



# Регулярные выражения



Олег Булыгин



**Олег Булыгин**

- Преподаватель на курсах “Основы языка программирования Python”, “Продвинутый Python”, “Python для анализа данных” в Нетологии
- IT-аудитор в ПАО “Сбербанк”

# О чём мы поговорим сегодня

3

1. Что такое регулярные выражения и чем они отличаются от простого поиска
2. Синтаксис регулярных выражений
3. Модуль re в Python
4. Примеры использования регулярных выражений

**Регулярные выражения (Regular Expressions, regex)** – это строки, содержащие совокупность *обычных символов* и *специальных метасимволов*, которые описывают определенный шаблон.

Эти шаблоны, к примеру, можно использовать для того, чтобы:

- находить и заменять что-то в текстовых данных;
- валидировать строковые поля.

Шаблоны могут быть настолько сложными, что другими способами определить их будет крайне трудно.

Во время изучения  
чего-то нового,  
я самозабвенно  
выдумываю  
невероятные  
ситуации, в которых  
это умение поможет  
мне спасти мир

О нет! Убийца должно  
быть последовал  
за ней в отпуск!



Но чтобы узнать где он, нам нужно  
прочитать 200 Мб писем в поисках  
чего-то похожего по формату с адресом!



Это безнадежно!

Всем расступиться



Я знаю регулярные  
выражения



**«Если у вас есть  
проблема, и вы  
собираетесь решать  
ее с использованием  
регулярных  
выражений, то у вас  
уже две проблемы.»\***

\*шутка из интернетов

\*\* это не шутка

# Синтаксис регулярных выражений

6

**Обычные символы** в регулярных выражениях интерпретируются как *обычные строковые символы*.

**Метасимволы** могут быть интерпретироваться как:

- *определенная группа обычных символов;*
- *специфика других символов (например, их количество или расположение).*

## Примеры метасимволов регулярных выражений – группы символов

7

<b>\d</b>	любая цифра. Аналогично [0-9]
<b>\D</b>	все, кроме цифры. Аналогично [^0-9]
<b>\w</b>	любая буква, цифра и символ подчеркивания
<b>\W</b>	все, кроме букв, цифр и символа подчеркивания
<b>\s</b>	любой пробельный символ, включая сам пробел: [ \t\n\r\f\v]
<b>\S</b>	все, кроме пробельных символов

## Примеры метасимволов регулярных выражений – квантификаторы

8

<b>*</b>	любое количество вхождений предыдущего символа (группы символов), от 0 до бесконечности
<b>+</b>	количество вхождений предыдущего символа (группы символов) от 1 до бесконечности
<b>?</b>	0 или 1 вхождение предыдущего символа (группы символов)
<b>{n}</b>	точное количество вхождений – n раз предыдущего символа (группы символов)
<b>{n, m}</b>	количество вхождений не менее n и не более m раз предыдущего символа (группы символов)



# Raw strings

9

**raw string** – подавляет значение *escape-последовательностей*, обозначается префиксом **r** перед обычной строкой.



Настоятельно рекомендуется использовать всегда при работе с регулярными выражениями для избежания проблем.

## Основные функции модуля **re**

10

***re.findall(pattern, string)*** – ищет во всем тексте, возвращает список всех найденных совпадений;

***re.split(pattern, string)*** – разделяет строку по заданному шаблону;

***re.match(pattern, string)*** – ищет по заданному шаблону в начале строки;

***re.search(pattern, string)*** – ищет во всем тексте, возвращает первое совпадение;

## Основные функции модуля **re**

11

***re.findall(pattern, string)*** – ищет во всем тексте, возвращает список всех найденных совпадений;

***re.split(pattern, string)*** – разделяет строку по заданному шаблону;

***re.match(pattern, string)*** – ищет по заданному шаблону в начале строки;

***re.search(pattern, string)*** – ищет во всем тексте, возвращает первое совпадение;

## Основные функции модуля **re**

12

***re.findall(pattern, string)*** – ищет во всем тексте, возвращает список всех найденных совпадений;

***re.split(pattern, string)*** – разделяет строку по заданному шаблону;

***re.match(pattern, string)*** – ищет по заданному шаблону в начале строки;

***re.search(pattern, string)*** – ищет во всем тексте, возвращает первое совпадение;

## Еще примеры метасимволов

13

.	любой символ
^	1) начало строки, 2) инвертирование («всё, кроме») при применении с [ ]
\$	конец строки
\	символ экранирования. Например, символ точки: \.
	логическое “ИЛИ”
[ ]	применяет “ИЛИ” к набору символов, любой из которых может встретиться в тексте <i>Например, [а-яёА-ЯЁ] – любая буква русского алфавита в любом регистре</i>

***re.sub(pattern, repl, string)*** – ищет шаблон в строке и заменяет его на указанную подстроку.

# Скобочные группы

14

Часть шаблона можно заключить в скобки (...). Это называется «**скобочная группа**».

Это позволяет:

1. поместить часть шаблона в отдельную структуру (группу);
2. если установить квантификатор после скобок, то он будет применяться ко всему содержимому скобки, а не к одному символу.



*квантификатор внутри группы и после нее – это совершенно разные вещи!*

При наличии подгрупп в шаблоне можно ссылаться на них через их порядковый номер при помощи 1, 2, 3, ...

Группам можно давать собственные имена при помощи (?P<имя>...)

# Non-capturing groups и lookarounds

15

**Non-capturing группы** позволяют найти группу, но не включать ее в требуемый шаблон

**Позиционная проверка (lookaround)** – используется при поиске шаблона, которому обязательно предшествует или следует другой шаблон.

<b>?:</b>	Выделить группу, но не включать ее в результат
<b>?=</b>	Положительный lookahead (ищет шаблон, если он находится до указанной группы)
<b>?!</b>	Отрицательный lookahead (ищет шаблон, если он не находится до указанной группы)
<b>?&lt;=</b>	Положительный lookbehind (ищет шаблон, если он находится после указанной группы)
<b>?&lt;!</b>	Отрицательный lookbehind (ищет шаблон, если он не находится после указанной группы)

# Сервисы для проверки регулярных выражений

16

## REGULAR EXPRESSION

3 matches, 67 steps (~1ms)

/ регулярн[a-я]+ выражен[a-я]+ |

/ gm

## TEST STRING

SWITCH TO UNIT TESTS ►

Что такое регулярные выражения и как их использовать?

Говоря простым языком, регулярное выражение – это последовательность символов, используемая для поиска и замены текста в строке или файле. Как уже было упомянуто, их поддерживает множество языков общего назначения: Python, Perl, R. Так что изучение регулярных выражений рано или поздно пригодится.

<https://regex101.com/>



<https://docs.python.org/3/library/re.html> – документация по регулярным выражениям

<https://regex101.com/> – тестер регулярных выражений

<https://tproger.ru/translations/regular-expression-python/> – хорошая понятная статья по регулярным выражениям

# Регулярные выражения

## Вопросы?



Олег Булыгин

Соцсеть  
[fb.com/obulygin91](https://fb.com/obulygin91)

Почта  
[obulygin91@ya.ru](mailto:obulygin91@ya.ru)