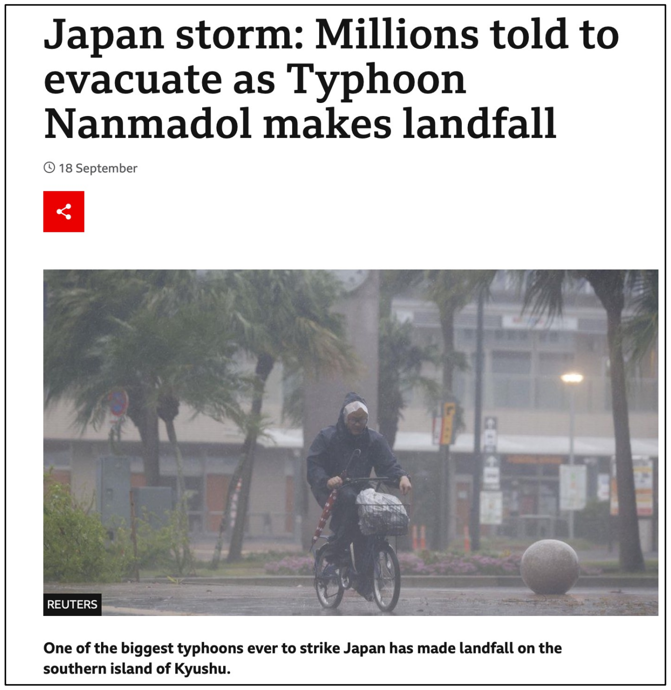
LING/C SC/PSYC 438/538

Lecture 6

Sandiway Fong

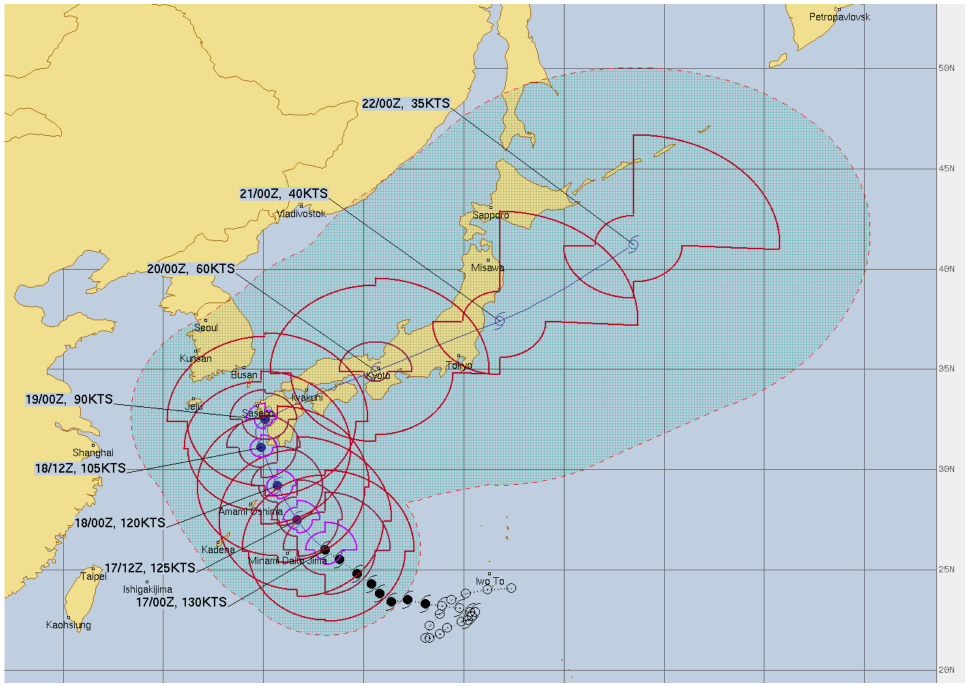
Last week



Japan storm: Millions told to evacuate as Typhoon Nanmadol makes landfall

18 September

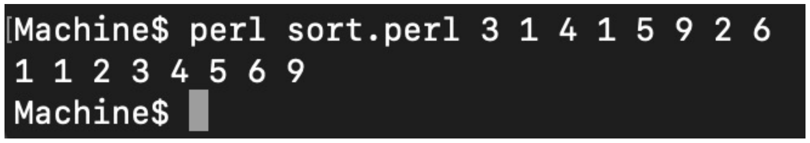
One of the biggest typhoons ever to strike Japan has made landfall on the southern island of Kyushu.



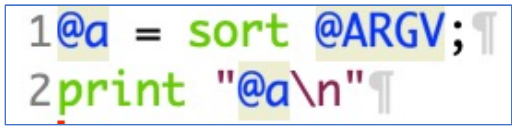
Today's Topics

- Homework 5 Review
- More Perl (with some comparisons to Python)
    - Hope you have been reading perlintro!

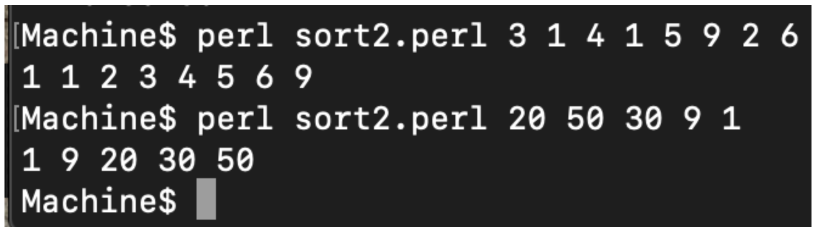
Homework 5: Question 2 Review



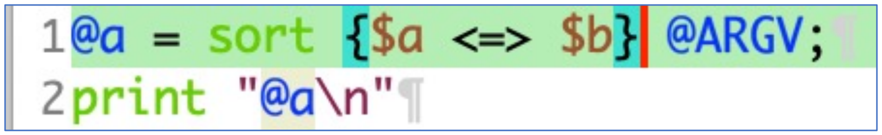
```
Machine$ perl sort.perl 3 1 4 1 5 9 2 6
1 1 2 3 4 5 6 9
Machine$ 
```



```
@a = sort @ARGV;
print "@a\n"
```

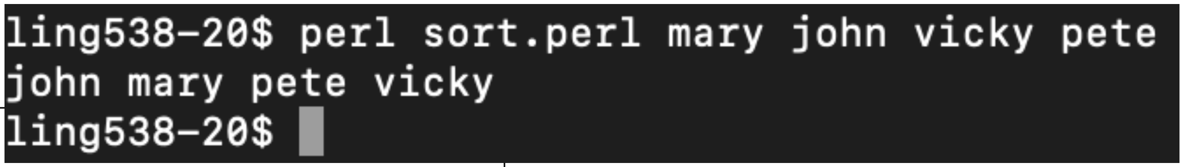


```
Machine$ perl sort2.perl 3 1 4 1 5 9 2 6
1 1 2 3 4 5 6 9
Machine$ perl sort2.perl 20 50 30 9 1
1 9 20 30 50
Machine$ 
```

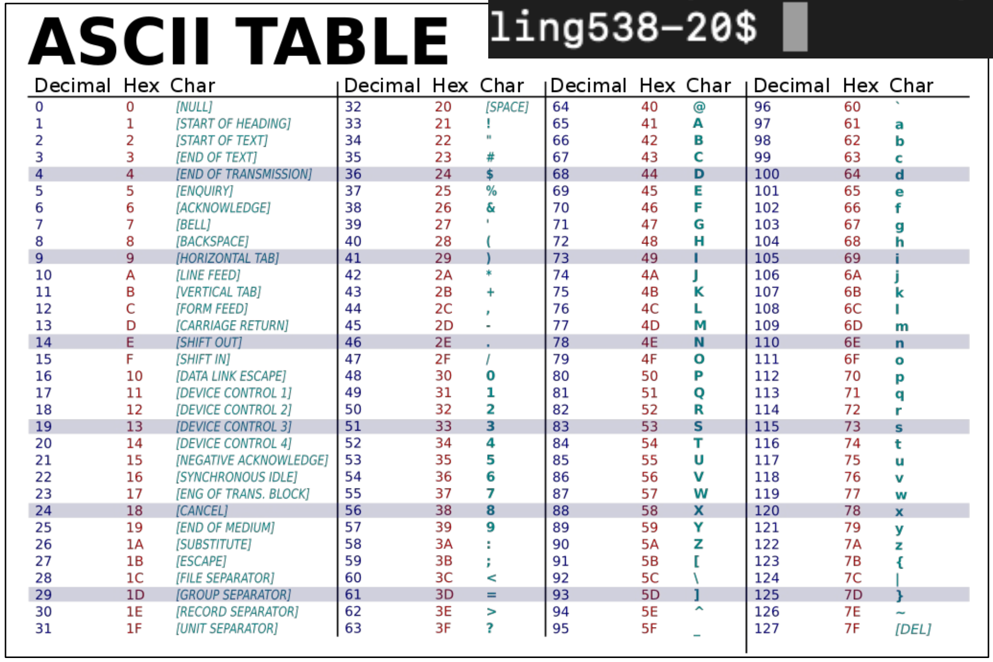


```
@a = sort {$a <=> $b} @ARGV;
print "@a\n"
```

Homework 5: Question 2 Review



```
ling538-20$ perl sort.perl mary john vicky pete
john mary pete vicky
ling538-20$ 
```

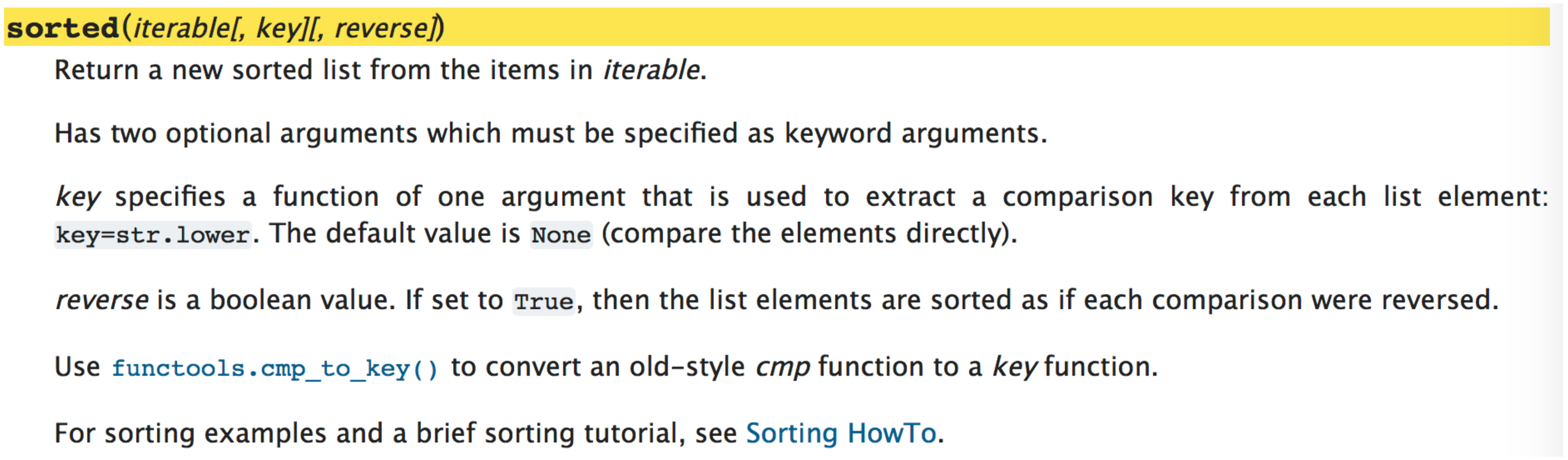


ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

asciitable.com

Python: sorting



sorted(*iterable[, key][, reverse]*)

Return a new sorted list from the items in *iterable*.

Has two optional arguments which must be specified as keyword arguments.

key specifies a function of one argument that is used to extract a comparison key from each list element: `key=str.lower`. The default value is `None` (compare the elements directly).

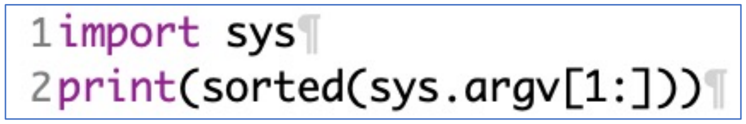
reverse is a boolean value. If set to `True`, then the list elements are sorted as if each comparison were reversed.

Use `functools.cmp_to_key()` to convert an old-style *cmp* function to a *key* function.

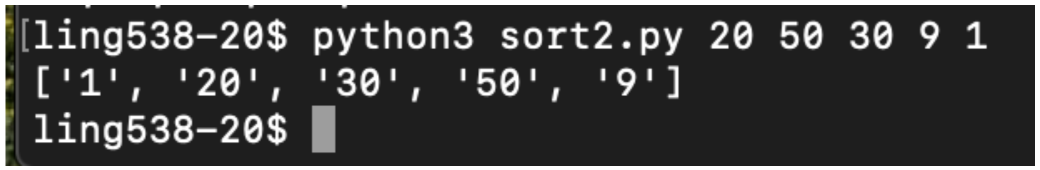
For sorting examples and a brief sorting tutorial, see Sorting HowTo.

Homework 5 Review

sort2.py

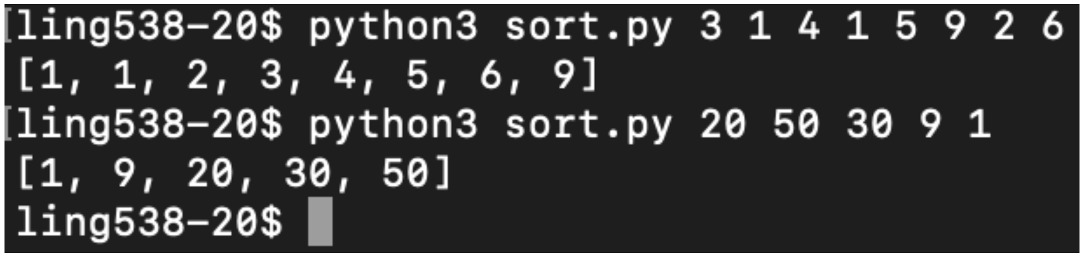


```
import sys
print(sorted(sys.argv[1:]))
```



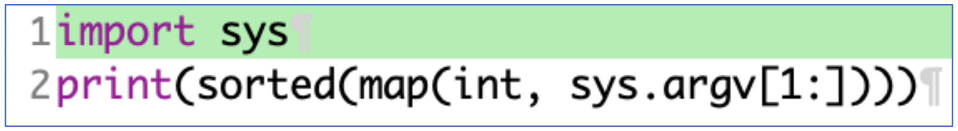
```
ling538-20$ python3 sort2.py 20 50 30 9 1
['1', '20', '30', '50', '9']
ling538-20$ 
```

Homework 5 Review

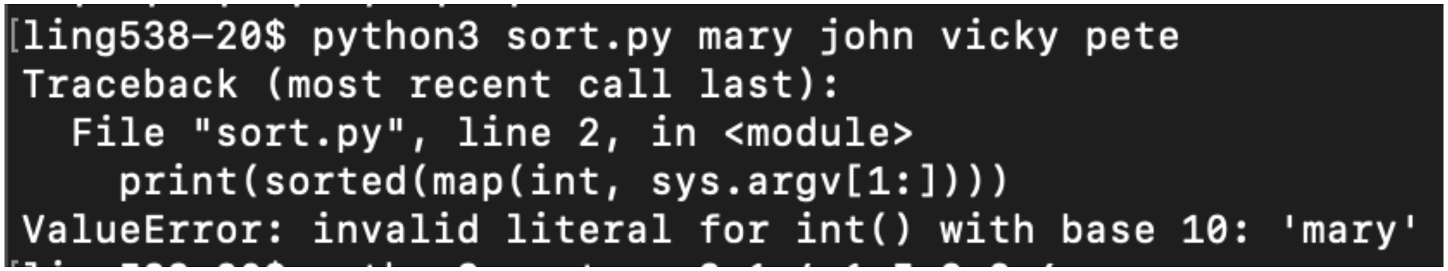


```
ling538-20$ python3 sort.py 3 1 4 1 5 9 2 6
[1, 1, 2, 3, 4, 5, 6, 9]
ling538-20$ python3 sort.py 20 50 30 9 1
[1, 9, 20, 30, 50]
ling538-20$
```

sort.py



```python
import sys
print(sorted(map(int, sys.argv[1:])))
```



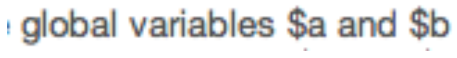
```
ling538-20$ python3 sort.py mary john vicky pete
Traceback (most recent call last):
  File "sort.py", line 2, in <module>
    print(sorted(map(int, sys.argv[1:])))
ValueError: invalid literal for int() with base 10: 'mary'
```

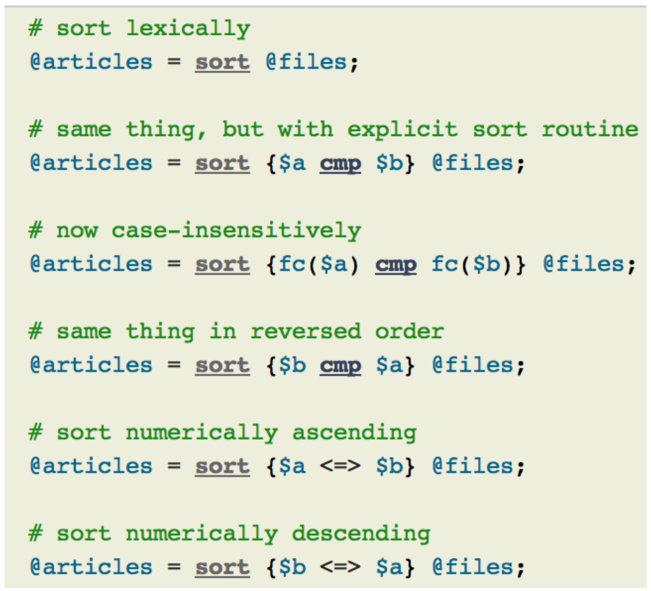
Perl Arrays: sorting revisited

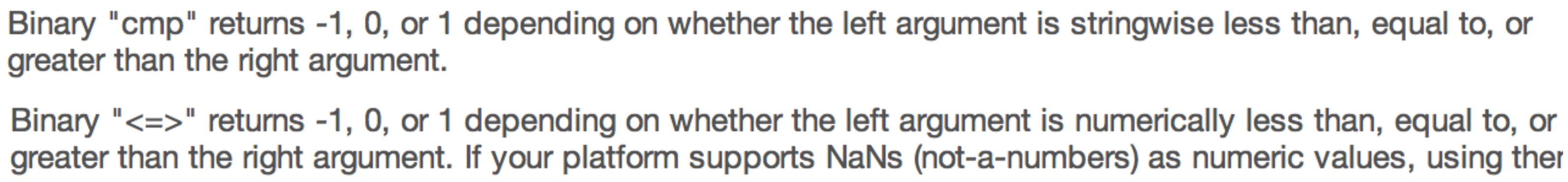
Numeric sort?

- **Idea**: to pass an inline comparison function as parameter…
- **Note**: function fc (fold case)

```
use feature 'fc';
```

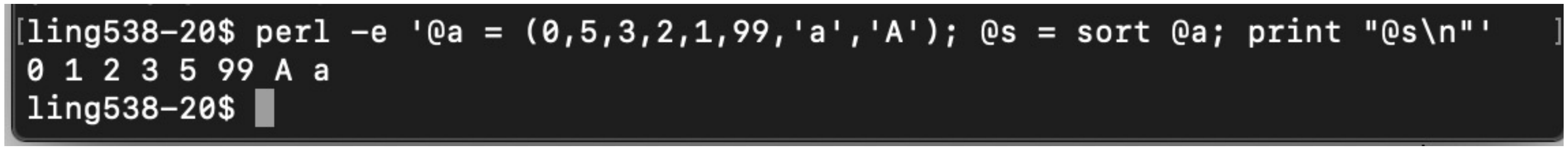

global variables $a and $b


```
# sort lexically
@articles = sort @files;

# same thing, but with explicit sort routine
@articles = sort {$a cmp $b} @files;

# now case-insensitively
@articles = sort {fc($a) cmp fc($b)} @files;

# same thing in reversed order
@articles = sort {$b cmp $a} @files;

# sort numerically ascending
@articles = sort {$a <=> $b} @files;

# sort numerically descending
@articles = sort {$b <=> $a} @files;
```

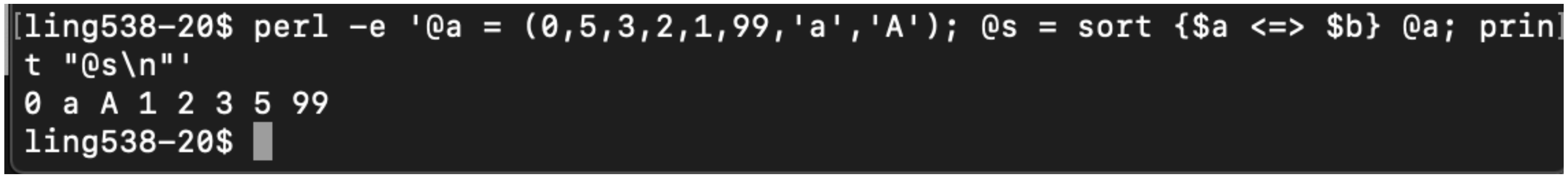

Binary "cmp" returns -1, 0, or 1 depending on whether the left argument is stringwise less than, equal to, or greater than the right argument.

Binary "<=>" returns -1, 0, or 1 depending on whether the left argument is numerically less than, equal to, or greater than the right argument. If your platform supports NaNs (not-a-numbers) as numeric values, using ther

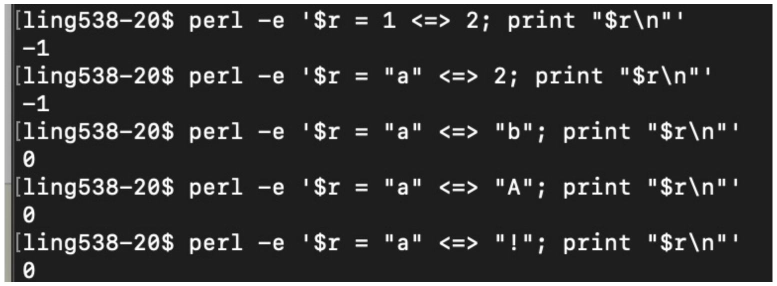
Perl Arrays: mixed sorting



```
[ling538-20$ perl -e '@a = (0,5,3,2,1,99,'a','A'); @s = sort @a; print "@s\n"'    ]
0 1 2 3 5 99 A a
ling538-20$
```

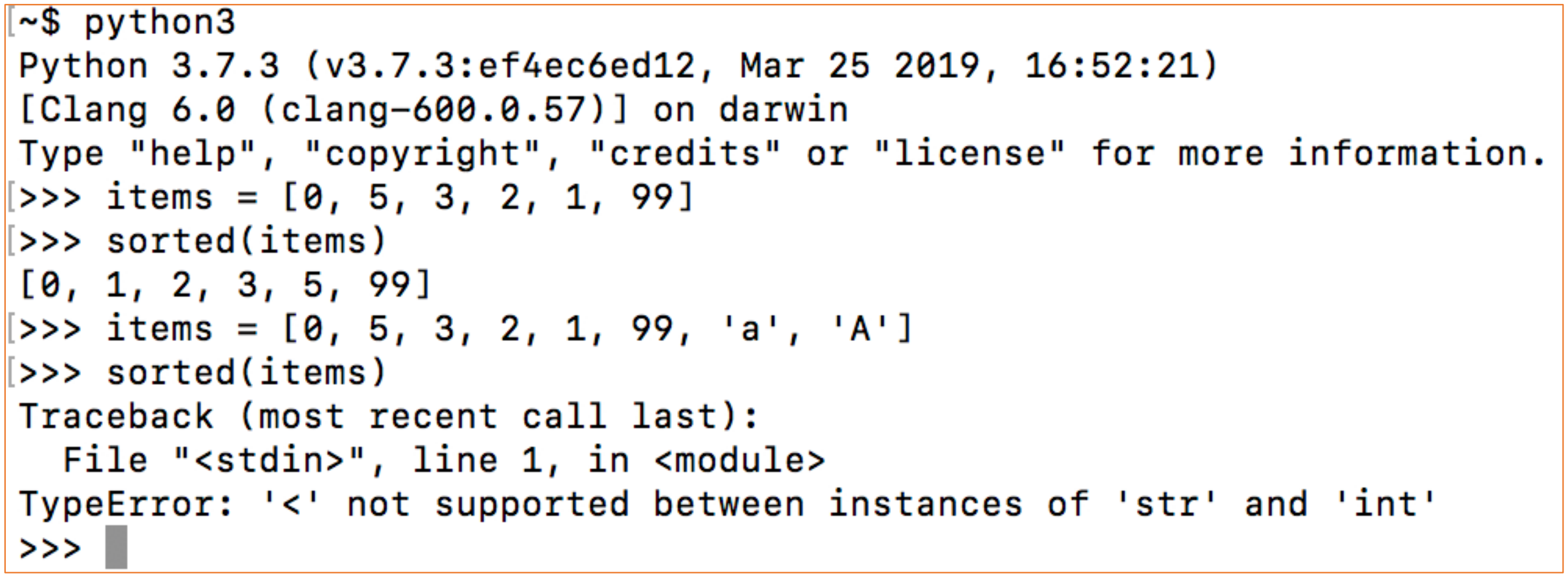


```
[ling538-20$ perl -e '@a = (0,5,3,2,1,99,'a','A'); @s = sort {$a <=> $b} @a; prin]
t "@s\n"'
0 a A 1 2 3 5 99
ling538-20$
```



```
[ling538-20$ perl -e '$r = 1 <=> 2; print "$r\n"'
-1
[ling538-20$ perl -e '$r = "a" <=> 2; print "$r\n"'
-1
[ling538-20$ perl -e '$r = "a" <=> "b"; print "$r\n"'
0
[ling538-20$ perl -e '$r = "a" <=> "A"; print "$r\n"'
0
[ling538-20$ perl -e '$r = "a" <=> "!"; print "$r\n"'
0
```

Python: sorting



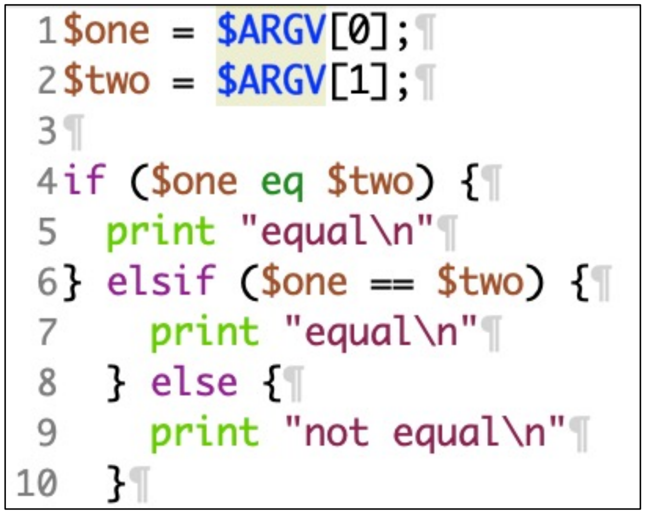
```
~$ python3
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 16:52:21)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> items = [0, 5, 3, 2, 1, 99]
>>> sorted(items)
[0, 1, 2, 3, 5, 99]
>>> items = [0, 5, 3, 2, 1, 99, 'a', 'A']
>>> sorted(items)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '<' not supported between instances of 'str' and 'int'
>>> 
```

Homework 5: Question 1 Review

Exercise highlights **Implicit Coercion (of datatypes)**

- test whether they're equal or not, e.g.
  - `$ARGV[0] $ARGV[1]`
  - `perl hw5q1.perl 1 1.0`
  - `equal`
  - `perl hw5q1.perl 1 0.1e1`
  - `equal`
  - `perl hw5q1.perl windy Windy`
  - `not equal`
  - What happens with this? `perl hw5.perl`

Homework 5: Question 1 Review

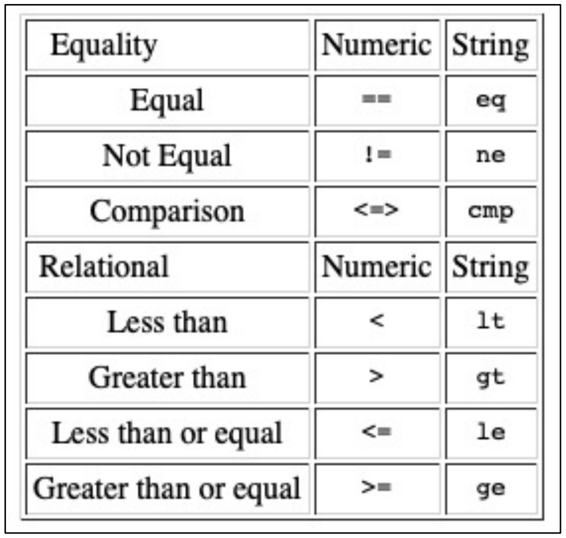


```
$one = $ARGV[0];
$two = $ARGV[1];

if ($one eq $two) {
  print "equal\n"
} elsif ($one == $two) {
    print "equal\n"
  } else {
    print "not equal\n"
  }
```

- I'd expect you to write something like this
  - You don't know the data type, number or string
  - test string equality first
  - if it fails, test number equality
- But if you do this, there's a surprise

Perl equality (numeric and string)

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

Homework 5: Question 1 Review

- Here's the problem:

```
perl hw5q1.perl windy windy
equal
perl hw5q1.perl windy Windy
equal
perl hw5q1.perl windy calm
equal
```

- A real headache
- Solution?
  - either test for data type first manually
    - `use Scalar::Util "looks_like_number";`
    - `looks_like_number($ARGV[0])`
  - or turn on warnings!
    - `use warnings;`

```
perl hw5q1w.perl windy 0.0
Argument "windy" isn't numeric in
numeric eq (==) at hw5q1w.perl line 7.
equal
```

More Implicit Coercion

- Example:
  - the following program prints
    3 is my number
  - . is the string concatenation operator

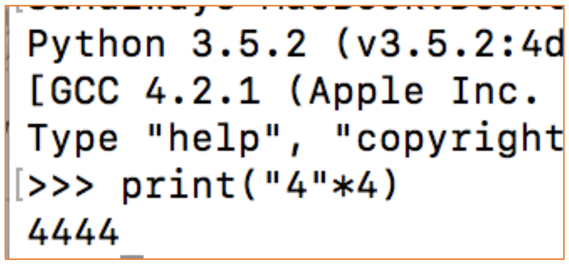
```
my @a = qw(one, two, three);
my $string = @a." is my number";
print "$string\n";
```

Note: qw = quote word

Implicit Coercion

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

16



```
Python 3.5.2 (v3.5.2:4d
[GCC 4.2.1 (Apple Inc.
Type "help", "copyright
>>> 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"' (*in list context*)

4 4 4 4

- @a = (4) x 4

(4, 4, 4, 4)

split

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

Conditionals and Looping

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

- **Ungraded Exercise:**
  - look up the equivalents in Python (www.python.org)
  - do they always exist?

Conditionals and Looping

- **Numeric comparison**

```
==  equality
!=  inequality
<   less than
>   greater than
<=  less than or equal
>=  greater than or equal
```

- **String comparison**

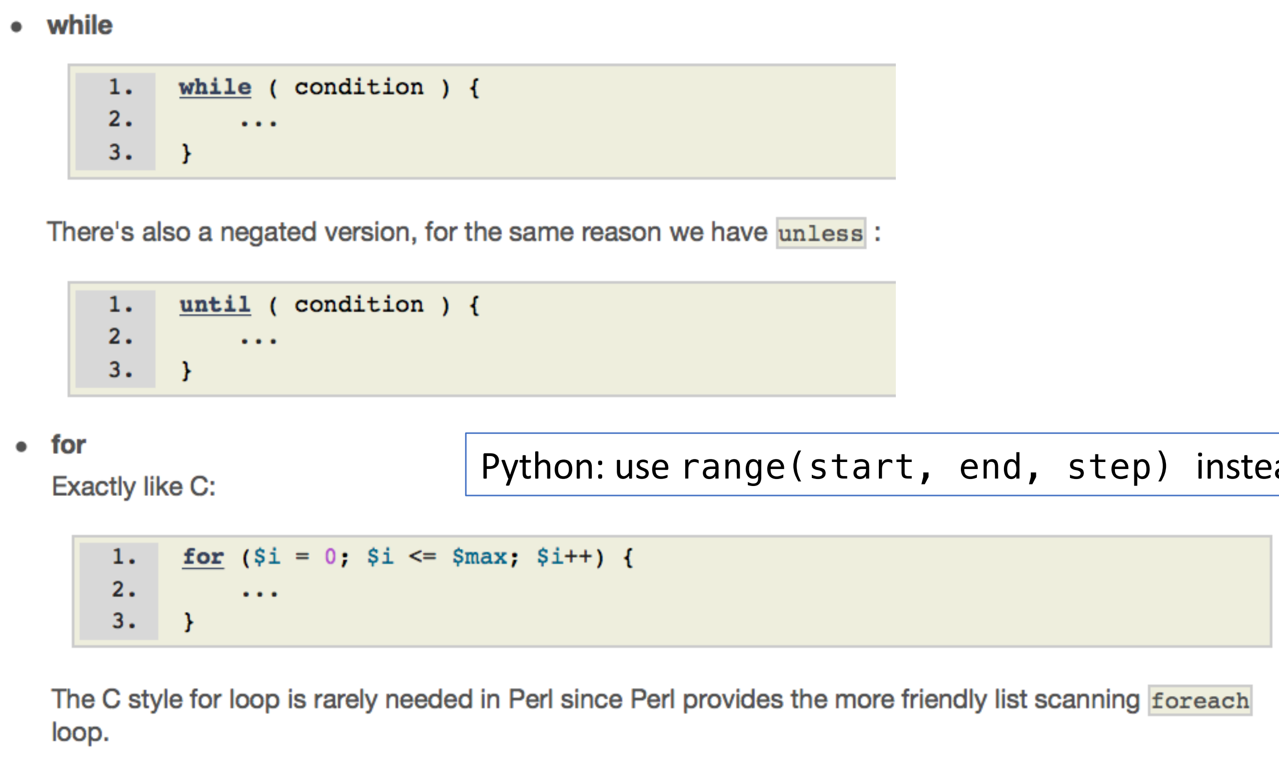
```
eq  equality
ne  inequality
lt  less than
gt  greater than
le  less than or equal
ge  greater than or equal
```

(Why do we have separate numeric and string comparisons? Because we don't have special variable types, and Perl needs to know whether to sort numerically (where 99 is less than 100) or alphabetically (where 100 comes before 99).

- **Boolean logic**

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

General Looping



- **while**

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

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

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

- **for**

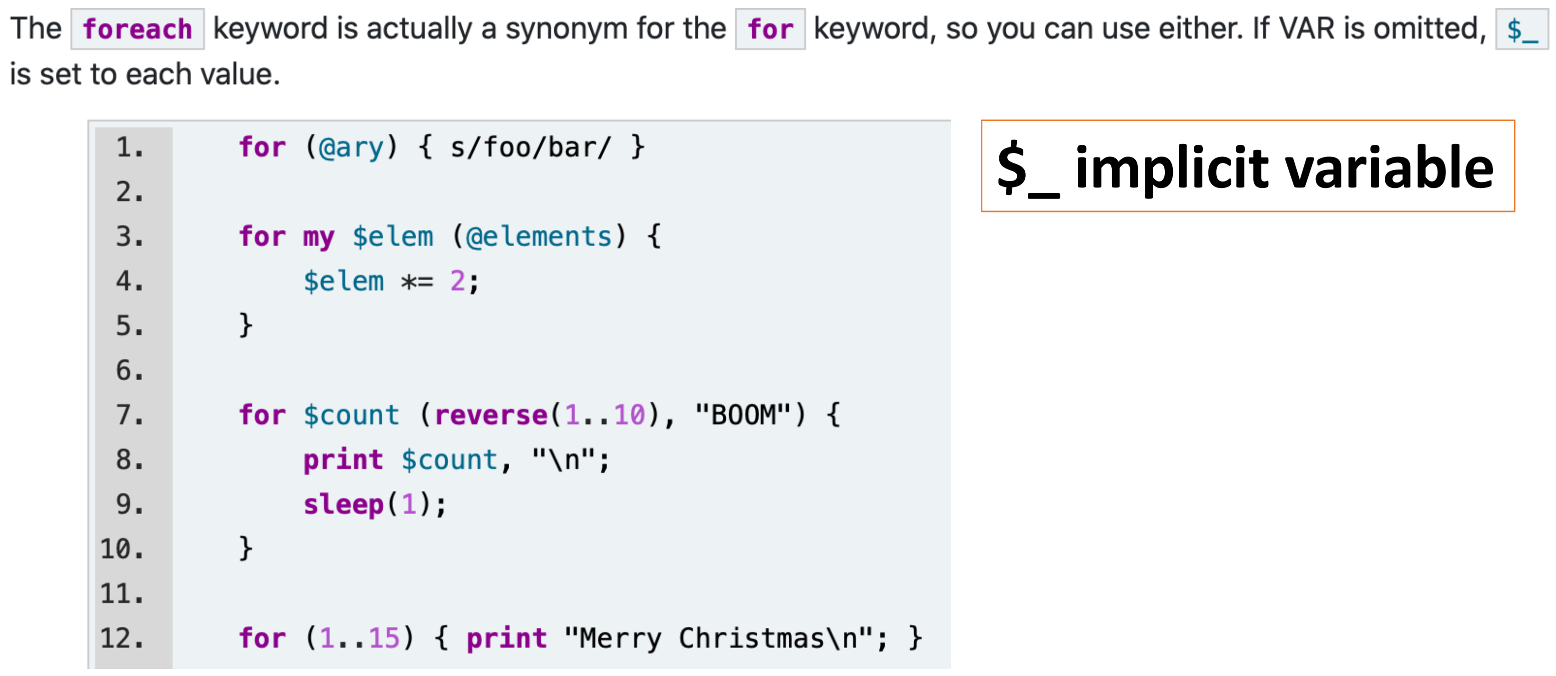
Exactly like C:

```
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.

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

General Looping



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"; }
```

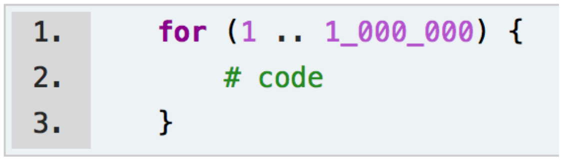
$_ implicit variable

General Looping

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

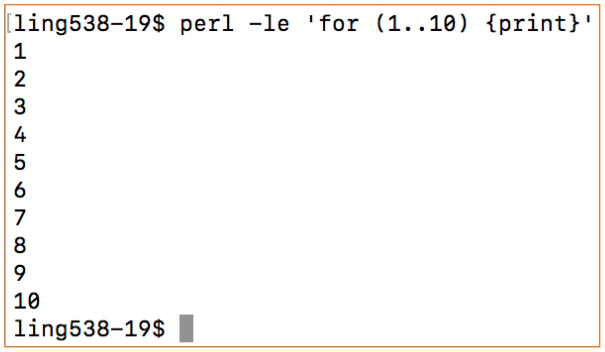
```
1 4 9 16 25
```

Perl list ranges



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

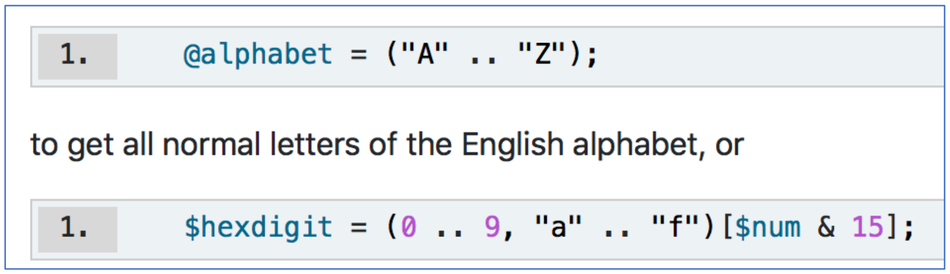
iterates setting **$_** (*the default variable*)
from 1, 2, .., 1000000



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

Python equivalent:

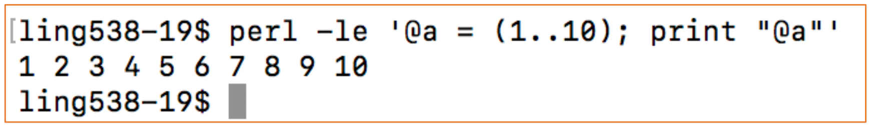
- `for i in range(1,1000001):`
  `   # code`



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

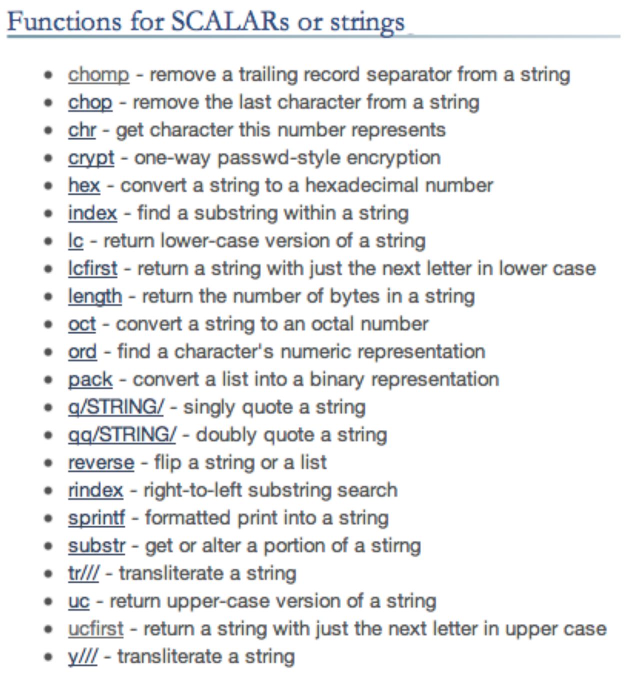
to get all normal letters of the English alphabet, or

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



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

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

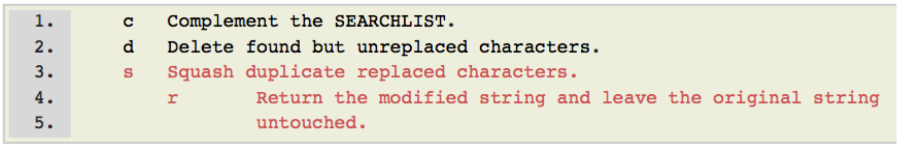
```
@a = split " ", $line;
```

Note: multiple spaces ok with " " variant

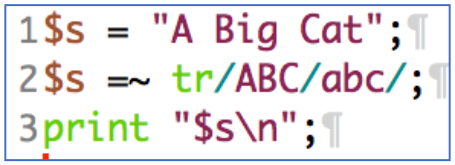
Perl: useful string functions

Transliterate:

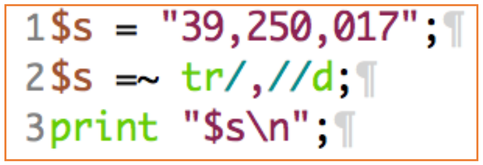
- destructive operation!
- `tr/`*`matchingcharacters`*`/`*`replacementcharacters`*`/`*`modifiers`*
- modifiers are optional:



```
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.
```



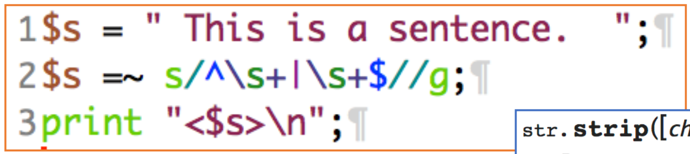
```perl
$s = "A Big Cat";
$s =~ tr/ABC/abc/;
print "$s\n";
```



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

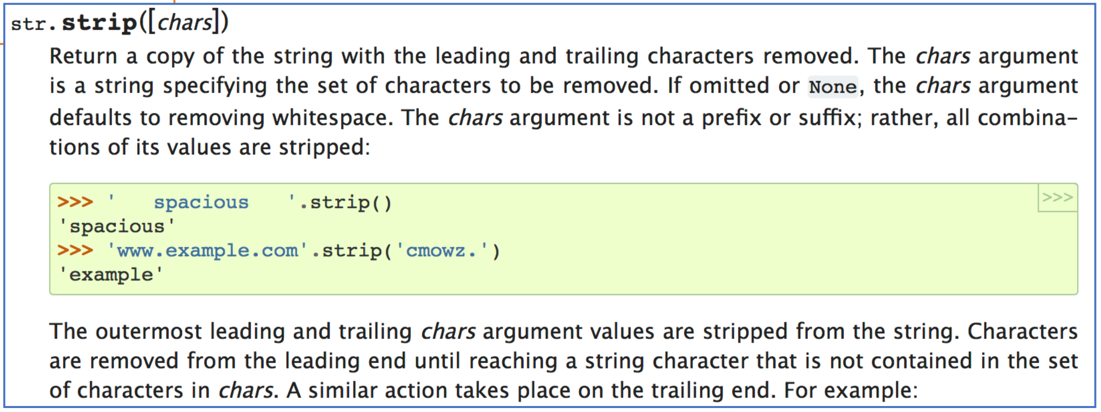
Perl: useful string functions

- 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:



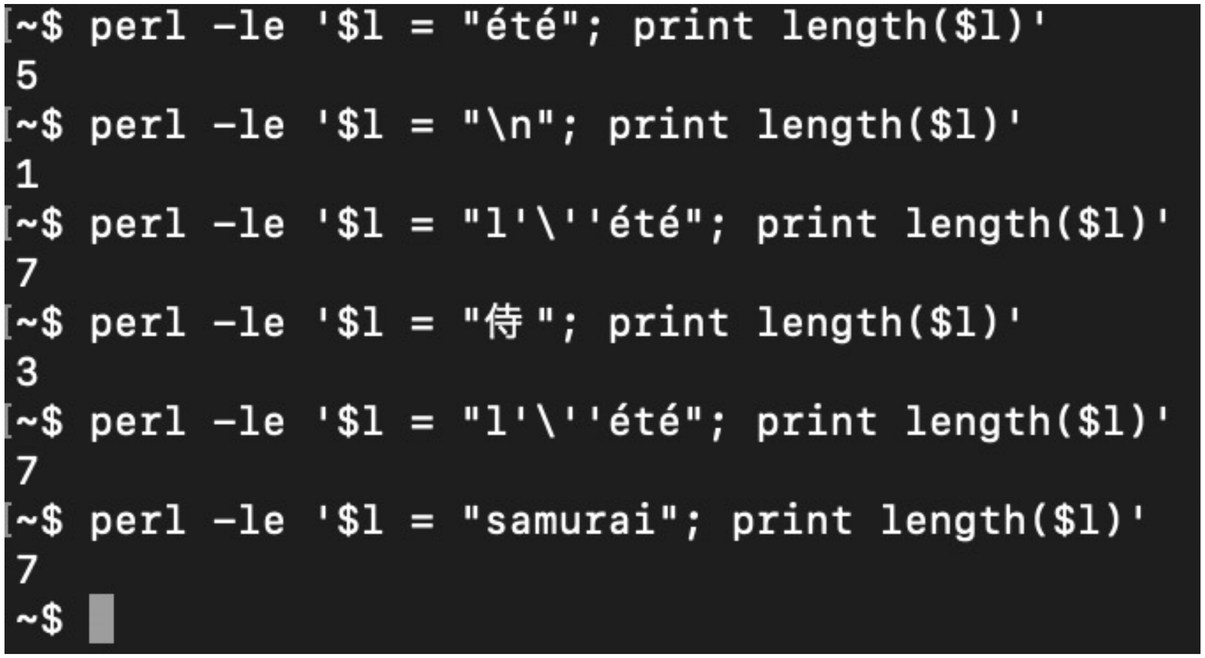
`str.`**`strip`**`([`*`chars`*`])`

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:

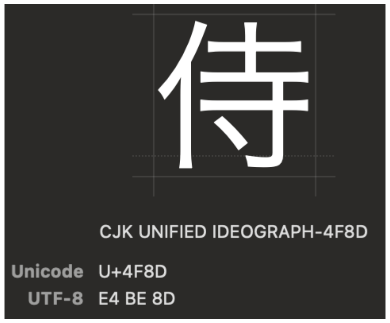
```python
>>> '   spacious   '.strip()
'spacious'
>>> 'www.example.com'.strip('cmowz.')
'example'
```

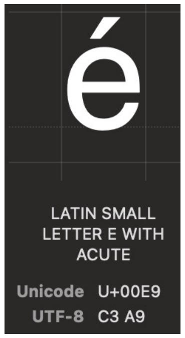
The outermost leading and trailing *chars* argument values are stripped from the string. Characters are removed from the leading end until reaching a string character that is not contained in the set of characters in *chars*. A similar action takes place on the trailing end. For example:

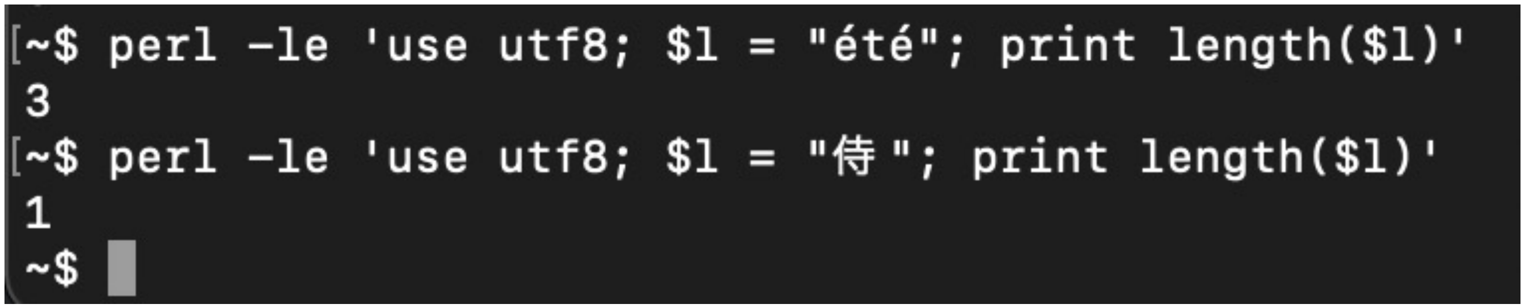
Terminal: newline ("\n") is of length 1 in Perl.



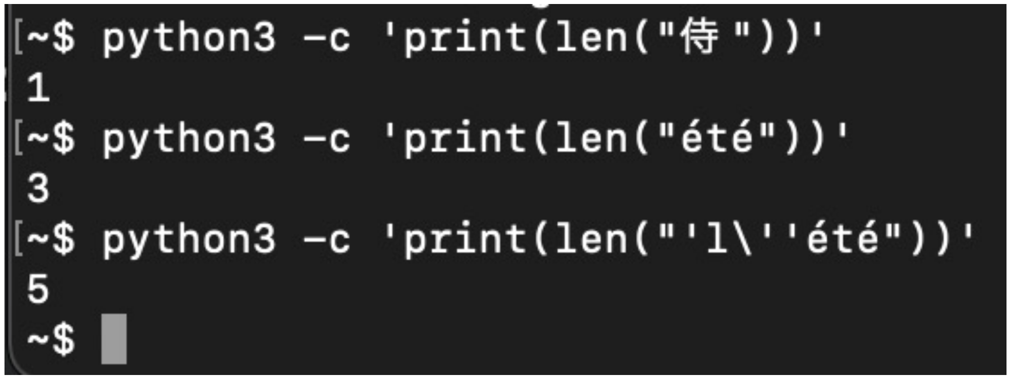
```
[~$ perl -le '$l = "été"; print length($l)'
5
[~$ perl -le '$l = "\n"; print length($l)'
1
[~$ perl -le '$l = "l'\''été"; print length($l)'
7
[~$ perl -le '$l = "侍"; print length($l)'
3
[~$ perl -le '$l = "l'\''été"; print length($l)'
7
[~$ perl -le '$l = "samurai"; print length($l)'
7
~$ 
```





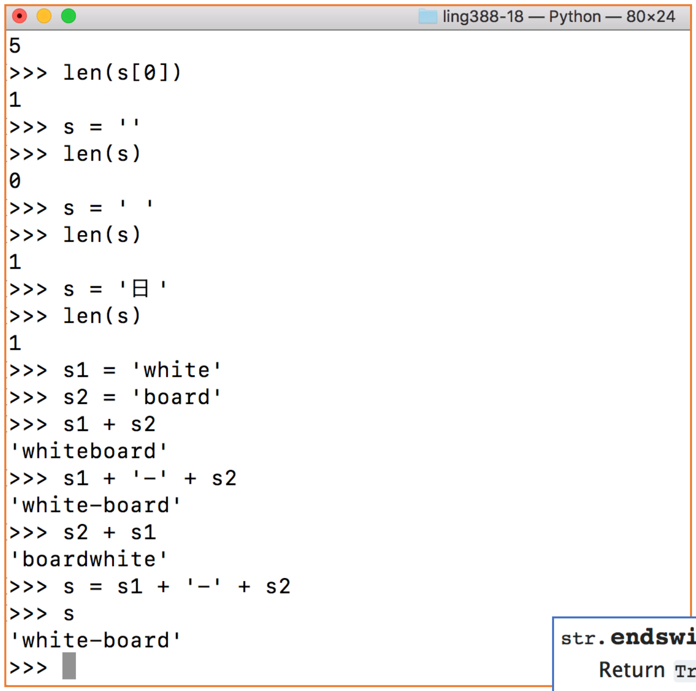


```
[~$ perl -le 'use utf8; $l = "été"; print length($l)'
3
[~$ perl -le 'use utf8; $l = "侍"; print length($l)'
1
~$ 
```



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

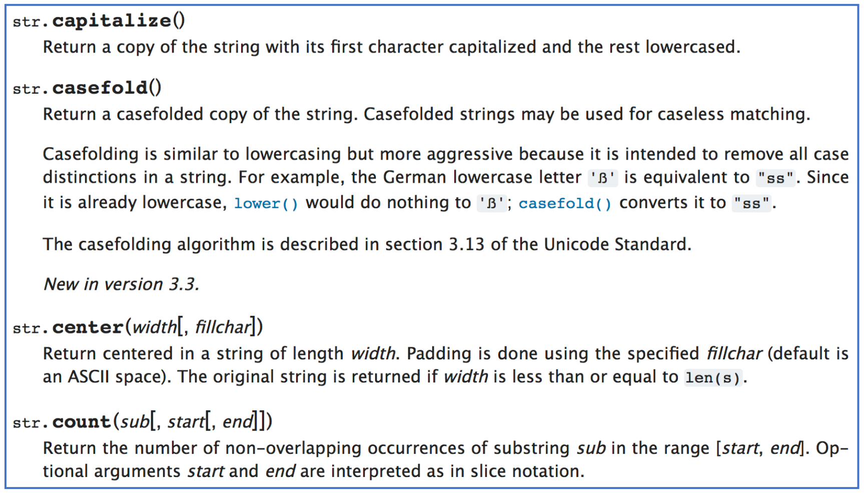
Python: strings



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.

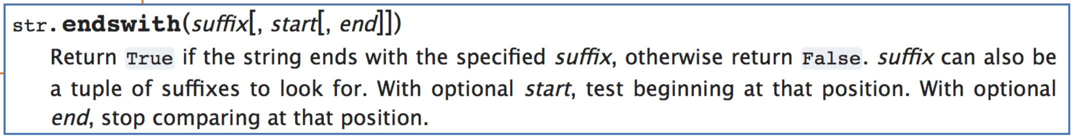
New in version 3.3.

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.