

Регулярные выражения

Регулярные выражения

В компьютерной терминологии «регулярное выражение» (его еще называют *регехр* или *регех*, 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

О регулярных выражениях

- Очень мощные
- Регулярные выражения сами по себе напоминают язык программирования
- Пишутся с помощью специальных символов

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

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

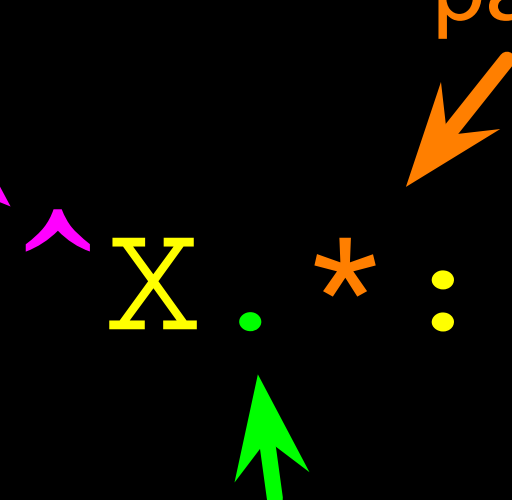
```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-DSPAM-Confidence: 0.8475
X-Content-Type-Message-Body: text/plain
```

Начало строки

Множество

раз

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

Множество

раз

Любой символ

X.*:

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)
[]
```

Жадные квантификаторы

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

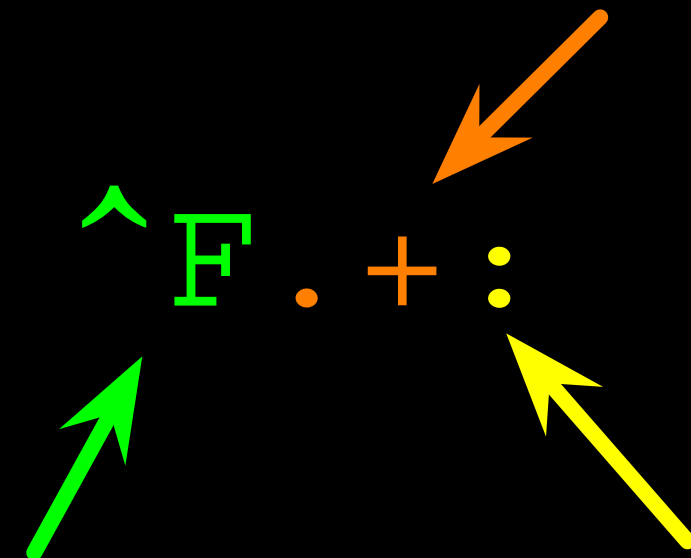
```
>>> import re
>>> x = 'From: Using the : character'
>>> y = re.findall('^F.+:', x)
>>> print(y)
['From: Using the :']
```

Почему не просто
'From:' ?

Первый символ
совпадения - буква F

Последний символ
совпадения - :

Один или более
СИМВОЛОВ



Ленивые квантификаторы

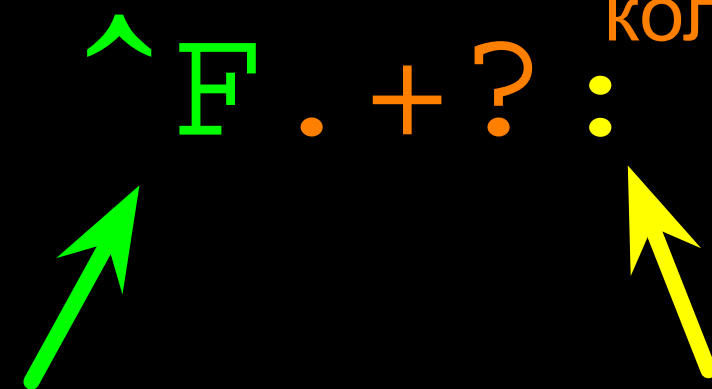
Но не все квантификаторы регулярных выражений жадные!
Добавьте символ `?`, это немного охладит пыл `+` и `*`...

```
>>> 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

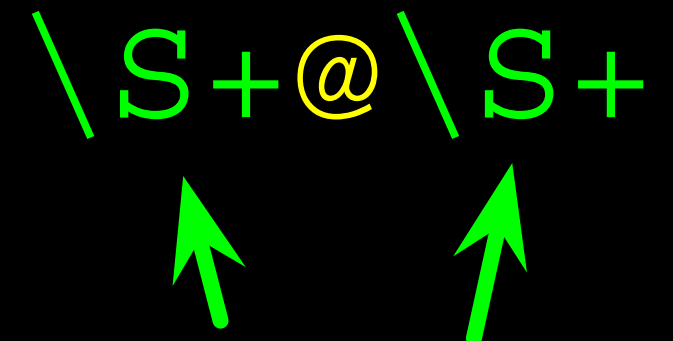


Тонкая настройка извлечения строк

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. The first arrow points to the `\S+` before the `@`, and the second arrow points to the `\S+` after the `@`.

Как минимум
один
непробельный
СИМВОЛ

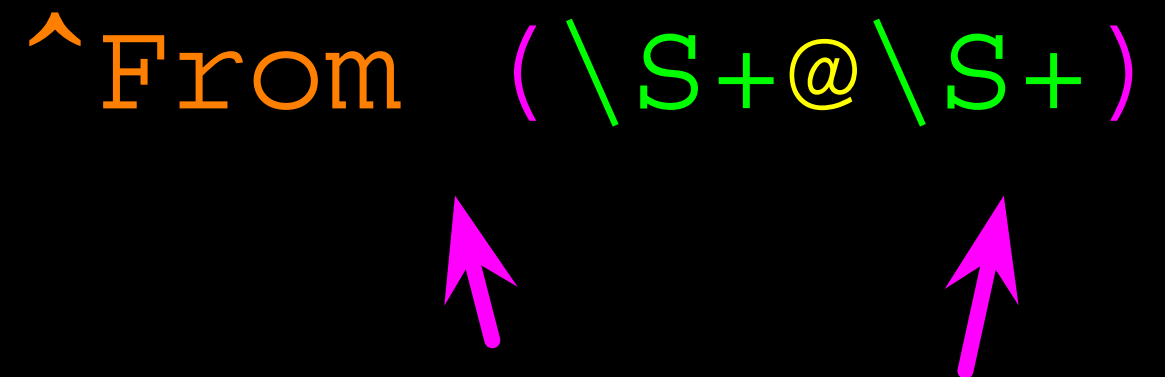
Тонкая настройка извлечения строк

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 regex pattern `^From (\S+@\S+)` in orange. A pink arrow points to the start of the match, which is the character `^`. Another pink arrow points to the end of the match, which is the closing parenthesis `)`. The text `From` is in orange, and the parentheses and the email address `\S+@\S+` are in green.

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('@([^\s]*)', lin)
```

```
print(y)
```

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

```
'@([^\s]*)'
```



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('@([^\s]*)', lin)
print(y)
```

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

'@([^\s]*)'

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('@([^\s]*)', lin)
print(y)
```

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

'@([^\s]*)'



Извлечь пробельные символы

Или так...

```
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 .*@([^]*)'. A green arrow points from the start of the string to the '^' character. A purple arrow points from the word 'From' to the 'F' character.

Начиная с начала строки, ищем подстроку 'From'

Или так...

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

```
import re
```

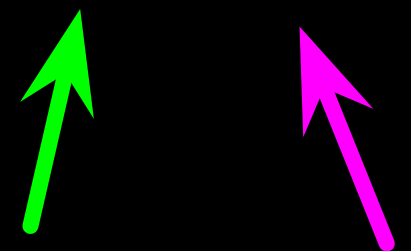
```
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
```

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

```
print(y)
```

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

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



Пропустим часть символов, пока не встретим символ @

Или так...

```
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

Или так...

```
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

Или так...

```
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



Экранирование символа

Чтобы **отменить (экранировать)** специальное значение символа регулярного выражения, поставьте перед ним обратную косую черту `\`

```
>>> import re
>>> x = 'Мы только что получили $10.00 за
печенье.'
>>> y = re.findall('\$[0-9.]+', x)
>>> print(y)
['$10.00']
```

