

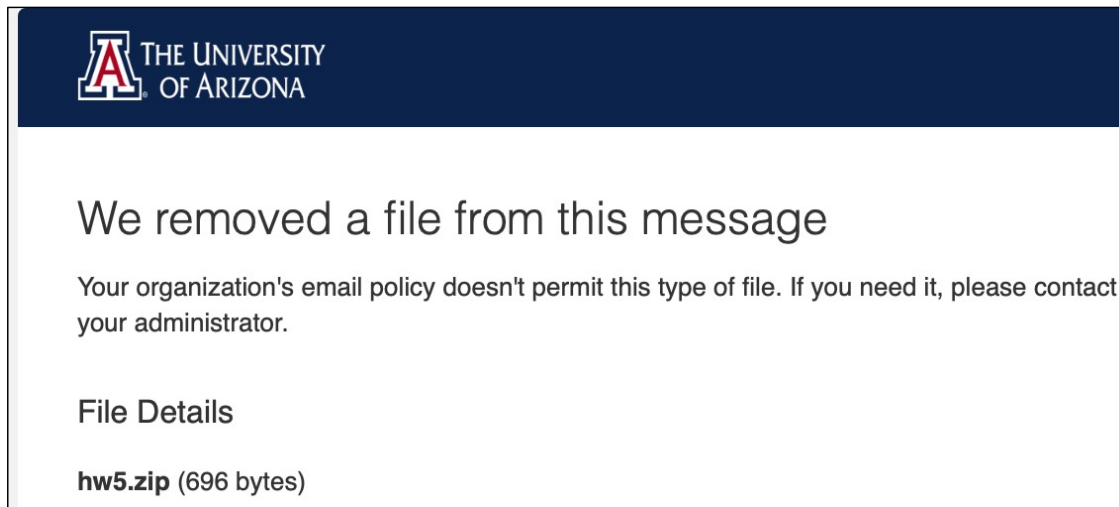
LING/C SC/PSYC 438/538

Lecture 7

Prof. Sandiway Fong

Today's Topics

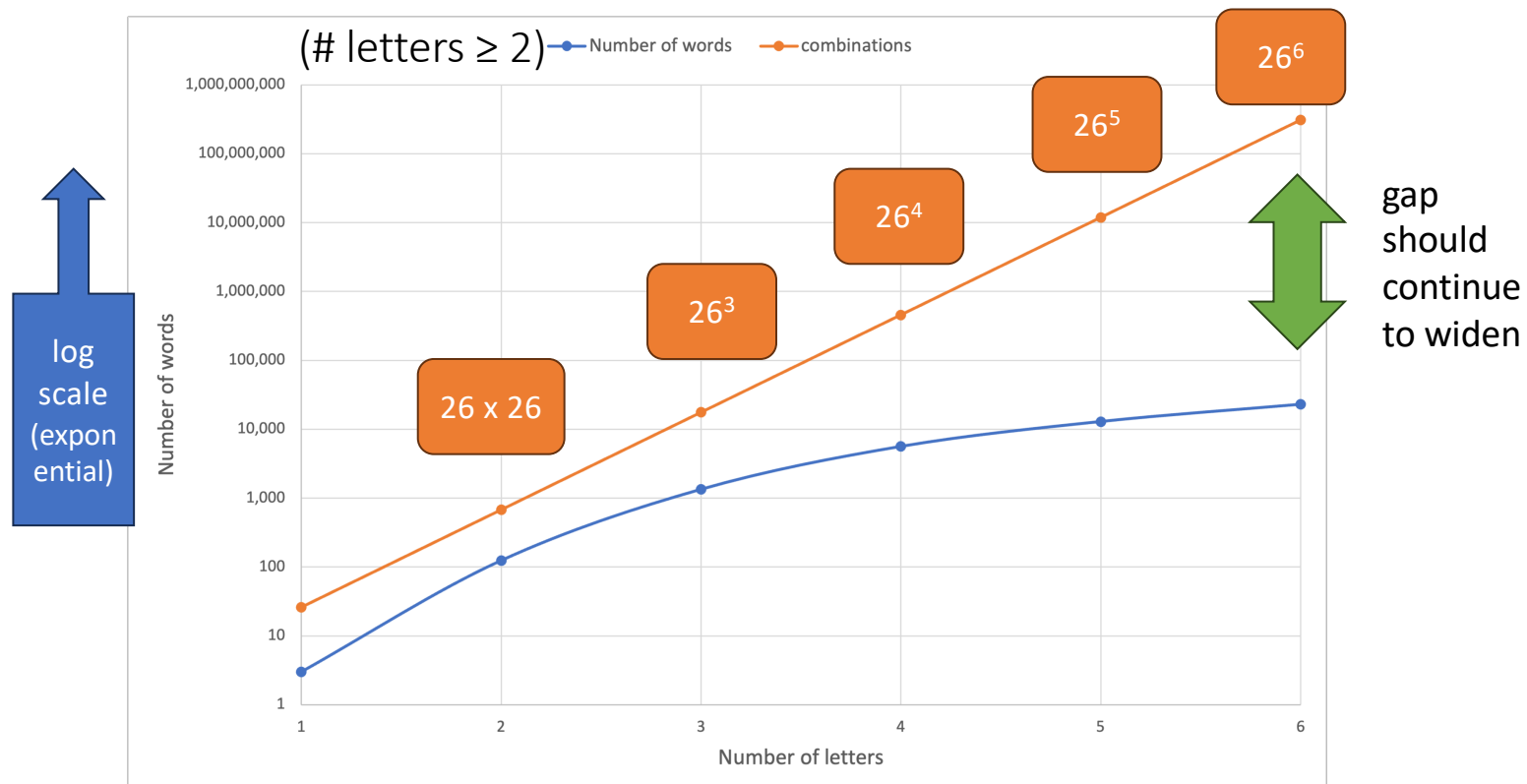
- Homework 5 Review
 - best to send a pdf file (.zip doesn't always work)



Today's Topics

- Worked file I/O example:
 - `falconheavylaunch.txt`
 - I'll be assuming Bash/Zsh (*from macOS or Ubuntu*)
- Sorting

A particular set of Scrabble Words

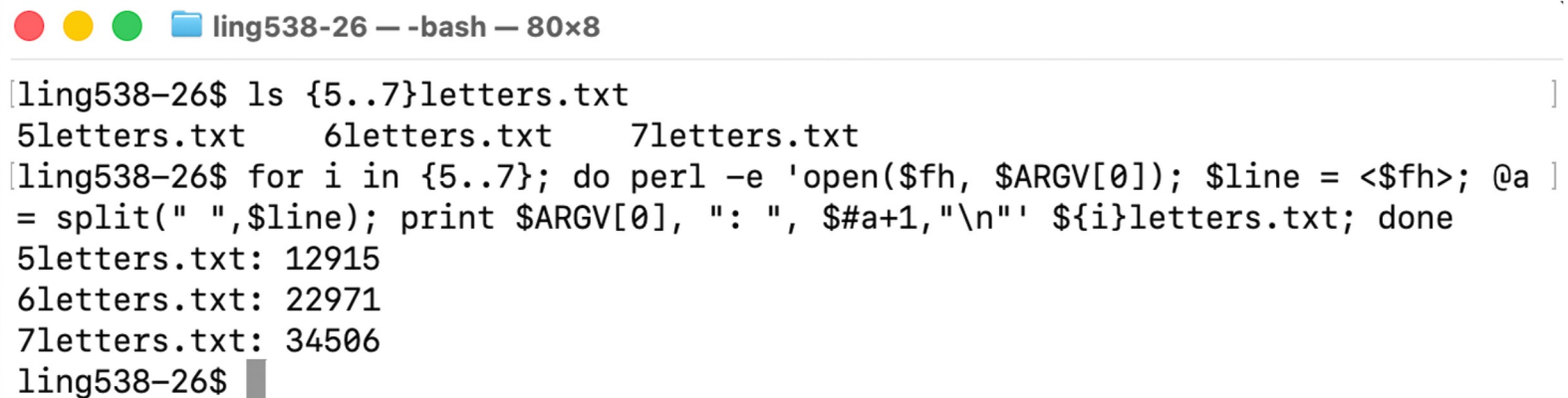


Scrabble Words Statistics



Homework 5 Review

- Question 1: print out the number of words.
- Bash:



```
ling538-26 — -bash — 80x8
[ling538-26$ ls {5..7}letters.txt
5letters.txt    6letters.txt    7letters.txt
[ling538-26$ for i in {5..7}; do perl -e 'open($fh, $ARGV[0]); $line = <$fh>; @a
= split(" ", $line); print $ARGV[0], ": ", $#a+1, "\n" ' ${i}letters.txt; done
5letters.txt: 12915
6letters.txt: 22971
7letters.txt: 34506
ling538-26$
```

Homework 5 Review

- Question 1: print out the number of words.

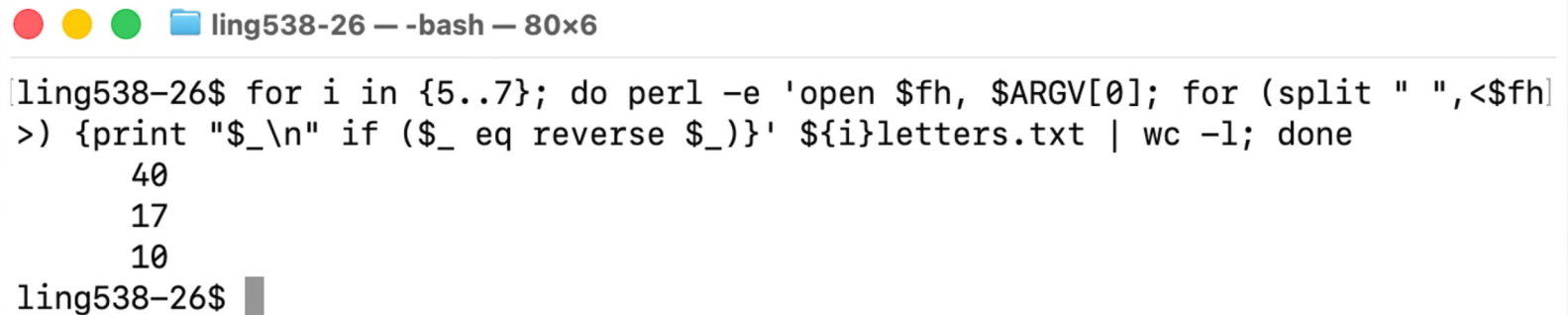
- Example:

```
$ perl -e 'open($fh, $ARGV[0]); ...' 5letters.txt
```

```
$ for i in {5..7}; do echo ${i}letters.txt; done  
5letters.txt  
6letters.txt  
7letters.txt
```

Homework 5 Review

- Question 2: which words spell the same forwards as backwards?



A terminal window titled "ling538-26 — -bash — 80x6" displays a bash script execution. The script iterates over indices 5, 6, and 7, running a perl command to process "letters.txt" and output the word count of palindromes. The results are 40 for index 5, 17 for index 6, and 10 for index 7. The prompt "ling538-26\$" is shown at the bottom.

```
ling538-26$ for i in {5..7}; do perl -e 'open $fh, $ARGV[0]; for (split " ",<$fh  
>) {print "$_\n" if ($_ eq reverse $_)}' ${i}letters.txt | wc -l; done  
40  
17  
10  
ling538-26$
```

Homework 5 Review

Is it true for $n = 8, 9, 10 \dots$?

How long before the streak is broken?

- **Results:**

- ALULA ANANA AYAYA CIVIC DEKED DELED DERED DEWED
KAIK KAYAK LAHAL LAVAL LEMEL LEVEL MADAM MALAM
MINIM QAJAQ RADAR REFER ROTOR SAGAS SAMAS SEDES
SELES SEMES SENES SERES SEXES SHAHS SIMIS SIRIS
SOLOS STATS STETS STOTS SULUS SUSUS TENET TOROT
- ARAARA ATAATA DEGGER DENNED HAJJAH HALLAH MALLAM
MARRAM PULLUP REDDER SELLES SERRES SESSES SUCCUS
TALLAT TERRET TIRBIT
- DEIFIED HADEDAH HALALAH REIFIER REPAPER REVIVER
ROTATOR SEITIES SEMEMES SLAHALS

Perl: file I/O

- What does this code do?

```
perl -e 'open $f,  
"falconheavylaunch.txt"; while (<$f>)  
{print((split " ")[0],"\n"))}'
```

- *try it!*

```
perl -e 'open $f,  
"falconheavylaunch.txt"; while (<$f>)  
{print((split " ")[0],"\n"))}' | wc -l
```

- *reports 49 lines*

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

- *reports 669 words*



Perl: file I/O

And another bit more. Looking ahead:

- %freq (is a hash table, or key/value dict – Python)
 - \$freq{*word*} will hold the number of times *word* occurs
 - keys %freq will be the list of keys, i.e. *words*, recorded
 - values %freq will be the list of values, i.e. *count for each word*, recorded
-
- `perl -e 'open $f, "falconheavylaunch.txt"; while (<$f>) {for (split " ") {$freq{$_}++}};'`

Perl: hash tables

perlintro

Hashes

A hash represents a set of key/value pairs. A `%` sigil indicates a variable is a hash:

```
my %fruit_color = ("apple", "red", "banana", "yellow");
```

You can use whitespace and the `=>` operator to lay them out more nicely:

```
my %fruit_color = (  
    apple => "red",  
    banana => "yellow",  
    lime  => "green",  
);
```

To get at hash elements:

```
$fruit_color{"apple"};      # gives "red"
```

Perl: file I/O

- Unfortunately:
 - `perl -e 'open $f, "falconheavylaunch.txt"; while (<$f) {for (split " ") {$freq{$_}++}}; print %freq'`
- pretty-printing for hash tables is non-existent:

```
print %freq'  
system1low-Earth1use1towards1from1world's1explosion1have1an2"That1slightly1Hubble's1clear1t  
hrough1Earth1performance.1how1satellites2Telescope,1has2planets1over1rocket6most4for5requir  
ed1of19mammoth1company1origami-like1year.1By2fit1essentially1up1unable1radio1is7Musk3low1de  
liver2be5several1hours1across1next2constellation1whimsical1companies1long1Saturn,1Delta1inc  
ident1event.1proposed1cost,1hit1Their1sea.1buses1chances1broadband1your1soundtrack1limited1  
heading1touchdown1"That's1space-suited1Sun1challenges1controlled1with3Bigger,1third3it4inte  
lligence1advance1Mr4"It'll2after3exciting1putting1their1confirmed1all3"I1said2road.1It2Davi  
d1successfully1south1engines,1that's2much1since1stationed1cherry-red1far1his3I've1water1Suc
```

Perl Arrays: sorting

- Example:

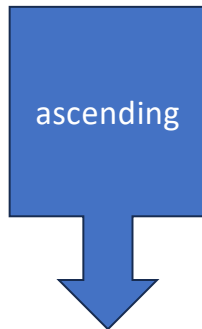
- perl -e '@a=sort @ARGV; print "@a\n"' 3 2 99 7 13 100

- 100 13 2 3 7 99

- perl -e '@a=sort @ARGV; print "@a\n"' mary 77 john vicky Pete

- 77 Pete john mary vicky

- Standard order:

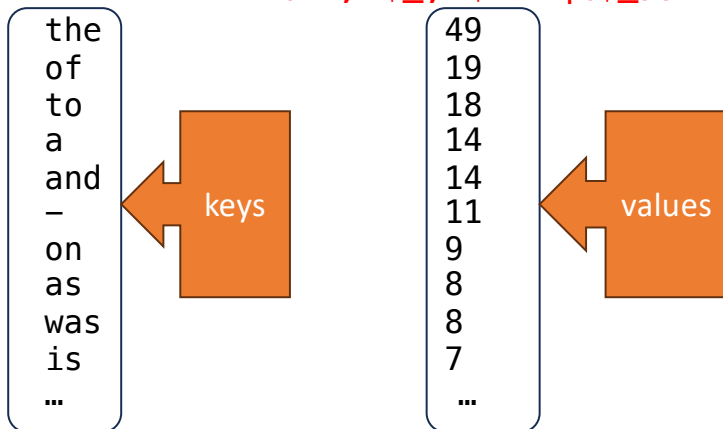


ASCII TABLE			
Decimal	Hex	Char	
0	0	[NULL]	
1	1	[START OF HEADING]	
2	2	[START OF TEXT]	
3	3	[END OF TEXT]	
4	4	[END OF TRANSMISSION]	
5	5	[ENQUIRY]	
6	6	[ACKNOWLEDGE]	
7	7	[BELL]	
8	8	[BACKSPACE]	
9	9	[HORIZONTAL TAB]	
10	A	[LINE FEED]	
11	B	[VERTICAL TAB]	
12	C	[FORM FEED]	
13	D	[CARRIAGE RETURN]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

Perl: file I/O

- Print the word frequency table sorted in descending order:

- `sort {$freq{$b} <=> $freq{$a}} keys %freq`
- `perl -e 'open $f, "falconheavylaunch.txt"; while (<$f) {for (split " ")`
`{ $freq{$_}++}; for (sort {$freq{$b} <=> $freq{$a}} keys %freq) {printf`
`"%-20s %s\n", $_, $freq{$_}}' | head -10`



Perl Arrays: sorting

<https://perldoc.perl.org/functions/sort>

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 them

Perl: file I/O

- Save the sorted output into a file `sorted.txt`
 - `perl -e 'open $f, "falconheavylaunch.txt"; while (<$f) {for (split " ") {$freq{$_}++}}; for (sort {$freq{$b} <=> $freq{$a}} keys %freq) {printf "%-20s %s\n", $_, $freq{$_}}' > sorted.txt`
- Then show it in 3 column format
 - Terminal command `pr -3t sorted.txt`

```
$ pr -3t sorted.txt
the          49 US          2 soundtrack      1
of           19 reporters  2 Ocean.          1
to           18 But        2 tonnes          1
a            14 had         2 but             1
and          14 vehicle    2 much           1
-            11 Musk.       2 Mars'          1
on           9  It         2 without        1
was          8  capable    2 Hubble's       1
as           8  orbit      2 designed       1
is           7  By         2 Kennedy.       1
```

Perl: file I/O

Strictly speaking, we'd like to remove prefix/suffix punctuation,

- i.e. avoid accumulating "words" like:

"It'll	2
--------	---

""

sea.	1
------	---

literally."	1
-------------	---

- We haven't covered regex yet:

- ```
for $w (split " ") {
 $w=~s/^(^\\W+) | (\\W+)$//g;
 $freq{$w}++
}
```

# Formatted printing

```
printf "%-20s %s\n", $_, $freq{$_}
```

Perl's `sprintf` permits the following universally-known conversions:

|                 |                                                                         |
|-----------------|-------------------------------------------------------------------------|
| <code>%%</code> | a percent sign                                                          |
| <code>%c</code> | a character with the given number                                       |
| <code>%s</code> | a string                                                                |
| <code>%d</code> | a signed <code>integer</code> , in decimal                              |
| <code>%u</code> | an unsigned <code>integer</code> , in decimal                           |
| <code>%o</code> | an unsigned <code>integer</code> , in octal                             |
| <code>%x</code> | an unsigned <code>integer</code> , in hexadecimal                       |
| <code>%e</code> | a floating-point number, in scientific notation                         |
| <code>%f</code> | a floating-point number, in fixed decimal notation                      |
| <code>%g</code> | a floating-point number, in <code>%e</code> or <code>%f</code> notation |

## flags

one or more of:

|       |                                                                                                                                     |
|-------|-------------------------------------------------------------------------------------------------------------------------------------|
| space | prefix non-negative number with a space                                                                                             |
| +     | prefix non-negative number with a plus sign                                                                                         |
| -     | left-justify within the field                                                                                                       |
| 0     | use zeros, not spaces, to right-justify                                                                                             |
| #     | ensure the leading "0" for any octal,<br>prefix non-zero hexadecimal with "0x" or "0X",<br>prefix non-zero binary with "0b" or "0B" |

<https://perldoc.perl.org/functions/sprintf>

# Perl: file I/O

- Alternative: use shell command sort (instead of Perl sort):
  - `sort -k 2nr` *sort by key field #2 numerically (n) in reverse order (r)*
  - <https://man7.org/linux/man-pages/man1/sort.1.html>
  - `perl -e 'open $f, "falconheavylaunch.txt"; while (<$f) {for (split " ") {$freq{$_}++}}; for (keys %freq) {print $_," ", $freq{$_}, "\n"}}' | sort -k 2nr | pr -t5`

```
the 49 reporters 2 With 1 ever 1 military. 1
of 19 rocket, 2 about 1 exciting 1 missed 1
to 18 said 2 across 1 experimental mission, 1
a 14 satellites 2 advance 1 explosion 1 moons. 1
and 14 strapped 2 allied 1 far 1 much 1
- 11 that's 2 around 1 fascinating 1 not 1
on 9 told 2 attempt 1 final 1 occasions, 1
as 8 upper-stage 2 available. 1 fire 1 occurring 1
was 8 vehicle 2 back 1 first 1 old 1
is 7 where 2 batches 1 fit 1
Falcon 6 "I 1 becomes 1 five 1 open 1
The 6 "That 1 been 1 flight 1 or 1
rocket 6 "That's 1 believes 1 folded 1 orbit. 1
... origami-like
```

# Python: Files

- Like all other programming languages, uses a file handle, called **file variable**: `open()`
- `infile = open("file.txt","r")`      `outfile = open("results.txt","w")`

```
>>> with open('workfile') as f:
... read_data = f.read()
```

`<filevar>.read()` Returns the entire remaining contents of the file as a single (potentially large, multi-line) string.

`<filevar>.readline()` Returns the next line of the file. That is all text up to *and including* the next newline character.

`<filevar>.readlines()` Returns a list of the remaining lines in the file. Each list item is a single line including the newline character at the end.

```
infile = open(someFile, 'r')
for line in infile.readlines():
 # process the line here
infile.close()
```

same as

```
infile = open(someFile, 'r')
for line in infile:
 # process the line here
infile.close()
```

# Python: Files

- <https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files>

For reading lines from a file, you can loop over the file object. This is memory efficient, fast, and leads to simple code:

```
>>> for line in f:
... print(line, end='')
...
This is the first line of the file.
Second line of the file
```

If you want to read all the lines of a file in a list you can also use `list(f)` or `f.readlines()`.

# Python: Files

```
>>> i = open("falconheavylaunch.txt", 'r')
>>> type(i)
<class '_io.TextIOWrapper'>
>>> text = i.read()
>>> len(text)
3962
>>> text[:100]
"Elon Musk's Falcon Heavy rocket launches successfully\nBy Jonathan Amos BBC Sci
ence Correspondent\n\nUS"
>>> sentences = i.readlines()
>>> len(sentences)
0
>>> i.close()
```

# Python: Files

```
[>>> i = open("falconheavylaunch.txt",'r')
[>>> sentences = i.readlines()
[>>> len(sentences)
49
[>>> sentences[0]
"Elon Musk's Falcon Heavy rocket launches successfully\n"
[>>> sentences[1]
'By Jonathan Amos BBC Science Correspondent\n'
[>>> sentences[2]
'\n'
[>>> sentences[3]
'US entrepreneur Elon Musk has launched his new rocket, the Falcon Heavy, from t
he Kennedy Space Center in Florida.\n'
>>> █
```

# Python: sorting

<https://docs.python.org/3/howto/sorting.html>

**`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](#).