

Perl Regular Expressions, Matching and Substitutions

The HTML notes have lots more details,
examples and exercises (with solutions)

EIW - Perl Regular Expressions

1

Text Processing

- Pattern Matching.
 - Text Processing
 - Useful when dealing with HTML
 - Useful when dealing lots of things!
- Powerful language used to express patterns:
 - regular expressions.
 - roots in CS theory...

EIW - Perl Regular Expressions

2

Things Perl Can Do Easily

(once you know how to use regular expressions)

- Find out if a string contains some specific pattern.
- Parse a string
 - extract components from a string according to some syntactical rules.
- Replace parts of a string with other strings.

EIW - Perl Regular Expressions

3

Regular Expressions

- Cryptic language, but possible to master!
- You can do lots even if you only know the basics.
- An example:

```
($proto,$host,$uri) = /(^[^:]+):\/\/([^\s]+)\/(.*)/;
```

EIW - Perl Regular Expressions

4

Perl Match Operator

- Tells whether a *pattern* is matched by a string.
 - returns boolean value (indicating true or false in the Perl way).
- The pattern is a *regular expression*
- The string is usually in `$_`

EIW - Perl Regular Expressions

5

Match Operator Example

```
/foo/
```

- Will be true if "foo" is a substring of `$_`
 - "food is great" "blah foo"
- Will be false if there is no "foo" in `$_`
 - "nothing here" "blah fo o"

EIW - Perl Regular Expressions

6

foofinder.pl

```
# read one line at a time into $_
while (<>) {
  # see if the string in $_ contains "foo"
  if (/foo/) {
    # the string matches "foo", so print it
    print;
  }
}
```

EIW - Perl Regular Expressions

7

Defining Patterns

- Most characters match themselves (once).
 - 'a' matches an 'a', ...
- A sequence of characters matches itself (once).
 - 'foo' matches 'foo', ...
- '.' matches any single character!
 - period. dot.

EIW - Perl Regular Expressions

8

. is a wildcard

/f./ matches any string containing an **f** followed by any other character.
matches "hi there **foo**", but not "def"

/a.b/ matches any string containing an 'a' followed by any single character, followed by a 'b'
matches "123**axbdef**" or "**a3b**"

EIW - Perl Regular Expressions

9

What if we want to match '!' ?

- Throw in a slash (called "escaping").

`/f\./` matches any string that contains an 'f' followed by a period.

will match "d. e. r. f.", but not "fred"

EIW - Perl Regular Expressions

10

Character Class

- Sometimes we want to be able to match any character in a set of characters.
- Put a list of the characters in your set inside square brackets:
 - `[abc]` matches any single character that is an 'a', 'b' or 'c'.
 - `[abcdefghijklmnopqrstuvwxyz]` matches any lowercase alphabetic char.

EIW - Perl Regular Expressions

11

Examples

`/[aeiou]/` matches any string that contains a lowercase vowel.

`/a[bc]/` matches any string that contains an 'a' followed by either a 'b' or 'c'.

`/[aA][lL]/` any string with "al" in upper or lower case (any combination).

EIW - Perl Regular Expressions

12

Negated Character Class

- Make the first character in the square brackets '^'. Now it means "any character not in this set."
- `[^abc]` any character except 'a', 'b' or 'c'.
- `[^aeiou]` any character other than a lowercase vowel.

EIW - Perl Regular Expressions

13

Examples

`/[^a]/` matches any string that contains a letter other than 'a'. Won't match "aaa"

`/a[^bc]/` matches any string that contains an 'a' followed by something other than a 'b' or 'c'.
will match "dave", not "abac"

EIW - Perl Regular Expressions

14

Character Class Shortcuts

- You can do this: `[a-z]` or `[0-9]` or `[a-zA-Z]` or even `[a-cf-k0-5B-K]`
- Perl has some shortcuts for commonly used character classes.
 - `\d` is a shortcut for `[0-9]` (any digit).
 - `\D` is a shortcut for `[^0-9]`

EIW - Perl Regular Expressions

15

More Shortcuts

`\s` is a shortcut for any whitespace [`\r\n\t\f`]

`\S` is a shortcut for anything except whitespace.

`\w` is a shortcut for [`a-zA-Z0-9_`]

`\W` is a shortcut for [`^a-zA-Z0-9_`]

EIW - Perl Regular Expressions

16

Exercises

- any string that contains an "a" or "b" followed by any 2 characters followed by an "a" or a "b". The strings "axxb", "alfa" and "blka" match, and "ab" does not.
- any string that contains a 5 digit integer.

EIW - Perl Regular Expressions

17

`*`, `+` and `?`

- These characters do not match themselves.
- These modify whatever precedes them in the regular expression.
- `*` means "0 or more of the previous thing"
- `+` means "1 or more of the previous thing"
- `?` means "0 or 1 of the previous thing".

EIW - Perl Regular Expressions

18

* Examples

`/abc*/` matches any string that contains an 'a' followed by a 'b' followed by 0 or more 'c's

matches "hi ab" or "foo blabcccccc xy"

`/<.*>/` matches anything inside angle brackets, like "<HEAD>" or "<P>".

EIW - Perl Regular Expressions

19

Trick Question

- Give me a string that will NOT match this:

`/X*/`

EIW - Perl Regular Expressions

20

+ Examples

`/a+[01]/` matches any string that contains one or more 'a's followed by either '0' or '1'

`/ab+c/` an 'a' followed by one or more 'b's followed by a 'c'.

matches "abbbc" or "abc", but not "ac"

EIW - Perl Regular Expressions

21

? Example

`/A[0-9]?B/` matches any string that contains an 'A' followed by a 'B' with possibly a single digit between.

matches "A0B" or "AB", not "A123B"

Grouping with Parentheses

- You can put part of a regular expression inside parentheses
- *, + and ? will operate on everything inside the parentheses.

`/(a[0-9]+)/`

Matches "Hello a1a3", but not "aaa 0 abc"

Important Matching Property

- Whenever Perl sees '*' or '+' it will match as much as it can!
- This becomes an issue later when we use regular expressions with the substitute operator.

Examples

`/a(bc)*d/` matches "ad" or "abcbcd blah",
but not "abdcdbd".

`/([Ww]indows ?XP)+/` matches
"windowsXP" or "I Love Windows XP" or
"windowsexpwindowsexp", but not
"Windows 98"

EIW - Perl Regular Expressions

25

Exercises

- Match any Perl scalar variable name.
- An HTML Anchor Tag
``

EIW - Perl Regular Expressions

26

Grouping and Memory

- Perl can *remember* the part of a string that matches a *group* in a regular expression (a group something in parentheses).
- We can find out what part matched a group.
- We can use this part again in the regular expression.

EIW - Perl Regular Expressions

27

Extracting parts that match

```
($first,$second) = /(\w+)\s+(\w+)/;
```

- **\$first** will be set to whatever part of **\$_** matches the first *word* (group in the regular expression).
- If **\$_** is "Hello Dave", **\$first** would get "Hello" and **\$second** would get "Dave".

EIW - Perl Regular Expressions

28

What does this do?

```
($proto,$host,$uri) =  
/([^\:]+):\\\/([^\\/]+)\\\/(.*)/;
```

EIW - Perl Regular Expressions

29

Referencing Matches

- We can reference the *memorized* chunks of the string that matched a group.

\1 means "whatever part of the string matched the first group".

\2 means the second group, ...

We can reuse these inside regular expressions!

EIW - Perl Regular Expressions

30

Example

`/([abc])x\1/` matches any string that contains "axa" or "bxb" or "cxc".

`/(\w+).*\1/` matches any string that contains a substring of alphanumeric characters that is repeated.

matches "Dave is average".

EIW - Perl Regular Expressions

31

Exercises

- Any word (no whitespace) that contains a double letter, like "book" or "cookie".
- Any string that contains an HTML tag and its corresponding end tag. Should match "<H2>Hello</H2>", but not "<P>foo"

EIW - Perl Regular Expressions

32

Alternation

- The "|" character means "or" inside a regular expression.

`/EIW|CompOrg|WebSys/`

matches any string that contains either "EIW" or "CompOrg" or "WebSys"

EIW - Perl Regular Expressions

33

Anchors

- Some times you want to match something only if it is found at the beginning or end of a string.
- '^' means "beginning of string" if it's the first character in a regular expression.
- '\$' means "end of a string" if it's the last character in a regular expression.

EIW - Perl Regular Expressions

34

Anchor Examples

`/^[A-Z]/` matches any string that begins with a capital letter.

`/(w+).*\1$/` matches any string that begins and ends with the same word.
matches "hi blah hi", not "hi blah".

EIW - Perl Regular Expressions

35

Substitutions

- There is another Perl operator that uses regular expressions.
 - allows you to replace any part of a string that matches a regular expression with something else.
- "replace all H2 tags with H3 tags"

EIW - Perl Regular Expressions

36

Simple Example

```
s/Dave/Joe/;
```

replaces the first occurrence of "Dave" with "Joe" (in \$_).

EIW - Perl Regular Expressions

37

General form of the substitution operator

```
s/regexp/newtext/modifiers;
```

regexp is a regular expression.

newtext is a text string.

modifiers can change how this does its job.

EIW - Perl Regular Expressions

38

Substituting

- Whatever part of \$_ is matched by the regular expression is removed and replaced with the new text (only one replacement is made).

```
$_="Cookies are good";  
s/Cookies/Pizza/;  
# now $_ is "Pizza is good"
```

EIW - Perl Regular Expressions

39

Modifiers

- 'g' means "global". Make as many substitutions as possible. Without 'g' only the first match is replaced.
- 'i' means "case insensitive". The matching does not care about the case of alphabetic characters. ('a' matches 'A').

EIW - Perl Regular Expressions

40

Example

```
$_ = "MIT is a great place to clean  
your mit, but you need to wear  
mittens or you will be committed";
```

```
s/MIT/RPI/gi;
```

```
# now $_ is "RPI is a great place  
to clean your RPI, but you need to  
wear RPIitens or you will be  
comRPited"
```

EIW - Perl Regular Expressions

41

Exercise

- Write a Perl program that removes all HTML tags (anything that looks like an HTML tag - you don't need to check each tag name) from a file and prints out the resulting file.

EIW - Perl Regular Expressions

42

One Solution

```
while (<>) {  
    s/<[^>]*>/g;  
    print;  
}
```

EIW - Perl Regular Expressions

43

Using your own variables

- The =~ operator tells perl to use your scalar variable instead of \$_:

```
if ($foo =~ /TR>|TD>/) {  
    $foo =~ s/(T[RD])\1 BGCOLOR=RED/g;  
}
```

EIW - Perl Regular Expressions

44

There is more...

- Check the HTML lecture notes for even more regular expression fun (we have not covered everything).

EIW - Perl Regular Expressions

45
