

Регулярные выражения

Глава 11

Пайтон для всех
www.py4e.com



Регулярные выражения

В компьютерной терминологии «регулярное выражение» (его еще называют *редехр* или *редех*, сокр. «регулярка») — мощное и гибкое средство для сопоставления строк текста, например, определенных символов, слов или наборов символов.

Регулярное выражение написано на формальном языке, который может интерпретироваться обработчиком регулярных выражений

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

Регулярные выражения

Умный подход к анализу и сопоставлению
строк, основанный на использовании
метасимволов

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



Википедия
Свободная энциклопедия

[Заглавная страница](#)
[Рубрикация](#)
[Указатель А—Я](#)
[Избранные статьи](#)
[Случайная страница](#)
[Текущие события](#)

[Участие](#)
[Сообщить об ошибке](#)
[Сообщество](#)
[Форум](#)
[Свежие правки](#)
[Новые страницы](#)
[Справка](#)
[Пожертвовать](#)

[Инструменты](#)
[Ссылки сюда](#)
[Связанные правки](#)
[Служебные страницы](#)
[Постоянная ссылка](#)

Вы не представились системе [Обсуждение](#) [Вклад](#) [Создать учётную запись](#) [Войти](#)

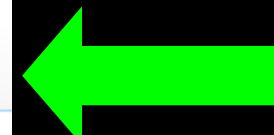
Статья [Обсуждение](#)

[Читать](#)

[Править](#)

[Править код](#)

[История](#)



Регулярные выражения

Материал из Википедии — свободной энциклопедии

[\[править \]](#) [\[править код \]](#)

Регуля́рные выраже́ния (*англ.* *regular expressions*) — формальный язык поиска и осуществления манипуляций с **подстроками** в тексте, основанный на использовании метасимволов (**символов-джокеров**, *англ.* *wildcard characters*). Для поиска используется строка-образец (*англ.* *pattern*, по-русски её часто называют «шаблоном», «маской»), состоящая из символов и метасимволов и задающая правило поиска. Для манипуляций с текстом дополнительно задаётся строка замены, которая также может содержать в себе специальные символы.

Содержание [\[скрыть\]](#)

- 1 Возможности
- 2 История
- 3 В теории формальных языков
- 4 Синтаксис
 - 4.1 Представление символов
 - 4.1.1 Обычные символы (литералы) и специальные символы (метасимволы)
 - 4.1.2 Любой символ
 - 4.1.3 Символьные классы (наборы символов)
 - 4.2 Позиция внутри строки
 - 4.3 Специальные символы
 - 4.4 Обозначение группы
 - 4.5 Перечисление
 - 4.6 Квантификация (поиск последовательностей)

«Умные» функции «Искать» / «Поиск»

О регулярных выражениях

- Очень мощные и немного загадочные
- Забавные, как только поймешь их
- Регулярные выражения сами по себе напоминают язык программирования
- Пишутся с помощью специальных символов
- Компактные, это своего рода язык «старой школы»

Во время изучения
чего-то нового,
я самозабвенно
выдумываю
невероятные
ситуации, в которых
это умение поможет
мне спасти мир

О нет! Убийца должно
быть последовал
за ней в отпуск!

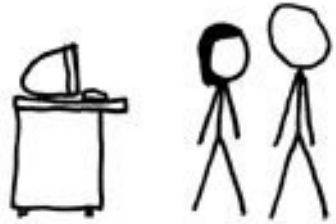


Но чтобы узнать где он, нам нужно
прочитать 200 Мб писем в поисках
чего-то похожего по формату с адресом!

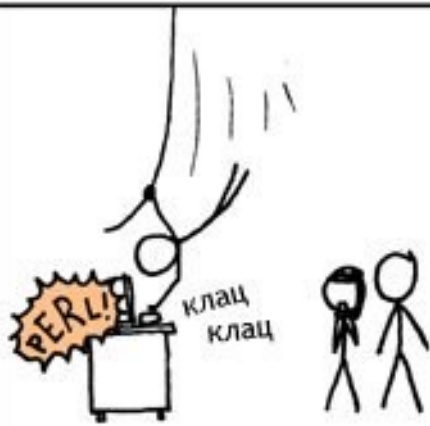


Это безнадежно!

Всем расступиться



Я знаю регулярные
выражения



<http://xkcd.com/208/>

Регулярные выражения: краткое РУКОВОДСТВО

<code>^</code>	Начало всего текста или начало строки текста
<code>\$</code>	Конец всего текста или конец строки текста
<code>.</code>	Один любой символ, кроме новой строки <code>\n</code>
<code>\s</code>	Любой пробельный символ
<code>\S</code>	Любой непробельный символ
<code>*</code>	Повторяет символ ноль или более раз
<code>*?</code>	Повторяет символ ноль или более раз (не жадный квантификатор)
<code>+</code>	Повторяет символ ноль или более раз
<code>+?</code>	Повторяет символ ноль или более раз (не жадный квантификатор)
<code>[aeiou]</code>	Любой из символов, перечисленных в наборе
<code>[^XYZ]</code>	Любой символ, не указанный в данном наборе
<code>[a-z0-9]</code>	Набор символов может включать диапазон
<code>(</code>	Указывает начало извлечения строки
<code>)</code>	Указывает конец извлечения строки

Модуль регулярных выражений

- Прежде чем вы сможете использовать в своей программе регулярные выражения, необходимо импортировать библиотеку, используя команду `import re`
- Используя `re.search()`, можно проверить, соответствует ли строка регулярному выражению, аналогично использованию метода `find()` для строк
- Вы можете использовать `re.findall()` для извлечения частей строки, которые соответствуют регулярному выражению, аналогично комбинации метода `find()` и среза: `var[5:10]`

Использование `re.search()`, как `find()`

```
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if line.find('From:') >= 0:
        print(line)
```

```
import re

hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('From:', line) :
        print(line)
```

Использование `re.search()`, как `startswith()`

```
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if line.startswith('From:') :
        print(line)
```

```
import re

hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('^From:', line) :
        print(line)
```

Мы гибко настраиваем то, что нужно найти, добавляя специальные символы в строку

Метасимволы (символы-джокеры)

- Символ **.** (точка) означает один любой символ
- Символ ***** (звездочка) означает «ноль или более повторений»

```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-DSPAM-Confidence: 0.8475
X-Content-Type-Message-Body: text/plain
```

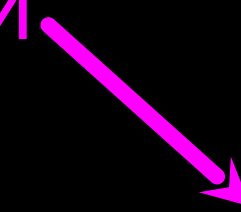
Начало строки

Множеств

о раз

^ X . * :

Любой символ



Тонкая настройка соответствия

В зависимости от «чистоты» данных и целей вашего приложения, вам может понадобиться немного сузить диапазон соответствия

```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-Plane is behind schedule: two weeks
X-: Very short
```

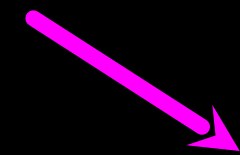
Начало строки

Множеств

0 раз

^ X . * :

Любой символ




Тонкая настройка соответствия

В зависимости от «чистоты» данных и целей вашего приложения, вам может понадобиться немного сузить диапазон соответствия

```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-: Very Short
X-Plane is behind schedule: two weeks
```

Начало строки

Один или более раз

^X-\S+:

Любой непробельный символ

Сопоставление и извлечение данных

- `re.search()` возвращает значение True/False в зависимости от того, соответствует ли строка регулярному выражению
- Если необходимо извлечь совпадающие строки, используем `re.findall()`

`[0-9]+`



Одна или более цифр

```
>>> import re
>>> x = '2 моих любимых числа - 19 и 42'
>>> y = re.findall('[0-9]+', x)
>>> print(y)
['2', '19', '42']
```

Сопоставление и извлечение данных

`re.findall()` возвращает список из нуля или более подстрок, соответствующих регулярному выражению

```
>>> import re
>>> x = '2 моих любимых числа - 19 и 42'
>>> y = re.findall('[0-9]+', x)
>>> print(y)
['2', '19', '42']
>>> y = re.findall('[AEIOU]+', x)
>>> print(y)
[]
```

Осторожно: Жадные квантификаторы

Квантификаторы (* и +) называют «жадными», так как в некоторых реализациях регулярным выражениям с ними соответствует максимально длинная строка из возможных

```
>>> import re
>>> x = 'From: Using the : character'
>>> y = re.findall('^F.+:', x)
>>> print(y)
['From: Using the :']
```

Почему не
просто 'From:' ?

Первый символ
совпадения - буква F

Один или более
СИМВОЛОВ

^F

.

+

:

Последний
символ
совпадения - :

Ленивые квантификаторы

Но не все квантификаторы регулярных выражений жадные!

Добавьте символ `?`, это немного охладит пыл `+` и `*`...

```
>>> import re
>>> x = 'From: Using the : character'
>>> y = re.findall('^F.+?:', x)
>>> print(y)
['From:']
```

Первый символ
совпадения - буква F

Последний
символ
совпадения - :

Один или более
символов, но
минимально
возможное
количество

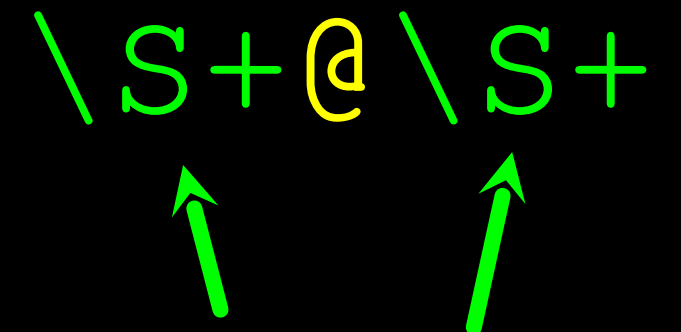
`^F` . `+?` `:`

Тонкая настройка извлечения строк

Вы можете точнее настроить поиск совпадения для `re.findall()` и отдельно указать, какая часть совпадения должна быть извлечена, используя круглые скобки

From `stephen.marquard@uct.ac.za` Sat Jan 5 09:14:16 2008

```
>>> y = re.findall('\S+@\S+', x)
>>> print(y)
['stephen.marquard@uct.ac.za']
```



The diagram shows the regular expression `\S+@\S+` in green. Two green arrows point upwards from the text below to the `\S+` parts of the expression, indicating that these parts are the ones being extracted.

Как минимум
один
непробельный
символ

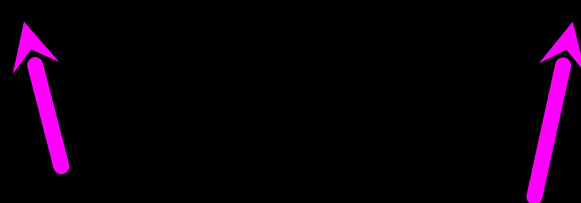
Тонкая настройка извлечения строк

Круглые скобки не являются частью совпадения, они лишь сообщают, где **начинается** и **заканчивается** извлечение строки

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

```
>>> y = re.findall('\S+@\S+', x)
>>> print(y)
['stephen.marquard@uct.ac.za']
>>> y = re.findall('^From (\S+@\S+)', x)
>>> print(y)
['stephen.marquard@uct.ac.za']
```

^From (**\S+@\S+**)



The diagram shows the text '^From (\S+@\S+)' with two blue arrows pointing upwards. The first arrow points to the start of the match, which is the character '^'. The second arrow points to the end of the match, which is the closing parenthesis ')' of the group '(\S+@\S+)'. This illustrates that the parentheses in the regex are used to define the boundaries of the match, but they are not part of the actual extracted string.

Примеры анализа строк

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
>>> data = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
>>> atpos = data.find('@')
>>> print(atpos)
21
>>> sppos = data.find(' ', atpos)
>>> print(sppos)
31
>>> host = data[atpos+1 : sppos]
>>> print(host)
uct.ac.za
```

Извлечение
имени хоста,
используя метод
find и срез
строки

Шаблон двойного разделения

Иногда бывает необходимо сначала разделить строку одним образом, а затем взять один из получившихся кусков и разделить его ещё раз

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

```
words = line.split()  
email = words[1]  
pieces = email.split('@')  
print(pieces[1])
```

```
stephen.marquard@uct.ac.za  
['stephen.marquard', 'uct.ac.za']  
'uct.ac.za'
```

Версия с регулярным выражением

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

```
import re  
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'  
y = re.findall('@([ ^ ]*)', lin)  
print(y)
```

```
['uct.ac.za']
```

'@([^]*)'

Просматривать строку пока не встретится символ

@

Версия с регулярным выражением

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

```
import re
```

```
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
```

```
y = re.findall('@([ ^ ]*)', lin)
```

```
print(y)
```

```
['uct.ac.za']
```

'@([^]*)'

Захватить непробельные
СИМВОЛЫ

Ноль или более
СИМВОЛОВ

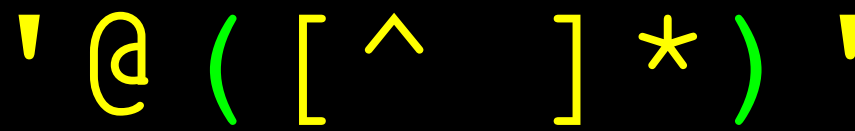
Версия с регулярным выражением

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

```
import re  
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'  
y = re.findall('@([ ^ ]*)', lin)  
print(y)
```

```
['uct.ac.za']
```

'@([^]*)'



Извлечь непробельные символы

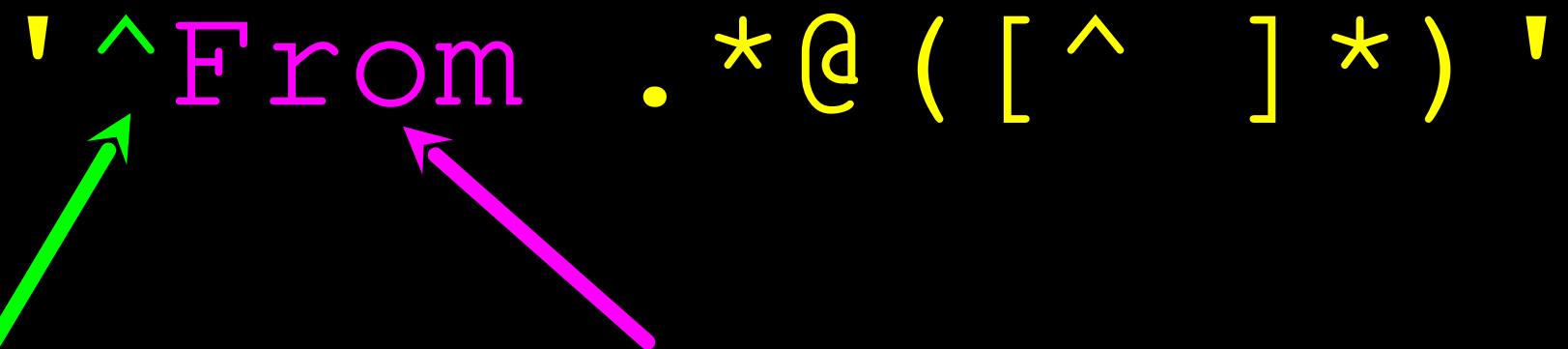
Сделаем еще круче

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
y = re.findall('^From .*@([ ^]*)', lin)
print(y)
```

```
['uct.ac.za']
```

'^From .*@([^]*)'

A diagram illustrating the regex pattern '^From .*@([^]*)'. The pattern is written in yellow text. A green arrow points from the '^' character to the text 'Начиная с начала строки, ищем подстроку 'From '' below. A purple arrow points from the 'From' text to the same text below.

Начиная с начала строки, ищем подстроку 'From '

Сделаем еще круче

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
y = re.findall('^From .*@([ ^]*)', lin)
print(y)
```

```
['uct.ac.za']
```

'^From .*@([^]*)'

Пропустим часть символов, пока не встретим символ @

Сделаем еще круче

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

```
import re
```

```
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
```

```
y = re.findall('^From .*@([ ^ ]*)', lin)
```

```
print(y)
```

```
['uct.ac.za']
```

```
'^From .*@([ ^ ]*)'
```

Начало извлечения

Сделаем еще круче

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
y = re.findall('^From .*@([ ^ ]*)', lin)
print(y)
```

['uct.ac.za']

'^From .*@([^]*)'

Захватить
непробельные символы

Захватить их как
можно больше

Сделаем еще круче

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

```
import re  
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'  
y = re.findall('^From .*@([ ^ ]*)', lin)  
print(y)
```

```
['uct.ac.za']
```

```
'^From .*@([ ^ ]+)'
```



Конец извлечения

Проверка на спам

```
import re
hand = open('mbox-short.txt')
numlist = list()
for line in hand:
    line = line.rstrip()
    stuff = re.findall('^X-DSPAM-Confidence: ([0-9.]*)', line)
    if len(stuff) != 1: continue
    num = float(stuff[0])
    numlist.append(num)
print('Maximum:', max(numlist))
```

X-DSPAM-Confidence: 0.8475

python ds.py

Maximum: 0.9907

Резюме

- Регулярные выражения — загадочный, но очень мощный язык для сопоставления строк и извлечения элементов из этих строк
- Регулярные выражения содержат специальные символы, которые являются управляющими конструкциями



Авторы / Благодарности



... Insert new Contributors and Translations here

Авторские права на эти слайды принадлежат Чарльзу Р. Северансу (www.dr-chuck.com), 2010 г., Школе Информации Мичиганского Университета и open.umich.edu, и доступны по лицензии Creative Commons Attribution 4.0 License.

Пожалуйста, сохраняйте этот слайд во всех копиях этого документа, в соответствии с требованиями Лицензии. Если вы внесли изменения, добавьте свое имя или организацию в список участников на этой странице.

Исходная разработка: Чарльз Северанс, Школа Информации Мичиганского Университета.

Перевод выполнила Фомкина Виолетта.

... Добавьте сюда новых авторов и переводчиков