Введение в регулярные выражения

Дмитрий Колодезев, ООО Промсофт
(Рабочая версия 01.09.2024)

Швейцарский нож для текста

- Проверка на совпадение
- Поиск подстроки в строке
- Предобработка (замена/удаление)
- Анонимизация (очистка от персональных данных)
- Токенизация (разбиение на токены)
- Разбор строк по шаблонам
- Построение признаков

Они везде

```
import random
from contextlib import contextmanager
from copy import deepcopy
import re

from functools import partial


def _split_tokenizer(x):  # noqa: F821
    # type: (str) -> List[str]
    return x.split()


def _spacy_tokenize(x, spacy):
    return [tok.text for tok in spacy.tokenizer(x)]


_patterns = [r'\'',
             r'\"',
             r'\.',
             r'<br \/>',
```

https://pytorch.org/text/_modules/torchtext/data/utils.html

В каждом утюге

Source code for nltk.tokenize.regexp

```
# Natural Language Toolkit: Tokenizers
#
# Copyright (C) 2001-2022 NLTK Project
# Author: Edward Loper <edloper@gmail.com>
#         Steven Bird <stevenbird1@gmail.com>
#         Trevor Cohn <tacohn@csse.unimelb.edu.au>
# URL: <https://www.nltk.org>
# For license information, see LICENSE.TXT

r"""
Regular-Expression Tokenizers

A ``RegexpTokenizer`` splits a string into substrings using a regular expression.
For example, the following tokenizer forms tokens out of alphabetic sequences,
money expressions, and any other non-whitespace sequences:
```

https://www.nltk.org/_modules/nltk/tokenize/regexp.html

Строки в python

- R — строки, в которых \ это просто \
- Чтобы в глазах не рябило

```
P<response_time>\d+)|-) \[(?P<datetime>\d{2}\/\w+\/\d{4}:\d{2}:\d{2}:\d{2} \+\d{4})\] \"(?P<request_type>\w+) (?P<request
P<response_time>\d+)|-) \[(?P<datetime>\d{2}\/\w+\/\d{4}:\d{2}:\d{2}:\d{2} \+\d{4})\] \"(?P<request_type>\w+) (?P<request


P<response_time>\d+)|-) \[(?P<datetime>\d{2}\/\w+\/\d{4}:\d{2}:\d{2}:\d{2} \+\d{4})\] \"(?P<request_type>\w+) (?P<request
P<response_time>\d+)|-) \[(?P<datetime>\d{2}\/\w+\/\d{4}:\d{2}:\d{2}:\d{2} \+\d{4})\] \"(?P<request_type>\w+) (?P<request

```

```
In [1]: r"мама\tмыла\траму" == "мама\\tмыла\\траму"
Out[1]: True
```

Библиотека re

```
1 [x for x in dir(re) if x[:1]!= '_']
```

```
['A',
 'ASCII',
 'DEBUG',
 'DOTALL',
 'I',
 'IGNORECASE',
 'L',
 'LOCALE',
 'M',
 'MULTILINE',
 'RegexFlag',
 'S',
 'Scanner',
 'T',
 'TEMPLATE',
 'U',
 'UNICODE',
 'VERBOSE',
 'X',
 'X',
 'compile',
 'copyreg',
 'enum',
 'error',
 'escape',
 'findall',
 'finditer',
 'fullmatch',
 'functools',
 'match',
 'purge',
 'search',
 'split',
 'sre_compile',
 'sre_parse',
 'sub',
 'subn',
 'template']
```

Возьмем какой-нибудь текст

```
1 import requests
2 text = requests.get("https://open-data-science-lab.github.io/archive").text
```

```
1 text
```

```
'<!DOCTYPE html>\n<html lang="en"><head>\n  <meta charset="utf-8">\n  <meta http-equiv="X-
UA-Compatible" content="IE=edge">\n  <meta name="viewport" content="width=device-width, in
itial-scale=1"><!-- Begin Jekyll SEO tag v2.8.0 -->\n<title>Архив новостей | OpenDataScien
ce Lab</title>\n<meta name="generator" content="Jekyll v3.9.0" />\n<meta property="og:titl
e" content="Архив новостей" />\n<meta property="og:locale" content="en_US" />\n<meta name
="description" content="open-data-science-lab.github.io" />\n<meta property="og:descriptio
n" content="open-data-science-lab.github.io" />\n<link rel="canonical" href="https://open-
data-science-lab.github.io/archive" />\n<meta property="og:url" content="https://open-data
-science-lab.github.io/archive" />\n<meta property="og:site_name" content="OpenDataScience
Lab" />\n<meta property="og:type" content="website" />\n<meta name="twitter:card" content
="summary" />\n<meta property="twitter:title" content="Архив новостей" />\n<script type="a
pplication/ld+json">\n{"@context":"https://schema.org","@type":"WebPage","description":"op
en-data-science-lab.github.io","headline":"Архив новостей","url":"https://open-data-scienc
e-lab.github.io/archive"}</script>\n<!-- End Jekyll SEO tag -->\n<link rel="stylesheet" hr
```

Какой-нибудь сайт

OpenDataScience Lab

Архив новостей

Архив новостей

- **2022-03-29** Вторничные анонсы: Туториал по регулярным выражениям
- **2022-03-10** Первый "Проектный день" в ODS Lab
- **2022-03-02** Про гранты. Как и зачем подавать заявку на конкурс
- **2022-02-24** Проект Covid Causal Inference
- **2022-02-24** Проект Board Game Assistant
- **2022-02-23** Board Game Assistant, Covid Causal Inference, Pytest и все-все-все
- **2022-02-21** Вторая волна проектов

re.findall

```
1 re.findall(r'\d{4}-\d{2}-\d{2}', text)
```

```
['2022-03-29',
 '2022-03-10',
 '2022-03-02',
 '2022-02-24',
 '2022-02-24',
 '2022-02-23',
 '2022-02-21']
```

Что это еще за \d

- \d — любая цифра
- \d{4} — четыре цифры подряд
- \d{4}-\d{2}-\d{2}
  - Четыре цифры
  - Дефис
  - Две цифры
  - Дефис
  - Две цифры
- 2022-03-30

re.findall

```
1 re.findall(r'<a href="([^"]+)"', text)
```

```
['./%D0%B0%D0%BD%D0%BE%D0%BD%D1%81/2022/03/29/tue_announcement.html',
 './2022/03/10/project_day.html',
 './2022/03/02/meetup.html',
 './2022/02/24/covid_causal.html',
 './2022/02/24/board-game-assistant.html',
 './2022/02/23/meetup.html',
 './2022/02/21/second_wave.html']
```

Что это еще за [^"]+

- [^"] — любой символ кроме "
- [^"]+ — один или больше таких символов
- "[^"]+" — в кавычках
- "([^"]+)" — то, что в скобках, выдать
- r'<a href="([^"]+")' — начинается с <a href

```
1 re.findall(r"[мпа]+", "мама и папа мыли раму")
```

```
['мама', 'папа', 'м', 'ам']
```

Символьные классы

- Буква/цифра обозначает саму себя
- . любой символ (если DITALL - включая \n)
- ^ начало строки (если MULTILINE - каждой)
- $ конец строки (если MULTILINE - каждой)
- [что-то тут] любой из символов внутри скобок
- \s пробельный символ [\t\n\r\f\v] см вики
- \S не \s [^ \t\n\r\f\v]
- \b начало или конец слова
- \B не \b

Видишь символ? А он есть

```
1  re.findall(r"\b", "мама мыла раму")
2
```

```
['', '', '', '', '', '']
```

```
1  re.findall(r"\B", "мама мыла раму")
```

```
['', '', '', '', '', '', '', '', '']
```

```
1  re.findall(r"\Bм\S+", "мама мыла раму")
2
```

```
['ма', 'му']
```

Если строк много

- \A — начало текста, \Z - конец

```
1 re.findall("\A.{5}", text)
```

```
['<!DOC']
```

```
1 re.findall("^.{5}", text)
```

```
['<!DOC']
```

```
1 re.findall("^.{5}", text, re.MULTILINE)
```

```
['<!DOC',
 '<html',
 '  <me',
 '  <me',
```

Еще символьные классы

- \w — все что может быть внутри слова
- \W — все что не может быть внутри слова

```
1 re.findall(r"\w+", "мама мыла Ford")
```

```
['мама', 'мыла', 'Ford']
```

```
1 re.findall(r"\w+", "мама мыла Ford", re.ASCII)
```

```
['Ford']
```

RE.LOCALE ;(

```
1  P = r"\w+".encode()
2  S = "DON'T DO THAT НИКОГДА".encode()
3  re.findall(P, S, re.LOCALE)
```

```
[b'DON', b'T', b'DO', b'THAT']
```

re.LOCALE

Make `\w`, `\W`, `\b`, `\B` and case-insensitive matching dependent on the current locale. This flag can be used only with bytes patterns. The use of this flag is discouraged as the locale mechanism is very unreliable, it only handles one “culture” at a time, and it only works with 8-bit locales. Unicode matching is already enabled by default in Python 3 for Unicode (str) patterns, and it is able to handle different locales/languages. Corresponds to the inline flag `(?L)`.

https://docs.python.org/3/library/re.html#re.LOCALE

Диапазоны

- [a-z] любой из символов от a до z
- [a-zA-Z0-9] цифра или латинская буква
- [0-9-] цифра или дефис
- \d цифра [0-9]
- \D не цифра [^0-9]

Метасимволы

- . ^ $ * + ? { } [ ] \ | ( )
- Нужно экранировать
- Внутри [] нужно экранировать только []
- $ внутри символьного класса — просто символ
- Не первый ^ внутри [] — просто символ
- Последний - внутри [] — просто символ

Флаги

- ASCII, A — только latin1
- DOTALL, S — . совпадает с \n

```
1 len(re.findall(".*", text))
```

```
190
```

```
1 len(re.findall(".*", text, re.DOTALL))
```

```
2
```

```
1 re.findall(".*", "\n", re.DOTALL)
```

```
['\n', '']
```

Другие флаги

- IGNORECASE, I — игнорировать регистр
- MULTILINE, M - ^ и $ в каждой строке
- LOCALE, L — не надо
- VERBOSE, X — комментарии и пробелы

```
1  P = """\d{4}    # Год
2  -\d{2}          # Месяц
3  -\d{2}          # День"""
4  re.findall(P, text, re.VERBOSE)
```

```
['2022-03-29',
 '2022-03-10',
 '2022-03-02',
```

Выбор |

- Совпадает любое из подвыражений
- Может быть использовано внутри групп
- Проверяется слева направо
- До первого совпадения

```
1 re.findall(r"мама|папа|дети", "мама мыла раму")
```

```
['мама']
```

```
1 re.findall(r"мама|мама мыла", "мама мыла раму")
```

```
['мама']
```

Квантификаторы

- * - 0 или больше
- + - 1 или больше
- ? - 0 или 1 раз
- {m} — m раз
- {m,} — m раз или больше
- {,n} — n раз или меньше
- {m, n} — от m до n раз

```
1 re.findall(r"мама?", "мама мыла маму")
```

```
['мама', 'мам']
```

Жадность

- Ищется максимально возможное совпадение

```
1  re.findall(r"м.*а", "мама мыла раму")
```

```
['мама мыла ра']
```

```
1  len(re.findall(r"<.*>", text, re.DOTALL))
```

```
1
```

Нежадные квантификаторы

- ? после квантификатора означает нежадность
- Нежадные квантификаторы пытаются совпасть с как можно меньшей длиной

```
1 len(re.findall(r"<.*>", text, re.DOTALL))
```

```
1
```

```
1 len(re.findall(r"<.*?>", text, re.DOTALL))
```

```
114
```

Жадный/Нежадный

- Жадный a+ => все подряд идущие a
- Нежадный a+? => одно a

```
1 txt = "aaaaaaaaaaaaaaaa"
2 re.search(r'a+', txt)
```

```
<_sre.SRE_Match object; span=(0, 16), match='aaaaaaaaaaaaaaaa'>
```

```
1 txt = "aaaaaaaaaaaaaaaa"
2 re.search(r'a+?', txt)
```

```
<_sre.SRE_Match object; span=(0, 1), match='a'>
```

Regex-убийцы

```
1 %time re.findall(r'^(A+|X)*B', "A"*3 + "C")
```

```
CPU times: user 24 µs, sys: 1 µs, total: 25 µs
```

```
1 %time re.findall(r'^(A+|X)*B', "A"*10 + "C")
```

```
CPU times: user 698 µs, sys: 0 ns, total: 698 µs
```

```
1 %time re.findall(r'^(A+|X)*B', "A"*20 + "C")
```

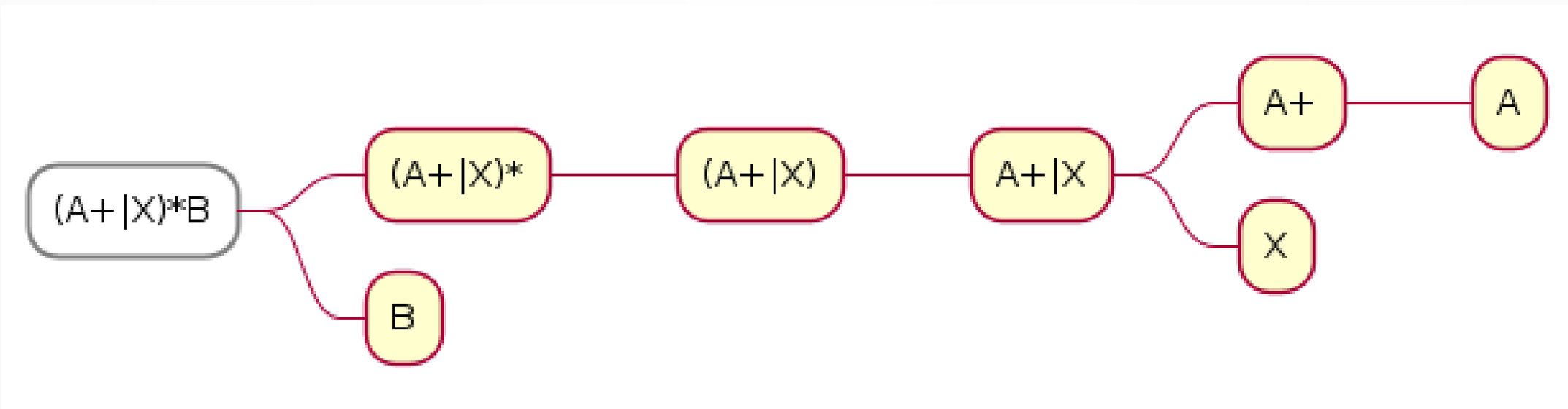
```
CPU times: user 109 ms, sys: 0 ns, total: 109 ms
```

```
1 %time re.findall(r'^(A+|X)*B', "A"*25 + "C")
```

```
CPU times: user 2.75 s, sys: 6.61 ms, total: 2.76 s
```

https://www.rexegg.com/regex-explosive-quantifiers.html

Дерево разбора



По шагам

Еще пример

```
txt = '<!DOCTYPE html>'
%time re.findall(r"^((.*)|(.*))*>>", txt)
```

```
CPU times: user 2.51 s, sys:
7.83 ms, total: 2.52 s
Wall time: 2.51 s
```

re.finditer

```
1 ri = re.finditer(r'\d{4}-\d{2}-\d{2}', text)
2 ri
```

```
<callable_iterator at 0x7f8644fcd828>
```

```
1 [x for x in ri]
```

```
[<_sre.SRE_Match object; span=(2912, 2922), match='20
22-03-10'>,
 <_sre.SRE_Match object; span=(3031, 3041), match='20
22-03-02'>,
 <_sre.SRE_Match object; span=(3173, 3183), match='20
22-02-24'>,
```

re.finditer

```
1 x = next(ri)
2 x
```

```
<_sre.SRE_Match object; span=(2769, 2779), match='202
2-03-29'>
```

```
1 [x for x in dir(x) if x[:1] != '_']
```

```
['end',          'lastindex',
 'endpos',       'pos',
 'expand',       're',
 'group',        'regs',
 'groupdict',    'span',
 'groups',       'start',
 'lastgroup',    'string']
```

Match object

- К группам мы еще вернемся
- Группа 0 — все что совпало

```
1 x.group(0), x.span()
```

```
('2022-03-29', (2769, 2779))
```

```
1 text[x.span()[0]: x.span()[1]]
```

```
'2022-03-29'
```

re.search

- Для поиска подстроки в строке
- Ищет первое совпадение в строке
- Возвращает match объект

```
1 m = re.search(r'\d{4}-\d{2}-\d{2}', text)
2 m
```

```
<_sre.SRE_Match object; span=(2769, 2779), match='202
2-03-29'>
```

```
1 m.group(0)
```

```
'2022-03-29'
```

re.match

- Совпадает только с началом строки

```
1  re.match("\d+", text)
```

```
1  re.match("<!\w+", text)
```

```
<_sre.SRE_Match object; span=(0, 9), match='<!DOCTYP
E'>
```

re.fullmatch

- Совпадает только с полным текстом
- Будьте внимательны:

```
1 re.fullmatch("<!.*>", text, re.DOTALL)
```

```
1 re.fullmatch("<!.*>", text.strip(), re.DOTALL)
```

```
<_sre.SRE_Match object; span=(0, 4340), match='<!DOCT
YPE html>\n<html lang="en"><head>\n  <meta >
```

```
1 re.fullmatch("<!.*>\n", text, re.DOTALL)
```

```
<_sre.SRE_Match object; span=(0, 4341), match='<!DOCT
YPE html>\n<html lang="en"><head>\n  <meta >
```

Группа захвата

- Все что в скобках раскладывается в «группы»
- Группа 0 — все совпавшее

```
1 m = re.search(r'(\d{4})-(\d{2})-(\d{2})', text)
2 m
```

```
<_sre.SRE_Match object; span=(2769, 2779), match='202
2-03-29'>
```

```
1 m.group(0), m.group(1), m.group(2), m.group(3)
```

```
('2022-03-29', '2022', '03', '29')
```

Группа захвата

- Группы могут быть вложенными

```
1 m = re.search(r'(((\d{4})-\d{2})-\d{2})', text)
2 m
```

```
<_sre.SRE_Match object; span=(2769, 2779), match='202
2-03-29'>
```

```
1 m.group(0), m.group(1), m.group(2), m.group(3)
```

```
('2022-03-29', '2022-03-29', '2022-03', '2022')
```

Группы и выбор

```
1  P = r'(мама|папа) мыла?'
2  m = re.search(P, "мама мыла раму")
3  m.group(0), m.group(1)
```

```
('мама мыла', 'мама')
```

Группа, которая не смогла

```
1  P = r'(мама|(папа)) мыла?'
2  m = re.search(P, "мама мыла раму")
3  m.group(0), m.group(1), m.group(2)
```

```
('мама мыла', 'мама', None)
```

```
1  P = r'((мама)|папа) мыла?'
2  m = re.search(P, "мама мыла раму")
3  m.group(0), m.group(1), m.group(2)
```

```
('мама мыла', 'мама', 'мама')
```

Match.expand

- Подставляет захваченные группы в шаблон
- Как re.sub со ссылкой на группу

```
m = re.search(r"(\w+)", "мама мыла Ford", re.ASCII)
m.expand(r"Их машина - \1")
```

```
'Их машина - Ford'
```

Скобки без захвата

- Если нужны скобки, но не группа

```
1 P = r'(?:мама|папа) (мыла?)'
2 m = re.search(P, "мама мыла раму")
3 m.group(0), m.group(1)
```

```
('мама мыла', 'мыла')
```

Комментарии

- Если нужно временно убрать часть регулярки

```
1 P = r'(?:мама|папа)(?# мыла?)'
2 m = re.search(P, "мама мыла раму")
3 m.group(0)
```

```
'мама'
```

```
1 P = r'(?#мама|папа) мыла'
2 m = re.search(P, "мама мыла раму")
3 m.group(0)
```

```
' мыла'
```

re.compile

- Можно скомпилировать регулярку
- Для скорости или для удобства

```
1 rd = re.compile(r'(\d{4})-(\d{2})-(\d{2})')
2 rd
```

```
re.compile(r'(\d{4})-(\d{2})-(\d{2})', re.UNICODE)
```

```
1 rd.findall
```

```
<function SRE_Pattern.findall(string=None, pos=0, end
pos=9223372036854775807, *, source=None)>
```

SRE_Pattern

```
1 rd.search(text)
```

```
<_sre.SRE_Match object; span=(2769, 2779), match='202
2-03-29'>
```

```
1 rd.findall(text)
```

```
[('2022', '03', '29'),
 ('2022', '03', '10'),
 ('2022', '03', '02'),
```

SRE_Pattern

```
1 [x for x in dir(rd) if x[:1] != '_']
```

```
['findall',    'pattern',
 'finditer',   'scanner',
 'flags',      'search',
 'fullmatch',  'split',
 'groupindex', 'sub',
 'groups',     'subn']
 'match',
```

```
1 rd.pattern
```

```
'(\\d{4})-(\\d{2})-(\\d{2})'
```

re.purge

- Библиотека re кеширует скомпилированное
- Размер кеша в re._MAXCACHE
- Как только заполняется, сбрасывается ВЕСЬ
- Можно сбросить руками

```
1 re._MAXCACHE
```

```
512
```

re.sub

- Поиск и замена

```
1 txt = """Контактные телефоны
2 7(903)930-17-24, 8(495)117-03-99"""
```

```
1 P = r'[78]\(\d\d\d\)\d\d\d-\d\d-\d\d'
2 re.findall(P, txt)
```

```
['7(903)930-17-24', '8(495)117-03-99']
```

```
1 re.sub(P, '#PHONE#', txt)
```

```
'Контактные телефоны\n#PHONE#, #PHONE#'
```

Обратные ссылки в re.sub

```
1  txt = "мама мыла раму"
2  re.sub('мама', 'Оля', txt)
```

```
'Оля мыла раму'
```

```
1  re.sub(r'(мама) (мыла) (раму)', r'\3 \2 \1', txt)
```

```
'раму мыла мама'
```

```
1  re.sub('(мама|папа|Вера) (мыл)а?', r'папа \2', txt)
```

```
'папа мыл раму'
```

Функции в re.sub

- Можно передать функцию

```
1  def phone_anon(m):
2      if m.group(0).startswith('7'):
3          return "#PHONE7#"
4      elif m.group(0).startswith('8'):
5          return "#PHONE8#"
6      else:
7          return "#ERROR#"
```

```
1  re.sub(P, phone_anon, txt)
```

```
'Контактные телефоны\n#PHONE7#, #PHONE8#'
```

Функции в re.sub

```
1  import random
2  def repl(m):
3      tmp = list(m.group(2))
4      random.shuffle(tmp)
5      return m.group(1) + "".join(tmp) + m.group(3)
6
7  txt = "The quick brown fox jumps over the lazy dog"
8  re.sub(r"(\w)(\w+)(\w)", repl, txt)
```

```
'The qucik brwon fox jpums oevr the lazy dog'
```

re.split

- Можно резать на токены

```
1 re.split('[ ,\n]+', txt)
```

```
['Контактные', 'телефоны', '7(903)930-17-24', '8
(495)117-03-99']
```

re.escape

- Все спецсимволы, см документацию

```
1  P = r'[78]\(\d\d\d\)\d\d\d-\d\d-\d\d'
2  P
```

```
'[78]\\(\\d\\d\\d\\)\\d\\d\\d-\\d\\d-\\d\\d'
```

```
1  re.escape(P)
```

```
'\\[78\\]\\\\\\(\\\\d\\\\d\\\\d\\\\\\)\\\\d\\\\d
\\\\d\\-\\\\d\\\\d\\-\\\\d\\\\d'
```

Именованные группы

- (?P<ИМЯ>регулярное_выражение)

```
1 P = r'(?P<year>\d{4})-(?P<month>\d{2})-(?P<day>\d{2})'
2 m = re.search(P, text)
3 m.group(0), m.groups()
```

```
('2022-03-29', ('2022', '03', '29'))
```

```
1 m.groupdict()
```

```
{'year': '2022', 'month': '03', 'day': '29'}
```

backreference

- В выражении можно ссылаться на совпавшие ранее группы, по номеру или по имени
- \1 или(?P=ИМЯ)

```
1 m = re.search(r'<([^>]+?)>.*</\1>', text)
2 m.group(0)
```

```
'<title>Архив новостей | OpenDataScience Lab</title>'
```

```
1 m = re.search(r'<(?P<tag>[^>]+?)>.*</(?P=tag)>', text)
2 m.group(0)
```

```
'<title>Архив новостей | OpenDataScience Lab</title>'
```

lookahead

- (?=...) Проверяем что там впереди

```
1  text = "мама мыла раму"
2  m = re.search(r'\w+(?=\sмыла)', text)
3  m.group(0)
```

```
'мама'
```

negative lookahead

- Негативный предпросмотр

```
1 text = "мама мыла раму, девочка пела песню"
2 re.findall(r'(мама|девочка)\s(?!мыла)', text)
```

```
['девочка']
```

lookbehind

- (?<=...) Перед совпавшим текстом есть
- (?<!...) Перед совпавшим текстом нет
- Квантификаторы нельзя

```
1  text = "возьми 100 грамм глины, 300 грамм золы, перемешай."
2  re.findall(r'(?<=\d\d\d\s)\w+', text)
```

```
['грамм', 'грамм']
```

Флаги внутри выражения

- (?aiLmsux)
  - Задает флаги внутри выражения, а не в параметрах функции. Не группа.
  - Действует на все выражение.
  - Должно стоять в начале выражения.
- (?aiLmsux-imsx:...)
  - Добавляет (до дефиса) или снимает (после) флаги.
  - Действует только на подвыражение. Не группа.

Флаги внутри выражения

```
1 re.search(r'Yandex\.Metrika', text)
```

```
<_sre.SRE_Match object; span=(3684, 3698), match='Yandex.Metrika'>
```

```
1 re.search(r'(?i)yandex\.Metrika', text)
```

```
<_sre.SRE_Match object; span=(3684, 3698), match='Yandex.Metrika'>
```

```
1 re.search(r'(?i:yandex)\.Metrika', text)
```

```
<_sre.SRE_Match object; span=(3684, 3698), match='Yandex.Metrika'>
```

```
1 re.search(r'(?i:yandex)\.metrika', text)
```

Паттерн выбора

- (?(id/name)yes-pattern|no-pattern)

```
1 txt = "<info@promsoft.ru>"
2 re.match(r'(<)?(\w+@\w+(?:\.\w+)+)(?(1)>|$)', txt)
```

```
<_sre.SRE_Match object; span=(0, 18), match='<info@promsoft.ru>'>
```

```
1 txt = "info@promsoft.ru"
2 re.match(r'(<)?(\w+@\w+(?:\.\w+)+)(?(1)>|$)', txt)
```

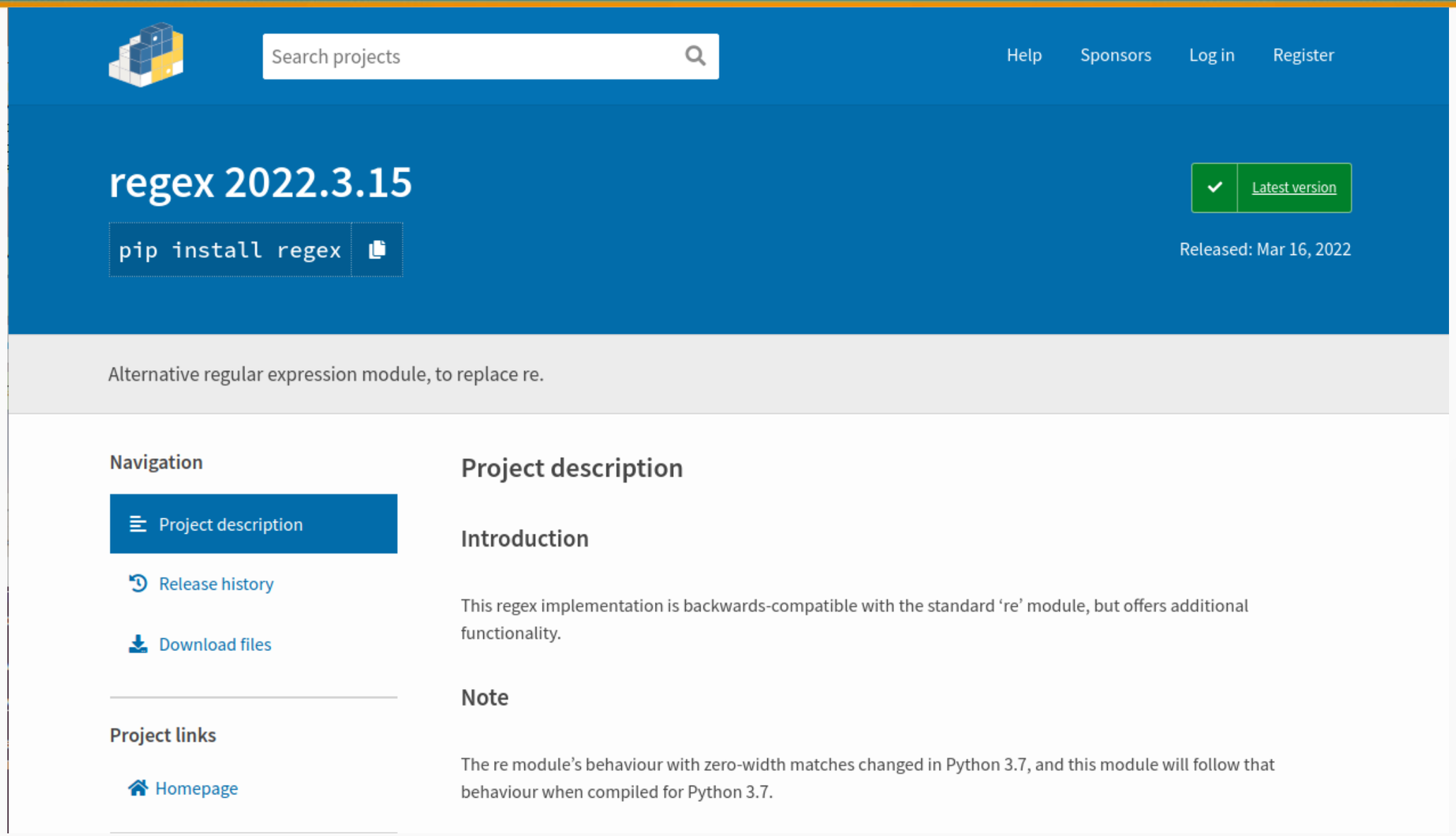
```
<_sre.SRE_Match object; span=(0, 16), match='info@promsoft.ru'>
```

```
1 txt = "<info@promsoft.ru"
2 re.match(r'(<)?(\w+@\w+(?:\.\w+)+)(?(1)>|$)', txt)
```

Ограничения regex

- Lock на процесс (GIL) в re
  - Библиотека regex не блокирует процесс
- Не работает произвольная вложенность
  - Невозможно написать regex для проверки скобок
  - Возможно в некоторых диалектах
- Экспоненциальная сложность
- Очень трудно поддерживать

Regex на стероидах



Стероиды

- Обратно совместима с re
- Операции над множествами
- Нечеткие совпадения
- Освобождает GIL во время матчинга
- Частичный матчинг
- Рекурсивные шаблоны
- Классы Unicode
- Таймауты

Версии

- V0 — совместима с re
- V1 — новые возможности
- Можно задать флагами V0 V1
- Можно задать inline (?V0) (?V1)

```
import regex
regex.DEFAULT_VERSION == regex.V0
```

```
True
```

Regex set operations

- -- && || ~~

```
txt = 'The quick brown fox jumps over'
re.findall(r'[[a-z]--[aeiou]]', txt)
```

```
[]
```

```
regex.findall(r'[[a-z]--[aeiou]]+', txt, regex.V0)
```

```
[]
```

```
regex.findall(r'[[a-z]--[aeiou]]+', txt, regex.V1)
```

```
['h', 'q', 'ck', 'br', 'wn', 'f', 'x', 'j', 'mps', 'v', 'r']
```

https://unicode.org/reports/tr18/#Subtraction_and_Intersection

Полезное

- Документация re
- Regular Expressions (Regex) Tutorial
- Библиотека regex
- Регулярные выражения, Фридл Джеффри
- regex в MongoDB
- regex в MySQL
- regex в JS

Вопросы

Слайды тут

dkolodezev

promsoft

d_key

dmitry_kolodezev

https://kolodezev.ru/download/regex.pdf