нетология

РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

ЕЛЕНА НИКИТИНА

Елена Никитина

Руководитель проектов ГК «Геоскан»

Shorstko Shorstko Shorstko

План занятия

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

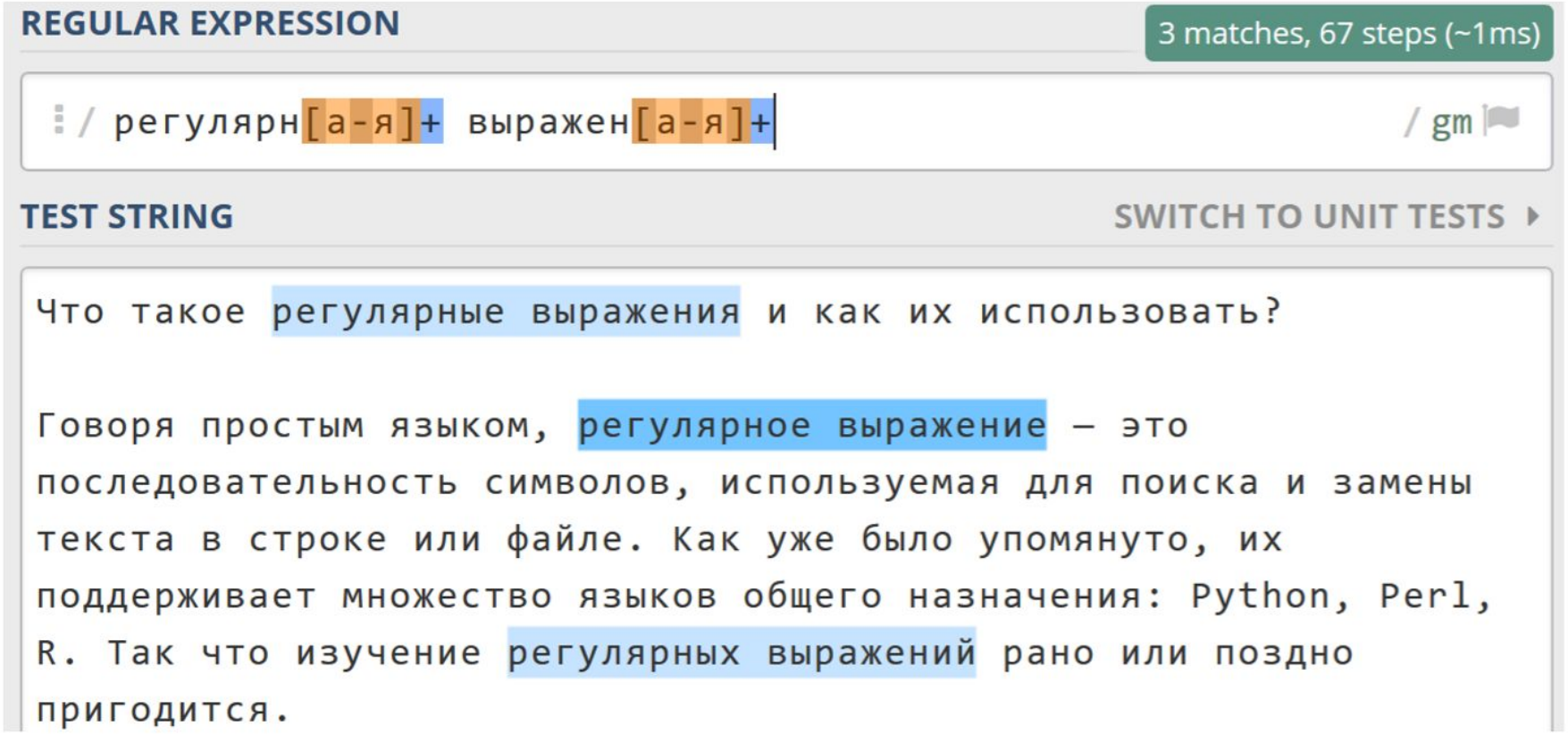
Регулярные выражения

Регулярные выражения (Regular Expressions, regex) – это простой язык «поисковых запросов» для поиска шаблонов в тексте.

Например, можно:

- найти и заменить любое количество пробелов на один,
- найти все e-mail'ы или телефоны в тексте, даже если они записаны по-разному,
- находить все слова и фразы независимо от окончаний (число, род, падеж...).

Как работают регулярные выражения



https://regex101.com/

Синтаксис. Спецсимволы

.	любой символ
^	1) начало строки, 2) инвертирование («всё, кроме»)
$	конец строки
*****	любое количество вхождений, от 0 до бесконечности
+	количество вхождений от 1 до бесконечности
?	0 или 1 вхождение {n} точное количество вхождений – n раз
{n, m}	количество вхождений не менее n и не более m раз
****	символ экранирования. Например, символ точки: \.

Синтаксис. Спецсимволы

[]	набор символов, любой из которых может встретиться в тексте. *Например, [а-яёА-ЯЁ] — любая буква русского алфавита в любом регистре*
\d	любая цифра. Аналогично [0-9]
\D	все, кроме цифры. Аналогично [^0-9]
\w	(для unicode) любая буква, цифра и символ подчеркивания. Для ASCII то же самое, но работает только для латинских букв
\W	все, кроме букв, цифр и символа подчеркивания
\s	любой пробельный символ, включая сам пробел: [\t\n\r\f\v]
\S	все, кроме пробельных символов
(...)	группировка символов (вместо **...** — текст или регулярное выражение)

Основные функции модуля re

- **re.match**(pattern, string, flags=0)

  Ищет по заданному шаблону в начале строки
- **re.search**(pattern, string, flags=0)

  Ищет во всем тексте, возвращает первое совпадение
- **re.findall**(pattern, string)

  Ищет во всем тексте, возвращает список всех найденных совпадений
- **re.compile**(pattern, flags=0)

  «Компилирует» регулярное выражение в объект. Если планируете использовать это регулярное выражение много раз
- **re.split**(pattern, string, maxsplit=0, flags=0)

  Разделяет строку по заданному шаблону
- **re.sub**(pattern, repl, string)

  Ищет шаблон в строке и заменяет его на указанную подстроку

re.match, re.search

```
 7   pattern = re.compile("регулярн[а-я]+ выражен[а-я]+")
 8   print("=== re.match ===")
 9   print(re.match(pattern, text))
10   result = re.match("Что", text)
11   print(result)
12   print(result.group(0))
13   print("First: {}, last: {}".format(result.start(), result.end()))
14   print("=== re.search ===")
15   print(re.search(pattern, text))
```

```
=== re.match ===
None
<_sre.SRE_Match object; span=(0, 3), match='Что'>
Что
First: 0, last: 3
=== re.search ===
<_sre.SRE_Match object; span=(10, 30), match='регулярные выражения'>
```

re.findall + re.compile

```
1   import re
2
3   pattern = re.compile("регулярн[а-я]+ выражен[а-я]+")
4   text = "Что такое регулярные выражения и как их использовать?\
5   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."
6
7   result = pattern.findall(text)
8   print(result)
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
['регулярные выражения', 'регулярное выражение', 'регулярных выраже
ний']
```

re.split

```
3   text = "Что такое регулярные выражения и как их использовать?\
4   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."
5
6   pattern = re.compile("регулярн[а-я]+ выражен[а-я]+")
7   print("=== re.split ===")
8   result = re.split("[.?]", text)
9   print(result)
10  print("Всего предложений: {}".format(len(result)))
```

```
=== re.split ===
['Что такое регулярные выражения и как их использовать', 'Говоря просты
м языком, регулярное выражение – это последовательность символов, испол
ьзуемая 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 приго
дится', '']
Всего предложений: 5
```

re.sub

```
3   text = "Что такое регулярные выражения и как их использовать?\
4   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."
5
6   pattern = re.compile("регулярн[а-я]+ выражен[а-я]+")
7   print("=== re.sub ===")
8   result = pattern.sub("RegEx", text)
9   print(result)
```

```
=== re.sub ===
Что такое RegEx и как их использовать?Говоря простым языком, RegEx – это пос
ледовательность символов, используемая для поиска и замены текста в строке и
ли файле. Как уже было упомянуто, их поддерживает множество языков общего на
значения: Python, Perl, R. Так что изучение RegEx рано или поздно пригодится
```

Флаги

re.A, re.ASCII	ASCII-диапазон символов вместо Юникода
re.U, re.UNICODE	Использование диапазонов Юникода. Работает по умолчанию, можно не назначать
re.I, re.IGNORECASE	Игнорировать регистр символов
re.M, re.MULTILINE	Разбивать текст на строки при обработке. Нужен, в основном, для функций re.match и re.search
re.S, re.DOTALL	По умолчанию символ точки означает любой символ, кроме символа новой строки \n. Если назначить этот флаг, ограничение снимается

Модификаторы

```
3   text = "Что такое регулярные выражения и как их использовать?\
4   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."
5
6   pattern = "[\w]+"
7   result = re.findall(pattern, text, re.U)
8   print(result)
```

```
['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', 'Per
l', 'R', 'Так', 'что', 'изучение', 'регулярных', 'выражений', 'рано', 'или', 'поздно', 'пригодит
ся']
```

Полезные ссылки

- Документация по регулярным выражениям: https://docs.python.org/3/library/re.html
- Тестер регулярных выражений: https://regex101.com/
- Хорошая понятная статья по регулярным выражениям: https://tproger.ru/translations/regular-expression-python/

Домашнее задание

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задаём в чате Slack!
- Задачи можно сдавать по частям.
- Зачёт по домашней работе проставляется после того, как приняты **все задачи**.

нетология

Задавайте вопросы и напишите отзыв о лекции!

ЕЛЕНА НИКИТИНА

 Shorstko

 Shorstko

 Shorstko