







Программная инженерия. Разработка ПО (Python для продвинутых специалистов. Машинное обучение)

Модуль: Введение в Python для машинного обучения

Лекция 8: Регулярные выражения для очистки и нормализации данных







Содержание лекции

ПЕРЕДОВАЯ
ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

- Базовые элементы регулярного выражения
- Регулярные выражения в Базе данных
- Регулярные выражения в Python
- Регулярные выражения в Unix
- Практическая часть



Почему полезны регулярные выражения



Регулярное выражение (Regular expression) – это строка, которая задает шаблон поиска подстрок в заданном тексте.

С помощью регулярных задач решается несколько классов задач:

1. Валидация значения

- Как убедиться в корректности введенного пользователем телефона/email?
- Как провалидировать переданную дату/время или ір-адрес?
- Как проверить пароль на соответствие правилам безопасности?

2. Работа с текстом (поиск, извлечение, редактирование)

- Как найти все повторяющиеся значения в текстовом файле размером 1Gb?
- Как удалить или заменить определенные части в файле или множестве файлов?
- Как проверить лог файл на сервере на предмет ошибок в заданную дату?

3. Подготовка и очистка данных

- Как отфильтровать не валидные записи из текстового файла или таблицы?
- Как проверить файл заданного формата на пустые значения?
- Как достать нужные параметры из URL-адреса?

Базовые элементы регулярного

ВЫРАЖЕНИЯ1. Группы символов

- - [abcABC123] любой из символов
 - [a-zA-Z0-9] любой из диапазона символов
 - [^0-9] любой НЕ из диапазона символов
 - (ABC|DEF) любой из группы ИЛИ любой из другой группы
- Специальные знаки
 - ^ начало строки
 - \$-конец строки
 - . любой символ
 - \- символ экранирования
 - \w − буквы, цифры и знак _
 - \W НЕ буквы, НЕ цифры и НЕ знак _
 - /d цифры
 - \D НЕ цифры
 - \s любой пробельный символ (пробел, неразрывный пробел, табуляция)
 - \S НЕ пробельный символ (пробел, неразрывный пробел, табуляция)
 - \b граница слова (работает не везде)
 - \B НЕ граница слова (работает не везде)
 - •[-+]?Лबсь симвыновисла



- Квантификаторы
 - + один и больше
 - * ноль и больше
 - ? ноль или одно
 - {n} ровно n
 - {n,m} от n до m
 - {n,} не менее n
 - {,m} не более m

Регрулярки в Базе данных



Функция

regexp_like('Year of 2023', '\d+') ИЛИ 'Year of 2023' ~ '\d+'

regexp_count('Year of 2023', $\cdot \cdot \cdot d+'$)

1. regexp_substr('Year of 2023', '\d+', [flags])
2. regexp_match('Year of 2023', '\d+', [flags])

regexp_instr('Year of 2023', '\d+')

regexp_replace('Year of 2023', '\d+', 'Dragon')

<u>Назначение</u>

Поиск по регулярному выражению (возвращает «истина» или «ложь»).

Подсчет количества найденных вхождений регулярного выражения.

Возвращает подстроку, найденную по регулярному выражению. match умеет работать с групповыми выражениями

Возвращает позицию найденной подстроки по регулярному выражению.

Заменяет все вхождения, найденные по регулярному выражению на подстроку.

SQLite нативно НЕ поддерживает регулярные выражение, требуется установка <u>sqlean-regexp</u>. В этом случае синтаксис будет следующим:

SELECT * FROM table WHERE column_name REGEXP '^abc';

Регрулярки в Pyhon



Для работы с регулярками в Python используется модуль **re**

Вот базовые функции, доступные в Python:

- re.search(pattern, string) ищет в строке string первую строчку, подходящую под шаблон pattern
- re.fullmatch(pattern, string) проверяет, подходит ли строка string под шаблон pattern
- **re.split(pattern, string)** разбивает строку *string* на элементы списка, в качестве разделителя используется pattern
- re.findall(pattern, string) находит в строке string все непересекающиеся шаблоны pattern
- re.finditer(pattern, string) вернет итератор по всем непересекающимся шаблонам pattern в строке string
- re.sub(pattern, repl, string) заменит в строке string все непересекающиеся щаблоны pattern на repl

```
import re

pattern = r'\d\d\D\d\d'
string = r'Телефон 123-12-12'

match = re.search(pattern, string)
print(match)
```



Регрулярки в Unix



B Unix системах регулярные выражения поддерживаются нативно во многих базовых командах, таких как:

- grep
- find
- sed
- И тд.

```
root@28f8a4979f84:# cat phones.txt
(185) 136-1035
(95) 213-1874
(37) 207-2639
(285) 227-1602
(275) 298-1043
(107) 204-2197
(799) 240-1839
(218) 750-7390
(114) 776-2276
(7012) 219-308

root@28f8a4979f84:# grep -Ev '^\([0-9]{3}) [0-9]{4}$' phones.txt
(95) 213-1874
```

(37) 207-2639 (7012) 219-308



Много материалов по регуляркам



- Введение в регулярные выражения
- Оригинальная документация Python
- Очень подробный и обстоятельный материал
- Разные сложные трюки и тонкости с примерами
- Он-лайн отладка регулярок
- Он-лайн визуализация регулярок
- Тренировка-развлечение
- Платформа проверки регулярных выражений





Создайте запрос, который позволяет определить корректность электронной почты.

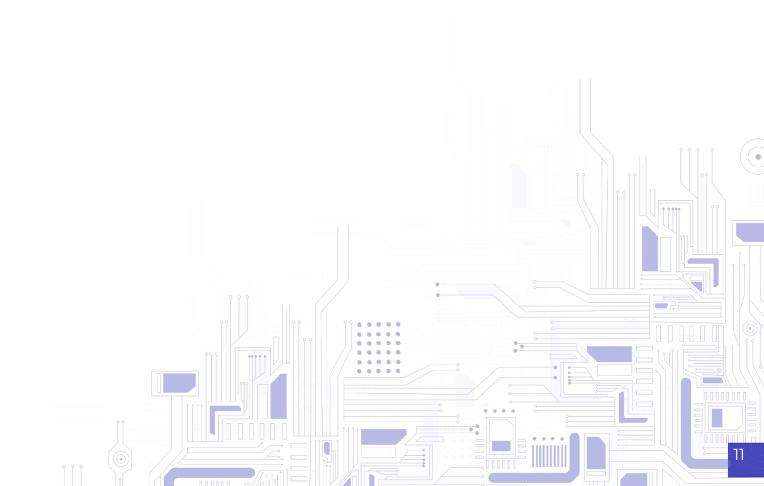




Создайте запрос, который позволяет определить корректность электронной почты.

Вероятнее всего, вы напишите что-то вроде:

'^.*@.*\.[a-z]{2,10}\$'





Создайте запрос, который позволяет определить корректность электронной почты.

Что встречается на практике:

 $(?:[a-z0-9!\#\%\&'*+/=?^{-}]+(?:\.[a-z0-9!\#\%\&'*+/=?^{-}]+)^*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\[\x01-\x09\x0b\x0c\x0e-\x7f])^*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9])?\.)+[a-z0-9](?:[x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\.)+[x01-\x09\x0b\x0c\x0e-\x7f])+)\])$





Создайте запрос, который позволяет определить корректность электронной почты.

Что встречается на практике:

 $(?:[a-z0-9!\#\%\&'*+/=?^{_}\{|]^{-}+(?:\.[a-z0-9!\#\%\&'*+/=?^{_}\{|]^{-}+)^*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\[\x01-\x09\x0b\x0c\x0e-\x7f])^*")@(?:(?:[a-z0-9](?:[a-z0-9-]^*[a-z0-9])?\].)+[a-z0-9](?:[a-z0-9])?\] (?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\].)+[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\])$

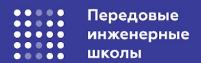
Важно помнить:

- Применяйте регурялки только там, где это действительно необходимо (например, не стоит использовать регулярки для парсинга файлов формата XML, HTML и тд.)
- Плохо написанное регулярное выражение снижает производительность
- Регулярные выражения как правило write-only

Юмор:

Некоторые люди, когда сталкиваются с проблемой: «Это просто, я решу её с помощью регулярных выражений!»
Теперь у них две проблемы.











Спасибо за внимание



