Day 10: Regular Expressions II

Suggested reading: *Learning Perl* (6th Ed.)

Chapter 8: Matching with Regular Expressions

Chapter 9: Processing Text with Regular Expressions

Homework Review

Matching

Thus Far…

letters, `!`, `@`, …	match self
`^`	start of string
`$`	end of string
`.`	any character
`\`***x***	*x*, without special meaning
`*`	match preceding, 0–*n*
`+`	match preceding, 1–*n*
`?`	match preceding, 0–1
`{`***n***`,`***m***`}`	match preceding, *n*–*m*
`[…]` & `[^…]`	character classes
`\d`, `\D`, `\w`, `\W`, `\s`, `\S`	digit, word char, whitespace
`\b`, `\B`	word boundary (or not)

Greedy vs. Non-Greedy Matches

.* greedy match

```
/a.*z/      azimuth, dazzle, waltz, abuzz, a.*z
            a, z, apples, buzz, Azimuth
```

.*? non-greedy match

```
/a.*?z/     azimuth, dazzle, waltz, abuzz, a.*z
            a, z, apples, buzz, Azimuth
```

Append **?** to *****, **+**, **?**, **{}**

When to Use Non-Greedy Matching

- With paired delimiters; e.g., (…), […], {…}, "…", '…'
- When pattern might occur multiple times in string

```
<img src="/icon.png" alt="icon">
```

```
/src=".+"/
```

Matches: `<img src="/icon.png" alt="icon">`

```
/src=".+?"/
```

Matches: `<img src="/icon.png" alt="icon">`

```
/src="[^"]+"/
```

Matches: `<img src="/icon.png" alt="icon">`

Groups

(…) groups *and remembers* parts of a match

`/(in){2}/`	`dining, feminine` `in, ini, nine, (in){2}`
`/^(pre)?te/`	`tend, pretend, test, pretest` `steam, present, a test, ^(pre)?te`

Groups (from matches) can be referenced by **\#**

`/^(.)(.)\2\1$/`	`anna, deed, maam, noon` *not much else…*

Alternatives

aaa|bbb matches all of "aaa" ***or*** all of "bbb"

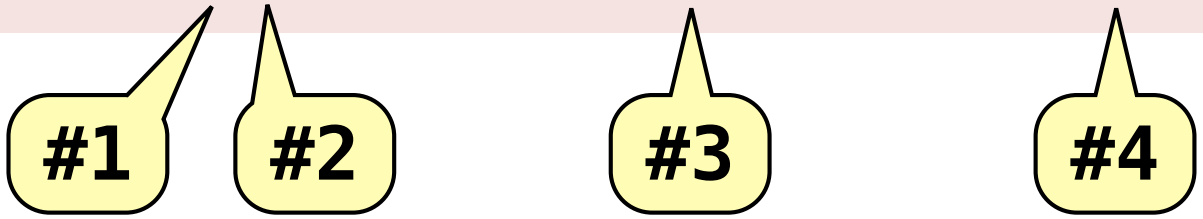
/here|hear/	**here, hear, there, heart, gatherer** **her, haer, heer, ear, herhear**

use **( )** when **|** applies *only to part* of whole pattern

/d(og|im|ay)/	**dog, dim, day, dime, Tuesdays** **dom, diy, dig, ayd, d(og|im|ay)**

Groups and alternatives are powerful (if complex)

```
/^((1[0-2])|([1-9])):([0-5]\d)$/
```



Modifiers

Ignoring Letter Case

`//i` Ignore case in matching

`/cat/`	`cat, a cat, catalog, scatter, tomcat` `Cat, a Cat, Cathy, TomCat`
`/cat/i`	`cat, Cat, Cathy, tomcat, TomCat` `dog`

Commenting Regular Expressions

//x Whitespace and comments allowed in RE
Quote both with \ to include in RE itself

```
print if /
  ^                 # start of string
  (                 # group hour part ($1)
    (1[0-2])        # hours option 1: 10-12 ($2)
    |
    ([1-9])         # hours option 2: 1-9 ($3)
  )                 # end of hour group
  :
  ([0-5]\d)         # minutes: 00-59 ($4)
  $                 # end of string
/x;
```

Matching Across Lines

`//m` Treat string as multiple lines

`^`, `$` match start, end of any line within string

`//s` Treat string as single line

`.` matches newline (which normally it does not)

```
/<name>\s*(.*?)\s*<\/name>/is

<person><name>
  Tim
  Cartwright
  </name><office>
4265</office></person>
```

Back to Perl

Testing For a Match

- A match operation yields true or false
- Specify string with =~ or !~ (or else uses $_)

```
while (<INPUT_FILE>) {
    next if /^\s*$/;
    my $lower = lc($_);
    print $lower if $lower !~ m{^\s*title:};
}
```

- In list context, yields group (substring) matches

```
my @parts = ($in =~ /(\d{4})-(\d\d)-(\d\d)/);
exit unless @parts;
my ($year, $month, $day) = @parts;
```

Looping Through Matches

- Add //g modifier to get all matches in string

```
my $file = join('', <INPUT_FILE>);

while ($file =~ /"(.+?)"/sg) {
    print "Quote: $1\n";
}
```

Splitting

Split a string using a string separator:

```
my @items = split ',', $string;
```

```
'cat'          => ( 'cat' )
'cat, dog'     => ( 'cat', ' dog' )
'1,2 ,3 , 4' => ( '1', '2 ', '3 ', ' 4' )
```

Split a string using a regular expression separator:

```
my @items = split /\s*,\s*/, $string;
```

```
'cat, dog'     => ( 'cat', 'dog' )
'1,2 ,3 , 4' => ( '1', '2', '3', '4' )
```

Extracting

- Upon match, groups are saved in **$1**, **$2**, **$3**, …

```
print "Enter your birthday (YYYY-MM-DD): ";
my $date = <STDIN>;
my ($year, $month, $day);
if ($date =~ /(\d{4})-(\d{2})-(\d{2})/) {
    ($year, $month, $day) = ($1, $2, $3);
}
```

- Make sure match succeeded!

```
if ($xml =~ m{<name>\s*(.*?)\s*</name>}im) {
    $name = $1;
} else {
    print "Could not extract name from XML.\n";
}
```

Substituting

`s/…/…/` substitutes a match with a string… *once*

```
$text = 'red and green and blue and cyan';
$text =~ s/and/or/;

=> 'red or green and blue and cyan'
```

`s/…/…/g` substitutes all matches in the string

```
$text = 'red and green and blue and cyan';
$text =~ s/and/or/g;

=> 'red or green or blue or cyan'
```

Can reference groups in replacement string

```
$html =~ s,<i>(.*?)</i>,<em>$1</em>,g;
```

Substitution Examples

Remove all vowel characters, regardless of case:

```
$string =~ s/[aeiou]//ig;
```

Make every line of code a comment, if not already:

```
while (<INPUT>) {
    s/^(\s*)([^#\s])/$1# $2/;
    print;
}
```

Do something if any substitutions occurred:

```
if ($text =~ s/Prof\w* Cartwright/Tim/g) {
    print "Incorrect title purged!\n";
}
```

Variable Interpolation

Perl interpolates variables into a regular expression ***before*** processing the expression itself

Interpolation in the pattern:

```
$dir_pattern = '.*/';
$path =~ s|^$dir_pattern|/home/cat/|;
```

Interpolation in the replacement:

```
$home_dir = '/home/cat';
$path =~ s|^.*/|$home_dir/|;
```

Now *YOU* Can Do This:

http://xkcd.com/208/

More Resources for Regular Expressions

- [Madcat] ***Mastering Regular Expressions***, Friedl
- [Madcat] ***Perl Cookbook***
- Google for patterns
  - Can be very helpful
  - Do you trust what you find?
  - Understand assumptions, limitation, etc.
  - Use as inspiration, not as copy-and-paste solution

Homework

- Parse a file of test results
  - Filter out lines of interest
  - Extract useful data
  - Change some data for readability
  - Save data in a useful data structure
- Report on tallies of test results by test module
- Use regular expressions and a good data structure!