Регулярные выражения

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

Регулярные выражения – это небольшой язык встроенный в Python, который позволяет производить поиск подстроки в строке, разбивать строку на подстроки или заменять часть строки.

Например, мы хотим заменить все буквы «ё» в строке, на букву «е». Конечно, мы можем это сделать с помощью 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)

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

import re

re – это библиотека для работы с регулярными выражениями в Python. sub(pattern, replace, string) – метод, предоставляемый библиотекой, 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']

Порой, требуется что-то сложнее, чем работа с подстрокой. 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. Это так называемые «грязные» (raw) строки.

При рассмотрении строк мы говорили про escape-последовательности, набор символов в которой несет определенное значение. Однако, если мы хотим, чтобы, например, строка «\n» читалась как два отдельных символа, необходимо экранировать слеш, 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.