Perl Text-processing language

CONCEPTS

Start scripts with use strict; use warnings; (or use v5.36;) and declare every variable with my.

Sigils

Sigil	Holds	Element access
$x	scalar	$x
@a	array	$a[0]
%h	hash	$h{key}
&f	sub	f(...)
\@a	reference	$r->[0]

Input files or STDIN
↓ -n / -p loop
$_ current line
↓ -a / -F
@F split fields
↓ your code
Output print or -i
↓

Only double-quoted strings interpolate variables and escapes; single quotes are literal.

Context decides results: an array in scalar context gives its length, in list context its elements.

ONE-LINERS

perl -e '*<code>*'
Run code given on the command line

perl -ne '*<code>*'
Run code once per input line

perl -pe '*<code>*'
Per line, then print $_

perl -i[.bak] -pe '*<code>*' *<file>*
Edit files in place

perl -l
Chomp input lines, add \n to print

perl -a -F*<sep>*
Split each line into @F

VARIABLES AND DATA

my $x = *<value>*;
Declare a lexical scalar

VARIABLES AND DATA (cont.)

my @a = (*<1>*, *<2>*, *<3>*);
Declare an array

my %h = (*<k>* ⇒ *<v>*);
Declare a hash

$a[0] / $a[-1]
First / last element

scalar @a / $#a
Element count / last index

push @a, *<x>* / pop @a
Add / remove at the end

$h{*<k>*}
Hash element

keys %h / values %h
All keys / all values

STRINGS AND OPERATORS

"Hi $name @{[f()]}"
Interpolate variables and expressions

== ≠ < > ⟺
Numeric compare

eq ne lt gt cmp
String compare

REGEX

$s =~ /*<re>*/
Test for a match

$s =~ s/*<re>*/*<new>*/g
Substitute all

($a,$b) = $s =~ /(\d+)-(\d+)/
Capture into variables

$1 $2 $&
Capture groups, whole match

my @all = $s =~ /*<re>*/g
All matches in list context

/i /x /m /s
Ignore case, extended, multiline, dot-all

CONTROL FLOW AND SUBS

if (*<c>*) {...} elsif (*<c>*) {...} else {...}
Conditional

CONTROL FLOW AND SUBS (cont.)

***<stmt>* if *<c>*; / *<stmt>* unless *<c>*;**
Postfix conditional

for my $x (@a) {...}
Loop over a list

while (*<c>*) {...}
Loop on a condition

next / last / redo
Skip / exit / repeat iteration

sub *<name>* { my ($a,$b) = @_; return ...; }
Define a sub

FILES AND I/O

open(my $fh, '<', $f) or die "$!";
Open for reading

open(my $fh, '>', $f) / '>>'
Open for write / append

while (my $l = *<$fh>*) {...}
Read line by line

print $fh "text\n";
Write to a handle

-e -f -d -r -s $path
File tests: exists, file, dir, readable, size

MODULES AND CPAN

use *<Module>*;
Load a module

cpanm *<Module>*
Install from CPAN

perldoc *<Module>*
Read module docs

RECIPES

Sum a column

```
perl -lane '$s += $F[2]; END
{ print $s }' <file>
```

Rename across many files

```
perl -i.bak -pe 's/\bfoo\b/bar/g'
*.txt
```

Word frequency

```
perl -lne '$c{$_}++ for /\w+/g;
END { print "$c{$_} $_" for sort
keys %c }' <file>
```