TOOL

Regular Expressions Reference Guide

Regex techniques increase your preprocessing efficiency because they allow you to search for complex patterns in one pass through a document. In this course, you used regex to find patterns and also replace one pattern with another. These skills are a good foundation for performing NLP, as well as for allowing you to grow your regex skills outside of this course. Remember, even seasoned programmers need to consult reference materials for regex at times; with practice, you will become more comfortable with regex rules. Use this tool as a guide to using regex, both throughout this certificate and when you use NLP techniques on your own data.

Pattern Matching Rules

Use the following table to find specific pattern matching rules. Recall that you can combine these rules to search for complex patterns.

Pattern Matching Rule	What It Searches For
`.`	Any single character. When inside character class brackets, such as `[.]`, it matches a period only.
`^`	The start of the string.
`$`	The end of the string.
`\b`	Word boundary. For example, `r'\bthing\b'` matches the word `'thing'` (surrounded by spaces or punctuation), but not 'nothing' or any other word with a subword `'thing'`.
`?`	Zero or one of the previous pattern.
`*`	Any number of repeated cases of the previous pattern. Example: `Hi!*` matches zero or more `!`. So, `Hi`, `Hi!`, `Hi!!`, `Hi!!!`, ... are matched, but `Hi` is not.

Pattern Matching Rule	What It Searches For
`+`	One or more numbers of repeated cases of the previous pattern Example: `Hi!+` one or more `!`. So, `Hi!`, `Hi!!`, `Hi!!!` ... are matched, but `Hi` is not.
`[]`	A character class. Example: `[0-9a.]`matches any digit, letter **“a”** or a period.
`[^]`	Any character excluded from the square brackets containing symbols after `^`.
`|`	Any pattern on the left **or** the right of the pipe symbol.
`\d` or `[0-0]`	Any decimal digit. The dash represents the range of digits.
`\D` or `[^0-9]`	Any non-decimal digit.
`\s`	Whitespace characters, including `‘ ’`, `‘\t’`, `‘\n’`, `‘\r’`.
`\S`	Non-whitespace characters.
`\w` or `[a-zA-Z0-9]`	Any alphanumeric character.
`\W` or `[a-zA-Z0-9]`	Any non-alphanumeric character.

Pattern Matching Rule	What It Searches For
`()`	A match group. Example: `(he|we|they)` matches any of the listed pronouns in a string.

Escaping Pattern Searches

If you do not end your regex search, every character you write in your code will be part of your search. To disable the effect of these special patterns, use escaping by prefixing patterns with a backslash `\`. For example:

1. `\.` escapes period's super powers and makes it match a period only.
2. `\[` escapes the start of the character class brackets and simply matches a square bracket.
   - Example: `[ab]` matches `a` or `b`, but `\[ab\]` literally matches `[ab]` string
3. `\?` literally matches a question mark, not any single character.
   - Example: `H?` matches `Hi`, `H1`, `H.`, and any other character following letter `H`, but `H\?` literally matches `H?` only.
4. `\+` literally matches a plus sign, not at least one preceding character.

Methods Covered in this Course

There are basic methods from the `re` library covered in this course are: `search(), split(), findall(), sub(), IGNORECASE()` and `finditer().`

More Regex Resources

For more on using the `re` module in Python-3, see the documentation.