Регулярные выражения

Глава 11

Пайтон для всех
www.py4e.com

Регулярные выражения

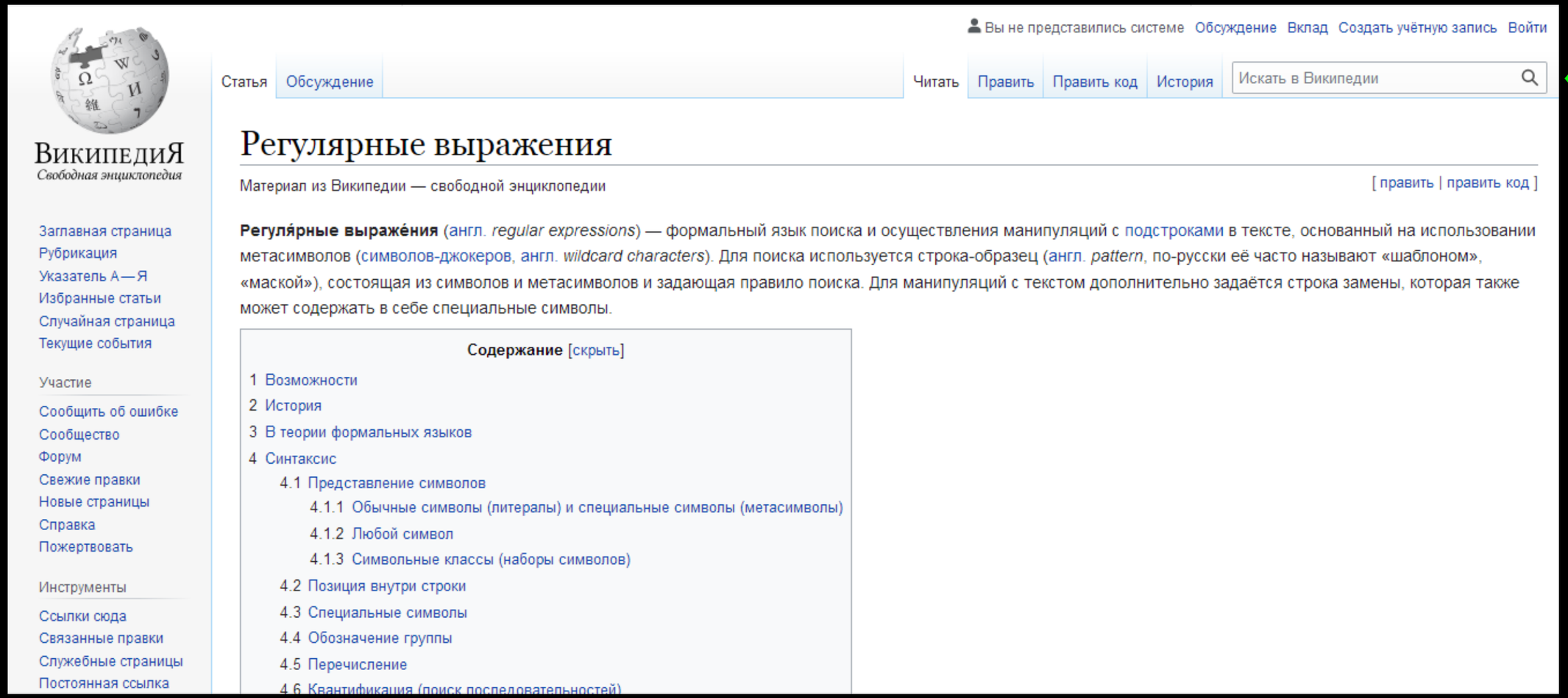
В компьютерной терминологии «регулярное выражение» (его еще называют *regexp* или *regex, сокр. «регулярка»*) — мощное и гибкое средство для сопоставления строк текста, например, определенных символов, слов или наборов символов. Регулярное выражение написано на формальном языке, который может интерпретироваться обработчиком регулярных выражений

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

Регулярные выражения

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

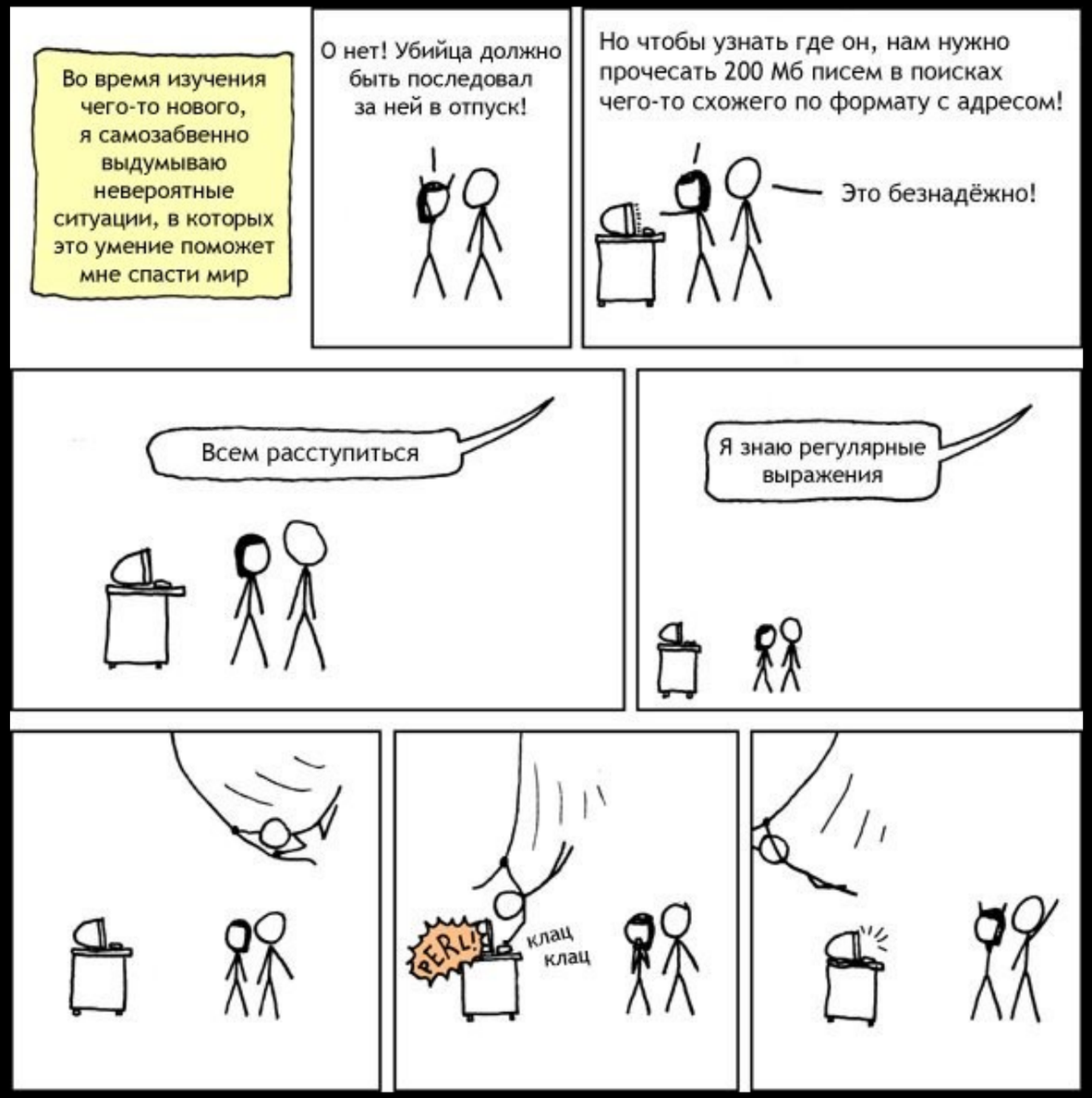
https://ru.wikipedia.org/wiki/Регулярные_выражения



«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»



http://xkcd.com/208/

Регулярные выражения: краткое руководство

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

https://www.py4e.com/lectures3/Pythonlearn-11-Regex-Handout.txt

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)
```

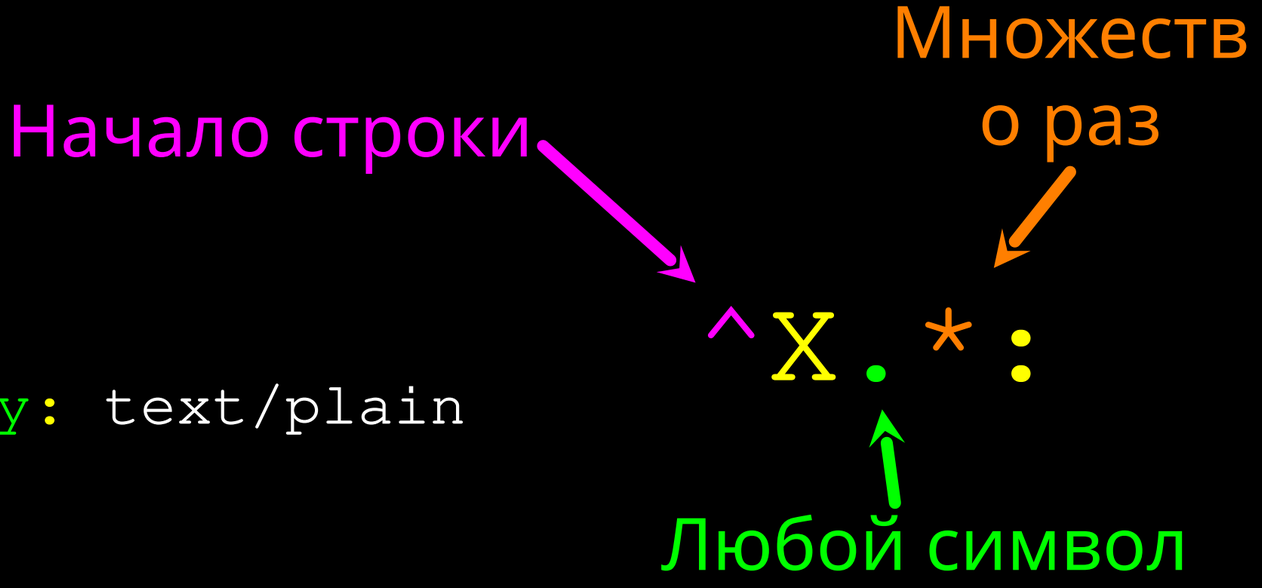
```
import re

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»

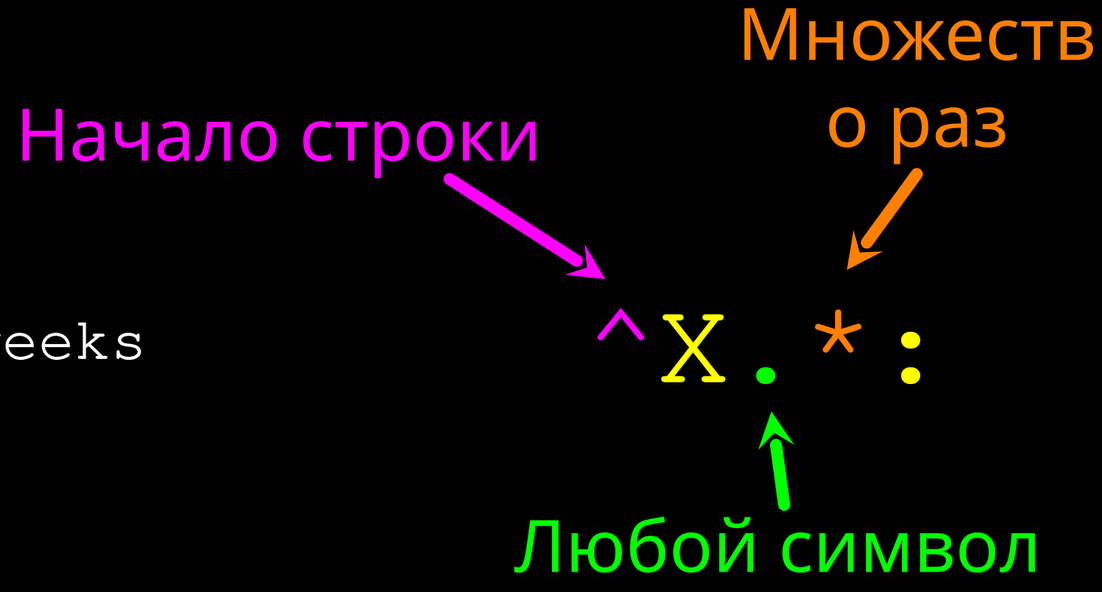
```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-DSPAM-Confidence: 0.8475
X-Content-Type-Message-Body: text/plain
```



Тонкая настройка соответствия

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

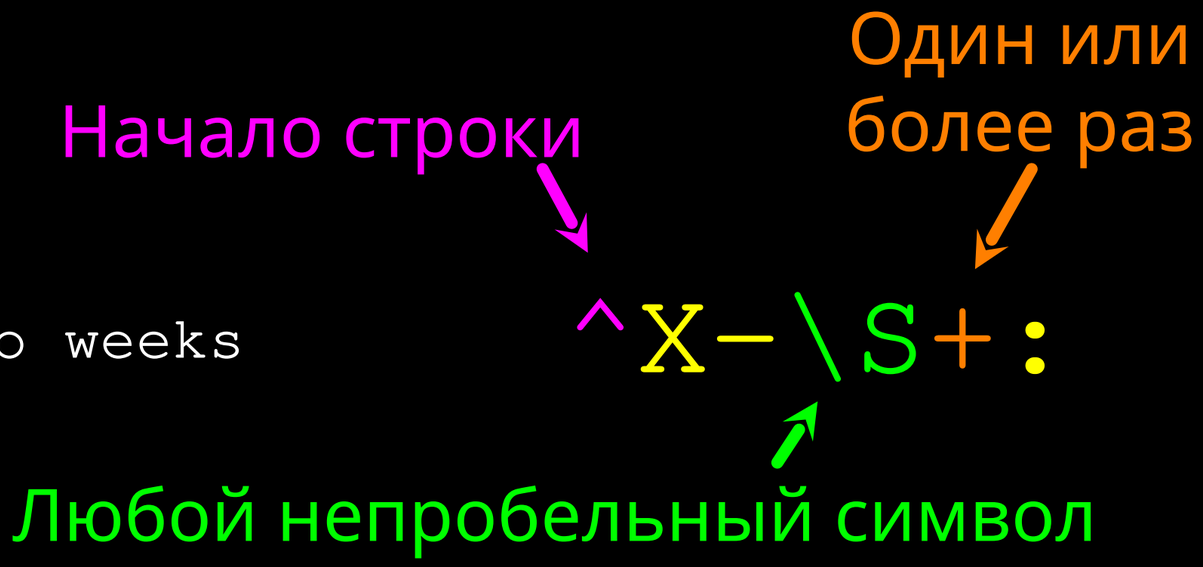
```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-Plane is behind schedule: two weeks
X-: Very short
```



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
```



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 :']
```

Один или более символов

^F.+:

Первый символ совпадения - буква F

Последний символ совпадения - :

Почему не просто 'From:' ?

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:']
```

^F.+?:

Один или более символов, но минимально возможное количество

Первый символ совпадения - буква F

Последний символ совпадения - :

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']
```

\S+@\S+

Как минимум один непробельный символ

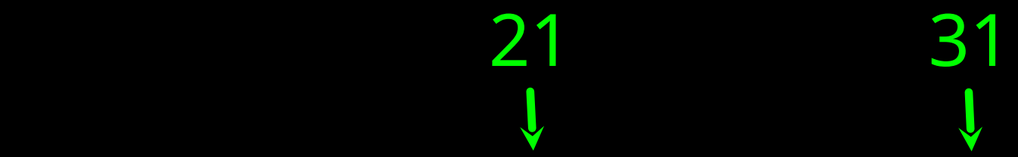
Тонкая настройка извлечения строк

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']
```

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
```

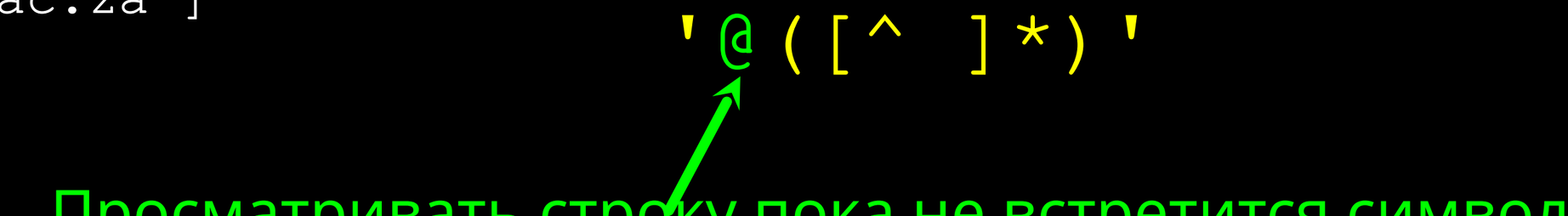
```
words = line.split()
email = words[1]
pieces = email.split('@')
print(pieces[1])
```

```
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
```

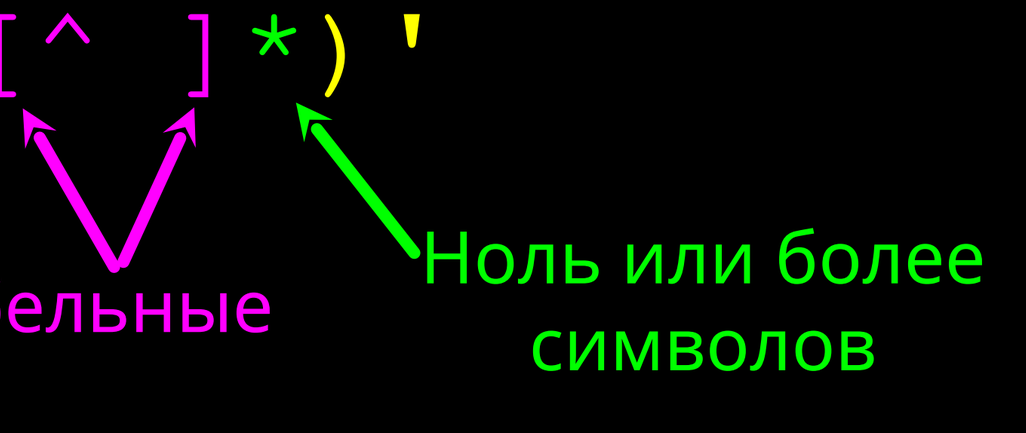
```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('@([^ ]*)',lin)
print(y)

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



Версия с регулярным выражением

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

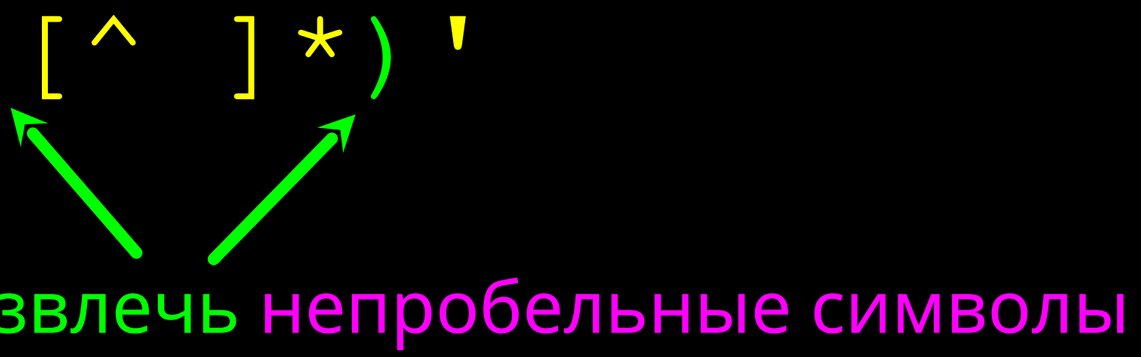
```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('@([^ ]*)',lin)
print(y)

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



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']
```



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']
```

Сделаем еще круче

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

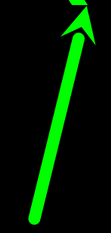
```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('^From .*@([^ ]*)',lin)
print(y)

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

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

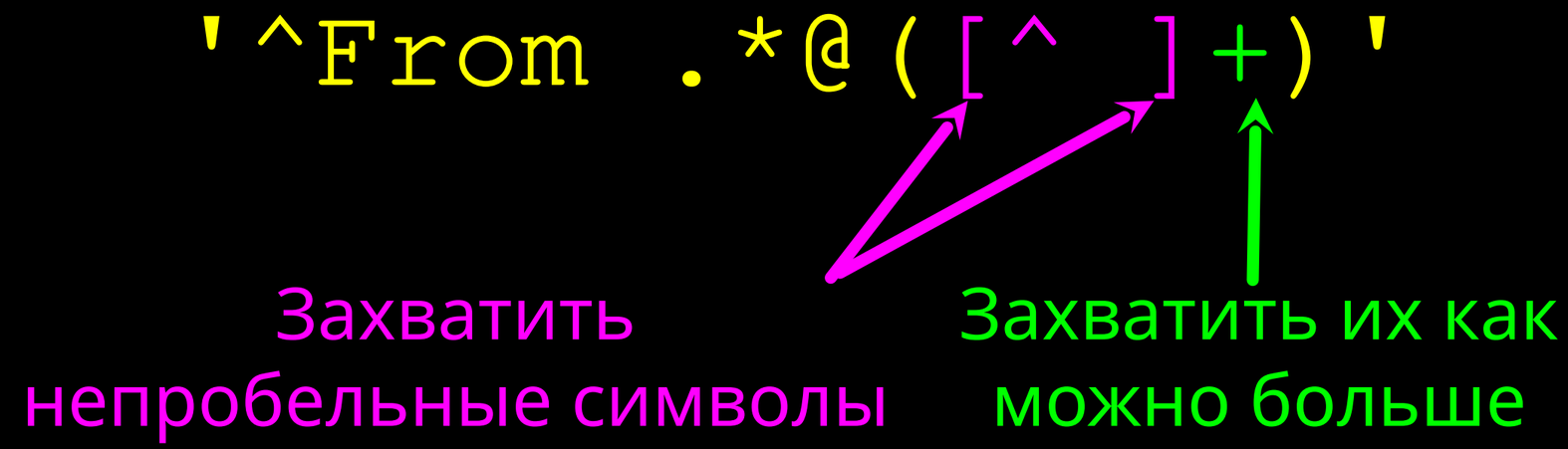


Начало извлечения

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']
```

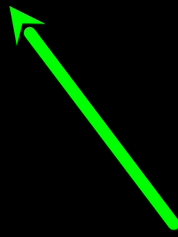


Сделаем еще круче

```
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
```

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']
```

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

Исходная разработка: Чарльз Северанс, Школа Информации Мичиганского Университета.

Перевод выполнила Фомкина Виолетта.

… Добавьте сюда новых авторов и переводчиков

...