LING/C SC/PSYC 438/538

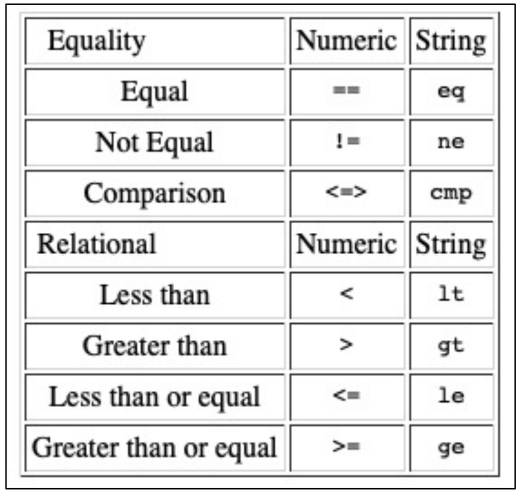
Lecture 6

Sandiway Fong

Today's Topics

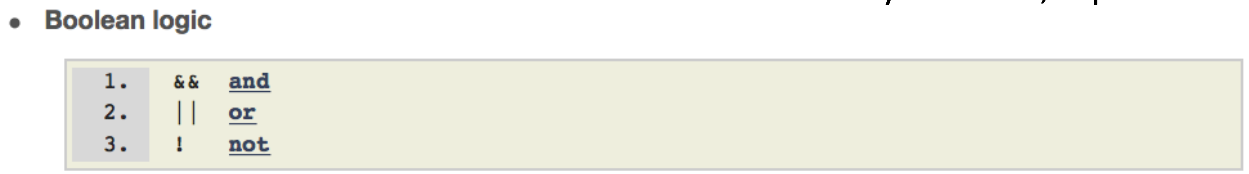
- perlintro:
  - equality, coercion, useful functions on strings
  - file input/output (I/O)
- Homework 5 (due Sunday midnight)

Equality



Equality	Numeric	String
Equal	`==`	`eq`
Not Equal	`!=`	`ne`
Comparison	`<=>`	`cmp`
Relational	Numeric	String
Less than	`<`	`lt`
Greater than	`>`	`gt`
Less than or equal	`<=`	`le`
Greater than or equal	`>=`	`ge`

returns 1 for true and "" for false



- Boolean logic

```
&&  and
||  or
!   not
```

- Conditionals:

  `if` (*comparison*) { *statement* }
  `else` { *statement* }
  `elsif` (*comparison*) { *statement* }

- `if ( @a < 10 ) { print "Small array\n" } else { print "Big array\n" }`
- **Note**:
  - @a here is in **scalar context** = size of array
- `unless ( @a > 10 ) { print "@a\n" }`
- **Note**:
  - if size of array *a* is ≤ 10, it prints the contents of array *a*

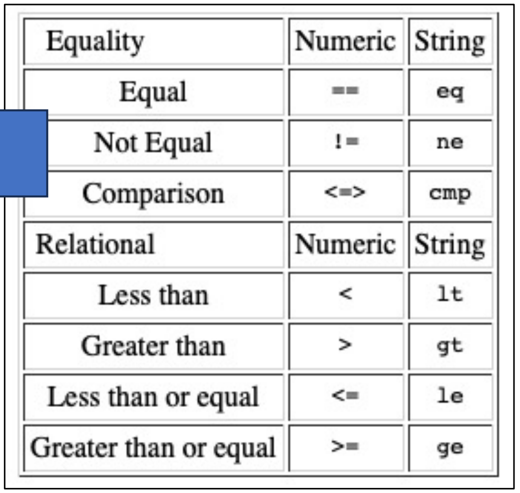
Perl equality (numeric and string)

- What does eq do?
- `perl -e '$x = "windy" eq "Windy"; print "$x\n"'`

- `perl -e 'use feature fc; $x = "windy" eq fc("Windy"); print "$x\n"'`
- 1
- `perl -e '$x = "windy" eq "windy"; print "$x\n"'`
- 1
- `perl -e '$x = 0.0 eq 1; print "$x\n"'`

implicit coercion

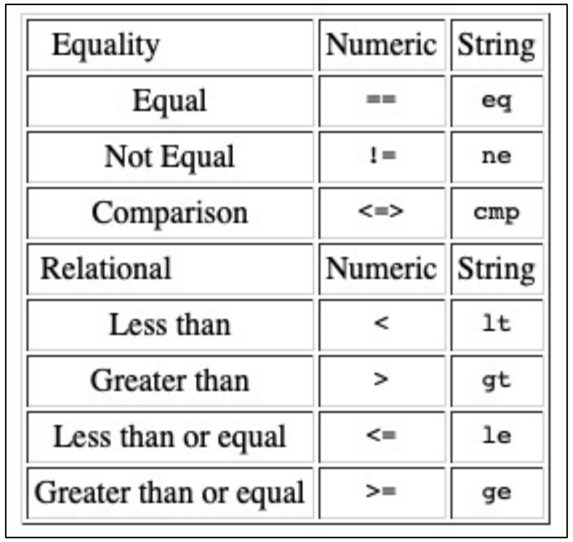
- `perl -e '$x = 0.0 eq 0; print "$x\n"'`
- 1
- `perl -e '$x = 0.0 eq "0"; print "$x\n"'`
- 1
- `perl -e '$x = 0.0 eq "windy"; print "$x\n"'`



Equality	Numeric	String
Equal	==	eq
Not Equal	!=	ne
Comparison	<=>	cmp
Relational	**Numeric**	**String**
Less than	<	lt
Greater than	>	gt
Less than or equal	<=	le
Greater than or equal	>=	ge

Perl equality (numeric and string)

- What does == do?
  - `perl -e '$x = 0.0 == 1; print "$x\n"'`
  - `perl -e '$x = 0.0 == 0; print "$x\n"'`
  - 1
  - `perl -e '$x = 0.0 == "0"; print "$x\n"'`
  - 1
  - `perl -e '$x = 0.0 == "windy"; print "$x\n"'`
  - 1



Equality	Numeric	String
Equal	`==`	`eq`
Not Equal	`!=`	`ne`
Comparison	`<=>`	`cmp`
Relational	Numeric	String
Less than	`<`	`lt`
Greater than	`>`	`gt`
Less than or equal	`<=`	`le`
Greater than or equal	`>=`	`ge`

Perl equality (numeric and string)

- Turn on warnings:

```
$ perl -e 'use warnings; "windy" == 0'
Useless use of numeric eq (==) in void context at -e line 1.
Argument "windy" isn't numeric in numeric eq (==) at -e line 1.
```

- Data type testing:
  - `use Scalar::Util "looks_like_number";`

```
$ perl -e 'use Scalar::Util "looks_like_number"; print
looks_like_number($ARGV[0]), "\n"' 1.23

1

$ perl -e 'use Scalar::Util "looks_like_number"; print
looks_like_number($ARGV[0]), "\n"' windy


$
```

More Implicit Coercion

- Example:
  - the following program

```
my @a = qw(one, two, three);
my $string = @a." is my number";
print "$string\n";
```

Note: qw = quote word, cf.
`("one", "two", "three")`

  - prints

    3 is my number

  - Note: . (*period*) is the string concatenation operator (*scalar context*)

Implicit Coercion

- `perl -e 'print "4" * 4, "\n"'`

```
16
```

```
$ python3
>>> print("4"*4)
4444
>>>
```

- `perl -e 'print "4" x 4, "\n"'` (“x” *is the repetition operator*)

```
4444
```

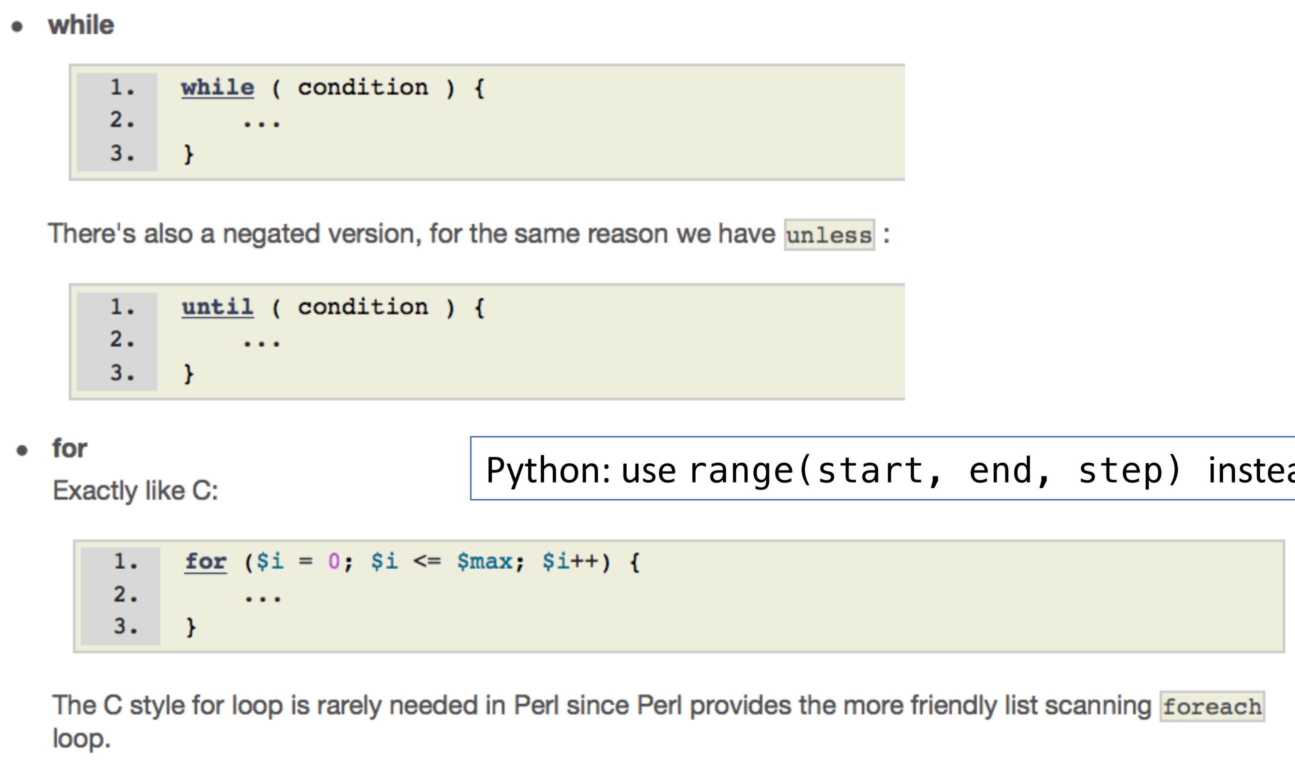
- `perl -e '@a = (4) x 4; print "@a\n"'` (*list context*)

```
4 4 4 4
```

- `@a = (4) x 4`

```
(4, 4, 4, 4)
```

General Looping: while and for



- **while**

```
while ( condition ) {
    ...
}
```

There's also a negated version, for the same reason we have `unless` :

```
until ( condition ) {
    ...
}
```

- **for**

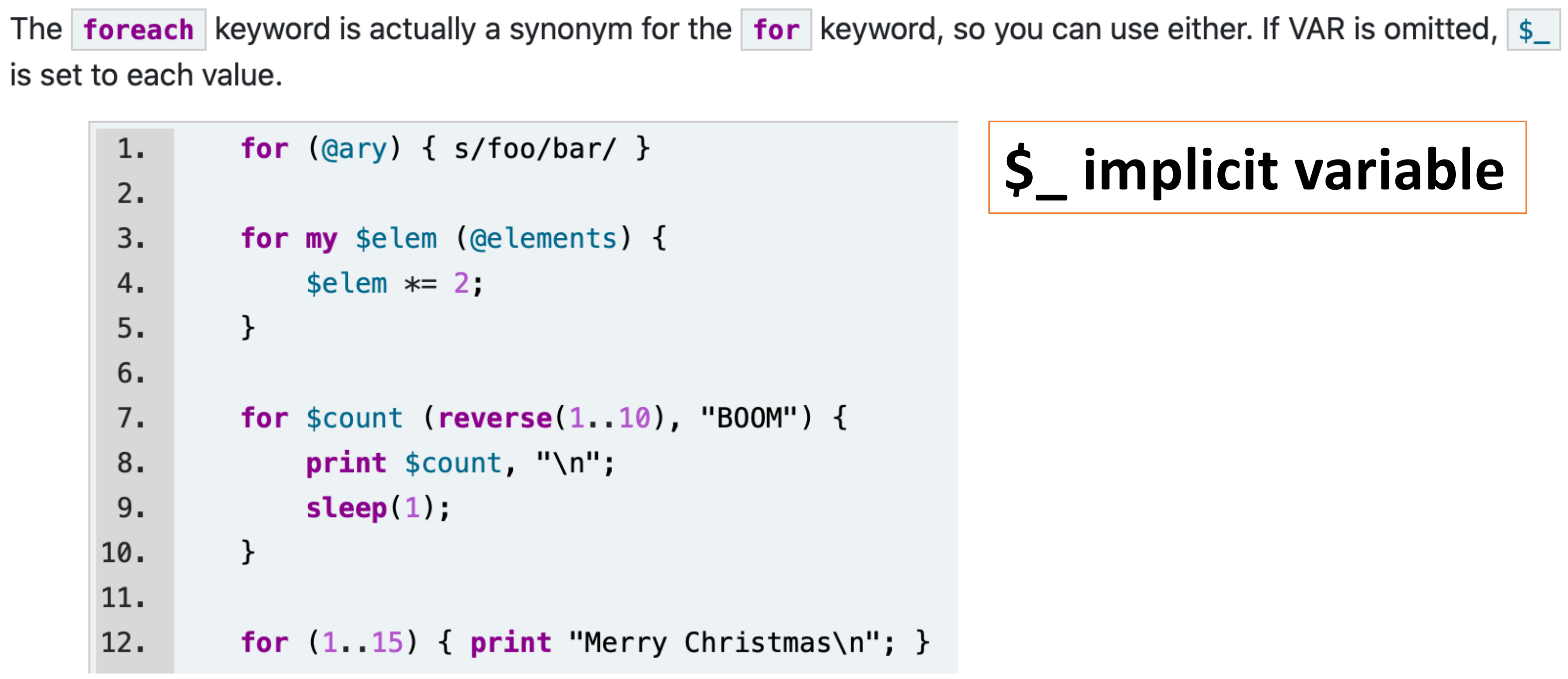
Exactly like C:

Python: use `range(start, end, step)` instead. Or Numpy `arange()`

```
for ($i = 0; $i <= $max; $i++) {
    ...
}
```

The C style for loop is rarely needed in Perl since Perl provides the more friendly list scanning `foreach` loop.

General Looping: foreach



The `foreach` keyword is actually a synonym for the `for` keyword, so you can use either. If VAR is omitted, `$_` is set to each value.

```perl
    for (@ary) { s/foo/bar/ }

    for my $elem (@elements) {
        $elem *= 2;
    }

    for $count (reverse(1..10), "BOOM") {
        print $count, "\n";
        sleep(1);
    }

    for (1..15) { print "Merry Christmas\n"; }
```

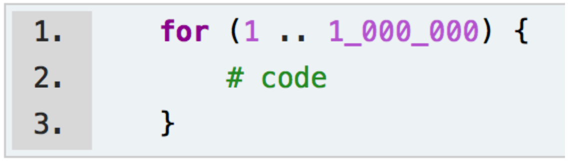
$_ implicit variable

General Looping

- `perl -e 'for (@ARGV) {print $_ * $_, " "}' 1 2 3 4 5`

```
1 4 9 16 25
```

Perl list ranges

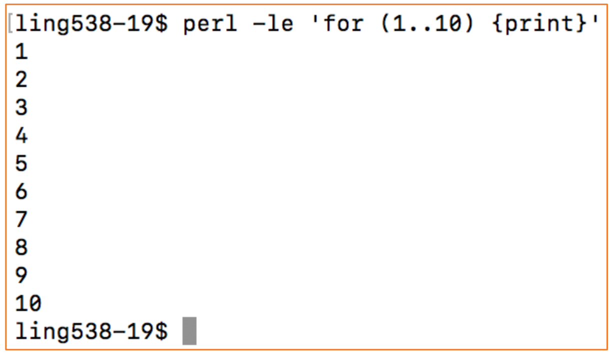


```
for (1 .. 1_000_000) {
    # code
}
```

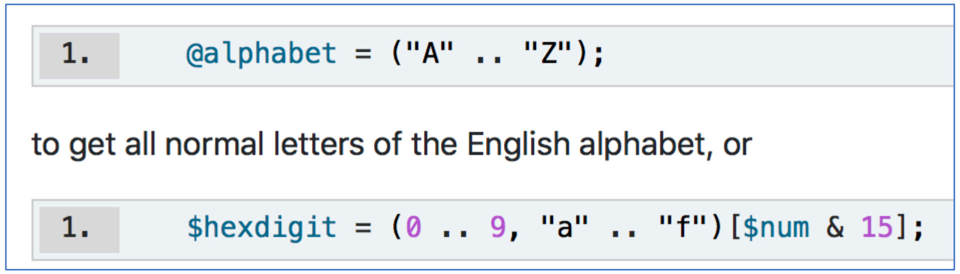
iterates setting **$_** (*the default variable*) from 1, 2, .., 1000000

Python equivalent:

- `for i in range(1,1000001):`
  `  # code`



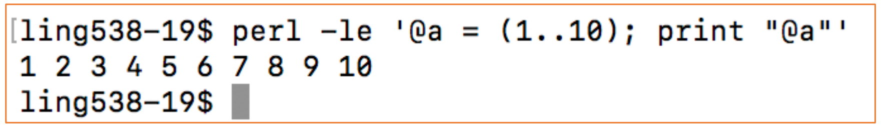
```
[ling538-19$ perl -le 'for (1..10) {print}'
1
2
3
4
5
6
7
8
9
10
ling538-19$ 
```



```
@alphabet = ("A" .. "Z");
```

to get all normal letters of the English alphabet, or

```
$hexdigit = (0 .. 9, "a" .. "f")[$num & 15];
```

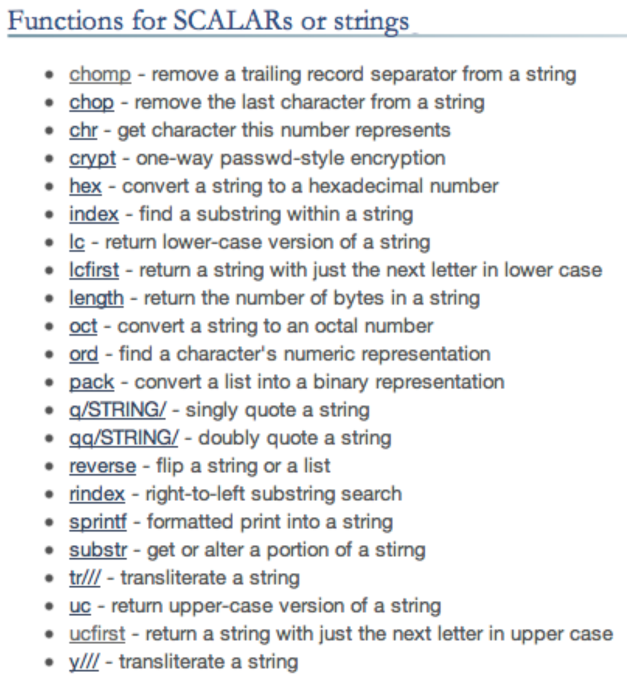


```
[ling538-19$ perl -le '@a = (1..10); print "@a"'
1 2 3 4 5 6 7 8 9 10
ling538-19$ 
```

split

- https://perldoc.perl.org/functions/split.html
- Compare:
  - `@a = split " ", "this is a sentence."`
  - `@a = split //, "this is a sentence."`
- Exercise: what is the size of array @a?

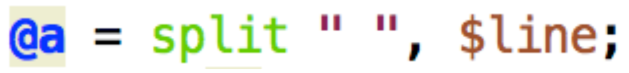
Perl: useful string functions



Functions for SCALARs or strings

- chomp - remove a trailing record separator from a string
- chop - remove the last character from a string
- chr - get character this number represents
- crypt - one-way passwd-style encryption
- hex - convert a string to a hexadecimal number
- index - find a substring within a string
- lc - return lower-case version of a string
- lcfirst - return a string with just the next letter in lower case
- length - return the number of bytes in a string
- oct - convert a string to an octal number
- ord - find a character's numeric representation
- pack - convert a list into a binary representation
- q/STRING/ - singly quote a string
- qq/STRING/ - doubly quote a string
- reverse - flip a string or a list
- rindex - right-to-left substring search
- sprintf - formatted print into a string
- substr - get or alter a portion of a stirng
- tr/// - transliterate a string
- uc - return upper-case version of a string
- ucfirst - return a string with just the next letter in upper case
- y/// - transliterate a string

- chomp (useful with file I/O) vs. chop



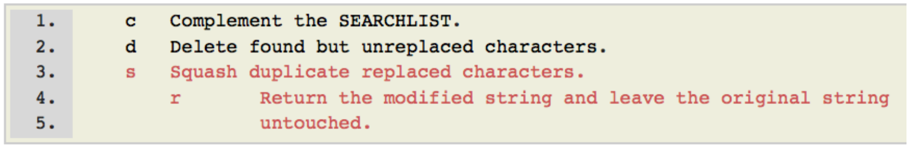
```
@a = split " ", $line;
```

Note: multiple spaces ok with " " variant

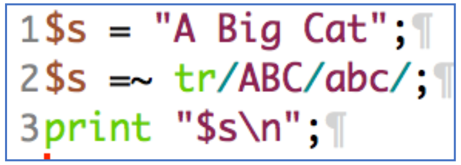
Perl: useful string functions

Transliterate:

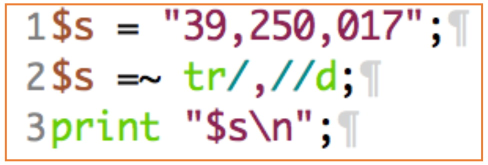
- destructive operation!
- `tr/`*`matchingcharacters`*`/`*`replacementcharacters`*`/`*`modifiers`*
- modifiers are optional:



```
c   Complement the SEARCHLIST.
d   Delete found but unreplaced characters.
s   Squash duplicate replaced characters.
    r       Return the modified string and leave the original string
            untouched.
```



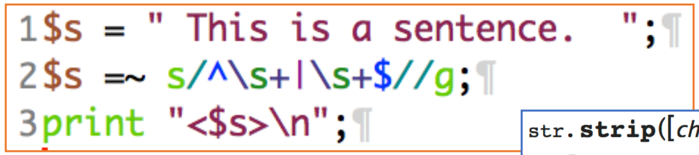
```perl
$s = "A Big Cat";
$s =~ tr/ABC/abc/;
print "$s\n";
```



```perl
$s = "39,250,017";
$s =~ tr/,//d;
print "$s\n";
```

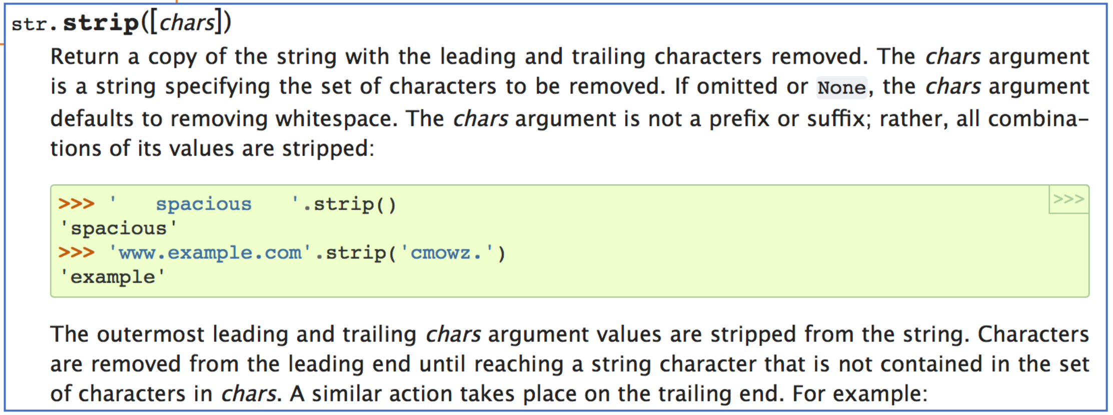
Perl: useful string functions

- Perl doesn't have a built-in t*rim-whitespace-from-both-ends-of-a-string* function.
- Can be mimicked using regex (*more later*)



```perl
$s = " This is a sentence.   ";
$s =~ s/^\s+|\s+$//g;
print "<$s>\n";
```

Python:



str.**strip**([*chars*])

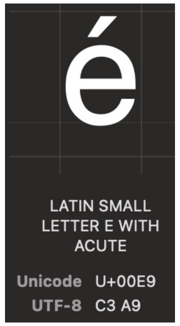
Return a copy of the string with the leading and trailing characters removed. The *chars* argument is a string specifying the set of characters to be removed. If omitted or `None`, the *chars* argument defaults to removing whitespace. The *chars* argument is not a prefix or suffix; rather, all combinations of its values are stripped:

```
>>> '   spacious   '.strip()
'spacious'
>>> 'www.example.com'.strip('cmowz.')
'example'
```

The outermost leading and trailing *chars* argument values are stripped from the string. Characters are removed from the leading end until reaching a string character that is not contained in the set of characters in *chars*. A similar action takes place on the trailing end. For example:

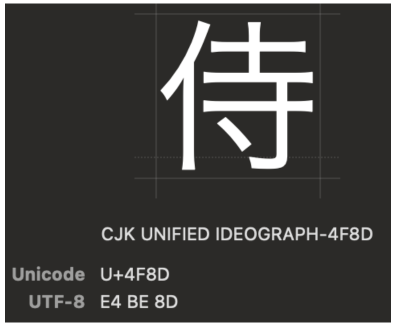
Terminal: newline ("\n") is of length 1 in Perl.

A note on string length


é

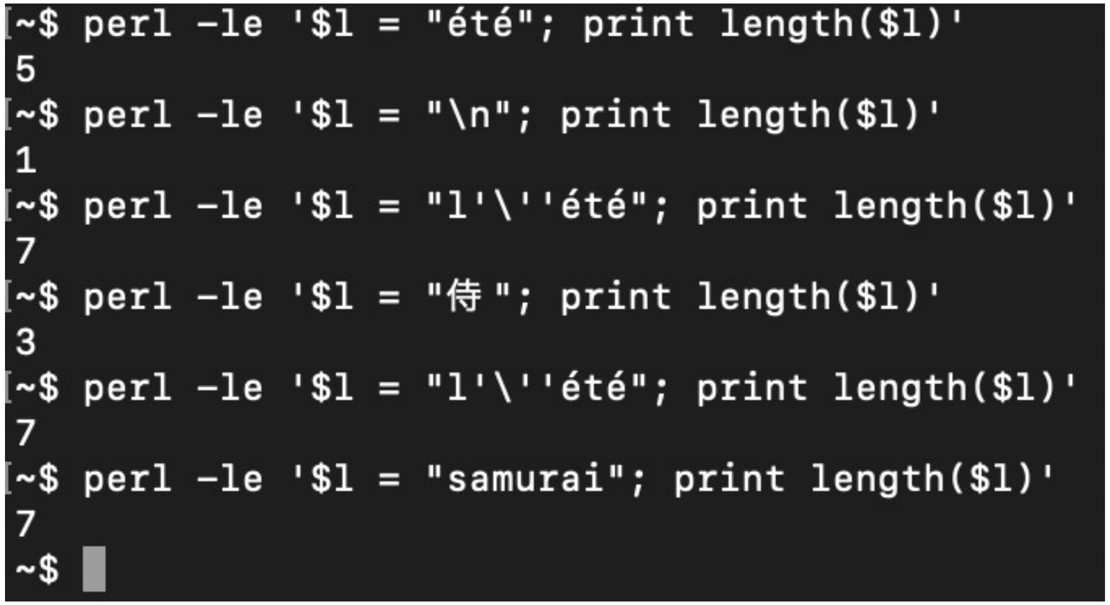
LATIN SMALL LETTER E WITH ACUTE

Unicode U+00E9
UTF-8 C3 A9

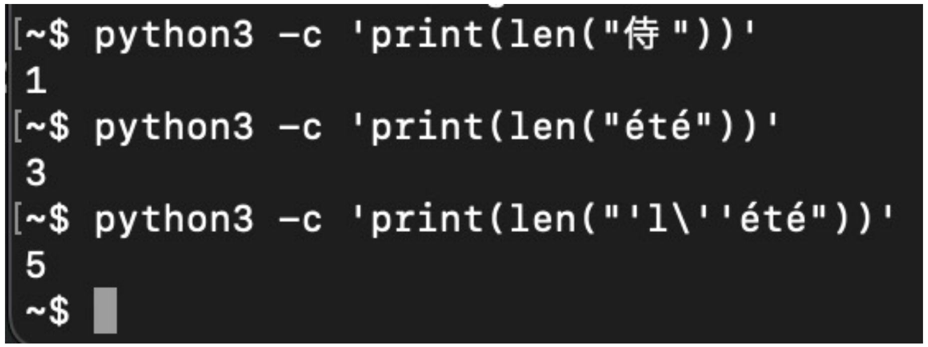

侍

CJK UNIFIED IDEOGRAPH-4F8D

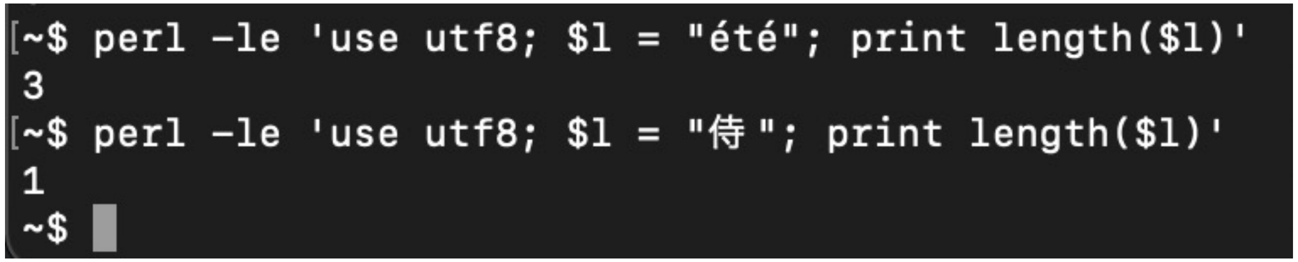
Unicode U+4F8D
UTF-8 E4 BE 8D


```
~$ perl -le '$l = "été"; print length($l)'
5
~$ perl -le '$l = "\n"; print length($l)'
1
~$ perl -le '$l = "l'\''été"; print length($l)'
7
~$ perl -le '$l = "侍"; print length($l)'
3
~$ perl -le '$l = "l'\''été"; print length($l)'
7
~$ perl -le '$l = "samurai"; print length($l)'
7
~$
```

Python: a note on string length



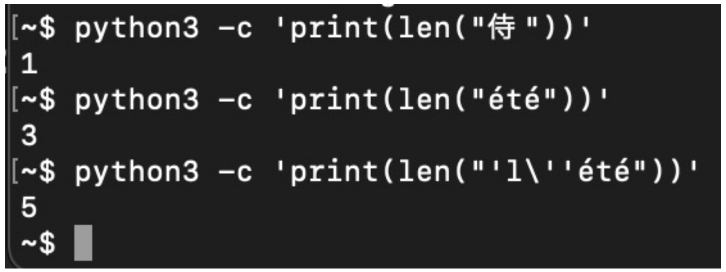
```
[~$ python3 -c 'print(len("侍"))'
1
[~$ python3 -c 'print(len("été"))'
3
[~$ python3 -c 'print(len("'l\''été"))'
5
~$ 
```



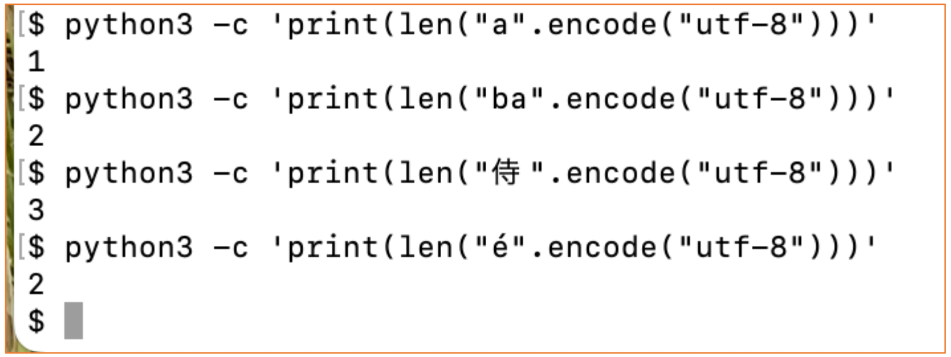
```
[~$ perl -le 'use utf8; $l = "été"; print length($l)'
3
[~$ perl -le 'use utf8; $l = "侍"; print length($l)'
1
~$ 
```

Python: bytes vs. characters

- (len) number of characters *vs.* number of bytes

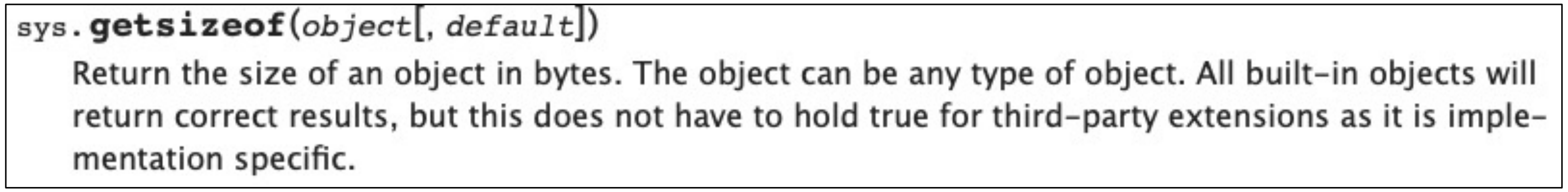


```
[~$ python3 -c 'print(len("侍"))'
1
[~$ python3 -c 'print(len("été"))'
3
[~$ python3 -c 'print(len("'l\''été"))'
5
~$ 
```



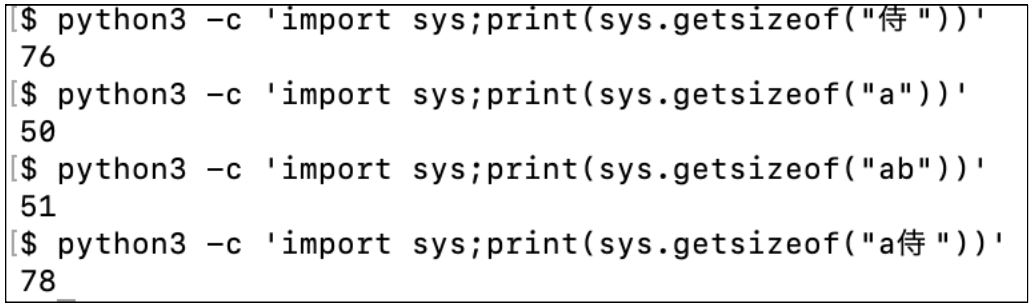
```
[$ python3 -c 'print(len("a".encode("utf-8")))'
1
[$ python3 -c 'print(len("ba".encode("utf-8")))'
2
[$ python3 -c 'print(len("侍".encode("utf-8")))'
3
[$ python3 -c 'print(len("é".encode("utf-8")))'
2
$ 
```

Python: size of object



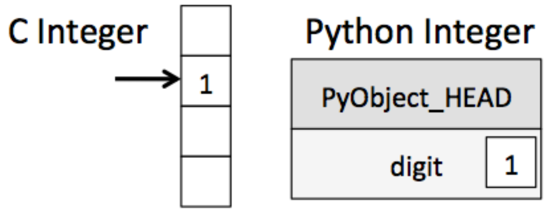
sys.getsizeof(*object*[, *default*])

Return the size of an object in bytes. The object can be any type of object. All built-in objects will return correct results, but this does not have to hold true for third-party extensions as it is implementation specific.

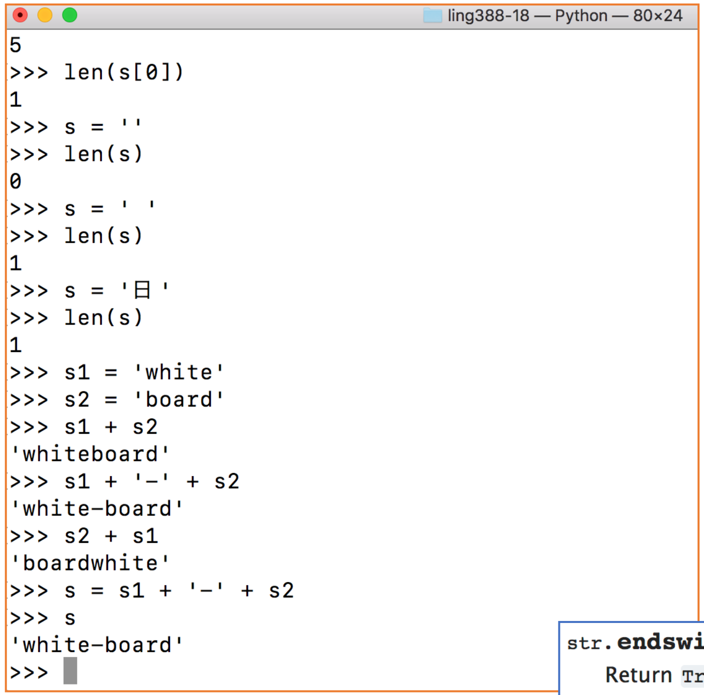


```
[$ python3 -c 'import sys;print(sys.getsizeof("侍"))'
76
[$ python3 -c 'import sys;print(sys.getsizeof("a"))'
50
[$ python3 -c 'import sys;print(sys.getsizeof("ab"))'
51
[$ python3 -c 'import sys;print(sys.getsizeof("a侍"))'
78
```

- Python: inefficient in data storage
  - overhead

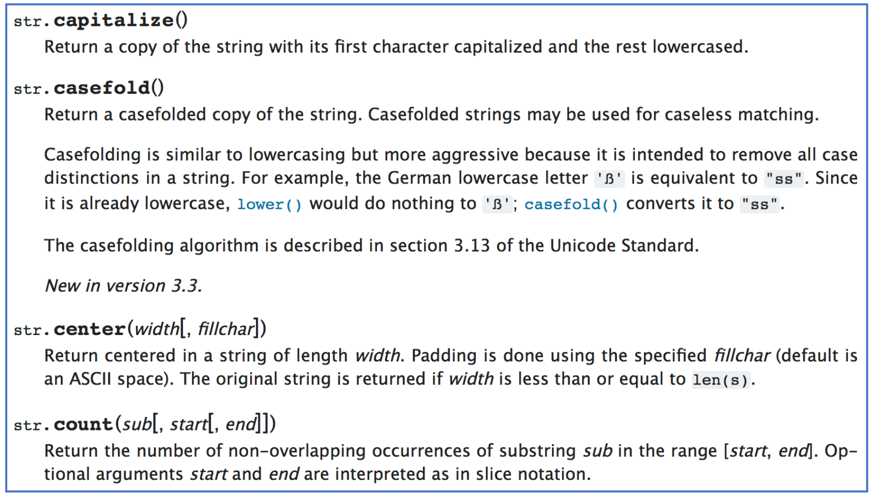


Python: strings


ling388-18 — Python — 80×24

```
5
>>> len(s[0])
1
>>> s = ''
>>> len(s)
0
>>> s = ' '
>>> len(s)
1
>>> s = '日'
>>> len(s)
1
>>> s1 = 'white'
>>> s2 = 'board'
>>> s1 + s2
'whiteboard'
>>> s1 + '-' + s2
'white-board'
>>> s2 + s1
'boardwhite'
>>> s = s1 + '-' + s2
>>> s
'white-board'
>>>
```

- Many methods that work on lists also work on strings


str.**capitalize**()

Return a copy of the string with its first character capitalized and the rest lowercased.

str.**casefold**()

Return a casefolded copy of the string. Casefolded strings may be used for caseless matching.

Casefolding is similar to lowercasing but more aggressive because it is intended to remove all case distinctions in a string. For example, the German lowercase letter `'ß'` is equivalent to `"ss"`. Since it is already lowercase, `lower()` would do nothing to `'ß'`; `casefold()` converts it to `"ss"`.

The casefolding algorithm is described in section 3.13 of the Unicode Standard.

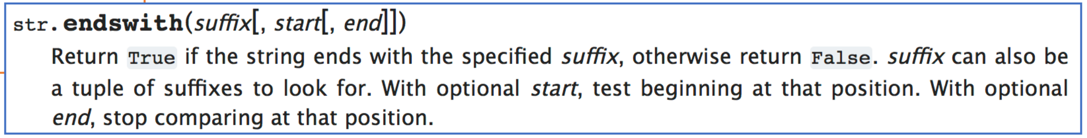
New in version 3.3.

str.**center**(*width*[, *fillchar*])

Return centered in a string of length *width*. Padding is done using the specified *fillchar* (default is an ASCII space). The original string is returned if *width* is less than or equal to `len(s)`.

str.**count**(*sub*[, *start*[, *end*]])

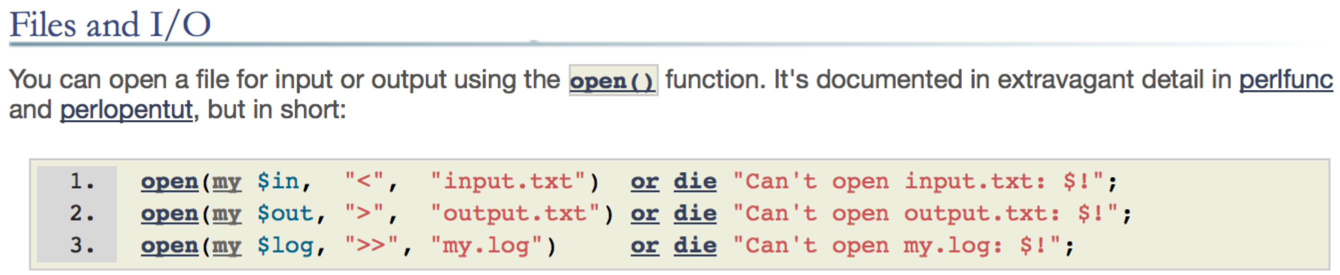
Return the number of non-overlapping occurrences of substring *sub* in the range [*start*, *end*]. Optional arguments *start* and *end* are interpreted as in slice notation.


str.**endswith**(*suffix*[, *start*[, *end*]])

Return `True` if the string ends with the specified *suffix*, otherwise return `False`. *suffix* can also be a tuple of suffixes to look for. With optional *start*, test beginning at that position. With optional *end*, stop comparing at that position.

Perl: file I/O

- Step 1: call open()



Files and I/O

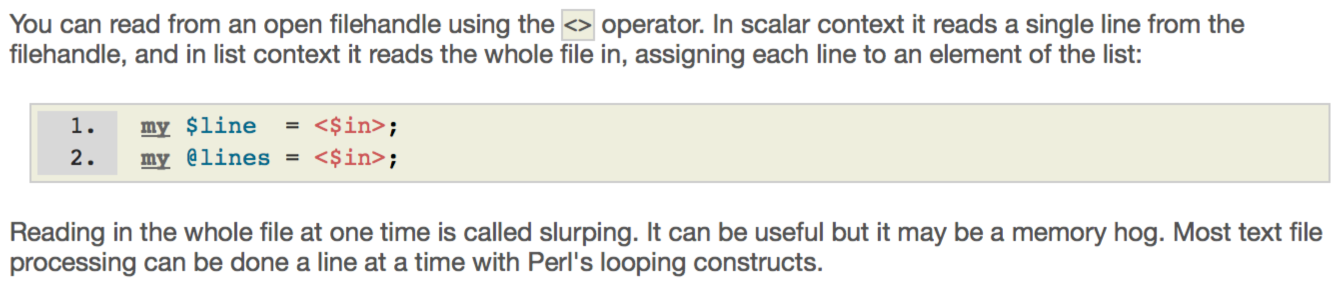
You can open a file for input or output using the `open()` function. It's documented in extravagant detail in perlfunc and perlopentut, but in short:

```perl
open(my $in,  "<",  "input.txt")  or die "Can't open input.txt: $!";
open(my $out, ">",  "output.txt") or die "Can't open output.txt: $!";
open(my $log, ">>", "my.log")     or die "Can't open my.log: $!";
```

Files: must be opened for reading “<“ or writing “>”
(overwrite or append mode “>>”)
Shell syntax: ***I/O redirection*** “<“ “>”
Opening a file creates a **file handle** (Perl variable)
– not to be confused with filename
Supply the **file handle** for read/write

Perl: file I/O

- Step 2: use the <> operator:



You can read from an open filehandle using the `<>` operator. In scalar context it reads a single line from the filehandle, and in list context it reads the whole file in, assigning each line to an element of the list:

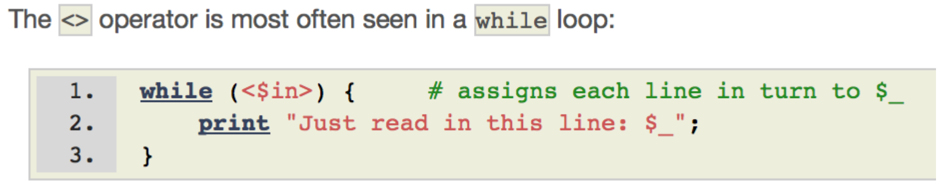
```perl
my $line  = <$in>;
my @lines = <$in>;
```

Reading in the whole file at one time is called slurping. It can be useful but it may be a memory hog. Most text file processing can be done a line at a time with Perl's looping constructs.

$in is the file handle instantiated by the open() call

Perl: file I/O

- Line by line:



The `<>` operator is most often seen in a `while` loop:

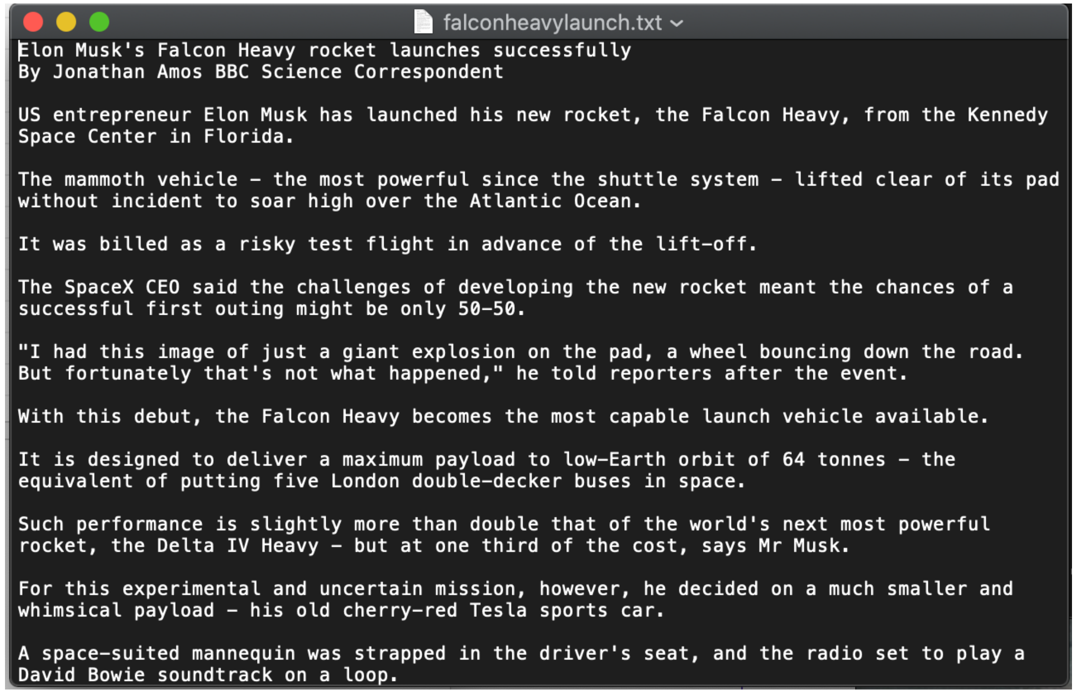
```perl
while (<$in>) {     # assigns each line in turn to $_
    print "Just read in this line: $_";
}
```

```perl
open($txtfile, $ARGV[0]) or die "File $ARGV[0] not found!\n";
while ($line = <$txtfile>) {
    print "$line";
}
close($txtfile)
```

Notes:

1. the command **$line = <$txtfile>** inside the condition reads in a line from the file referenced by the *file handle* **$txtfile**
2. and places that line into the variable **$line** (*including the newline at the end of the line*)
3. At the end of the file, **$line** is just an empty string (equivalent to false).
4. the filename is the first argument to the Perl program (arguments go in **@ARGV**).

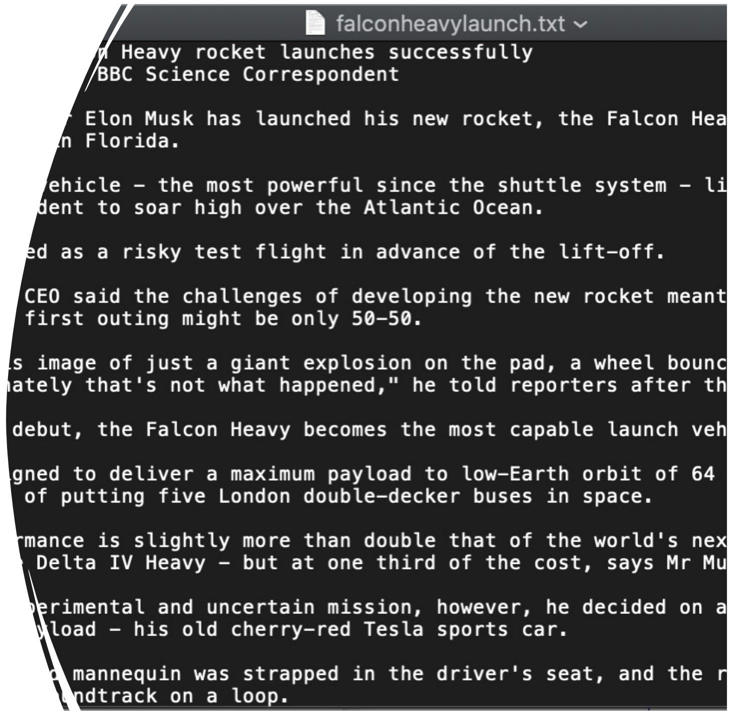
Perl: file I/O



```
Elon Musk's Falcon Heavy rocket launches successfully
By Jonathan Amos BBC Science Correspondent

US entrepreneur Elon Musk has launched his new rocket, the Falcon Heavy, from the Kennedy
Space Center in Florida.

The mammoth vehicle - the most powerful since the shuttle system - lifted clear of its pad
without incident to soar high over the Atlantic Ocean.

It was billed as a risky test flight in advance of the lift-off.

The SpaceX CEO said the challenges of developing the new rocket meant the chances of a
successful first outing might be only 50-50.

"I had this image of just a giant explosion on the pad, a wheel bouncing down the road.
But fortunately that's not what happened," he told reporters after the event.

With this debut, the Falcon Heavy becomes the most capable launch vehicle available.

It is designed to deliver a maximum payload to low-Earth orbit of 64 tonnes - the
equivalent of putting five London double-decker buses in space.

Such performance is slightly more than double that of the world's next most powerful
rocket, the Delta IV Heavy - but at one third of the cost, says Mr Musk.

For this experimental and uncertain mission, however, he decided on a much smaller and
whimsical payload - his old cherry-red Tesla sports car.

A space-suited mannequin was strapped in the driver's seat, and the radio set to play a
David Bowie soundtrack on a loop.
```

Perl: file I/O

- **What does this code do?**
- `perl -e 'open $f, "falconheavylaunch.txt"; while (<$f>) {print((split " ")[0],"\n")}'`
- `perl -e 'open $f, "falconheavylaunch.txt"; while (<$f>) {print((split " ")[0],"\n")}'  | wc -l`
- *reports 49 lines*



```
n Heavy rocket launches successfully
BBC Science Correspondent

 Elon Musk has launched his new rocket, the Falcon Hea
n Florida.

ehicle - the most powerful since the shuttle system - li
dent to soar high over the Atlantic Ocean.

ed as a risky test flight in advance of the lift-off.

CEO said the challenges of developing the new rocket meant
first outing might be only 50-50.

s image of just a giant explosion on the pad, a wheel bounc
ately that's not what happened," he told reporters after th

debut, the Falcon Heavy becomes the most capable launch veh

gned to deliver a maximum payload to low-Earth orbit of 64
 of putting five London double-decker buses in space.

rmance is slightly more than double that of the world's nex
 Delta IV Heavy - but at one third of the cost, says Mr Mu

erimental and uncertain mission, however, he decided on a
load - his old cherry-red Tesla sports car.

 mannequin was strapped in the driver's seat, and the r
ndtrack on a loop.
```

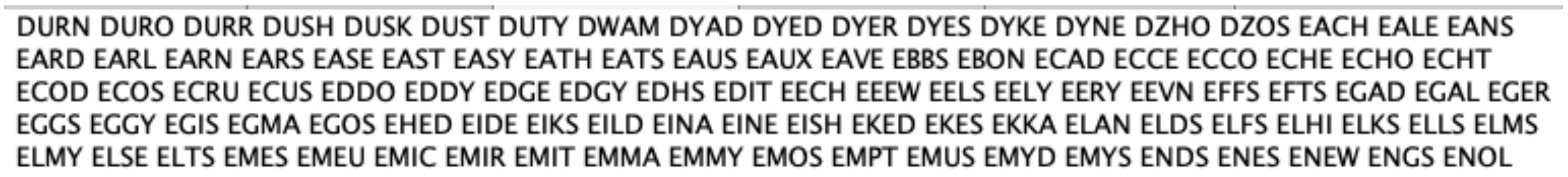
Perl: file I/O

- A bit more:
  - `perl -e 'open $f, "falconheavylaunch.txt"; while (<$f>) {@words = split " "; $sum+=@words}; print $sum'`
- Compare with:

```
wc falconheavylaunch.txt
      49      669     3973 falconheavylaunch.txt
```

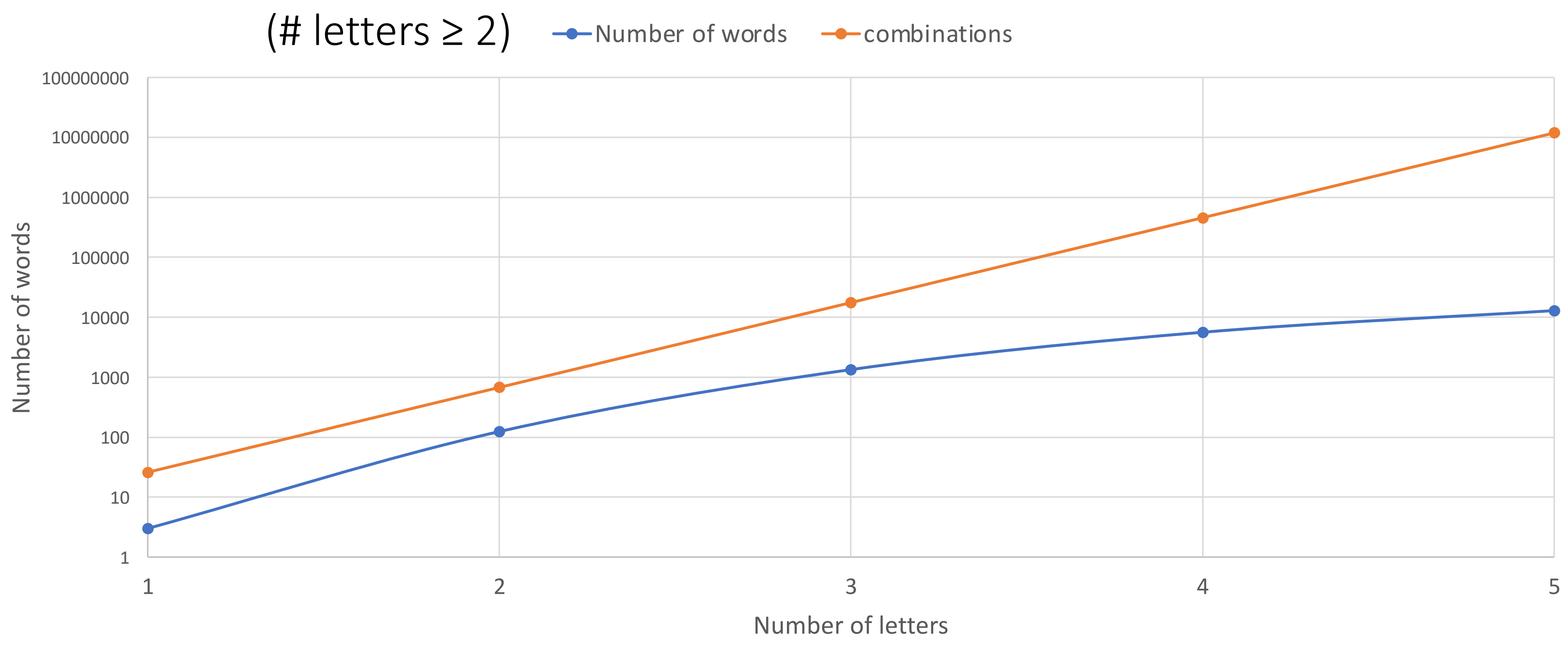
Homework 5

- There are 4 files on the course website:
  - `2letters.txt` words with 2 letters
  - `3letters.txt` *etc.*
  - `4letters.txt`
  - `5letters.txt`



```
DURN DURO DURR DUSH DUSK DUST DUTY DWAM DYAD DYED DYER DYES DYKE DYNE DZHO DZOS EACH EALE EANS
EARD EARL EARN EARS EASE EAST EASY EATH EATS EAUS EAUX EAVE EBBS EBON ECAD ECCE ECCO ECHE ECHO ECHT
ECOD ECOS ECRU ECUS EDDO EDDY EDGE EDGY EDHS EDIT EECH EEEW EELS EELY EERY EEVN EFFS EFTS EGAD EGAL EGER
EGGS EGGY EGIS EGMA EGOS EHED EIDE EIKS EILD EINA EINE EISH EKED EKES EKKA ELAN ELDS ELFS ELHI ELKS ELLS ELMS
ELMY ELSE ELTS EMES EMEU EMIC EMIR EMIT EMMA EMMY EMOS EMPT EMUS EMYD EMYS ENDS ENES ENEW ENGS ENOL
```

A particular set of Scrabble Words



Homework 5

- Question 1:
  - write a Perl one-liner or a Perl program that reads in one of the *n*`letter.txt` files and prints out the number of words.
  - HINT *for new programmers*: use `$ARGV[0]`. use `split`. Store the words in an array. Print the size of the array.
  - Show your program working on each *n*`letter.txt` for *n* = 2, 3, 4 and 5 at the terminal, e.g. screenshot.
  - What numbers did you get for *n* = 2, 3, 4 and 5?

Homework 5

- Question 2:
  - Which words in the *n*`letter.txt` files spell the same forwards as backwards?
  - Examples: YAY, WOW, DEED, NAAN
  - write a Perl one-liner or a Perl program that reads in one of the *n*`letter.txt` files and prints out the words that satisfy this condition.
  - Show your program working on each *n*`letter.txt` for *n* = 2, 3, 4 and 5 at the terminal, e.g. screenshot.
  - Report how many words you got for *n* = 2, 3, 4 and 5?
  - HINT *for new programmers*: use `foreach` to loop over your array (for each word), use `reverse` and eq to test.

Homework 5

- Instructions:
  - Put all your answers, screenshots in one PDF document
    - (*not* Word .docx or .doc)
  - email to me (sandiway@arizona.edu)
  - subject line: 438/538 Homework 5 YOUR NAME
  - Due date: by midnight Sunday