Perl Lesson 6

L615

Spring 2009

1. Regular expression matching
   - `/.../` — perform basic matches — same as `m//` (or `m()` or `m<>` or `m!!`)
   - RE syntax essentially the same as we saw before

```
$_ = "KL5-8890";

# for now, we'll match against the default $_ variable
if (/KL5-[0-9]{4}/) {
  print "I think this is a fake number\n";
}
elsif (m<KLM>) {
  print "Is there an airline involved?\n";
}
```

2. Options
   - Case-insensitive matching: `/i`

```
if (/kl5-[0-9]{4}/i) {
  print "I think this is a fake number\n";
}
```

   - Any character matching: `/s` — the dot (.) normally doesn't match newlines; the `/s` option allows newlines to be matched

```
$_ = "My favorite cop is Ponch\n\nBut some crazy people like John.";

if (/Ponch.*John/s) {
  print "Ponch is mentioned somewhere on this page before John\n";
}
```

   - Whitespace-added matching: `/x` — allows you to write cleaner regular expressions by treating whitespace as irrelevant

```
$_ = "The number on the wall was 867-5309, Tommy said";

if (/\b[0-9]{3}-[0-9]{4}\b/) {
  print "This looks like a telephone number\n";
}

# OR:
if (/
  \b       # word boundary
  [0-9]{3} # first 3 digits
  -        # customary hyphen
  [0-9]{4} # next 4 digits
  \b       # word boundary
/x) {
  print "This looks like a telephone number\n";
}
```

- Combining options: you can use the different options together

```
$_ = "My favorite cop is Ponch\n\nBut some crazy people like John.";

if (/
   ponch
   .*
   john
/six) {
  print "Ponch is mentioned somewhere on this page before John\n";
}
```

3. Binding operator: `=~`
   - Instead of matching against the default `$_`, we can match a variable to a pattern

```
my $tv_show = "The Brady Bunch";
if ($tv_show =~ /brady/i) {
  print "Who's your favorite Brady?\n";
}
```

4. Match Variables: by using parentheses, we can capture values and refer to them later

```
if (/\b(\w+)\s+(\w+)\b/) {
  print "word 1 was $1 and word 2 was $2\n";
}
```

   - Note that if a RE is unsuccessful in matching, the values of `$1` and `$2` stay as what they were before (which is why we like to refer to them only after we've matched something)