1. Regular expression matching

- `/.../` — perform basic matches — same as `m//` (or `m()`) or `m<>` or `m!!`)

```perl
$_ = "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";
}
elif (m<KLM>) {
  print "Is there an airline involved?\n";
}
```

2. Options

- Case-insensitive matching: `/i`

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

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

```perl
$_ = "The number on the wall was 867-5309, I heard";
if (/\b[0-9]{3}-[0-9]{4}\b/) {
  print "This looks like a telephone number\n";
}
```

# OR:

```perl
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";
}
```

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

```perl
my $tv_show = "CHiPs";
if ($tv_show =~ /chips/i) {
  print "Who's your favorite California Highway Patrol officer?\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)

5. RE interpolation: variables inside regular expressions

    $name = "tom";

    # reading through a piped-in file
    while (<>) {
        if (/\$name/) {
            print "Found name: \$_";  }
    }

6. Automatic match variables: a way to refer to the actually matched item vs. the surrounding context

• $& = matched string
• $' = material preceding matched string
• $' = material following matched string

    if ("larry, moe, and curly" =~ /\W+(moe)\W+/) {
        print "Before: \$'\n";
        print "Match: \$&\n";  # NB: not the same as $1
        print "After: \$'\n";
    }

7. Substitutions — s/// — REs allow us to specify a pattern and then replace it

• Global replacements — /g — by default, s/// replaces only the first match; /g allows you to replace all matches
• Delimiters — can change from / to something else (if, e.g., you want to refer to a string that uses '/')
• Options (Case Shifting) — as with regular RE matching, we can use the /i, /x, and /s modifiers

    $_ = "larry, moe, and curly";
    s/curly/shemp/;
    print "\$_\n";  # prints "larry, moe, and shemp"

    $_ = "larry, moe, and curly";
    s/(\w+)\W+(\w+)/$2 $1/;  # swaps words
    print "\$_\n";

    $_ = "larry, moe, and curly had curly hair";
    s/curly/shemp/g;
    print "\$_\n";

    $_ = "Larry, Moe, and Curly";
    s/(\w+)/\U$1/;  # makes first word uppercase
    print "\$_\n";
    s/(\w+)/\L$1/g;  # makes all words lowercase
    print "\$_\n";