Regex + scrapping

1.10.2020

regex

Регулярные выражения - язык описания шаблонов (patterns) для извлечения информации из текста.

Регулярка	Её смысл
`simple text`	В точности текст «simple text»
`\d{5}`	Последовательности из 5 цифр `\d` означает любую цифру `{5}` — ровно 5 раз
`\d\d/\d\d/\d{4}`	Даты в формате ДД/ММ/ГГГГ (и прочие куски, на них похожие, например, 98/76/5432)
`\b\w{3}\b`	Слова в точности из трёх букв `\b` означает границу слова (с одной стороны буква, а с другой — нет) `\w` — любая буква, `{3}` — ровно три раза
`[-+]?\d+`	Целое число, например, 7, +17, -42, 0013 (возможны ведущие нули) `[-+]?` — либо -, либо +, либо пусто `\d+` — последовательность из 1 или более цифр
`[-+]?(?:\d+(?:\.\d*)?|\.\d+)(?:[eE][-+]?\d+)?`	Действительное число, возможно в экспоненциальной записи Например, 0.2, +5.45, -.4, 6e23, -3.17E-14. См. ниже картинку.

Write only code

```
(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-
\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0
-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:25[0-5]|
2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?|[a-z0-9-]*[a-z0-9]:
(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\])
```

regex

- Поиск точного совпадения.
- Поиск шаблонного совпадения.
- Возможность введения переменных.
- Жадность.
- Метасимволы.

regex

Точное совпадение:

Строка на вход «aaa bbb ccc»

шаблон: r"a"

на выходе: [a, a, a]

regex

[] - символ «или»

Строка на вход «aaa bbb ccc»

шаблон: r"[ab]"

на выходе: [a, a, a, b, b, b]

regex

() - группировка

Строка на вход «aaa bbb ccc»

шаблон: r"(aaa) bbb"

на выходе: [aaa]

regex

Если группа не нужна, то в группе ставим :?

Строка на вход «aaa bbb ccc»

шаблон: r"(aaa) (?:bbb)"

на выходе: [aaa]

regex

Метасимволы

- \d - все цифры [0-9]
- \w - все буквы [a-Za-Я]
- \s - пробельные символы
- \b - слова

regex

Метасимволы

- \D - Не цифры [ô-9]
- \W - Не буквы [â-Za-Я]
- \S - Не пробельные символы
- \B - Не слова

regex

Шаблон	Описание	Пример	Применяем к тексту
.	Один любой символ, кроме новой строки \n.	м.л.ко	молоко, малако, Им0л0коИхлеб
\d	Любая цифра	СУ\d\d	СУ35, СУ111, АЛСУ14
\D	Любой символ, кроме цифры	926\D123	926)123, 1926-1234
\s	Любой пробельный символ (пробел, табуляция, конец строки и т.п.)	бор\sода	бор ода, бор ода, борода
\S	Любой непробельный символ	\S123	X123, я123, !123456, 1 + 123456
\w	Любая буква (то, что может быть частью слова), а также цифры и _	\w\w\w	Год, f_3, qwert
\W	Любая не-буква, не-цифра и не подчёркивание	сом\W	сом!, сом?

regex

`[..]`	Один из символов в скобках, а также любой символ из диапазона `a-b`	`[0-9][0-9A-Fa-f]`	12, 1F, 4B
`[^..]`	Любой символ, кроме перечисленных	`<[^>]>`	<1>, <a>, <>
`\d≈[0-9]`, `\D≈[^0-9]`, `\w≈[0-9a-zA-Z a-яA-ЯёЁ]`, `\s≈[ \f\n\r\t\v]`	Буква “ё” не включается в общий диапазон букв! Вообще говоря, в `\d` включается всё, что в юникоде помечено как «цифра», а в `\w` — как буква. Ещё много всего!		
`[abc-]`, `[-1]`	если нужен минус, его нужно указать последним или первым		
`[*` `[(+\\\]\t]`	внутри скобок нужно экранировать только] и \		
`\b`	Начало или конец слова (слева пусто или не-буква, справа буква и наоборот). В отличие от предыдущих соответствует позиции, а не символу	`\bвал`	вал, перевал, Перевалка
`\B`	Не граница слова: либо и слева, и справа буквы, либо и слева, и справа НЕ буквы	`\Bвал`	перевал, вал, Перевалка
		`\Bвал\B`	перевал, вал, Перевалка

Check here

regex

Квантификаторы

Шаблон	Описание	Пример
`{n}`	Ровно n повторений	`\d{4}`
`{m,n}`	От m до n повторений включительно	`\d{2,4}`
`{m,}`	Не менее m повторений	`\d{3,}`
`{,n}`	Не более n повторений	`\d{,2}`

regex

Пример кватификатора

Строка на вход «На дворе - трава, на траве - дрова.»

шаблон: r"[а-я]{3,}"

на выходе: ['дворе', 'трава', 'траве', 'дрова']

Синонимы кватификаторов

Шаблон	Описание	Пример
?	Ноль или одно вхождение, синоним {0,1}	валы?
*	Ноль или более, синоним {0,}	су\d*
+	Одно или более, синоним {1,}	а\)+

regex

Жадность - по умолчанию регулярные выражения захватывают максимум символов, которые помещаются под шаблон.

Шаблон	Описание	Пример

`*?`
`+?`
`??`
`{m,n}?`
`{,n}?`
`{m,}?`
	По умолчанию квантификаторы *жадные* — захватывают максимально возможное число символов.
	Добавление `?` делает их *ленивыми*, они захватывают минимально возможное число символов
		`\(.*\)`
		`\(.*?\)`

Python

Функция	Её смысл
`re.search(pattern, string)`	Найти в строке `string` первую строчку, подходящую под шаблон `pattern`;
`re.fullmatch(pattern, string)`	Проверить, подходит ли строка `string` под шаблон `pattern`;
`re.split(pattern, string, maxsplit=0)`	Аналог `str.split()`, только разделение происходит по подстрокам, подходящим под шаблон `pattern`;
`re.findall(pattern, string)`	Найти в строке `string` все непересекающиеся шаблоны `pattern`;
`re.finditer(pattern, string)`	Итератор всем непересекающимся шаблонам `pattern` в строке `string` (выдаются `match`-объекты);
`re.sub(pattern, repl, string, count=0)`	Заменить в строке `string` все непересекающиеся шаблоны `pattern` на `repl`;

scraping

Скрапинг - процесс сбора информации с веб страниц.

Парсинг - процесс обработки текста, часто подразумевается разбор текста на составные части.

scraping

- requests + bs4 - низкоуровневый поиск.
- selenium - имитация клиента.
- scrapy - готовое решение для скрапинга, основано на ассинхронных запросах.

scraping

Из чего состоит веб страница

- html
- css
- js

scraping

Возможные проблемы:

- Скорость...
- Тайминги, фризы.
- AJAX страницы.