

NAYANA DAVIS

INTRO TO REGEX

What is Regex?

REGEX

Regular Expressions -- almost always just called "RegEx" -- are a specialized syntax for matching portions of text.

- ▶ You define a pattern, and search for parts of a string that match the pattern
- ▶ Effectively its own language
- ▶ Used in every major programming language (kind of like JSON)
- ▶ Designed to be a concise


REGEX

A regular expression is a pattern of text that consists of ordinary characters (for example, letters a through z) and special characters, known as metacharacters. The pattern describes one or more strings to match when searching a body of text. The regular expression serves as a template for matching a character pattern to the string being searched.

Regular Expression	Example Matches	Examples that do not Match
"123"	"12345", "abc123def", "abc123def", "123"	"1a2b3c", "321"
"b"	"b", "abc", "bread"	"xyz", "c", "B"

SPECIAL CHARACTERS: "."

- ▶ The special character "." is used to indicate that any character can be put in its place

Regular Expression	Example Matches	Examples that do not Match
"a.c"	"aac", "abc", "acc", "alchemy", "branch"	"add", "crash"
".t"	"bat", "habit", "oat"	"at", "it", "to" 

BEGINNINGS & ENDS OF STRINGS

- ▶ We can use "^" to match the start of a string and "\$" to match the end of string.
- ▶ "^a" will match all strings that start with "a"
- ▶ "a\$" will match all strings that end with "a"

YOU DO: 3 MINUTES

Assign a regular expression that is 7 characters long and matches every string in `strings`, but not `bad_string`, to the variable `regex`.

```
1 strings = ["better not put too much",  
            "butter in the", "batter"]  
2 bad_string = "We also wouldn't want it  
               to be bitter"
```


TEXT

ANSWER

```
regex = "^b.tter"
```

ASKREDDIT DATA

- ▶ DEMO
- ▶ We'll be working with a dataset of the top 1000 posts on AskReddit in 2015

RE MODULE

- ▶ In Python, we use the re module to work with regular expressions
- ▶ In the module's documentation you'll find a list of special characters we can use to refine a regular expression.
- ▶ <https://docs.python.org/3/library/re.html>

READING AND PRINTING THE DATASET

- ▶ Let's use the csv module to read and then print our dataset
- ▶ The csv module implements classes to read and write tabular data in CSV format.
- ▶ `reader()` returns a reader object which will iterate over lines in the given csvfile.

TESTING FOR MATCHES

- ▶ With `re.search(regex, string)`, we can check if string is a match for regex
- ▶ If it is, it will return a match object. If it isn't, it will return `None`.

ACCOUNTING FOR INCONSISTENCIES

- ▶ In regular expressions, square brackets are used to indicate that any character within the square bracket can fill the space.
- ▶ For example, the regular expression "[bcr]at" would be matched by strings with the substrings "bat", "cat", and "rat", but nothing else.
- ▶ We can account for the inconsistency of people writing "of Reddit" versus "of reddit" using square brackets. []

ESCAPING SPECIAL CHARACTERS

- ▶ In regular expressions, escaping a character means indicating that you don't want the character to do anything special, and that it should be treated as any other character would be.
- ▶ We use "\" (backslash) to escape characters in regular expressions.
- ▶ If we used ".\$", it would match all strings that contain any character, since "." has that special meaning. Instead, we must escape the "." with a backslash, and our regular expression becomes "\\.\$".

YOU DO: 5 MINUTES

- ▶ In our dataset, some people tag serious posts as "[Serious]", and others as "[serious]". We should account for both capitalizations.
- ▶ Refine the code to count how many posts have either "[Serious]" or "[serious]" in their title.
- ▶ Assign the count to `serious_count`.

YOU DO: 5 MINUTES

- ▶ In our dataset, some users have tagged their post using "(Serious)" or "(serious)". We should account for both square brackets and parenthesis. We can do this using square bracket notation, and escaping "[", "]", "(", and ")" with the backslash.
- ▶ Refine the code to count how many posts are tagged as serious, using either square brackets or parenthesis.
- ▶ Assign the count to `serious_count`.

MULTIPLE REGULAR EXPRESSIONS

- ▶ To combine regular expressions, we use the "|" character.

YOU DO: 10 MINUTES

- ▶ Use the "^" character to count how many posts have the serious tag at the beginning of their title. Assign this count to `serious_start_count`.
- ▶ Use the "\$" character to count how many posts have the serious tag at the end of their title. Assign this count to `serious_end_count`.
- ▶ Use the "|" character to count how many posts have the serious tag at either the beginning or end of their title. Assign this count to `serious_count_final`.