Ruby

Interpolation, Code Blocks
Regular Expressions

Interpolation

Evaluate expressions inside strings

```
score = 100
"My score is #{x + 50}"
```

Must use double-quotes

Code Blocks

```
3.times {puts 'Repeat…'}
```

Method in Fixnum class

Method's parameter is a *code block*

A *code block* is an unnamed function

```
animals = ['lions', 'tigers', 'bears', 'rabbits']
animals.each {|a| puts a}
```

Array iterator method

Parameter assigned next value from each

Code Blocks and lambda

Code blocks can be stored in variables using the `lambda` method

```
show = lambda {|x| puts x}
animals.each {|a| show.call a}
```

Code Blocks and yield

`yield` passes control to the code block

```
def mytimes(i)
   while i > 0
      i = i - 1
      yield
   end
End

mytimes(4) {puts "Repeat..."}
```

Regular Expressions

Used extensively in Ruby scripts for pattern matching

- *Matching*
  - Test if a string contains a regular expression
- *Substituting*
  - Replace part of a string with a new string

Both use the match operator: `=~`

Some Standard Quantifiers

Pattern	Description
a	match a
a*	match zero or more a's
.	match any character
.*	match zero or more of any character
[a-m]	match characters a through m only
[^n-z]	do not match letters in n to z
[a-m]*	match zero or more letters in a to m
[a-m]+	match one or more letters a to m
^	match anchored at the beginning of the line
$	match anchored at the end of the line

Some Extensions

\t	matches a tab character
\d	same as [0-9]
\D	same as [^0-9]
\s	matches white space (space or tab)
\S	matches anything but white space
\w	same as [0-9a-zA-Z_]
\W	same as [^0-9a-zA-Z_]
.+	same as ..*
a{n,m}	at least *n* a's, not more than *m* a's
a?	zero or one a

Regex Examples

Delimit regular expression patterns with / or %r{}

Match year numbers in '1970s & '1980s

```
/19[78]\d/
```

Match an HTML tag

```
/<\w+>/
```

Match an Ada identifier

```
/[a-zA-Z](_?[a-zA-Z0-9])*/
```

Match a zip code

```
%r{\d{5}(-\d{4})?}
```

Matching Examples

```
p1 = /19[78]\d/
puts "Disco time" if '1983' =~ p1
p2 = Regexp.new('<\w+>')
puts 'tag' if p2 =~ "<html>"
p3 = /[a-zA-Z](_?[a-zA-Z0-9])*/
puts "Not an Ada id" unless p3 =~ 'Good_id'
p4 = %r{\d{5}(-\d{4})?}
puts "Zip" if '08240-0195' =~ p4
```

Selecting Substring Matches

() selects a substring of a regular expression

Each substring match is assigned to a variable $k, where k = 1, 2, 3, …

Example:

Break up the components of an email address in the form user@host.domain

```
if addr =~ /(\S+)@(\S+)\.(\S+)/
   user, host, tld = $1, $2, $3
end
```

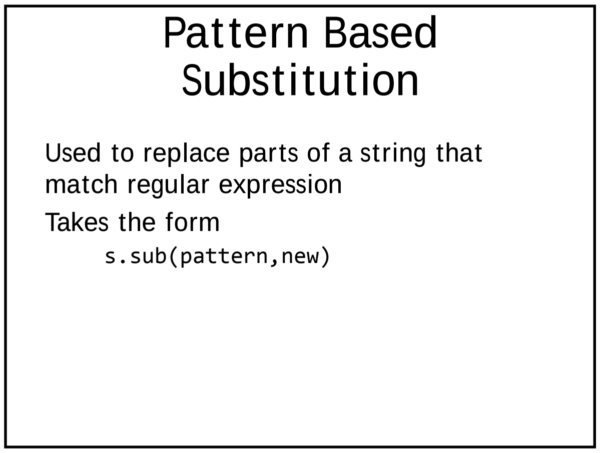
Substring Matches

```
"10:45am" =~ /((\d\d?):(\d\d))(..)/
puts "Time is #{$1}, hour is #{$2}, \
      minute is #{$3}, AM/PM is #{$4}"
```

Match identical beginning and ending HTML tags <xyz> ... </xyz> on a single line

```
if line =~ %r{<(\w+)>(.*)</\1>}
   content = $2
end
```

\1 is used to refer to a previously matched substring in the regular expression itself

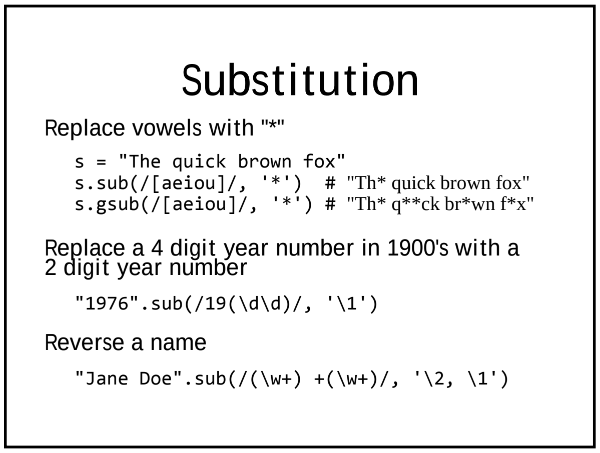


Pattern Based Substitution

Used to replace parts of a string that match regular expression

Takes the form

```
s.sub(pattern,new)
```



Substitution

Replace vowels with "*"

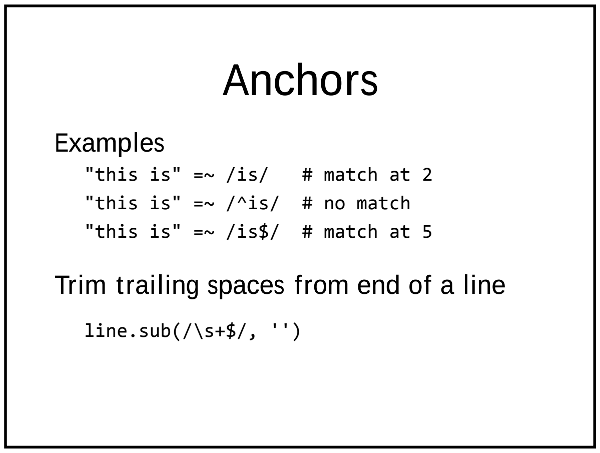
```
s = "The quick brown fox"
s.sub(/[aeiou]/, '*')  # "Th* quick brown fox"
s.gsub(/[aeiou]/, '*') # "Th* q**ck br*wn f*x"
```

Replace a 4 digit year number in 1900's with a 2 digit year number

```
"1976".sub(/19(\d\d)/, '\1')
```

Reverse a name

```
"Jane Doe".sub(/(\w+) +(\w+)/, '\2, \1')
```



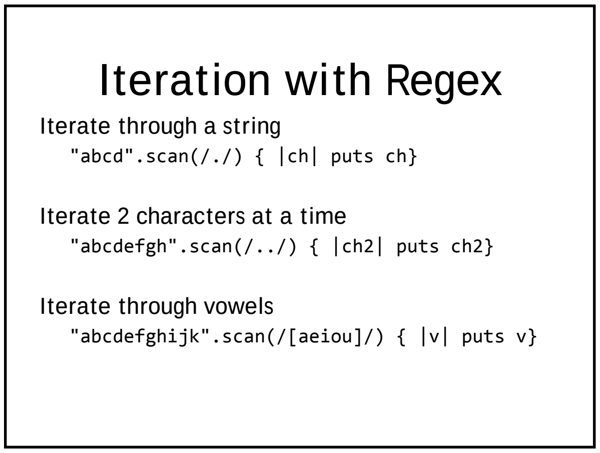
Anchors

Examples

```
"this is" =~ /is/   # match at 2
"this is" =~ /^is/  # no match
"this is" =~ /is$/  # match at 5
```

Trim trailing spaces from end of a line

```
line.sub(/\s+$/, '')
```



Iteration with Regex

Iterate through a string

```
"abcd".scan(/./) { |ch| puts ch}
```

Iterate 2 characters at a time

```
"abcdefgh".scan(/../) { |ch2| puts ch2}
```

Iterate through vowels

```
"abcdefghijk".scan(/[aeiou]/) { |v| puts v}
```