

COMP2041 Week 5 Tutorial

- use `#!/usr/bin/perl -w`
- to know the flags, use `perldoc perlrun`
- `-w` is used for warning
- `#` comment
- `$` scalar
- `@` array (indexed by integers)
- `%` hash (indexed by strings)
- `&` subroutine
- functions: `chomp`, `split`, `if`, `for`, `foreach`
- variables don't to be declared or initialised
- if not initialised, scalar is empty string or 0
- `$_` if there is no argument

operations

- `perldoc perlop`
- numeric: `==`, `!=`, `<=>` (returns -1 0 or 1), etc
- string: `eq`, `ne`, `lt`, etc
- append: `'.'`

stream

- `line = <STDIN>`
- `print STDOUT "Jess"`
- `open ...`
- `close ...`
- `<>` treats all command-line arguments as file names, opens and reads

control

- `foreach i (1..n)`

external commands

- use backticks
- `system "cmd"`
- `open F, "cmd|"`

special variables

- `$_`
- `@ARGV` list of command line arguments
- `$0` name of perl script

- `$i` matching string for ith regexp in pattern
- `$!` last error
- `$.` line number

start with test1 and test2

Question 1

a. same b. same c. same d. same e. same, string "42" f. same numeric, but compare string different g. it will interpolate "\$2" so "\$2.50" -> "0.50", and the other one is legit "\$2.50"

Question 2

```
print $#ARGV + 1, "\n";

print @ARGV + 0, "\n";

$x = @ARGV;

print "$x\n";
```

- perldoc perlvar
- `$#ARGV` is the maximum index of how many argv, so need to add 1

Question 3

```
while ($line = <>) {
    $line =~ s/[aiueo]//gi;
    print $line;
}

while (<>) {
    s/[aiueo]//gi;
    print;
}
```

the second one using `$_`

Question 4

`last` `break`, `next` `continue`

```
$n = 10;

if (@ARGV && $ARGV[0] =~ /[0-9]+/) {
```

```

    $n = shift @ARGV;
    $n =~ s/-//;
}

$i = 1;
while (<STDIN>) {
    last if ($i++ == $n);
    print;
}

@lines = <STDIN>;
$n = @lines if $n > @lines;
print @lines[0..$n-1];

```

Question 5

```

$ARGV[0] = "-" if @ARGV == 0;
foreach $file (@ARGV) {
    open(INPUT, "<$file") or die("$file: cannot be opened: $!\n");
    print "==> $file <==\n";
    $i = 0;
    while(<INPUT>) {
        last if ($i++ == $n);
        print;
    }
    close INPUT;
}

```

- Why don't we just use `<>` instead of `@ARGV` ?
- `INPUT` is just a name for the stream, you can give name anything. Usually capital letters.

Question 6

```

$pattern = shift @ARGV;

while($line = <>) {
    print $line if $line =~ /$pattern/;
}

while(<>) {
    print if /$pattern/;
}

```

Question 7

```

if ($ARGV[0] eq "-v") {
    $reverse = 1;
    shift @ARGV;
}

```

```

}

$pattern = shift @ARGV;

while (<>) {
    if ($reverse) {
        print if !/$pattern/;
    } else {
        print if /$pattern/;
    }
}

```

Question 8

```

if (@ARGV && $ARGV[0] eq "-n") {
    $show_line_number = 1;
    shift @ARGV;
}

while(<>) {
    printf "%6d  ", $. if ($show_line_number);
    print;
}

```

Question 10

```

#!/usr/bin/perl -w

$ARGV[0] = "-" if (!@ARGV);

foreach $file (@ARGV) {
    open INPUT, "<$file" or die;
    print reverse <INPUT>;
    close INPUT;
}

```