Регулярные выражения

Глава 11

Python for Informatics: Exploring Information
www.pythonlearn.com

Регулярные выражения

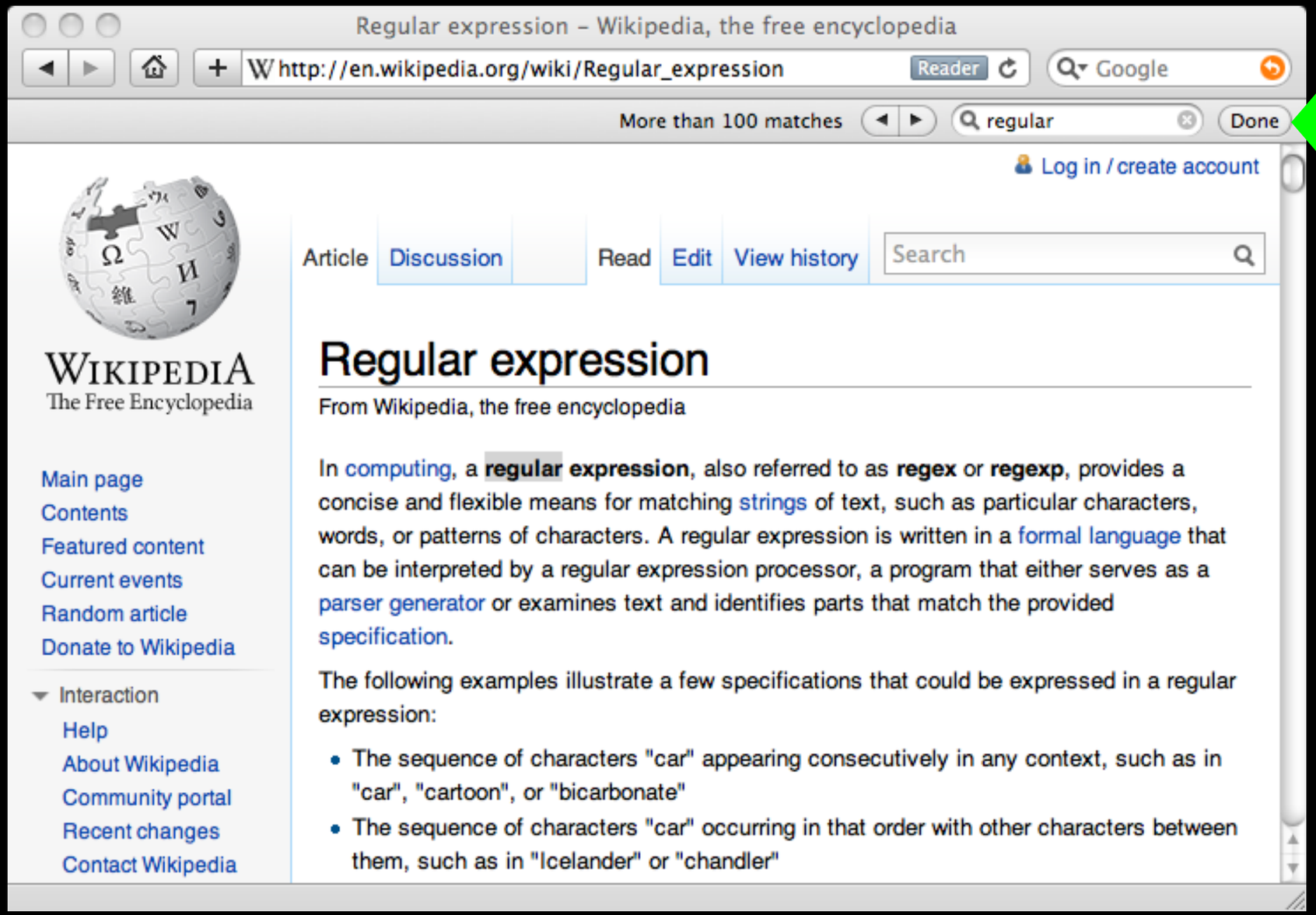
В вычислительной технике регулярное выражение (также "regex" или "regexp") представляет собой краткий и гибкий способ поиска строк текста, таких как конкретные символы, слова или набор символов. Регулярное выражение записывается на формальном языке, понятном программе-интерпретатору регулярных выражений.

http://en.wikipedia.org/wiki/Regular_expression

Регулярные выражения

Рациональное выражение с "шаблоном" для поиска и разбора строк

http://en.wikipedia.org/wiki/Regular_expression



Рациональный поиск

Понимание регулярных выражений

- Очень мощный и достаточно зашифрованный инструмент
- Интересный для тех, кто его понимает
- Регулярные выражения - это отдельный язык
- Язык "шаблонных" символов - программирование с символами
- Это своего рода “традиционный” (компактный) язык

http://xkcd.com/208/

Краткое руководство

^	Начало строки
$	Конец строки
.	Любой символ
\s	Пробел
\S	Любой непробельный символ
*	Повторяет символ ноль или более раз
*?	Повторяет символ ноль или более раз (нежадное совпадение)
+	Повторяет символ один или более раз
+?	Повторяет символ один или более раз (нежадное совпадение)
[aeiou]	Любой один из перечисленных символов
[^XYZ]	Любой один символ, кроме перечисленных
[a-z0-9]	Набор символов может быть указан как диапазон
(	Указывает начало извлечения строки
)	Указывает конец извлечения строки

Модуль регулярных выражений

- Перед тем как начать использовать регулярные выражения в программе, необходимо импортировать соответствующую библиотеку с помощью команды''import re''
- Чтобы проверить, соответствует ли строка регулярному выражению, можно ввести re.search(). Эта команда аналогична использованию метода find() для строк
- Чтобы извлечь часть строки, можно использовать re.findall(). Этот метод подобен использованию функции find() со срезом строки 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 настроили поиск, добавив к строке специальные символы

Шаблонные символы

- Точка соответствует любому символу
- Если добавить звездочку, то символ может повторяться “любое количество раз”

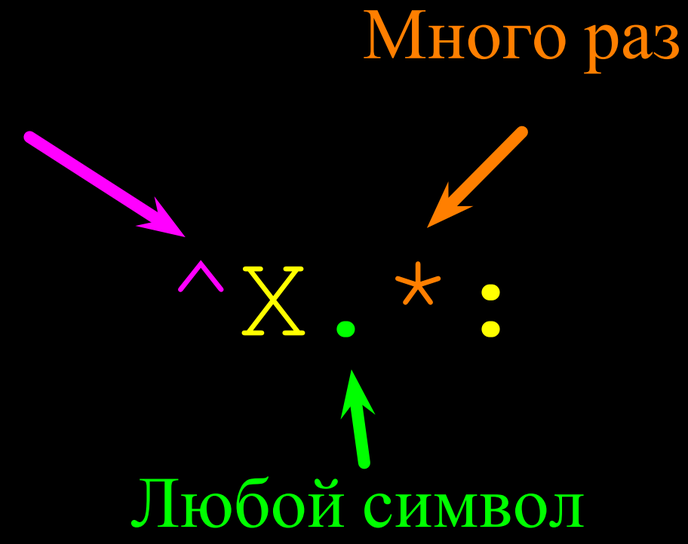
```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-DSPAM-Confidence: 0.8475
X-Content-Type-Message-Body: text/plain
```

```
^X.*:
```

Шаблонные символы

- Точка соответствует любому символу
- Если добавить звездочку, то символ может повторяться “любое количество раз”

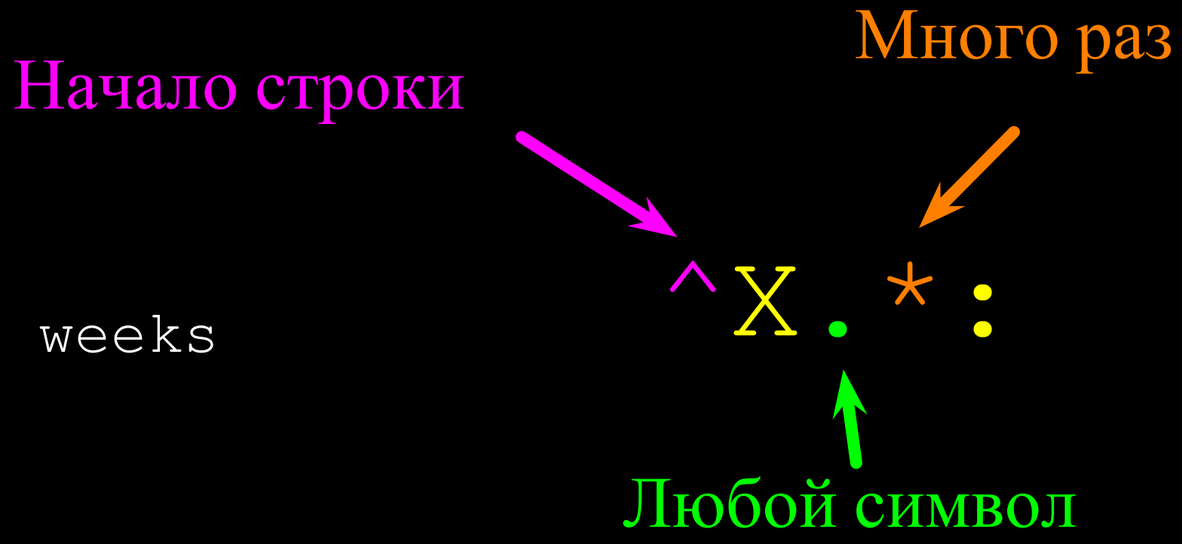
```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-DSPAM-Confidence: 0.8475
X-Content-Type-Message-Body: text/plain
```



Настройка совпадений

- В зависимости от формата данных и цели вашей программы, вам может потребоваться сузить поиск

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



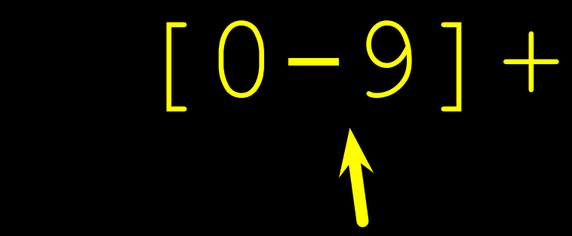
Настройка совпадений

- В зависимости от формата данных и цели вашей программы, вам может потребоваться сузить поиск

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

Поиск и извлечение данных

- Функция re.search() возвращает True (истинно)/False (ложно) в зависимости от соответствия строки регулярному выражению
- Для извлечения строки, соответствующей регулярному выражению, используется функция re.findall()



Одна или несколько цифр

```
>>> import re
>>> x = 'My 2 favorite numbers are 19 and 42'
>>> y = re.findall('[0-9]+',x)
>>> print y
['2', '19', '42']
```

Поиск и извлечение данных

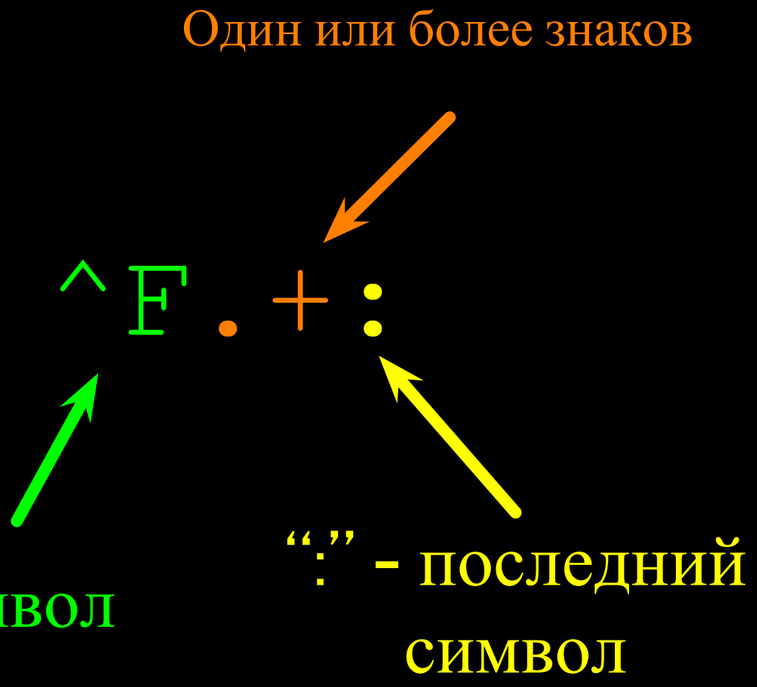
- При использовании функции re.findall() выдается список из нуля или более подстрок, соответствующих регулярному выражению

```
>>> import re
>>> x = 'My 2 favorite numbers are 19 and 42'
>>> y = re.findall('[0-9]+',x)
>>> print y
['2', '19', '42']
>>> y = re.findall('[AEIOU]+',x)
>>> print y
[]
```

Внимание! Жадное совпадение

- Символы повторения (* и +) соответствуют максимально длинной (жадной) строке из возможных

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



Почему не 'From:' ?

Нежадное совпадение

- Не все коды повторения в регулярных выражениях являются жадными! Если к регулярному выражению добавить знак вопроса, то знаки “+” и “*” немного расслабляются...

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

Один или более знаков, но нежадное совпадение

^F.+?:

“F” - первый символ

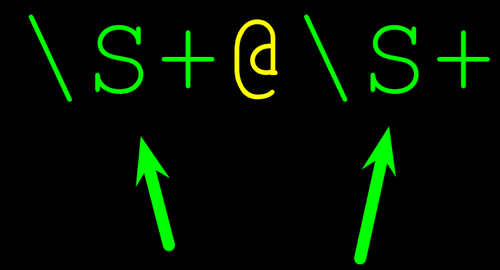
“:” - последний символ

Настройка извлечения строк

- Вы можете настроить поиск re.findall(), при помощи скобок указав, какую часть строки извлекать

```
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']
```



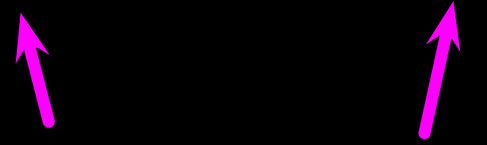
Настройка извлечения строк

- Сами скобки в поиск не включаются. Они только указывают начало и конец части строки для извлечения

```
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+)
```




```
>>> 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 с помощью поиска и среза строки

Двойной срез

- Иногда нам необходимо срезать строку, а затем взять одну из полученных частей и снова ее срезать

```
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]
```

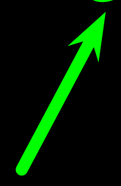
```
stephen.marquard@uct.ac.za
['stephen.marquard', 'uct.ac.za']
'uct.ac.za'
```

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']
```

'@([^]*)'

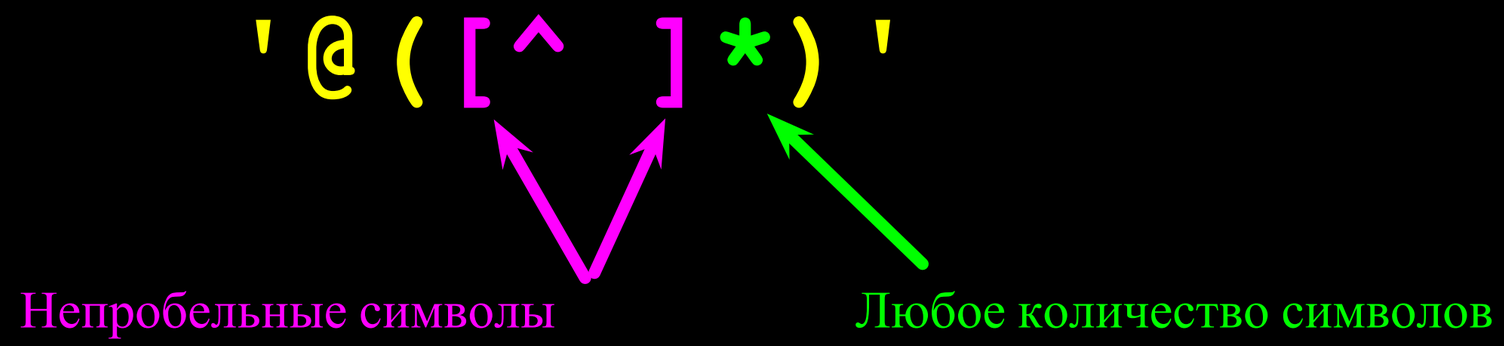


Выполнять поиск, пока не встретится символ “собачки”

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
```

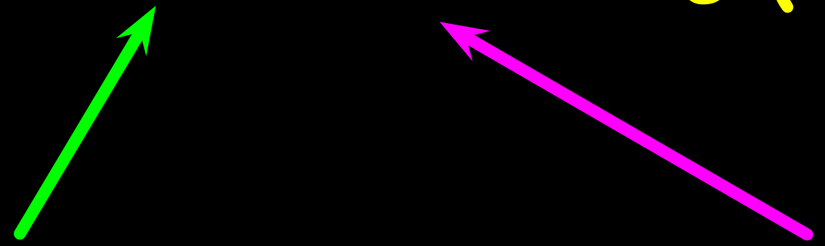
```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('@([^ ]*)',lin)
print y['uct.ac.za']
```

Вариант покруче

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



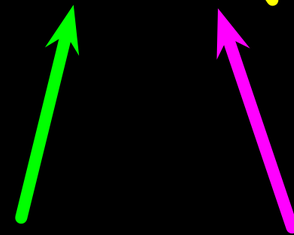
Поиск с начала строки выражения 'From '

Вариант покруче

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



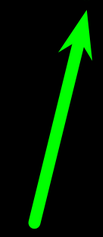
Пропустить все символы, пока не встретится “собачка”

Вариант покруче

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



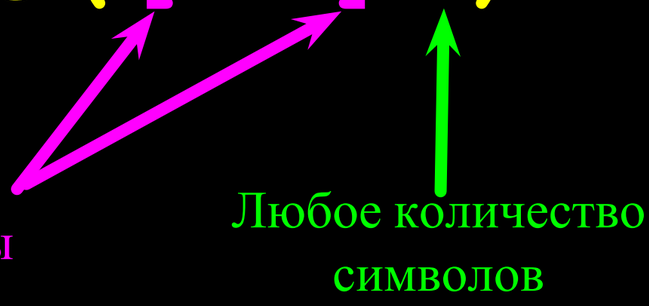
Начать извлечение

Вариант покруче

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

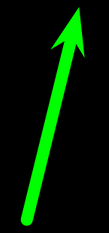


Вариант покруче

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



Остановить извлечение

Spam Confidence

```
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)
```

python ds.py
Maximum: 0.9907

Краткое руководство

^	Начало строки
$	Конец строки
.	Любой символ
\s	Пробел
\S	Любой непробельный символ
*	Повторяет символ ноль или более раз
*?	Повторяет символ ноль или более раз (нежадное совпадение)
+	Повторяет символ один или более раз
+?	Повторяет символ один или более раз (нежадное совпадение)
[aeiou]	Любой один из перечисленных символов
[^XYZ]	Любой один символ, кроме перечисленных
[a-z0-9]	Набор символов может быть указан как диапазон
(	Указывает начало извлечения строки
)	Указывает конец извлечения строки

Символ выхода

- Чтобы использовать знак регулярного выражения в качестве обычного символа, поставьте перед ним наклонную черту '\'

```
>>> import re
>>> x = 'We just received $10.00 for cookies.'
>>> y = re.findall('\$[0-9.]+',x)
>>> print y
['$10.00']
```

Обзор

- Регулярные выражения - это зашифрованный, но мощный язык для поиска строк и извлечения элементов из этих строк
- В регулярных выражениях имеются специальные символы для указания конкретного поиска

Благодарность / Содействие

Данная презентация охраняется авторским правом “Copyright 2010- Charles R. Severance (www.dr-chuck.com) University of Michigan School of Information” open.umich.edu и доступна на условиях лицензии 4.0 “С указанием авторства”. В соответствии с требованием лицензии “С указанием авторства" данный слайд должен присутствовать во всех копиях этого документа. При внесении каких-либо изменений в данный документ вы можете указать свое имя и организацию в список соавторов на этой странице для последующих публикаций.

Первоначальная разработка: Чарльз Северанс, Школа информации Мичиганского университета

Здесь впишите дополнительных авторов и переводчиков...

...