analyze weather trends in Brazil and predict the output of non-durable consumer goods at harvest time.

You will analyze a dataset that contains averages calculated based on rainfall, temperature, humidity, and wind metrics collected during the coffee growing season.

You will also analyze a dataset that contains Minas Gerais' crop output. **You will then combine these two datasets to explore how the weather influences coffee growth.**

Wednesday

Wednesday will entail:

- Hypothesis testing
- AB Testing on a real-world dataset