Text Processing

```
/^
(?:ftp|https?):\/\/
(?:
  (?:(?:[\w\.\-\+!$&'\(\)*\+,;=]|%[0-9a-f]{2})+:)*
  (?:[\w\.\-\+%!$&'\(\)*\+,;=]|%[0-9a-f]{2})+@
)?
(?:
  (?:[a-z0-9\-\.]|%[0-9a-f]{2})+
  |(?:\[(?:[0-9a-f]{0,4}:)*(?:[0-9a-f]{0,4})\])
)
(?::[0-9]+)?
(?:[\/|\?]
  (?:[\w#!:\.\?\+=&@$'~*,;\/\(\)\[\]\-]|%[0-9a-f]{
*)?
```

Text Manipulation

- Parsing text
  - extract tokens and understand their significance
- Text transformation
  - change all words to lower case
  - replace multiple consecutive spaces by a single space
- Regular expressions
  - identify text with specific structure

Strings

- s = “hello world”
- s = “””go home
  class dismissed”
- s = ‘gone with the wind’
- Strings are immutable.
- Strings are similar to sequences
  - s[3] # ‘n’
  - s[3:6] # ‘e wi’

Loops with String

```
s = "The quick fox jumps over the lazy dog"
for si in s:
    print si,
```

```
T h e   q u i c k   f o x   j u m p s   o v e r   t h e   l a z y   d o g
```

```
s = "The quick fox jumps over the lazy dog"
for si in s:
    print si
```

```
T
h
e

q
u
i
c
k

f
o
x

j
```

Splitting text

```
s = "The quick fox jumps over the lazy dog"
slist = s.split()
print slist
```

```
['The', 'quick', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
```

```
s = """The quick fox
jumps over
the lazy dog"""
slist = s.split()
print slist
slist2 = s.split(' ')
print slist2
```

```
['The', 'quick', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
['The', 'quick', 'fox', '\njumps', 'over', '\nthe', 'lazy', 'dog']
```

Joining and printing

```
a="The"
b="lazy"
c="fox"
print a+b+c
blank = "  "
print a+blank+b+blank+c+blank
print "%s %s %s" % (a,b,c)
print "Example: %li %s %ses\ndivided the bounty of\t%.2f" % (5,b,c,5.4567)
```

```
Thelazyfox
The lazy fox
The lazy fox
Example: 5 lazy foxes
divided the bounty of   5.46
```

Joining Lists

```
s = """The quick fox
jumps over
the lazy dog"""
slist = s.split()
print "LIST:              ",  slist
print "FOR:               ",
for si in slist:
    print si,
print
print "APPEND:            ",
news = ""
for si in slist:
    news += si
print news
print "JOIN:              ", "".join(slist)
print "JOIN with @:       ","@".join(slist)
print "JOIN with blank:   "," ".join(slist)
print "JOIN: with plus:   ","  +  ".join(slist)
```

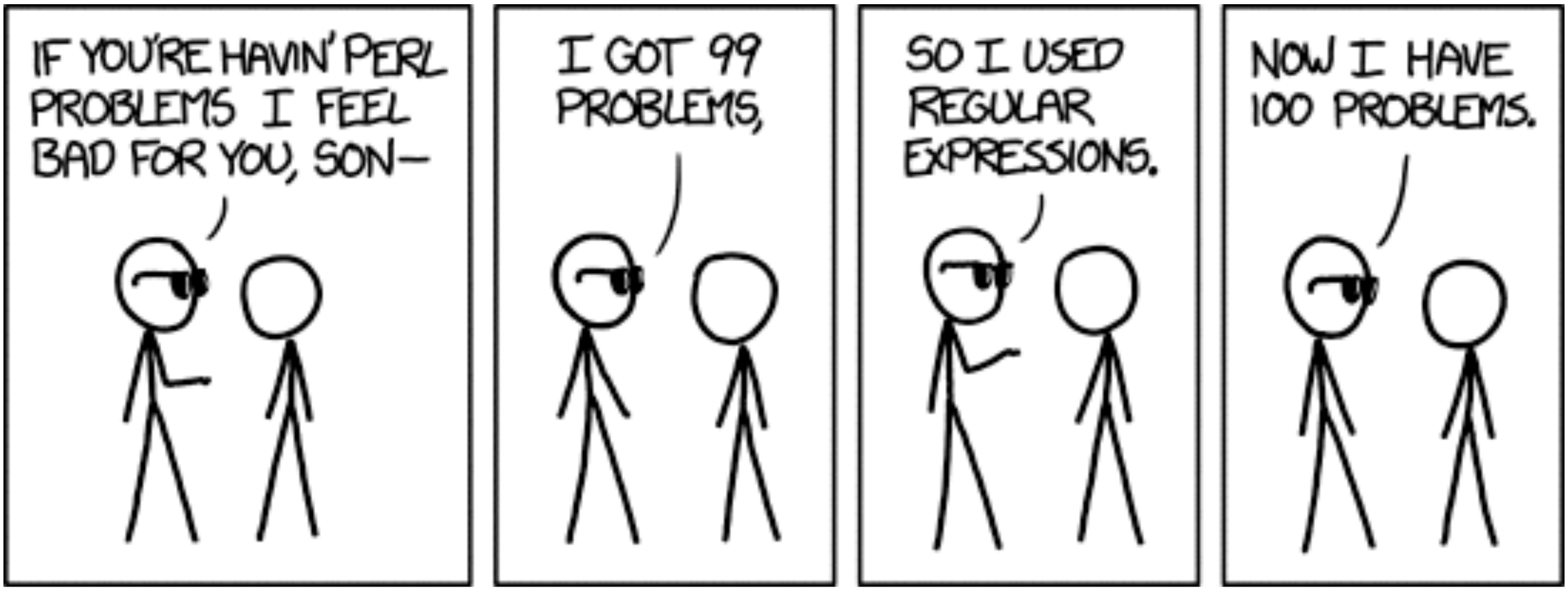
```
LIST:              ['The', 'quick', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
FOR:               The quick fox jumps over the lazy dog
APPEND:            Thequickfoxjumpsoverthelazydog
JOIN:              Thequickfoxjumpsoverthelazydog
JOIN with @:       The@quick@fox@jumps@over@the@lazy@dog
JOIN with blank:   The quick fox jumps over the lazy dog
JOIN: with plus:   The  +  quick  +  fox  +  jumps  +  over  +  the  +  lazy  +  dog
```

Regular Expressions

- Documentation
  - http://docs.python.org/library/re.html
- Tutorial
  - http://docs.python.org/howto/regex.html

- regular expression module
  - import re

```
'compile','copy_reg','error','escape','findall','finditer','
match','purge','search','split','sub',
```



Simple Searches

```
CAP = re.compile(r"[ZQ][uo][a-z]*")
CAP.findall(text)
```

```
['Zoologique',
 'Zoonomia',
 'Zoolog',
 'Zoologisch',
 'Quercus',
 'Quercus',
 'Quatrefages',
 'Zoological',
 'Quadrupeds',
 'Quadrupeds',
 'Quagga',
 'Quatrefages',
 'Quercus',
 'Quince']
```

```
import re
filename = "/Users/beerli/Documents/Work/talks/ISC-4304/misc/origin6th.txt"
f = open(filename,'rU')
text = f.read()
#tt = re.split('\.|"|\?|\!|-|_|\s|;|,|\*|\n',text.lower())
tt = re.split('\W',text.lower())
```

```
filter(None,tt)
```

```
['the',
 'project',
 'gutenberg',
 'ebook',
 'of',
 'on',
 'the',
 'origin',
 'of',
 'species',
 'by',
 'charles',
 'darwin',
 'this',
 'ebook',
 'is',
 'for',
 'the',
 'use',
```

Repetition

- pattern = “a*b” # 0 or more
- pattern = “a+b” # 1 or more
- pattern = “a?b” # 0 or 1
- pattern = “a{2}” # 2 copies of a

Special Forms

- . : any character
- $: end of string
- ^ : beginning of string
- a-z : “-” range of characters
- [a-c3-5] : any of the characters within [ ]
- [^a-c] : all characters *except* [abc]

Further abbreviations

- \d : digit character class: [0-9]
- \D : non-digit character
- \w : alphanumeric char
- \W: non-alphanumeric char
- \A : beginning of string
- etc.

Greediness

- By default, matching generates the longest possible match: *greedy* (*,+,?)
- It is possible to reverse the behavior to non-greedy (*?, +?, ??)

Greedy vs non-greedy

- tx = “abab c4 ab”
- re.search(“a.*b”, tx) # ==> “ab c4 ab”
- re.search(“a.?*b”, tx) # ==> “ab”
- re.search(“[ab]{2}.*[ab]”) ==> “abab c4 ab”

A few methods from re module

- re.search(pattern, str)
  - returns a *matchObject* for the *leftmost* substring
- re.sub(pattern, replace, str)
  - return string with pattern replaced by replace
- re.findall(pattern, str)
  - return a list of nonoverlapping patterns in string
- re.compile(pattern, flags)
  - compile the pattern for efficiency

Locating matches

```
pattern = re.compile(r'[Q][a-z]+')
for m in pattern.finditer(text):
    print m.start(), m.group()
```

```
139490 Quercus
139993 Quercus
676073 Quatrefages
1194386 Quadrupeds
1208959 Quadrupeds
1242547 Quagga
1242566 Quatrefages
1242602 Quercus
1242629 Quince
```

```
text[139490  : 139490   +50]
```

```
'Quercus robur has twenty-eight varieties, all of w'
```

re_show()

```
import re
def re_show(pat, s):
    print re.compile(pat, re.M).sub("{\g<0>}", s.rstrip()),'\n'

s = """Mary had a little lamb
    And everywhere that Mary
    went, the lamb was sure to go"""

re_show('a', s)          # letter 'a'
re_show(r'^Mary', s)     # beginning of line
re_show(r'Mary$', s)     # end of line
re_show(r'.a', s)        # any letter + 'a'
```

```
M{a}ry h{a}d {a} little l{a}mb
    And everywhere th{a}t M{a}ry
    went, the l{a}mb w{a}s sure to go

{Mary} had a little lamb
    And everywhere that Mary
    went, the lamb was sure to go

Mary had a little lamb
    And everywhere that {Mary}
    went, the lamb was sure to go

{Ma}ry {ha}d{ a} little {la}mb
    And everywhere t{ha}t {Ma}ry
    went, the {la}mb {wa}s sure to go
```

```
re_show(r'Wher|ever', s)
re_show(r'(Wher)(ever)', s)
re_show(r'(Wherever)', s)
re_show(r'(th).*(th)', s)
re_show(r'(th).*(th)+', s)
re_show(r'(th).*?(th)??', s)
re_show(r'(th).*(th)??', s)
```

```
Mary had a little lamb
    And {ever}ywhere that Mary
click to scroll output; double click to hide
```

```
Mary had a little lamb
    And everywhere that Mary
    went, the lamb was sure to go

Mary had a little lamb
    And everywhere that Mary
    went, the lamb was sure to go

Mary had a little lamb
    And everywhere that Mary
    went, the lamb was sure to go

Mary had a little lamb
    And everywhere that Mary
    went, the lamb was sure to go

Mary had a little lamb
    And everywhere {th}at Mary
    went, {th}e lamb was sure to go

Mary had a little lamb
    And everywhere {that Mary}
    went, {the lamb was sure to go}
```

re.search

```
str = "The fuzz is on the street"
m = re.search("[tT]he", str)
print m
print m.groups()
print m.group(0)
print re.findall("[tT]he", str)
m = re.search("(The).*(fu.*z)", str)
print "groups(): ", m.groups()
print "group(): ", m.group()
print "group(0): ", m.group(0)
print "group(1): ", m.group(1)
print "group(2): ", m.group(2)
```

```
<_sre.SRE_Match object at 0x10b5c59f0>
()
The
['The', 'the']
groups():  ('The', 'fuzz')
group():  The fuzz
group(0):  The fuzz
group(1):  The
group(2):  fuzz
```

Some Constant flags

- re.I : re.IGNORECASE :
- re.L : locale
- re.M : multiline
  - pattern match do not cross ‘\n’ boundaries)

Explorations

- Regular expressions offer much more than discussed