

Подсказка по регулярным выражениям

Шпаргалка с кратким содержанием:

.	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

Описание шаблонов

В столбце “Применение выражения к тексту” соответствия регулярному выражению выделяются бирюзовым цветом с подчёркиванием.

Источник (и бонусом - отличная статья по RegEx):

<https://habr.com/ru/post/349860/#Regulyarki>

Шаблон	Описание	Пример выражения	Применение выражения к тексту
.	Один любой символ, кроме новой строки \n.	м.л.ко	<u>молоко</u> , <u>малако</u> , И <u>м0л0ко</u> Ихлеб
\d	Любая цифра	су\d\d	<u>СУ35</u> , <u>СУ111</u> , АЛ <u>СУ14</u>
\D	Любой символ, кроме цифры	926\D123	<u>926)123</u> , <u>1926-1234</u>

\s	Любой пробельный символ (пробел, табуляция, конец строки и т.п.)	бор\sода	<u>бор ода</u> , <u>бор ода</u> , борода
\S	Любой непробельный символ	\S123	<u>X123</u> , <u>я123</u> , <u>!123</u> 456, 1 + 123456
\w	Любая буква (то, что может быть частью слова), а также цифры и _	\w\w\w	<u>Год</u> , <u>f_3</u> , <u>qwert</u>
\W	Любая не-буква, не-цифра и не подчёркивание	com\W	<u>com!</u> , <u>com?</u>
[...]	Один из символов в скобках, а также любой символ из диапазона a-b	[0-9][0-9A-Za-f]	<u>12</u> , <u>1E</u> , <u>4B</u>
[^...]	Любой символ, кроме перечисленных	<[^>]>	<u><1></u> , <u><a></u> , <>>
[abc-], [-1]	если нужен минус, его нужно указать последним или первым		
\b	Начало или конец слова (слева пусто или не-буква, справа буква и наоборот). В отличие от предыдущих соответствует позиции, а не символу	\bвал	<u>вал</u> , перевал, Перевалка
\B	Не граница слова: либо и слева, и справа буквы, либо и слева, и справа НЕ буквы	\Bвал	пере <u>вал</u> , вал, Перев <u>вал</u> ка

		<code>\Ввал\В</code>	перевал, вал, Пере <u>вал</u> ка
<code>*?</code> <code>+?</code> <code>??</code> <code>{m,n}?</code> <code>{,n}?</code> <code>{m,}?</code>	По умолчанию квантификаторы <i>жадные</i> — захватывают максимально возможное число символов. Добавление <code>?</code> делает их <i>ленивыми</i> , они захватывают минимально возможное число символов	<code>\ (. * \)</code> <code>\ (. * ? \)</code>	<u><code>(a + b) * (c + d) * (e + f)</code></u> <u><code>(a + b)</code></u> * (c + d) * (e + f)

Квантификаторы (указание количества повторений)

Шаблон	Описание	Пример выражения	Применение выражения к тексту
<code>{n}</code>	Ровно n повторений	<code>\d{4}</code>	1, 12, 123, <u>1234</u> , 12345
<code>{m,n}</code>	От m до n повторений включительно	<code>\d{2,4}</code>	1, <u>12</u> , <u>123</u> , <u>1234</u> , 12345
<code>{m,}</code>	Не менее m повторений	<code>\d{3,}</code>	1, 12, <u>123</u> , <u>1234</u> , <u>12345</u>
<code>{,n}</code>	Не более n повторений	<code>\d{,2}</code>	<u>1</u> , <u>12</u> , <u>123</u>
<code>?</code>	Ноль или одно вхождение, синоним <code>{0,1}</code>	валы?	<u>вал</u> , <u>валы</u> , <u>валов</u>
<code>*</code>	Ноль или более, синоним <code>{0,}</code>	<code>су\d*</code>	<u>су</u> , <u>су1</u> , <u>су12</u> , ...
<code>+</code>	Одно или более, синоним <code>{1,}</code>	<code>a \) +</code>	<u>a</u>), <u>a</u>)), <u>a</u>))), b <u>a</u>)]

Группы в регулярном выражении

При решении различных задач также часто используются круглые скобки (. . .)
Их значение заключается в двух функциях:

1) Эти скобки призваны сократить повторяющиеся группы внутри шаблонов.

Пример: MAC-адрес сетевого устройства обычно записывается как шесть групп из двух шестнадцатеричных цифр, разделённых символами '-' или ':'

01:23:45:67:89:ab

Без применения скобочных групп шаблон будет выглядеть так:

```
pattern = r'[0-9a-fA-F]{2}[:-][0-9a-fA-F]{2}[:-]' \
          r'[0-9a-fA-F]{2}[:-][0-9a-fA-F]{2}[:-]' \
          r'[0-9a-fA-F]{2}[:-][0-9a-fA-F]{2}'
```

Сгруппировав повторяющиеся части, можно с помощью квантификаторов задать количество их повторов:

```
pattern = r'[0-9a-fA-F]{2}(?:[:-][0-9a-fA-F]{2}){5}'
```

Что даже на таком простом примере позволило значительно сократить размер регулярного выражения.

Ещё одна сильная сторона подобных группировок в том, что теперь мы можем писать шаблон даже не зная точного количества групп, ведь в квантификаторе можно указать не только точное число, **но и отрезок**, на котором оно должно находиться

2) Используя скобки (. . .) с функциями **re.search()**, **re.fullmatch()** и **re.finditer()** в возвращенных match-объектах мы сможем получить доступ к информации по каждой группе, выделенной скобками, отдельно (часть подстроки, которая совпала этой группой и индексы)

Пример:

```
pattern = r'\s*([А-Яа-яЁё]+)(\d+)\s*'
string = r'--- 0пять45 ---'
matched = re.search(pattern, string)

print('Совпадение вернулось объектом', matched) # re.Match object
```

match[0] в обычном случае работает так же, как и **match.group()**

Но теперь с помощью **match[1]** и **match[2]** мы можем вернуть подстроки, совпадающие с первой группой - **([А-Яа-яЁё]+)** - это 'Опять' и со второй **(\d+)** - это '45'

```
print(f'Найдена подстрока >{match[0]}< с позиции {match.start(0)} до {match.end(0)}')
#           Найдена подстрока >   Опять45   < с позиции 3 до 16
print(f'Группа букв >{match[1]}< с позиции {match.start(1)} до {match.end(1)}')
#           Группа букв >Опять< с позиции 6 до 11
print(f'Группа цифр >{match[2]}< с позиции {match.start(2)} до {match.end(2)}')
#           Группа цифр >45< с позиции 11 до 13
```

Более подробно про группы в регулярных выражениях можно почитать в вышеупомянутой статье, в разделе “Группирующие скобки”:

https://habr.com/ru/post/349860/#Gruppiruyuschie_skobki_

Полезные ссылки

Официальная документация:

<https://docs.python.org/3/library/re.html>

Также есть отличный сайт для дебага и проверки регулярных выражений, обязательно попробуйте:

<https://regex101.com/>