Регулярные выражения

В дальнейшем мы будем часто использовать текстовые данные. Для обработки таких данных используются регулярные выражения.

Регулярные выражения – это компактная форма записи шаблона других строк. Регулярные выражения представлены небольшим языком, встроенным в Python и имеющим свой собственный синтаксис. Мощь регулярных выражений заключается в том, что они могут представлять шаблон неограниченного набора строк.

Регулярные выражения нужны для четырех конкретных целей:

1. Проверка: проверка данных на соответствие определённому шаблону;
2. Поиск: поиск подстрок в строке;
3. Поиск и замена: замена всего, что попадает под шаблон на указанную строку
4. Разбиение строк: разбиение строки по точкам совпадения с регулярным выражением.

Самое простое регулярное выражение – это обычный литерал, например «ё». Рассмотрим, как это работает.

Предположим, нам необходимо заменить все буквы «ё» в строке, на букву «е». Конечно, мы можем это сделать с помощью обычного for:

string = 'ёжик нёс яблоки под ёлку'  
print(string)  
string = list(string)  
  
for i in range(len(string)):  
 if string[i] == 'ё':  
 string [i] = 'е'  
  
string = ''.join(string)  
print(string)

Вывод на консоль:

ёжик нёс яблоки под ёлку

ежик нес яблоки под елку

А вот как это выглядит с использованием регулярных выражений:

import re  
  
string = 'ёжик нёс яблоки под ёлку'  
print(string)  
string = re.sub('ё', 'е', string)  
print(string)

Регулярные выражения позволяют описать «шаблон» строки, что позволяет сделать код более «читабельным» и упрощают работу в целом.

Итак, для того чтобы использовать регулярные выражения, необходимо подключить модуль re.

import re

re – это библиотека для работы с регулярными выражениями в Python.

Мы использовали метод sub(pattern, replace, string) – метод, предоставляемый библиотекой re, который принимает шаблон, который необходимо найти, подстроку, на которую будет заменять шаблон и саму строку, в которой будет производиться замена, и возвращает новую строку, в которой произведена замена.

Наиболее часто используемые методы из библиотеки re:

* re.search;
* re.findall;
* re.sub;
* re.split;

re.search(pattern, string) – метод, позволяющий найти первое вхождения шаблона в строку:

import re  
  
string = 'мама мыла раму'  
result = re.search('а', string)  
print(result.start(), result.end(), result.group())

Вывод на консоль:

1 2 a

Чтобы вывести содержимое результата, необходимо использовать метод group(). start() и end() показывают начальную и конечную позицию найденного шаблона.

re.findall(pattern, string) – метод, возвращающий список всех найденных шаблонов.

import re  
  
string = 'мама мыла раму'  
result = re.findall('а', string)  
print(result)

Вывод на консоль:

['а', 'а', 'а', 'а']

re.sub(pattern, string) – метод, разбивающий строку на подстроки по заданному шаблону.

import re  
  
string = 'мама мыла раму'  
result = re.split('а', string)  
print(result)

Вывод на консоль:

['м', 'м', ' мыл', ' р', 'му']

Часто вместо совпадения с одним символом необходимо отыскать совпадение с шаблоном, содержащим множество символов. Рассмотрим следующий пример:

import re  
  
string = '- Полей-ка цветы! В который раз прошу?! - грозно сказала мама.'  
print(string)  
string = re.sub(r'[-,.?!]+(\s|$)', ' ', string).strip()  
print(string)  
words = re.split(r'\s+', string)  
print(words)

Вывод на консоль:

- Полей-ка цветы! В который раз прошу?! - грозно сказала мама.

Полей-ка цветы В который раз прошу грозно сказала мама

['Полей-ка', 'цветы', 'В', 'который', 'раз', 'прошу', 'грозно', 'сказала', 'мама']

В этом примере мы хотим убрать из текста всю пунктуацию, а затем разбить его на отдельные слова. Для начала нужно разобраться, что за строки с символом r мы используем.

Строки, слева от которых стоит символ r – это так называемые «грязные» (raw) строки. При рассмотрении строк мы говорили про escape-последовательности, когда набор символов несет определенное значение. Однако, если мы хотим, чтобы, например, строка «\n» читалась как два отдельных символа, необходимо экранировать символ «\», поставив перед ним еще один «\».

print('Символ "\n" - это символ перевода строки.')

Вывод на консоль:

Символ "

" - это символ перевода строки.

print('Символ "\\n" - это символ перевода строки.')

Вывод на консоль:

Символ "\n" - это символ перевода строки.

Однако, это может затруднять понимание кода. Для этого к нам на помощь приходят «грязные» строки. Они позволяют пропускать все специальные символы, оставляя их в текстовом виде.

print(r'Символ "\n" - это символ перевода строки.')

Вывод на консоль:

символ "\n" - это символ перевода строки.

Итак, мы разобрались с «грязными» строками, теперь необходимо разобраться в значении строки r'[-,.?!]+(\s|$)'. Это наше регулярное выражение. Регулярное выражение может состоять как из обычных символов (алфавит, цифры, пунктуация и др.), так и из спец символов, помогающих делать регулярное выражение короче и понятнее.

Наиболее часто используемые символы:

|  |  |  |  |
| --- | --- | --- | --- |
| **Символ** | **Описание** | **Шаблон** | **Соответствие** |
| **Символы** | | | |
| […] | Один из символов в скобках (можно указать диапазон символов, например, [а-г] – то же самое, что и [абвг]) | [абв] | «а» в «чай» |
| [^…] | Любой символ, кроме тех, что в скобках | [^abc] | «ч», «й» в «чай» |
| \d | Любая цифра (то же что и [0-9]) | \d | «1» в «1-ый» |
| \w | Буква, цифра, или знак подчеркивания (то же что и [а-яёА-ЯЁa-zA-Z0-9\_]) | \w | «2», «5», «м», «с» в «2.5 м/с» |
| \s | Пробельный символ (пробел, табуляция, перенос строки и др.) | \s | « » в «а он?» |
| . | Один любой символ, кроме перевода строки | м. | «мы» в «мыла» |
| ^ | Начало строки | ^а. | «аб» в «аба» |
| $ | Конец строки | .а$ | «ба» в «аба» |
| **Символы повторения** | | | |
| ? | 0 или 1 вхождение шаблона слева | м?а | «ма», «ма», «а» в «маммаа» |
| + | 1 или более вхождений шаблона слева | м+а | «ма», «мма» в «маммаа» |
| \* | 0 или более вхождений шаблона слева | м\*а | «ма», «мма», «а» в «маммаа» |
| **Символы выбора** | | | |
| | | Соответствует либо левому, либо правому шаблону | аб|ба | «аб», «ба» в «абба» |
| (…) | Группирует выражение в скобках | (ма)+ | «мама» в «мама» |

Таким образом в r'[-,.?!]+(\s|$)' мы ищем символы «-», «,», «.», «?» и «!», которые входят в строку 1 или более раз (+), после которых либо стоит пробел (\s), либо конец строки($). Затем мы заменяем эти символы на пробелы, обрезаем пробелы справа и слева, и разбиваем строку по пробелу.