Python 3 Regex Playbook

Working with Regular Expressions in Python

Maaike van Putten
Trainer & Software Developer

@brightboost | www.brightboost.nl

Overview

Regex definition

Use cases for Regex

How to read Regex

- Basics
- Special characters
- Repetition and grouping

Python and Regex

- Functions that work with Regex
- The Match object

Familiar with Regex Already?

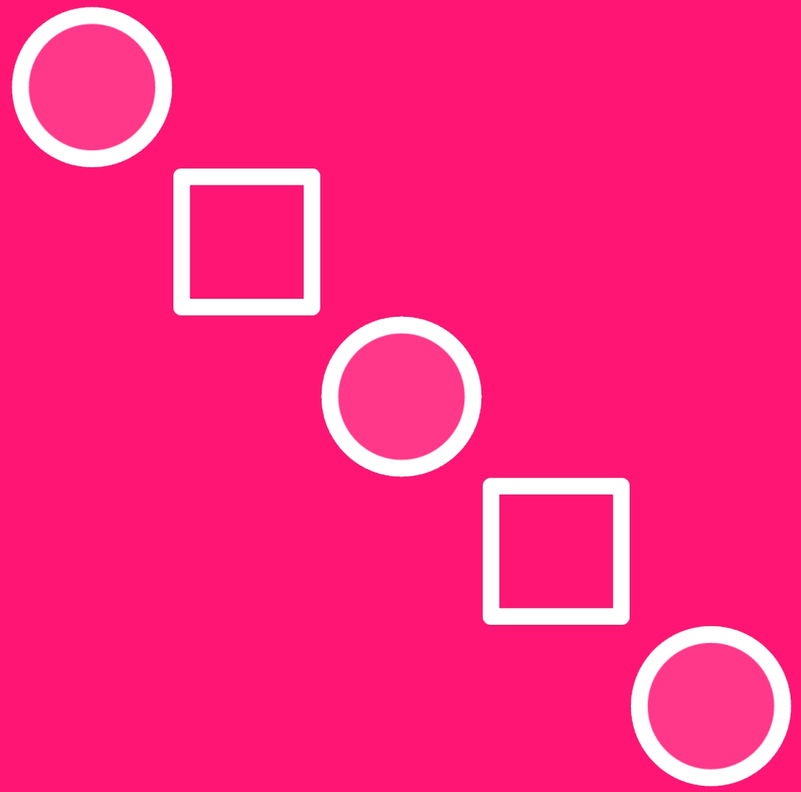
You can consider skipping (part of) this module.

Overview

What Regex is

Regex use cases

Basics of reading Regex



Regular Expression

Sequence of characters that represents a text pattern.

Checking if a string matches the regular expression.

Examples of Regex Use Cases

Search a text file for a specific pattern

Validating certain data types

Replacing matches to the pattern

Reading Regex can be intimidating

Reading Regex

From left to right.

Matching literal characters

Simple regex: “e”

Matches the first “e” in a string

Python Regex Playbook on Pluralsight

Match regex “e”

This contains the “e” and there’s a match.

Python Regex Playbook on Pluralsight

Match regex “sight”

The text contains “sight” and there’s a match.

Python Regex Playbook on Pluralsight

Match regex “sght”

Doesn’t contain “sght” and therefore there’s no match.

Python Regex Playbook on Pluralsight

Match regex “Sight”

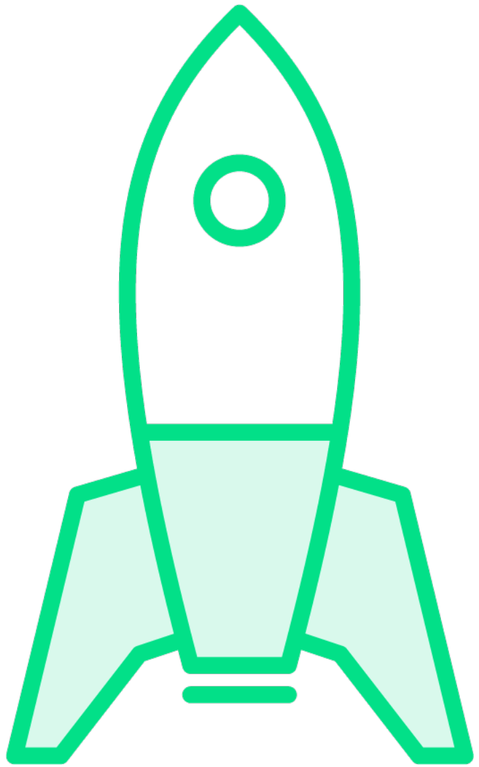
The text doesn’t contain “Sight”, Regex is case sensitive by default.

Special characters

Don’t represent a literal string but add extra abstraction to our Regex.

Overview

- Wildcard
- Anchors
- Escape character
- Character class
- Group characters



Higher complexity than searching for literal strings

Special characters have a special meaning

Specify a pattern instead of literal text

Different strings can match the same pattern

```
Python Regex is fun.
```

Match regex “f.n”

The dot matches any character, except for a new line. Since the text contains “fun”, there’s a match.

```
I’m a fan.
```

Match regex “f.n”

The dot matches any character, except for a new line. Since the text contains “fan”, there’s a match.

```
I’m a fan.
```

Match regex “f..n”

The dot matches any character, except for a new line. No match.

Pluralsight

Match regex “^sight”

^ matches the start of the string, no match here.

Pluralsight

Match regex “^Plural”

^ matches the start of the string, there’s a match here.

Pluralsight

Match regex “sight$”

$ matches the end of the string, there’s a match here.

Anchors

^ and $ are called anchors

Match a condition and not a character

Escape Character

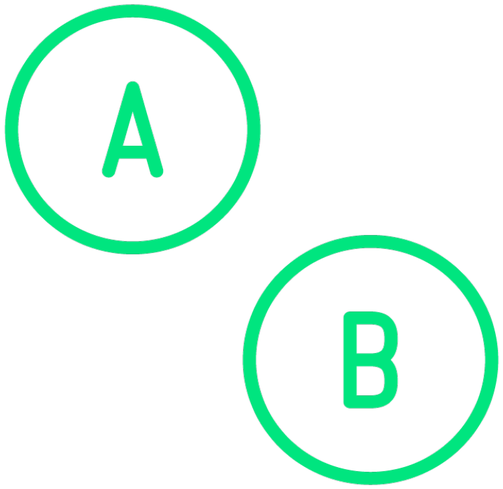
Necessary for looking for the literal value of special characters

\ is the escape character

Looking for “regex\.” will match to “I like regex.”

But it won’t match to “I like regex!”

Character Class



Match one character from a group of characters

For example, the “a” or the “e”

We can create a character class with square brackets: [ae]

Character class is sometimes called “a character set”

```
I’d like to apologise.
```

Match regex “apologi[sz]e”

This matches with apologize or apologise.

Range

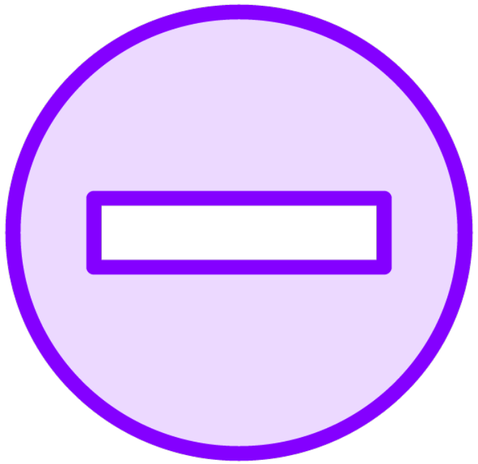
We can specify a range of characters using character classes

[a-z]

[a-g]

[a-gA-G]

Exclude Characters with Negating



Exclude characters or a range of them

Done by negating

[^abc]

In a character class, only the], \, ^ and – have a special meaning.

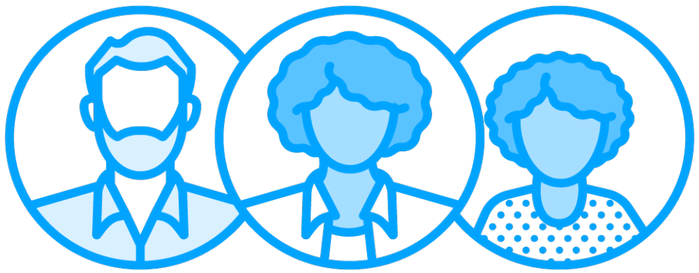
`\d`	◂ Matches with a decimal digit such as 3
`\D`	◂ Any non decimal digit, such as ’h’
`\s`	◂ Any whitespace character
`\S`	◂ Any non-whitespace character
`\w`	◂ Word characters: alphanumeric characters and underscores
`\W`	◂ Non-word characters

```
I prefer to print on A4.
```

Match regex “[aA]\d”

This matches, since it’s an A followed by a digit.

Character Group



Treat the group as a single unit

(abc)

(abc|def)

Repetition

We can repeat parts of our regex

We can specify the number of times it needs repeating as well

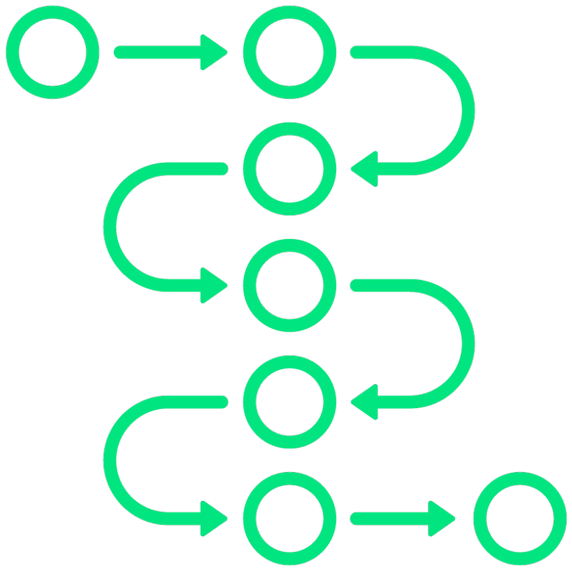
Overview

Repetition

Different ways of repeating

Examples

Repetition



Repeating a character or group of characters several times

Ways to achieve repetition:

- +
- *
- ?
- {n}
- {n,m}

`data.txt`

Match regex ".+\.txt"

This matches, since "data" matches with .+
And .txt matches with \.txt

```
.txt
```

Match regex “.*\.txt”

This matches, since “” (empty string) matches with .*
And .txt matches with \.txt

```
colou?r
```

Match regex “colour”

This matches, since “colour” has one u. The text “color” would also match.

```
(ha)*
```

Match regex “haha”

This matches, since “ha” is present 0 or more times.

```
(ha){2}
```

Match regex “haha”

This matches, since “ha” is present 2 times.

```
(ha){2,5}
```

Match regex “hahaha”

This matches, since “ha” is present 3 times.

We want to specify a hex color code.

Valid code is 6 characters and starts with a #.

```
^#\d{6}$
```

We want to specify a hex color code.

Valid code is 6 characters and starts with a #.

But A-F upper- and lowercase is also valid.

```
^#(\d|[A-Fa-f]){6}$
```

We want to specify a hex color code.

Valid code is 6 characters and starts with a #.

But # and only 3 characters is also valid.

```
^#((\d|[A-Fa-f]){3}){1,2}$
```

Overview

Python as the re library to deal with regex

Different functions:

- search
- match
- fullmatch
- findall
- sub and subn
- split
- escape
- compile

Search Function

Search scans through the provided string looking for the first match.

Accepts three parameters: pattern, string to be searched, and optional flags

Returns None if no match was found, returns the match object if a match was found

Match

Checks if the beginning of the string matches the pattern

Accepts three parameters: pattern, string to be searched, and optional flags

Returns None if no match was found, returns the match object if a match was found

Fullmatch Function

Checks if the full string matches the regex pattern

Accepts three parameters: pattern, string to be searched, and optional flags

Returns None if no match was found, returns the match object if a match was found

Findall Function

Searches a string for all non-overlapping matches

Accepts three parameters: pattern, string to be searched, and optional flags

Returns the results of matches

Sub and Subn Function

Search a string for all non-overlapping occurrences and replace them with a string

Accepts five parameters: pattern, string to be searched, replacement, optional max number of replacements, and optional flags

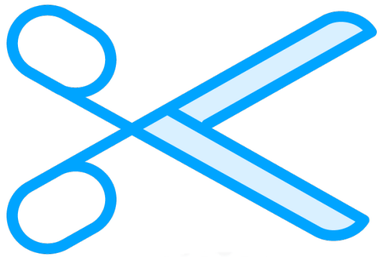
Sub doesn't return anything, subn returns a tuple containing the new string and number of replacements made

Split Function

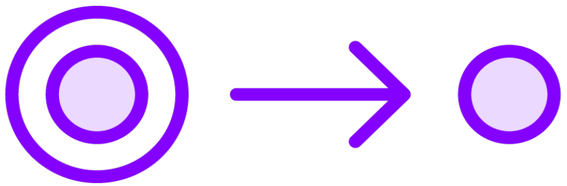
Splits a string at every occurrence of the pattern

Takes four parameters: pattern, string, optional max number of splits, and optional flags

Returns a list of the substrings



Escape Function



Escape is used to escape special characters so that they can be matched literally

Accepts one parameter: pattern

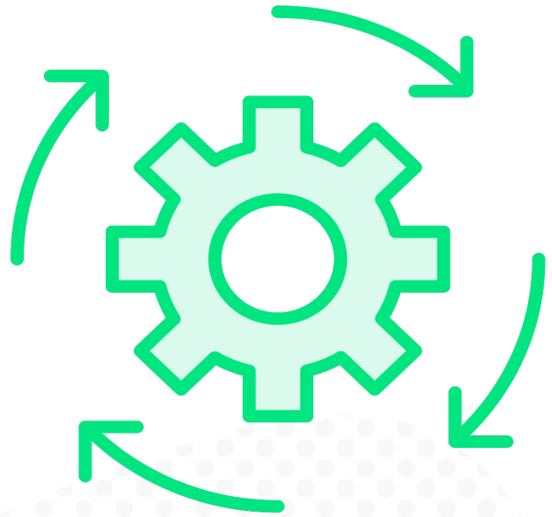
Returns the escaped pattern

Compile Function

Compile a pattern into a regex object

We can pass the object to other functions

Makes the code more efficient when we want to reuse a pattern



Overview

Match object

Methods on the match object

Demo

Match object

Always has a Boolean value of true

Has methods to get information about the match

Methods on the Match Object

string()

group()

groups()

start()

end()

span()

Demo

Match the pattern for a phone number to several strings

Inspect the match object

Overview

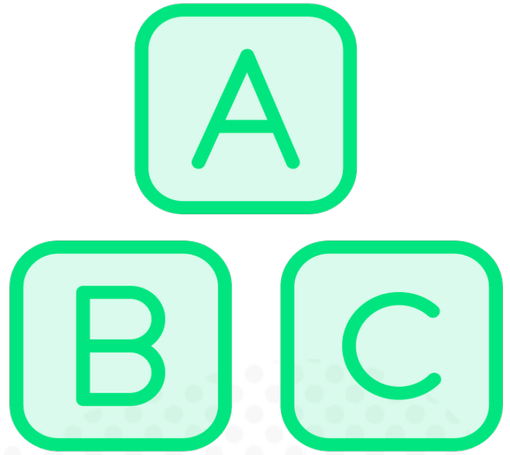
Regex flags

Different flags:

- Case insensitivity
- Multiline
- Dotall matching

Case Insensitivity

The re.I or re.IGNORECASE flag makes a pattern match case insensitive



Multiline

^ and $ match the start and end of each line, not the start and end of the entire string

Achieved with the flag: re.M or re.MULTILINE

Dotall matching

The flag re.S or re.DOTALL makes the . Character match any character including new lines