?????????? ??????????

Метасимволы

^	начало
$	конец
[]	список символов [1-6] перечисление, может комбинироваться порядок роли не играет [0-5a-fkl] [- если дефис в начале, то как символ ^ Если в начале, то инверсия правила. Если не в начале, то обычный символ
|	или. Желательно экранировать остальное скобками, например gr(e|a)y внутри могут быть полноценные регулярные выражения
.	Один любой символ, кроме новой строки \n.
?	0 или 1 вхождение предшествующего элемента
+	1 и более вхождений предшествующего элемента
*	0 и более вхождений предшествующего элемента
\w	Любая цифра или буква
\W	все, кроме буквы или цифры
\d	Любая цифра [0-9]
\D	все, кроме цифры
\xFF	шестнадцатеричное число
\s	Любой пробельный символ
\S	любой непробельный символ
\b	Граница слова
\A	начало текста
\t, \n, \r	Символ табуляции, новой строки и возврата каретки соответственно
{n,m}	От n до m вхождений {n} ровно n вхождений {n,} от n вхождений {,m} — от 0 до m Иногда нужно экранировать \{

Модификаторы

(?i:что-то)	интервальный модификатор, отключающий поиск с учетом регистра для "что-то"
(?:что-то)	Атомарная группировка проходит как обычно, но когда процесс поиска выходит за пределы конструкции (за закрывающую круглую скобку), все сохраненные состояния удаляются.
(?P<word>...)	Именованная группа, в данном случае имя - word m.group('word') - пример обращения к группе

Опережающая проверка

(?<=...)	Должно совпасть слева (Позитивная ретроспективная проверка), в Python - fixed length.
(?<!...)	Не должно совпасть слева (Негативная ретроспективная проверка), в Python не поддерживается.
(?=...)	Должно совпасть справа (Позитивная опережающая проверка).
(?!...)	Не должно совпасть справа (Негативная опережающая проверка).
\g<...>	группы при обратных ссылках, g<1> группа 1

.*? "ленивый" поиск
Атрибуты
\p{атрибут}
\p{L} \p{Letter} – символы, считающиеся буквами
\p{M} \p{Mark} – различные символы, существующие не самостоятельно, а лишь в сочетании с другими базовыми символами (диакритические знаки, рамки и т. д.)
\p{Z} \p{Separator} – символы, выполняющие функции разделителей, но не имеющие собственного визуального представления (разнообразные пробелы и т. д.)
\p{S} \p{Symbol} – различные декоративные элементы и знаки
\p{N} \p{Number} – цифры
\p{P} \p{Punctuation} – знаки препинания
\p{C} \p{Other} – прочие символы (редко используется при работе с обычным текстом)
Похоже, что python не поддерживает

Python

Флаги при обработке
re.compile('...', re.IGNORECASE) - игнор регистра
re.VERBOSE re.X Allow inline comments and extra whitespace.
re.IGNORECASE re.I Do case-insensitive matches.
re.MULTILINE re.M Allow anchors (^ and $) to match the beginnings and ends of lines instead of matching the beginning and end of the whole text.
re.DOTALL re.S Allow dot (.) to match any character, including a newline. (The default behavior of dot is to match anything, except for a newline.)

Нюансы python
В возвращаемом кортеже группа 0 - максимальная группа. Группа определяется скобками. Поэтому если внутри есть () для формирования регулярки, то необходимо всю регулярку брать в скобки.
Вместо r'gr(e|a)y' можно использовать f-строки, {} будет означать переменную
Пример:

```
m = re.finditer(r'gr(e|a)y', 'gray grey')
    for ma in m:
        print(ma.groups())
#('a',)
#('e',)
m = re.finditer(r'(gr(e|a)y)', 'gray grey')
    for ma in m:
        print(ma.groups())
#('gray', 'a')
#('grey', 'e')
```

По умолчанию символы повторения жадные (greedy). Если нужно отключить жадность, достаточно добавить знак вопроса после символов повторения: match = re.search(r'<.*?>', line)

re.sub(pattern, repl, string, count=0) Заменить в строке string все непересекающиеся шаблоны pattern на repl

---

Revision #3
Created 29 August 2025 03:21:38 by Admin
Updated 29 August 2025 14:29:40 by Admin