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 RegExp):
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`	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>, <>
`[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`	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

`*?`
`+?`
`??`
`{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)

Квантификаторы (указание количества повторений)

Шаблон	Описание	Пример выражения	Применение выражения 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)])

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 находится

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