Регулярные выражения

Регулярные выражения (англ. regular expressions) — система обработки текста, основанная на специальной системе записи образцов для поиска. Образец (англ. pattern), задающий правило поиска, по-русски также иногда называют «шаблоном», «маской».

Регулярные выражения - это, по существу, крошечный язык программирования. Используя его, указываются правила для множества возможных строк, которые нужно проверить; это множество может содержать английские фразы, или адреса электронной почты, или TeX команды, или все что угодно. С помощью регулярных выражений можно задавать вопросы, такие как «Соответствует ли эта строка шаблону?», или «Совпадает ли шаблон где-нибудь с этой строкой?». Можно также использовать регулярные выражения, чтобы изменить строку или разбить ее на части различными способами.

Очень часто регулярные выражения используются для того, чтобы проверить, является ли данная строка строкой в необходимом формате. Например, следующий regexp предназначен для проверки того, что строка содержит корректный e-mail адрес:

```
/^\w+([\.\w]+)*\w@\w((\.\w)*\w+)*\.\w{2,3}$/
```

Сутью механизма регулярных выражений является то, что они позволяют задать шаблон для **нечеткого** поиска по тексту. Например, если перед вами стоит задача найти в тексте определенное слово, то с этой задачей хорошо справляются и обычные функции работы со строками. Однако если вам нужно найти "то, не знаю что", о чем вы можете сказать только то, как **приблизительно** это должно выглядеть - то здесь без регулярных выражений просто не обойтись. Например, вам необходимо найти в тексте информацию, про которую вам известно только то, что это "*3 или 4 цифры после которых через пробел идет 5 заглавных латинских букв*", то вы сможете сделать это очень просто, воспользовавшись следующим регулярным выражением:

```
/\d{3,4}\s[A-Z]{5}/
```

Сейчас регулярные выражения используются многими текстовыми редакторами и утилитами для поиска и изменения текста на основе выбранных правил. Многие языки программирования уже поддерживают регулярные выражения для работы со строками. Например, Perl и Tcl имеют встроенный в их синтаксис механизм обработки регулярных выражений. Набор утилит (включая редактор sed и фильтр grep), поставляемых в дистрибутивах Unix, одним из первых способствовал популяризации понятия регулярных выражений.

Области применения

В рамках курса регулярные выражения будут использоваться для подготовки данных. Часто данные бывают представлены в «сыром» неструктурированном виде, не пригодном для обработки их алгоритмами Data Mining. Например, это может быть текст статьи, полученной из интернета или журнал выполнения компьютерной программы. Чтобы извлечь из «сырого» текста полезную информацию, и используются регулярные выражения.

В то же время следует помнить, что язык регулярных выражений относительно мал и ограничен, поэтому не все возможные задачи по обработке строк можно сделать с помощью регулярных выражений. Также существуют задачи, которые *можно* сделать с помощью регулярных выражений, но выражения оказываются слишком сложными. В этих случаях может быть лучше написать обычный код

на языке программирования, пусть он будет работать медленнее, чем разработанное регулярное выражение, но будет более понятен.

Теория

Регулярные выражения, как уже было сказано выше, представляют собой строку. Строка всегда начинается с символа разделителя, за которым следует непосредственно регулярное выражение, затем еще один символ разделителя и потом необязательный список модификаторов. В качестве символа разделителя обычно используется слэш ('/'). Таким образом в следующем регулярном выражении: `/\d{3}-\d{2}/m`, символ '/' является разделителем, строка '`\d{3}-\d{2}`' - непосредственно регулярным выражением, а символ '`m`', расположенный после второго разделителя - это модификатор.

Основой синтаксиса регулярных выражений является тот факт, что некоторые символы, встречающиеся в строке рассматриваются не как обычные символы, а как имеющие специальное значение (т.н. ***метасимволы***). Именно это решение позволяет работать всему механизму регулярных выражений. Каждый метасимвол имеет свою собственную роль в синтаксисе регулярных выражений.

Полные справочники по работе с регулярными выражениями в методических указаниях не приводятся, так как их легко можно найти в интернете, например, по приведённым ниже ссылкам:

https://ru.wikipedia.org/wiki/Регулярные_выражения

https://ru.wikibooks.org/wiki/Регулярные_выражения

Здесь можно поэкспериментировать с регулярными выражениями:

http://pythex.org/

По этой ссылке можно найти краткую шпаргалку по регулярным выражениям, которую удобно всегда держать перед глазами:

http://www.exlab.net/files/tools/sheets/regexp/regexp.pdf

При работе с регулярными выражениями нужно научиться думать в регулярных выражениях. Этот навык приходит с опытом, но нужно знать меру. Если слишком увлекаться регулярными выражениями и пытаться с их помощью решать все задачи по обработке текстов, получающиеся выражения будут слишком сложными и непригодными к пониманию и дальнейшему сопровождению.

Справка по регулярным выражениям в Python

В Python есть встроенная поддержка регулярных выражений, реализованная в модуле re. Для использования регулярных выражений в своих скриптах нужно импортировать данный модуль как показано ниже

```
import re
```

а затем использовать представляемые модулем функции. Описание функций можно найти по этой ссылке ниже, либо в справке Python:

https://tproger.ru/translations/regular-expression-python/

Задания на практику

Ниже описаны 10 различных заданий на регулярные выражения и таблица, в которой указаны задания для каждого варианта. Каждому студенту необходимо выполнить по 3 задания из предложенного ниже списка.

При выполнении заданий внимательно читайте формулировку, не усложняйте регулярные выражения. Они полезны только в том случае, если достаточно просты для понимания и эффективны. Часто на практике при подготовке данных используют сочетание регулярных выражений и кода на языке программирования.

Активно экспериментируйте с регулярными выражениями на сайте http://pythex.org/, либо напишите небольшой скрипт. Проверяйте выражения на всех приведённых тестах.

Задание 1:

Написать регулярное выражение, определяющее, является ли строка кодом цвета в 16-ричном формате.
Корректные значения: #03B63A, #FFF, #000
Некорректные значения: ABCDEF, 123, #GHIJKL

Задание 2:

Написать регулярное выражение для проверки адреса электронной почты. В электронном адресе допустимы латинские символы верхнего и нижнего регистров, цифры, точки, дефисы, подчёркивания.
Корректные значения: a@b.c, a-b@c.d.e, a-b_c.d@e_f-g.h
Некорректные значения: a+@b.c, a_b.c, a_b@.c-d

Задание 3:

Написать регулярное выражение для проверки адреса IP4. Проверку на превышение значений 255 включать не нужно.
Корректные значения: 192.168.0.1, 127.0.0.1, 0.0.0.0, 0.100.200.300
Некорректные значения: 192.168.0., a.b.c.d, 1234.2345.3456.4567

Задание 4:

Написать регулярное выражение, которое из текстовой строки выделяет положительное десятичное число. Число может содержать дробную часть, отделяемую точкой. Число должно отделяться от текста пробелами.
Корректные значения:
some text 5678.23 some text, some text 0 some text, some text 0.15 some text
Некорректные значения:
some text123some text, text 123,4 text, text -123.4 text

Задание 5:

Из HTTP-ссылки выделить подстроку, соответствующую доменам второго и верхнего уровней. Например, из ссылки http://vk.com/friends должна быть выделена подстрока vk.com. Нужно учитывать протоколы http и https. Домены третьего уровней и выше рассматривать не нужно.
Корректные значения: http://ya.ru/index.html, https://wikipedia.org/
Некорректные значения: ftp://some.server/, https://ru.wikipedia.org/

Задание 6:

Написать регулярное выражение, проверяющее номер телефона на соответствие российскому формату. Номер должен начинаться с +7 или 8, далее в скобках один из кодов: 909, 912, 922, затем цифры номера через дефис.
Корректные значения: +7(922)123-45-67, 8(922)123-45-67, +7(909)123-45-67
Некорректные значения: +7(923)123-45-67, +5(922)123-45-67, +7(922)1234567, +7(922)123456

Задание 7:

Из кода html-страницы выделить тег img (изображение) со всем его содержимым.

Корректные значения:

<img alt="title" src="http://test/image.png">

Некорректные значения:

<img alt="title">

Задание 8:

Написать регулярное выражение, проверяющее строку на соответствие даты заданному формату: dd.mm.yyyy. Дата в строке должна быть отделена от текста пробелами. Для числа и месяца отводится строго по 2 цифры, для года – 4 цифры. Проверку на допустимость чисел (не более 31) и месяца (не более 12) реализовывать не нужно.

Корректные значения: 12.03.2016, 02.05.1928

Некорректные значения: 12/03/2016, 12-03-2016, 2016.03.12, 2016 03 12, 5.3.2016

Задание 9:

Написать регулярное выражение, проверяющее строку на соответствие почтовому индексу в российском формате.

Корректные значения: 123456

Некорректные значения: 123 456, str123456str, 123a456b

Задание 10:

Написать регулярное выражение, проверяющее строку на соответствие допустимому имени файлов. Допустимыми являются имена, состоящие из букв, с расширениями png, jpg, jpeg, gif.

Корректные значения: test.png, test.jpeg, test.jpg, test.gif

Некорректные значения: test.php, test.exe, ~!@$%.png, <?php test.png ?>

Задание / Вариант	1	2	3	4	5	6	7	8	9	10
1	■				■				■	
2		■		■			■			
3			■					■		■
4	■				■	■				
5		■							■	■
6						■	■		■	
7			■	■				■		
8	■				■		■			
9				■				■		■
10		■	■			■				
11	■				■					■
12		■				■		■		
13			■	■			■			
14	■					■			■	
15		■			■			■		