Модуль Re (Regex)

Регулярные выражения

|  |  |
| --- | --- |
| **Import re** | <https://habr.com/ru/post/349860/>  <https://www.regexpal.com/> |
| **re.match(*pattern*, *string*)**  result = re.match(r'AV', 'AV Analytics Vidhya AV')  **print result.group(0)**  Результат:  AV  result = re.match(r'AV', 'AV Analytics Vidhya AV')  print result.start()  print result.end()  Результат:  0  2 | Ищет по заданному шаблону в начале строки. Например, если мы вызовем метод match() на строке «AV Analytics AV» с шаблоном «AV», то он завершится успешно. Однако если мы будем искать «Analytics», то результат будет отрицательный.  Result.group(0) – выводет найденный результат результат |
| **re.search(*pattern*, *string*)** | ищет не только в начале строки. В отличие от предыдущего, search() вернет объект, если мы попытаемся найти «Analytics». Метод search() ищет по всей строке, но возвращает только первое найденное совпадение. |
| **re.findall(*pattern*, *string*)**  result = re.findall(r'AV', 'AV Analytics Vidhya AV')  print result  Результат:  ['AV', 'AV'] | Этот метод возвращает список всех найденных совпадений. У метода findall() нет ограничений на поиск в начале или конце строки. Если мы будем искать «AV» в нашей строке, он вернет все вхождения «AV». Для поиска рекомендуется использовать именно findall(), так как он может работать и как re.search(), и как re.match(). |
| **re.split(*pattern*, *string*, *[maxsplit=0]*)**  result = re.split(r'y', 'Analytics')  print result  Результат:  ['Anal', 'tics']  result = re.split(r'i', 'Analytics Vidhya')  print result  Результат:  ['Analyt', 'cs V', 'dhya'] # все возможные участки.  result = re.split(r'i', 'Analytics Vidhya',maxsplit=1)  print result  Результат:  ['Analyt', 'cs Vidhya'] | разделяет строку по заданному шаблону. В примере мы разделили слово «Analytics» по букве «y». Метод split() принимает также аргумент maxsplit со значением по умолчанию, равным 0. В данном случае он разделит строку столько раз, сколько возможно, но если указать этот аргумент, то разделение будет произведено не более указанного количества раз. |
| **re.sub(*pattern*, *repl*, *string*)**  result = re.sub(r'India', 'the World', 'AV is largest Analytics community of India')  print result  Результат:  'AV is largest Analytics community of the World' | Этот метод ищет шаблон в строке и заменяет его на указанную подстроку. Если шаблон не найден, строка остается неизменной. |
| **re.compile(*pattern*, *repl*, *string*):**  pattern = re.compile('AV')  result = pattern.findall('AV Analytics Vidhya AV')  print result  result2 = pattern.findall('AV is largest analytics community of India')  print result2  Результат:  ['AV', 'AV']  ['AV'] | можем собрать регулярное выражение в отдельный объект, который может быть использован для поиска. Это также избавляет от переписывания одного и того же выражения..  Задает шаблон регулярного выражения в переменную.  shablon= re.compile(r"380\d\d\d\d\d\d\d\d\d")  shablon1= re.compile(r"\d\d\d\d\d\d\d\d\d\d\d\d")  shablon2= re.compile(r"(\d\d)(\d\d\d)(\d\d\d)(\d\d)(\d\d)")  Можно комбинировать со знаком или |. Искать будет Batmobile, Batman и т.д.  shablon= re.compile(r"Bat(man|mobile|copter")  Не обязательные символы шаблона  shablon= re.compile(r"Bat(wo)?man ")  Ищет Batman и Batwoman  Не обязательные элементы  (\d\d\d)?\d\d\d\d\d\d\d\d\d |
| |  |  | | --- | --- | | **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 | | если у нас нет определенного шаблона, и нам надо вернуть набор символов из строки, отвечающий определенным правилам? Такая задача часто стоит при извлечении информации из строк. Это можно сделать, написав выражение с использованием специальных символов. Вот наиболее часто используемые из них: |
| \(\d\d\d \d\d\) | Скобки должны экранироватся в регулярных выражениях. Если скобки не экранируются то выборка шаблона будет происходить по группам указанным в скобках |
| (\d\d\d)? | Выражение не обязательно |
| (wo)\*  Re.compile(r“Bat(wo)\*man”) | Ищет строку с 0 и больше вхождениями символов из скобки. Тоесть оно не обязательно |
| (wo)+  Re.compile(r“Bat(wo)+man”) | Обязательное вхождение символов из скобки от 1 и больше раз. |
| Re.compile(r“(He){3,5}”)  HeHeHe  HeHeHeHe  HeHeHeHeHe | Ищет повторяющиеся символы от 3 до 5 раз.  В данной редакции шаблон будет выбирать максимально длинные строки.  Тоесть если ему скормить строку HeHeHeHeHeHeHeHeHeHe то Пайтон вернет HeHeHeHeHe |
| Re.compile(r“(He){3,5}?”) | Ищет повторяющиеся символы от 3 до 5 раз.  В данной редакции шаблон будет выбирать максимально короткие строки.  Тоесть если ему скормить строку HeHeHeHeHe то Пайтон вернет HeHeHe |
| ? | Вопросительный знак в регулярных выражениях может использоваться двояко.   1. Указывать на не жадный поиск(ищет самые короткие допустимые строки) 2. Может указывать на не обязательную группу |
| \d+\s\w+ | Будет искать несколько цифр подряд, пробел и следующие за ним слово или Любая буква (то, что может быть частью слова), а также цифры и \_ |
| Re.compile(r“[аоэуеяиюАОЭЮИЯУЕ]”)  Re.compile(r“[A-Za-z0-9]”) | Ищет любые символы в строке указанные в кортеже.  Ищет символы из указаного диапазона |
| Re.compile(r“[^аоэуеяиюАОЭЮИЯУЕ]”)  Re.compile(r“[^A-Za-z0-9]”) | Ищет любые символы в строке НЕ указанные в кортеже.  Ищет символы НЕ из указаного диапазона |
| Re.compile(r“^Hello”)  Re.compile(r“Hello$”)  Re.compile(r“^Hello$”) | Строка должна начинатся с Hello  Строка должна заканчиваться шаблоном  Строка должна соответствовать шаблону |
| Re.compile(r“.има”) | . – Любой одиночный символ. Дима,Зима, Тима,Фима |
| Re.compile(r“Имя: (.\*) Фамилия: (.\*)”)  Re.compile(r“Имя: (.\*?) Фамилия: (.\*?)”) | Выберет две группы символов между словами имя и фамилия и после фамилия. Режим жадного поиска.  Режим не жадного поиска, ищет минимум текста |
| Re.compile(“ .\*”)  Re.compile(“ .\*”, re.DOTALL) | Парсит все символы до символа переноса новой строки.  Парсит все символы, включая перенос строки |
| Re.compile(“РоБокОп”, re.IGNORECASE) | Парсит игнорируя регистр |
| Длинная строка в несколько строчек регулярных выражений  re.compile(r"""  (\+38(\)|\(|\s|-)?|38(\)|\(|\s|-)?)?  (\d){3}  (\)|\(|\s|-)?  \d  (\)|\(|\s|-)?  \d  (\)|\(|\s|-)?  \d  (\)|\(|\s|-)?  \d  (\)|\(|\s|-)?  \d  (\)|\(|\s|-)?  \d  (\)|\(|\s|-)?  \d""", re.VERBOSE) | Многострочный режим |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **Функция** | **Её смысл** |
| re.search(pattern, string) | Найти в строке string первую строчку, подходящую под шаблон pattern; |
| re.fullmatch(pattern, string) | Проверить, подходит ли строка string под шаблон pattern; |
| re.split(pattern, string, maxsplit=0) | Аналог str.split(), только разделение происходит по подстрокам, подходящим под шаблон pattern; |
| re.findall(pattern, string) | Найти в строке string все непересекающиеся шаблоны pattern; |
| re.finditer(pattern, string) | Итератор всем непересекающимся шаблонам pattern в строке string(выдаются match-объекты); |
| re.sub(pattern, repl, string, count=0) | Заменить в строке string все непересекающиеся шаблоны pattern на repl; |

| **Константа** | **Её смысл** |
| --- | --- |
| re.ASCII | По умолчанию \w, \W, \b, \B, \d, \D, \s, \S соответствуют  все юникодные символы с соответствующим качеством.  Например, \d соответствуют не только арабские цифры,  но и вот такие: ٠١٢٣٤٥٦٧٨٩.  re.ASCII ускоряет работу,  если все соответствия лежат внутри ASCII. |
| re.IGNORECASE | Не различать заглавные и маленькие буквы.  Работает медленнее, но иногда удобно |
| re.MULTILINE | Специальные символы ^ и $ соответствуют  началу и концу каждой строки |
| re.DOTALL | По умолчанию символ \n конца строки не подходит под точку.  С этим флагом точка — вообще любой символ |

Примеры регулярных выражений

| **Регулярка** | **Её смысл** |
| --- | --- |
| simple text | В точности текст «simple text» |
| \d{5} | Последовательности из 5 цифр \d означает любую цифру {5} — ровно 5 раз |
| \d\d/\d\d/\d{4} | Даты в формате ДД/ММ/ГГГГ (и прочие куски, на них похожие, например, 98/76/5432) |
| \b\w{3}\b | Слова в точности из трёх букв \b означает границу слова (с одной стороны буква, а с другой — нет) \w — любая буква, {3} — ровно три раза |
| [-+]?\d+ | Целое число, например, 7, +17, -42, 0013 (возможны ведущие нули) [-+]? — либо -, либо +, либо пусто \d+ — последовательность из 1 или более цифр |
| [-+]?(?:\d+(?:\.\d\*)?|\.\d+)(?:[eE][-+]?\d+)? | Действительное число, возможно в экспоненциальной записи Например, 0.2, +5.45, -.4, 6e23, -3.17E-14. См. ниже картинку. |

| **Шаблон** | **Описание** | **Пример** | **Применяем к тексту** |
| --- | --- | --- | --- |
| . | Один любой символ, кроме новой строки \n. | м.л.ко | молоко, малако,  Им0л0коИхлеб |
| \d | Любая цифра | СУ\d\d | СУ35, СУ111, АЛСУ14 |
| \D | Любой символ, кроме цифры | 926\D123 | 926)123, 1926-1234 |
| \s | Любой пробельный символ (пробел, табуляция, конец строки и т.п.) | бор\sода | бор ода, бор ода, борода |
| \S | Любой непробельный символ | \S123 | X123, я123, !123456, 1 + 123456 |
| \w | Любая буква (то, что может быть частью слова), а также цифры и \_ | \w\w\w | Год, f\_3, qwert |
| \W | Любая не-буква, не-цифра и не подчёркивание | сом\W | сом!, сом? |
| [..] | Один из символов в скобках, а также любой символ из диапазона a-b | [0-9][0-9A-Fa-f] | 12, 1F, 4B |
| [^..] | Любой символ, кроме перечисленных | <[^>]> | <1>, <a>, <>> |
| \d≈[0-9],  \D≈[^0-9],  \w≈[0-9a-zA-Z  а-яА-ЯёЁ],  \s≈[ \f\n\r\t\v] | Буква “ё” не включается в общий диапазон букв! Вообще говоря, в \d включается всё, что в юникоде помечено как «цифра», а в \w — как буква. Ещё много всего! |  |  |
| [abc-], [-1] | если нужен минус, его нужно указать последним или первым |  |  |
| [\*[(+\\\]\t] | внутри скобок нужно экранировать только ] и \ |  |  |
| \b | Начало или конец слова (слева пусто или не-буква, справа буква и наоборот). В отличие от предыдущих соответствует позиции, а не символу | \bвал | вал, перевал, Перевалка |
| \B | Не граница слова: либо и слева, и справа буквы, либо и слева, и справа НЕ буквы | \Bвал | перевал, вал, Перевалка |
|  |  | \Bвал\B | перевал, вал, Перевалка |

Квантификаторы (указание количества повторений)

| **Шаблон** | **Описание** | **Пример** | **Применяем к тексту** |
| --- | --- | --- | --- |
| {n} | Ровно n повторений | \d{4} | 1, 12, 123, 1234, 12345 |
| {m,n} | От m до n повторений включительно | \d{2,4} | 1, 12, 123, 1234, 12345 |
| {m,} | Не менее m повторений | \d{3,} | 1, 12, 123, 1234, 12345 |
| {,n} | Не более n повторений | \d{,2} | 1, 12, 123 |
| ? | Ноль или одно вхождение, синоним {0,1} | валы? | вал, валы, валов |
| \* | Ноль или более, синоним {0,} | СУ\d\* | СУ, СУ1, СУ12, ... |
| + | Одно или более, синоним {1,} | a\)+ | a), a)), a))), ba)]) |
| \*? +? ?? {m,n}? {,n}? {m,}? | По умолчанию квантификаторы *жадные* —  захватывают максимально возможное число символов.  Добавление ? делает их *ленивыми*,  они захватывают минимально возможное число символов | \(.\*\) \(.\*?\) | (a + b) \* (c + d) \* (e + f) (a + b) \* (c + d) \* (e + f) |