PYTHON

РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

REGEX

- Регулярные выражения - это поиск строк по шаблону, используемый во многих инструментах обработки текста, применяемых в системном администрировании
- Попросту говоря, это своего рода крошечный язык программирования, предоставляющий множество инструментов. Регулярные выражения поддерживаются большинством современных языков программирования, в которых представлены различные по удобству и функционалу средства

МОДУЛЬ RE

- В этой лекции мы узнаем о самых основных особенностях применения RegEx в языке Python
- Для работы с регулярными выражениями в Python предусмотрен специальный модуль re. Прежде чем начать работу его нужно импортировать:
- import re

МОДУЛЬ VS БИБЛИОТЕКА

- Модуль — это часть функциональности, существующая отдельно от кода, который ее использует, модуль импортируется в код и используется (код использующий модуль обычно называют клиентским).
- Модуль обычно узко сфокусирован и отвечает за одну узкую задачу, такие задачи могут быть как простыми, например, модуль конвертации градусов Цельсия в Фаренгейт, так и сложными, например модуль по работе с GUI операционной системы.
- Библиотека — это набор разных методов и модулей.
- Можно сказать, что модуль — это простой инструмент для конкретной задачи, а библиотека — швейцарский нож.

Спец. символ	Зачем нужен
.	Задает **один** произвольный символ
[ ]	Заменяет символ из квадратных скобок
-	Задает один символ, которого не должно быть в скобках
[^]	Задает один символ из **не** содержащихся в квадратных скобках
^	Обозначает начало последовательности
$	Обозначает окончание строки
*	Обозначает произвольное число повторений одного символа
?	Обозначает строго одно повторение символа
+	Обозначает один символ, который повторяется несколько раз
|	Логическое **ИЛИ**. Либо выражение до, либо выражение после символа
\	Экранирование. Для использования метасимволов в качестве обычных
( )	Группирует символы внутри
{ }	Указывается число повторений предыдущего символа

ДЛЯ ЧЕГО ИСПОЛЬЗУЮТСЯ РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

- Для определения нужного формата,
- Например телефонного номера или email-адреса;
- Для разбивки строк на подстроки;
- Для поиска,
- Замены
- Извлечения символов;
- Для быстрого выполнения нетривиальных операций
- Синтаксис таких выражений в основном стандартизирован, так что вам следует понять их лишь раз, чтобы использовать в любом языке программирования.

REGEX

- re.match()
- re.search()
- re.findall()
- re.split()
- re.sub()
- re.compile()
- И т.д

REGEX

- re.match(pattern, string)
- Функция re.match() осуществляет поиск по заданному шаблону с начала строки
- Искомая подстрока найдена. Чтобы вывести её содержимое, применим метод group() (мы используем «r» перед строкой шаблона, чтобы показать, что это «сырая» строка в Python):

```
result = re.match(r'First', 'First Second Third')
print(result.group(0))
# OK
result = re.match(r'Second', 'First Second Third')
print(result)
# None
```

```
<re.Match object; span=(0, 5), match='First'>
First
None
```

REGEX

- re.search(pattern, string)
- Функция re.search() используется для поиска в строке первого вхождения заданного шаблона
- Метод search() ищет по всей строке, но возвращает только первое найденное совпадение.

REGEX

- re.findall(pattern, string)
- Если нам нужно найти все вхождения, следует использовать функцию re.findall(). В случае успеха она возвращает список, который содержит все искомые вхождения по порядку. В противном случае функция возвратит пустой список.

```
result = re.findall(r'First', 'First Second Third, First Second Third')
print(result)
```

```
['First', 'First']
PS C:\Users\gunbo\OneDrive\Documents\Code\Python>
```

REGEX

- re.split(pattern, string, [maxsplit=0])
- Функция re.split() разделяет строку по заданному шаблону. Например, разобьем строку на отдельные слова, разделенные пробелом:
- Кроме того, эта функция принимает аргумент maxsplit со значением, по умолчанию равным 0. В нашем примере она разделит строку на максимальное количество частей. Если же специально задать этот аргумент, то разделение будет осуществлено не более заданного количества раз.

```python
result = re.split(r'i', 'First Second Third, First Second Third')
print(result)
temp = 'First Second Third, First Second Third'
x = re.split("\s", temp)
print(x)
x = re.split(r'i', temp,maxsplit=1)
print(x)
x = re.split("\s", temp,maxsplit=1)
print(x)
```

```
['F', 'rst Second Th', 'rd, F', 'rst Second Th', 'rd']
['First', 'Second', 'Third,', 'First', 'Second', 'Third']
['F', 'rst Second Third, First Second Third']
['First', 'Second Third, First Second Third']
PS C:\Users\gunbo\OneDrive\Documents\Code\Python>
```

REGEX

- re.sub(pattern, repl, string, count=0)
- Функция re.sub() осуществляет поиск шаблона в строке, заменяя его указанной подстрокой. В случае отсутствия шаблона никаких изменений не происходит. Для этих целей предназначена re.sub.
- В качестве четвертого параметра функции можно задать количество совпадений, подлежащих изменению. По умолчанию этот параметр равен 0, то есть заменяются все найденные совпадения с шаблоном.

```python
result = re.sub(r'New-York', 'London', 'New-York is a capital of UK')
print (result)

💡
result = re.sub(r'New-York', 'London', 'New-York is a capital of UK and New-York is heart of UK',count=1)
print (result)
```

```
London is a capital of UK
London is a capital of UK and New-York is heart of UK
PS C:\Users\gunbo\OneDrive\Documents\Code\Python>
```

Метасимволы

Метасимвол – это символ с указанным значением.

Метасимвол	Описание	Пример
[]	Представляет собой набор символов.	“[[a-z]”
\	Он представляет собой особую последовательность.	“\r”
.	Сигнализирует о том, что какой-либо символ присутствует в определенном месте.	“Ja.v.”
^	Он представляет собой образец, присутствующий в начале строки.	«^ Java»
$	Присутствует в конце строки.	“point”
*	Представляет собой ноль или более вхождений шаблона в строку.	“hello*”
+	Он представляет собой одно или несколько вхождений шаблона в строку.	“hello+”
{}	Указанное количество вхождений шаблона в строку.	“java{2}”
|	Он представляет собой присутствие того или иного символа.	“java|point”
()	Объединение и группировка.	

Оператор	Описание
`.`	Один любой символ, кроме новой строки `\n`.
`?`	0 или 1 вхождение шаблона слева
`+`	1 и более вхождений шаблона слева
`*`	0 и более вхождений шаблона слева
`\w`	Любая цифра или буква (`\W` — все, кроме буквы или цифры)
`\d`	Любая цифра [0-9] (`\D` — все, кроме цифры)
`\s`	Любой пробельный символ (`\S` — любой непробельный символ)
`\b`	Граница слова
`[..]`	Один из символов в скобках (`[^..]` — любой символ, кроме тех, что в скобках)
`\`	Экранирование специальных символов (`\.` означает точку или `\+` — знак «плюс»)
`^` и `$`	Начало и конец строки соответственно
`{n,m}`	От `n` до `m` вхождений (`{,m}` — от 0 до `m`)
`a|b`	Соответствует `a` или `b`
`()`	Группирует выражение и возвращает найденный текст
`\t, \n, \r`	Символ табуляции, новой строки и возврата каретки соответственно

ЗАДАЧИ

- Вернуть каждый символ из строки используя “.” (точку)
- Вернуть каждый символ из строки без пробела
- Вернуть каждое слово с пробелом
- Вернуть каждое слово без пробела
- Вернуть первое слово
- Вернуть последнее слово

ЗАДАЧИ

- Вернуть каждые 2 последующих символа без пробела
- Вернуть два последовательных символа, используя символ границы слова (\b)
- Вернуть все символы после знака “@”
- Вернуть полную строку после @
- Вернуть только доменное имя
- Вернуть дату из строки

ДОПОЛНИТЕЛЬНЫЕ ССЫЛКИ

- https://docs.python.org/3/library/re.html