Регулярные выражения

1. Язык регулярных выражений в Python
2. Модуль для работы с регулярными выражениями

Литература

1. Марк Саммерфилд. Программирование на Python 3. (С. 525-552)
2. https://docs.python.org/3/library/re.html
3. http://www.rexegg.com/
4. https://habr.com/ru/post/349860/

1. **Язык регулярных выражений в Python**
2. Модуль для работы с регулярными выражениями

Проблема!

- Тип Строки позволяет выполнять над строками различные *операции*, в том числе *поиск, замену, вставку и удаление подстрок*.
- Но есть классы задач по обработке символьной информации, где стандартных возможностей явно не хватает: проверить текст на соответствие предопределенному шаблону (например, адресу электронной почты), поиск по шаблону *pet*. ???* и др.

Регулярное выражение – что это?

- Регулярное выражение - это компактная форма записи набора строк, отвечающих требованиям регулярного выражения.
- Регулярное выражение - это текстовая строка, которая описывает шаблон, который используется механизмом регулярных выражений для поиска фрагмента текста (или позиций) в исходном тексте, как правило, для проверки, поиска, замены или разделения.
- В состав Python входит модуль ***re***, с помощью которого можно создавать и использовать регулярные выражения.

Регулярное выражение	Значение
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 фрагменты, 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 +,

Регулярное выражение – для чего?

- *Проверка:* проверка соответствия фрагментов текста некоторым критериям, например, наличие символа обозначения валюты и последующих за ним цифр.

- *Поиск:* поиск подстрок, которые могут иметь несколько форм,

например, поиск подстрок «pet.png», «pet.jpg», «pet.jpeg» или «pet.svg», чтобы при этом не обнаруживались подстроки «carpet. png» и подобные ей.

Регулярное выражение – для чего?

- *Поиск и замена:* замена всего, что совпадает с регулярным выражением, и замена на указанную строку, например, поиск подстроки «устройство передвижения, движимое мускульной силой» и замена подстрокой «велосипед».

- *Разбиение строк:* разбиение строки по точкам совпадения с регулярным выражением, например, разбиение строки по подстроке «: » или «=».

Спецсимволы

- ✔ . ^ $ * + ? { } [ ] \ | ( )
- ✔ Для написания их просто как символов требуется их *экранировать*, для чего нужно поставить перед ними знак \.

Осторожно - регулярные выражения!

- ✔ Использовать только там, где они действительно приносят пользу, а не вред. Плохо написанные регулярные выражения работают медленно.
- ✔ Их сложно читать, особенно если *регулярка* написана не лично тобой пять минут назад.
- ✔ Даже небольшое изменение задачи (того, что требуется найти) приводит к значительному изменению выражения.
- ✔ Это *write only code* (код, который только пишут с нуля, но не читают и не правят).

Регулярные выражения: как?

- Регулярное выражение - это шаблон, по которому выполняется поиск соответствующего фрагмента текста.
- Язык описания регулярных выражений состоит из символов двух видов: обычных символов и метасимволов.
- Обычный символ представляет в выражении сам себя, а метасимвол - некоторый *класс символов*.

Наиболее употребительные метасимволы:

Use	To match any character
[set]	In that set
[^set]	Not in that set
[a–z]	In the a-z range
[^a–z]	Not in the a-z range
.	Any except \n (new line)
\char	Escaped special character

Примеры

Выражение	***Фрагмент***
c.t	***cat, cut, c#t, c{t***
c[aui]t	***cat, cut, cit***
c[a-c]t	***cat, cbt, cct***
c[^aui]t	***cbt, cct, c2t и т.д.***
c[^a-c]t	***cdt, cet, c%t***

УПРАВЛЯЮЩИЕ СИМВОЛЫ

Use	To match	Unicode
\t	Horizontal tab	\u0009
\v	Vertical tab	\u000B
\b	Backspace	\u0008
\e	Escape	\u001B
\r	Carriage return	\u000D
\f	Form feed	\u000C
\n	New line	\u000A
\a	Bell (alarm)	\u0007
\c char	ASCII control character	–

CHARACTER CLASSES

Use	To match character
\p{ctgry}	In that Unicode category or block
\P{ctgry}	Not in that Unicode category or block
\w	Word character. Любая буква (то, что может быть частью слова), а также цифры и _
\W	Non-word character. Любая не-буква, не-цифра и не подчёркивание
\d	Decimal digit
\D	Not a decimal digit. Любой символ, кроме цифры
\s	White-space character. Любой пробельный символ (пробел, табуляция, конец строки и т.п.)
\S	Non-white-space char. Любой непробельный символ

Примеры

Выражение	***Фрагмент***
c\wt	*cbt*, cct, c2t и т.д., но не c%t, c{t и т.д.
c\Wt	c%t, c{t, c. t , но не *cbt*, cct, c2t и т.д.
\s\w\w\w\s	Любое слово из трех букв, окруженное пробельными символами.
\s\S\S\S\s	Любые три непробельных символа, окруженные пробельными.
c\dt	c1t, c2t, c3t
c\Dt	Не c1t, c2t, c3t и т.д.

Уточняющие символы

Use	To specify position
^	At start of string or line
\A	At start of string (многострочная)
\z	At end of string
\Z	At end (or before \n at end) of string
$	At end (or before \n at end) of string or line
\G	Where previous match ended
\b	On word boundary
\B	Not on word boundary

Повторители или квантификаторы

Greedy	Lazy	Matches
*	*?	0 or more times
+	+?	1 or more times
?	??	0 or 1 time
{n}	{n}?	Exactly n times
{n,}	{n,}?	At least n times
{n,m}	{n,m}?	From n to m times

Про квантификаторы

По умолчанию квантификаторы *жадные* — захватывают максимально возможное число символов.

Добавление **?** делает их *ленивыми*, они захватывают минимально возможное число символов.

Выражение	Фрагмент
ca*t	ct, cat, caat, caaat
ca+t	cat, caat, caaat
ca?t	ct, cat
ca{3}t	caaat
(cat){2}	catcat
ca{3, }t	caaat, caaaat, caaaaaaat
(cat){2, }	catcat, catcatcat
ca{2, 4}t	caat, caaat, caaaat

1. **Язык регулярных выражений в Python**
2. Модуль для работы с регулярными выражениями

pattern, string

- Два параметра в функциях модуля регулярных выражений ***re*** :
  - Шаблон регулярного выражения для определения текста.
  - Текст, который будет проанализирован на соответствие шаблону регулярного выражения.

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 по 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;

Примеры регулярных выражений:

1) Слово math **math, «math»**
2) Номер телефона в формате xxx-xx-xx

@"\d\d\d-\d\d-\d\d" или r"\d{3}(-\d\d){2}

3) Номер автомобиля - **r"[A-Z]\d{3}[A-Z]{2}\d{2,3}RUS"**

Задание. Запишите регулярное выражение, соответствующее:

1. дате в формате дд.мм.гг или дд.мм.гггг
2. времени в формате чч.мм или чч:мм
3. целому числу (со знаком и без)
4. вещественному числу (со знаком и без, с дробной частью и без, с целой частью и без)
5. Найдите все натуральные числа (возможно, окружённые буквами);
6. Найдите все «слова», написанные капсом (то есть строго заглавными), возможно внутри настоящих слов (аааБББввв);
7. Найдите слова, в которых есть русская буква, а когда-нибудь за ней цифра;
8. Найдите все слова, начинающиеся с русской или латинской большой буквы (\b — граница слова);
9. Найдите слова, которые начинаются на гласную (\b — граница слова).

```
match = re.findall(r'[0-9]+\w', r'Телефон 123-12-12')
```