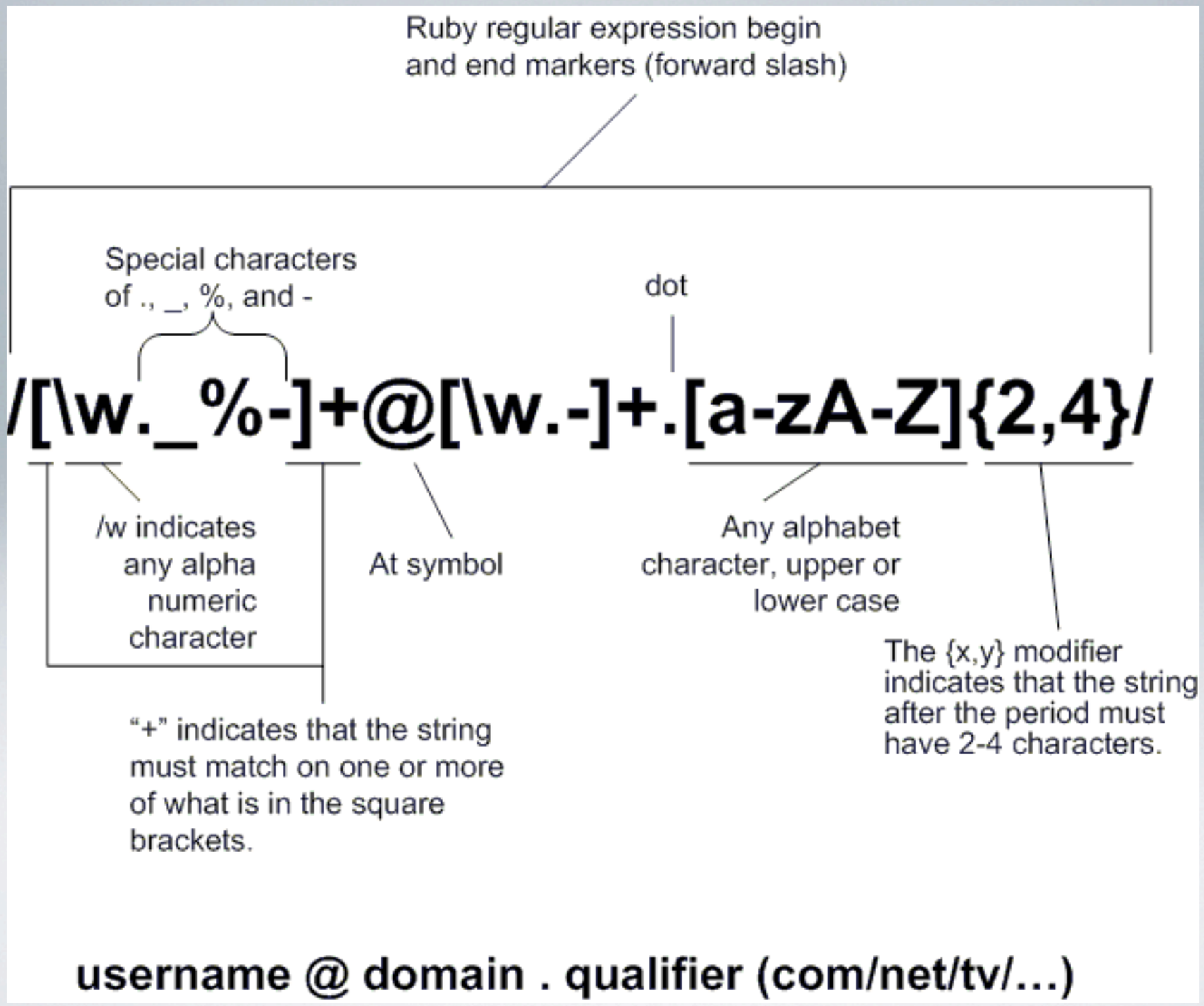
PATTERN MATCHING WITH REGULAR EXPRESSIONS

CS50 Seminar by 2012 TF John Mussman


WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP

THEY LOOK LIKE THIS



WHAT THEY DO

- Protocol for finding patterns in text
- String comparisons, selections, replacements
- Example - find all phone numbers ending in “54” in a directory

ALMOST KINDA LIKE

```
// attempt to read a word from the dictionary
// if we fail, we're either at the end of the file or we erred
if (fscanf(dictionary_file, "%45s", entry->word) != 1)
{
    free(entry);
    break;
}
```

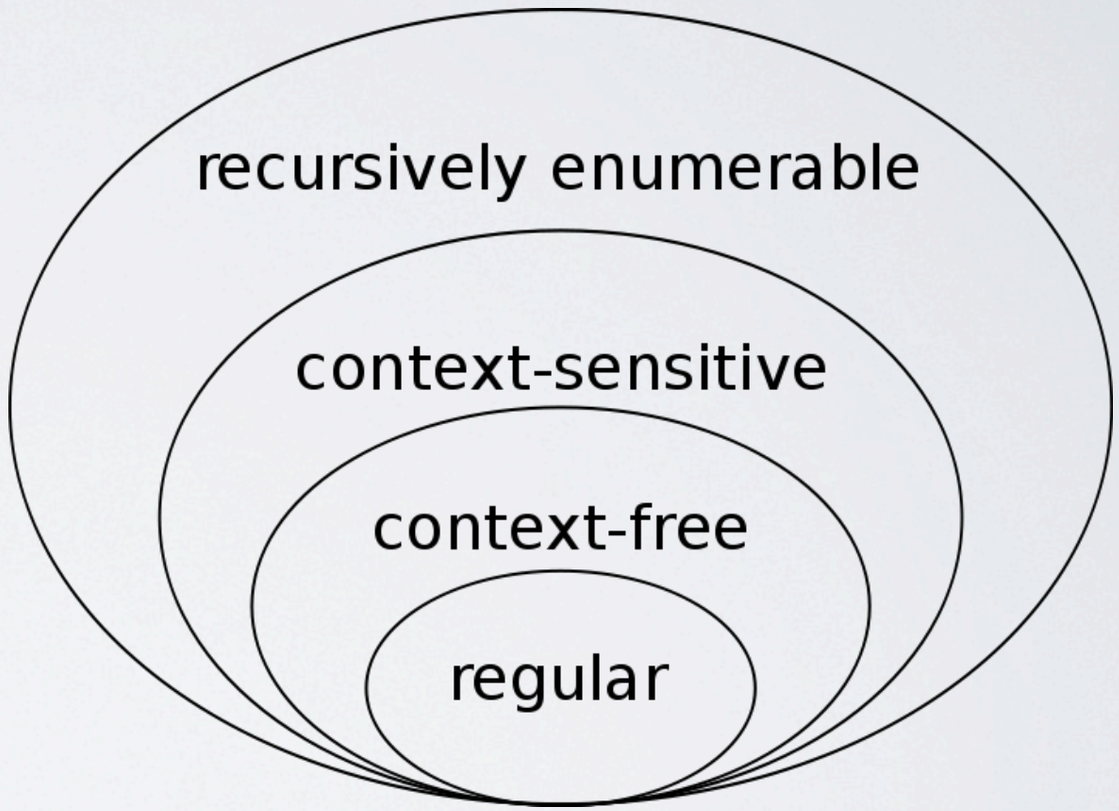
```
// requirements
require("constants.php");
require("functions.php");

// enable sessions
session_start();

// require authentication for most pages
if (!preg_match("{(?:login|logout|register)\.php$}", $_SERVER["PHP_SELF"]))
{
    if (empty($_SESSION["id"]))
    {
        redirect("login.php");
    }
}
```

USE REGEX[P] WHEN

- There is not already a dedicated parser for the language (XML, HTML)
- You're okay with any tradeoffs from using the algorithms - complexity versus accuracy
- The language is regular...enough

REGULARITY IN FORMAL LANGUAGE THEORY

- Regular expressions can be used to define regular languages using union, concatenation, and the Kleene star (*)
- Finite languages
- Star-free languages
- Cyclic languages



Chomsky hierarchy

A GLORIOUS HISTORY

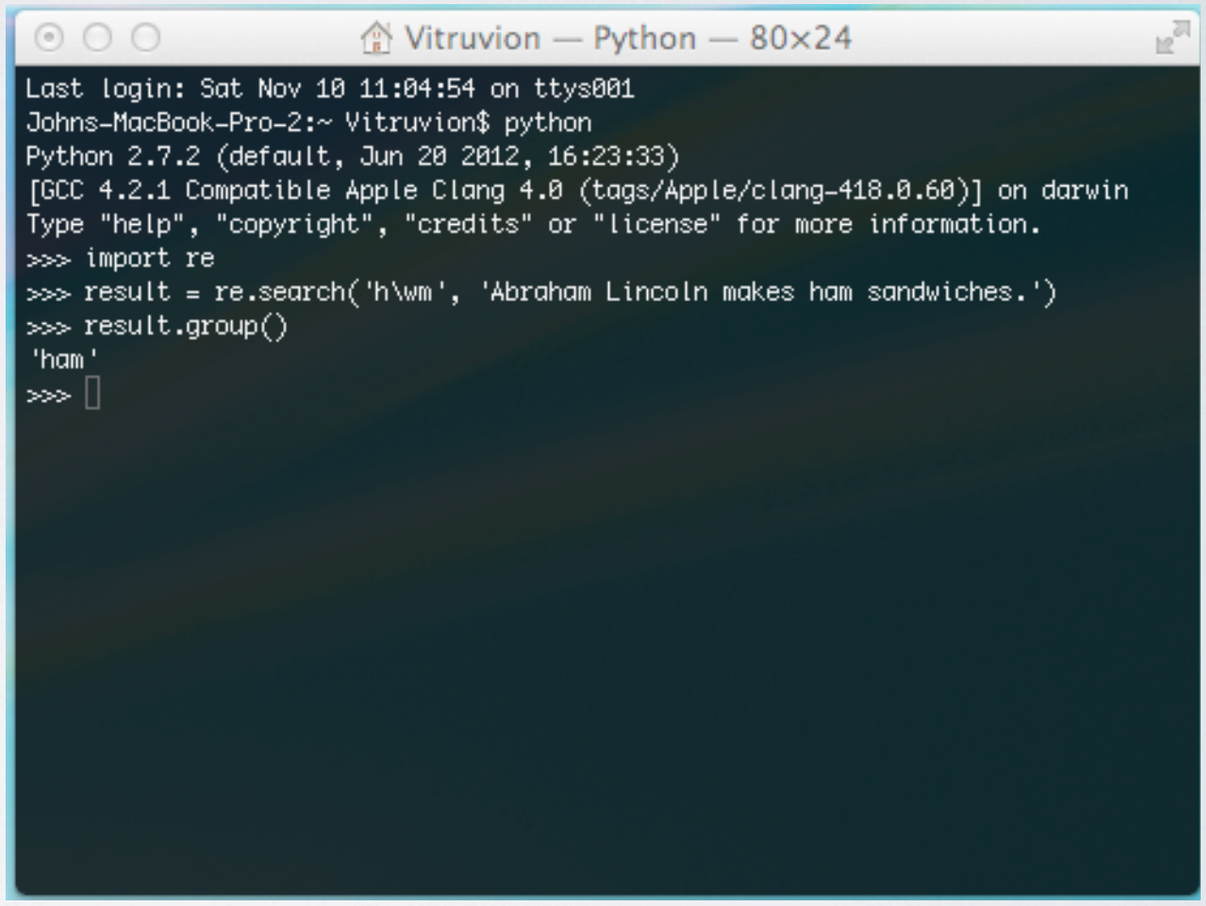
- 1950s - regular sets [Kleene]
- 1960s - grep [Thompson]
- 1980s - Perl [Wall]
- more recently - PCRE
- 2008 - first National Regular Expressions Day

ALGORITHMS

- Construct or simulate deterministic finite automaton (DFA) for regular expression of size *m* and string of size *n*
  - Construct: build phase $O(2^m)$ and run phase $O(n)$
  - Simulate: build phase implicit and run phase $O(m^2n)$
- Backtracking: depth-first recursive search for partial solutions - potentially slower (exponential), but greater flexibility
- Regular Expressions Denial of Service attacks exploit the potentially unbounded complexity of some regexen

PYTHON RE LIBRARY

- Regex is built into Python
- Python is pre-loaded on Macs and available at http://www.python.org/getit/
- Manual - docs.python.org/2/library/re.html
- Cheat sheet - https://github.com/tartley/python-regex-cheatsheet/blob/master/cheatsheet.rst



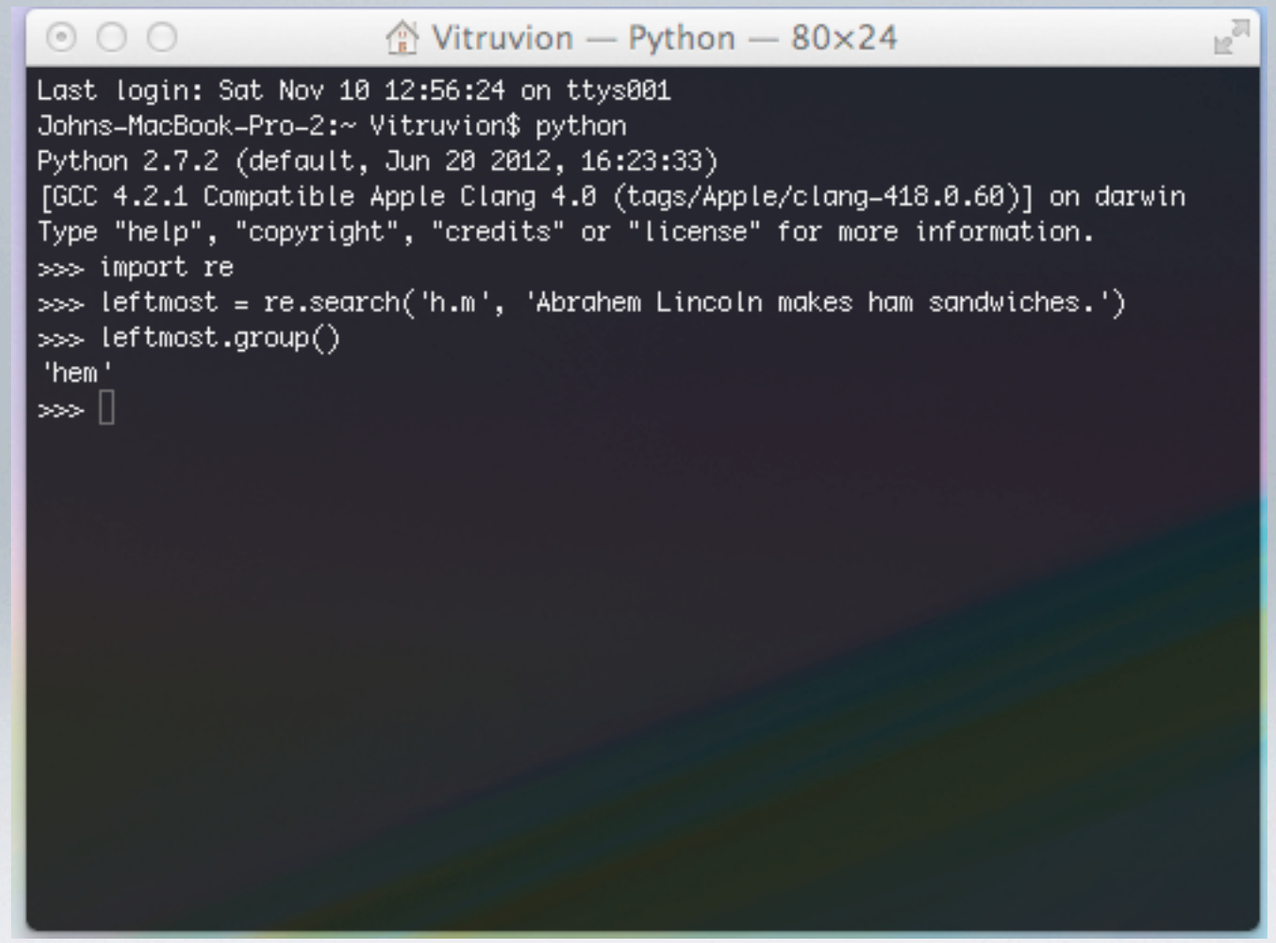
BASICS

- re.search()
- variable.group()
- r before strings

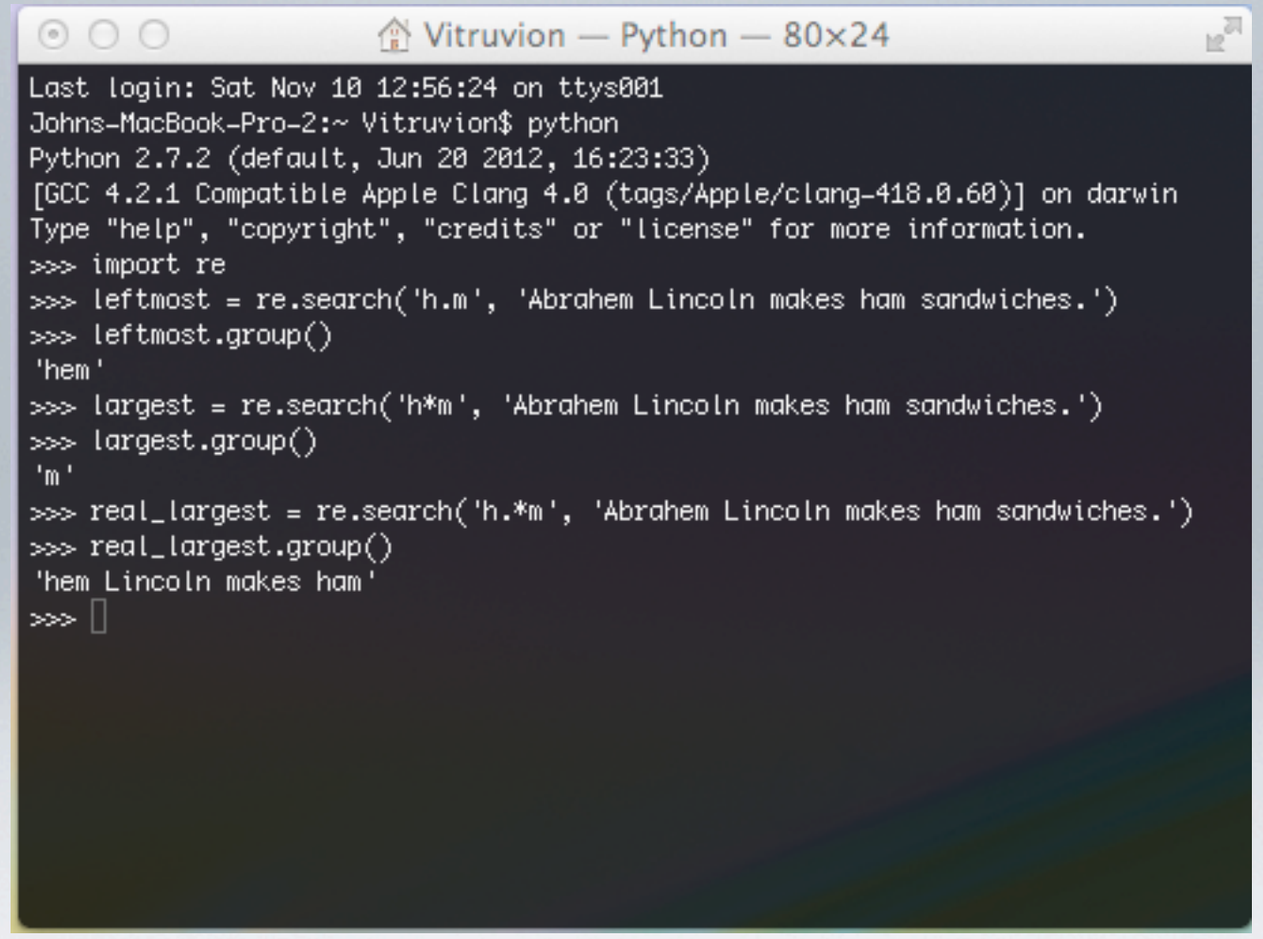
REPETITIVE PATTERNS

- '**ab***' a followed by any number N of b
- '**ab+**' a followed by N > 0 of b
- '**ab**?' a followed by 0 or 1 of b
- '**ab{N}**' a followed by N of b
- '**ab{M,N}**' a followed by between M and N of b

LEFTMOST


Vitruvion — Python — 80×24
Last login: Sat Nov 10 12:56:24 on ttys001
Johns-MacBook-Pro-2:~ Vitruvion$ python
Python 2.7.2 (default, Jun 20 2012, 16:23:33)
[GCC 4.2.1 Compatible Apple Clang 4.0 (tags/Apple/clang-418.0.60)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import re
>>> leftmost = re.search('h.m', 'Abrahem Lincoln makes ham sandwiches.')
>>> leftmost.group()
'hem'
>>>

LARGEST


Vitruvion — Python — 80×24
Last login: Sat Nov 10 12:56:24 on ttys001
Johns-MacBook-Pro-2:~ Vitruvion$ python
Python 2.7.2 (default, Jun 20 2012, 16:23:33)
[GCC 4.2.1 Compatible Apple Clang 4.0 (tags/Apple/clang-418.0.60)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import re
>>> leftmost = re.search('h.m', 'Abrahem Lincoln makes ham sandwiches.')
>>> leftmost.group()
'hem'
>>> largest = re.search('h*m', 'Abrahem Lincoln makes ham sandwiches.')
>>> largest.group()
'm'
>>> real_largest = re.search('h.*m', 'Abrahem Lincoln makes ham sandwiches.')
>>> real_largest.group()
'hem Lincoln makes ham'
>>>

SPECIAL CHARACTERS

- Escape using '****'
- '**.**' any character except newline
- '**\w**' any alpha character
- '**[abc]**' a, b, or c
- '**a|b**' a or b

POSITIONS

- '**^**' start of a string
- '**$**' end of a string

Oh noes!
Dis complecks patterns
is herting my brane!

USEFUL MODES

- re.match
- re.split
- re.findall
- re.groups

FUZZY MATCHING

- Gaius Julius Caesar
- Yulius Cesar
- G. Juliy Cezar

TOOLS SURFEIT

- http://regexpal.com/
- http://www.ultrapico.com/Expresso.htm
- O'Reilly's regex cookbooks

PRETTY MUCH THE SAME THING

- Unix - grep
- Perl - built-in
- C - PCRE
- PHP, Java, Ruby, Visual Basic, XML, JavaScript, ColdFusion, Oracle, R, ActionScript, Google Code Search...

REFERENCES

- Python - http://docs.python.org/2/library/re.html
- grep - http://pubs.opengroup.org/onlinepubs/9699919799/utilities/grep.html
- Perl - http://perldoc.perl.org/perlre.html
- Inspiration - http://blog.stevenlevithan.com/archives/10-reasons-to-learn-and-use-regular-expressions
- Formalism - http://en.wikipedia.org/wiki/Regular_language