

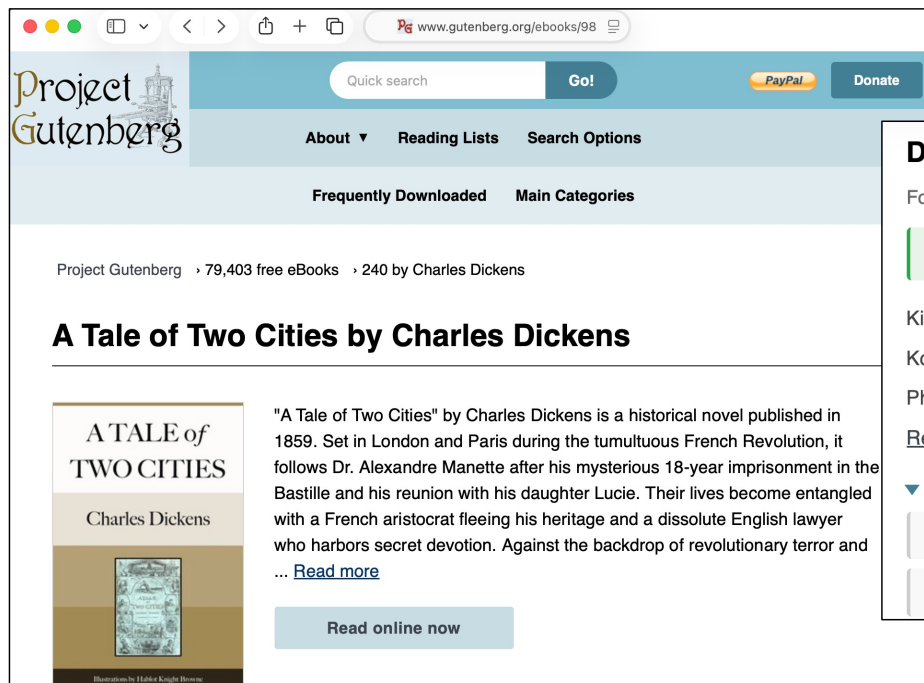
Lecture 8

# *408/508 Computational Techniques for Linguists*

# Today's Topics

1. re-cap + **ChatGPT** on the worked example (2cities.txt)
2. bc (command)
3. a note on positional parameters
- 4. Homework 4**
  - hope everyone is comfortable typing at the Terminal now
  - *a shell script program for you to write*

# Worked Example



The screenshot shows the Project Gutenberg website interface. At the top, there's a navigation bar with the Project Gutenberg logo, a search bar, and links for 'About', 'Reading Lists', and 'Search Options'. Below this, there are links for 'Frequently Downloaded' and 'Main Categories'. The main content area displays the title 'A Tale of Two Cities' by Charles Dickens. To the left is a book cover image. To the right is a description of the novel, followed by a 'Read online now' button.

Project Gutenberg › 79,403 free eBooks › 240 by Charles Dickens

## A Tale of Two Cities by Charles Dickens

"A Tale of Two Cities" by Charles Dickens is a historical novel published in 1859. Set in London and Paris during the tumultuous French Revolution, it follows Dr. Alexandre Manette after his mysterious 18-year imprisonment in the Bastille and his reunion with his daughter Lucie. Their lives become entangled with a French aristocrat fleeing his heritage and a dissolute English lawyer who harbors secret devotion. Against the backdrop of revolutionary terror and ... [Read more](#)

[Read online now](#)

### Download for free

For your e-reader or reading app — Kindle, Kobo, Apple Books, Calibre etc.

**EPUB3** ★ Recommended for most devices

   7.6 MB

Kindle → Use [Send-to-Kindle](#)

Kobo, Nook etc. → [Transfer via USB](#)

Phone, tablet or computer → Open in a reading app

[Reading guide](#)

### ▼ Other formats & older devices

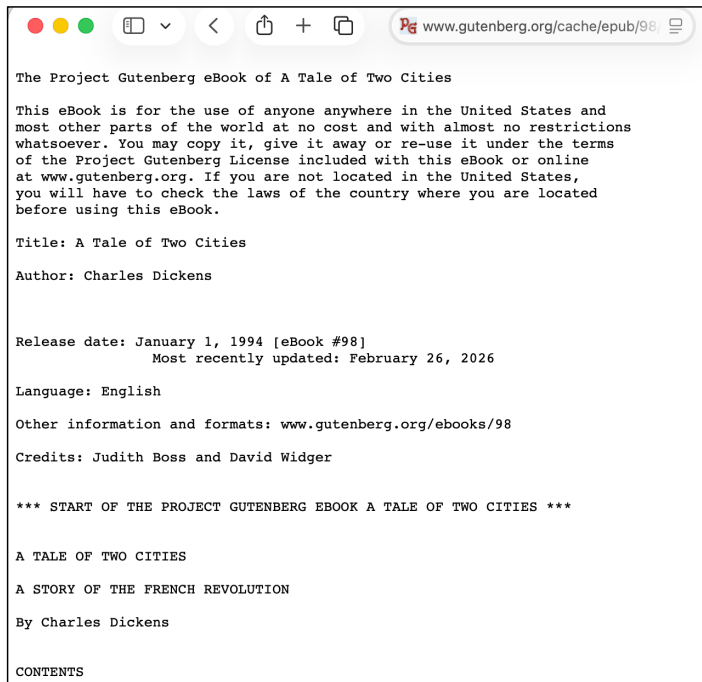
[Plain Text](#)

788 kB

[EPUB \(older e-readers\)](#)

   7.5 MB

# Worked Example



The Project Gutenberg eBook of A Tale of Two Cities

This eBook is for the use of anyone anywhere in the United States and most other parts of the world at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at [www.gutenberg.org](http://www.gutenberg.org). If you are not located in the United States, you will have to check the laws of the country where you are located before using this eBook.

Title: A Tale of Two Cities

Author: Charles Dickens

Release date: January 1, 1994 [eBook #98]  
Most recently updated: February 26, 2026

Language: English

Other information and formats: [www.gutenberg.org/ebooks/98](http://www.gutenberg.org/ebooks/98)

Credits: Judith Boss and David Widger

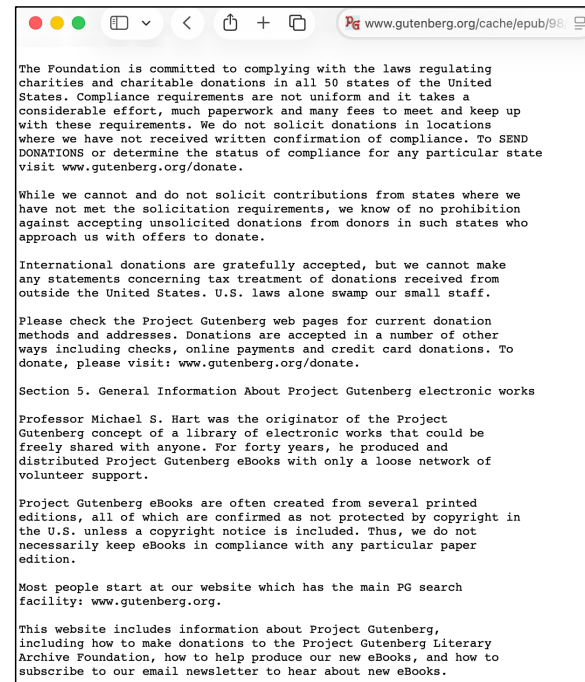
\*\*\* START OF THE PROJECT GUTENBERG EBOOK A TALE OF TWO CITIES \*\*\*

A TALE OF TWO CITIES

A STORY OF THE FRENCH REVOLUTION

By Charles Dickens

CONTENTS



The Foundation is committed to complying with the laws regulating charities and charitable donations in all 50 states of the United States. Compliance requirements are not uniform and it takes a considerable effort, much paperwork and many fees to meet and keep up with these requirements. We do not solicit donations in locations where we have not received written confirmation of compliance. To SEND DONATIONS or determine the status of compliance for any particular state visit [www.gutenberg.org/donate](http://www.gutenberg.org/donate).

While we cannot and do not solicit contributions from states where we have not met the solicitation requirements, we know of no prohibition against accepting unsolicited donations from donors in such states who approach us with offers to donate.

International donations are gratefully accepted, but we cannot make any statements concerning tax treatment of donations received from outside the United States. U.S. laws alone swamp our small staff.

Please check the Project Gutenberg web pages for current donation methods and addresses. Donations are accepted in a number of other ways including checks, online payments and credit card donations. To donate, please visit: [www.gutenberg.org/donate](http://www.gutenberg.org/donate).

Section 5. General Information About Project Gutenberg electronic works

Professor Michael S. Hart was the originator of the Project Gutenberg concept of a library of electronic works that could be freely shared with anyone. For forty years, he produced and distributed Project Gutenberg eBooks with only a loose network of volunteer support.

Project Gutenberg eBooks are often created from several printed editions, all of which are confirmed as not protected by copyright in the U.S. unless a copyright notice is included. Thus, we do not necessarily keep eBooks in compliance with any particular paper edition.

Most people start at our website which has the main PG search facility: [www.gutenberg.org](http://www.gutenberg.org).

This website includes information about Project Gutenberg, including how to make donations to the Project Gutenberg Literary Archive Foundation, how to help produce our new eBooks, and how to subscribe to our email newsletter to hear about new eBooks.

# Worked Example

- Fails!

- `cat 2cities.txt | tr '[:punct:]' ' ' | tr ' '\n' | sort | uniq -c | sort -rn | tail -n +2 | head -n 20 | awk '{print $2, $1}' | termgraph`

- Error:

Traceback (most recent call last):

File "/Library/Frameworks/Python.framework/Versions/3.14/bin/termgraph", line 6, in <module>

sys.exit(main())

~~~~~

File "/Library/Frameworks/Python.framework/Versions/3.14/lib/python3.14/site-packages/termgraph/termgraph.py", line 106, in main

data\_obj = Data.from\_file(args["filename"], args)

File "/Library/Frameworks/Python.framework/Versions/3.14/lib/python3.14/site-packages/termgraph/data.py", line 116, in from\_file

return cls(data, labels, categories)

File "/Library/Frameworks/Python.framework/Versions/3.14/lib/python3.14/site-packages/termgraph/data.py", line 45, in \_\_init\_\_

self.dims = self.\_find\_dims(data, labels)

~~~~~

File "/Library/Frameworks/Python.framework/Versions/3.14/lib/python3.14/site-packages/termgraph/data.py", line 128, in \_find\_dims

raise Exception(

    f"The inner dimensions of the data are different\nThe dimensions of {data[i - 1]} is different than the dimensions of {data[i]}"

)

Exception: The inner dimensions of the data are different

The dimensions of [] is different than the dimensions of [6769.0]

# Worked Example

- Works!

- `cat 2cities.txt | tr '[:punct:]' ' ' | tr ' ' '\n' | sort |  
 uniq -c | sort -rn | tail -n +3 | head -n 20 | awk '{print $2,  
 $1}' | termgraph`

26159

8077

6769 the

4469 and

3748 of

3293 to

2601 a

2340 in

1904 I

1770 his

# Worked Example

- ChatGPT:

I see the problem. Your pipeline is producing the **right kind of two-column data**, but the version of `termgraph` you have installed is parsing it incorrectly.

`termgraph` expects each input row to have a **label followed by one or more numeric values**. Its documentation explicitly supports space-separated input and stdin.

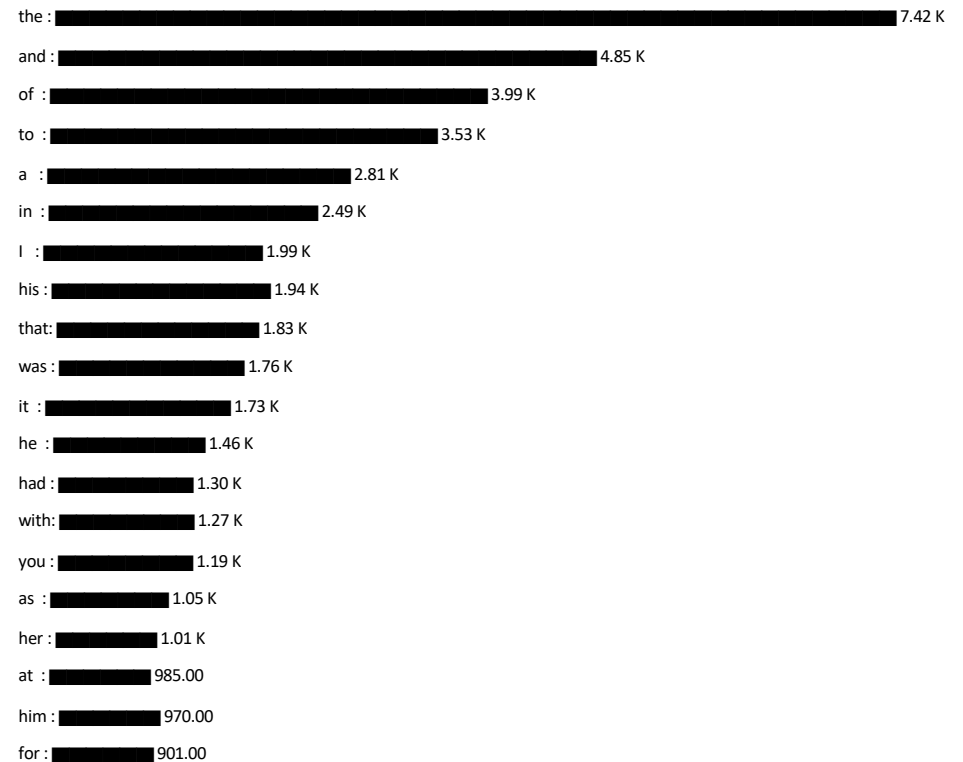
[🔗 Free offer](#)

## Even better: eliminate blank lines explicitly

I suspect `2cities.txt` contains blank lines, and your `tr ' ' '\n'` doesn't convert **tabs or other whitespace**. Try:

# Worked Example

```
tr '[:punct:]' ' ' < 2cities.txt |  
> tr -s '[:space:]' '\n' |  
> sort |  
> uniq -c |  
> sort -rn |  
> head -n 20 |  
> awk 'NF == 2 {print $2, $1}' |  
> termgraph
```



That handles **spaces, tabs, and newlines** and collapses repeated whitespace.

# Shell Arithmetic: use command **bc** instead

- Bash shell arithmetic, e.g. `((z= x+y))`, is [integer](#) only.
- What if you needed [floating point](#) numbers?

```
bc(1) bc(1)
NAME
  bc - An arbitrary precision calculator language
SYNTAX
  bc [ -hlwsqv ] [long-options] [ file ... ]
VERSION
  This man page documents GNU bc version 1.06.
DESCRIPTION
  bc is a language that supports arbitrary precision numbers with inter-
  active execution of statements. There are some similarities in the
  syntax to the C programming language. A standard math library is
  available by command line option. If requested, the math library is
```

man bc command brings up this page

- bc runs interactively

**Note:**

- `bc -l` loads the [math library](#) first

# command **bc**

- Examples:

- we know  $\tan(\pi/4) = 1$ , so  $(arctan) \tan^{-1}(1) = \pi/4$  ( $\pi/4$  in radians =  $45^\circ$ )
- function `a(radians)` computes `arctan` when `bc -l` is used
- Control-D (EOF) to exit `bc`

```
Machine$ bc -l
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
a(1)
.78539816339744830961
a(1)*4
3.14159265358979323844
^DMachine$
```

- can use `echo` with an expression and then pipe (`|`) it into `bc`

```
(base) ~$ echo "4*5" | bc
20
```

# command **bc**

- Bash shell trick:
  - `$(command)` is a modern synonym for ``command`` (``` = *backtick* character, not single quote `'`) which stands for **command substitution**;
  - it means run *command* and put its output here (*see next slide*).
- From man bc:
  - *the following [Terminal command] will assign the value of "pi" to the shell variable **pi**.*

```
[(base) ~$ pi=$(echo "scale=10; 4*a(1)" | bc -l)
[(base) ~$ echo $pi
3.1415926532
```

# command **bc**

[https://www.gnu.org/software/bash/manual/html\\_node/Command-Substitution.html#Command-Substitution](https://www.gnu.org/software/bash/manual/html_node/Command-Substitution.html#Command-Substitution)

## 3.5.4 Command Substitution

Command substitution allows the output of a command to replace the command itself. enclosed as follows:

`$(command)` `pi=$(echo "scale=10; 4*a(1)" | bc -l)`

or

``command`` `pi=`echo "scale=10; 4*a(1)" | bc -l``

```
(base) ~$ pi=`echo "scale=10; 4*a(1)" | bc -l`  
(base) ~$ echo $pi  
3.1415926532
```

# command **bc**

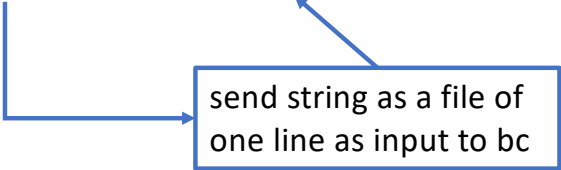
- [https://www.gnu.org/software/bash/manual/html\\_node/Bash-Builtins.html#index-echo](https://www.gnu.org/software/bash/manual/html_node/Bash-Builtins.html#index-echo)

**echo**

```
echo [-neE] [arg ...]
```

Output the *args*, separated by spaces, terminated with a newline.

```
pi=$(echo "scale=10; 4*a(1)" | bc -l)
```



send string as a file of  
one line as input to bc

## command **bc**

```
[Machine$ pi=$(echo "scale=10; 4*a(1)" | bc -l)
[Machine$ echo $pi
3.1415926532
```

- `pi` is a Bash shell variable here

```
[Machine$ echo "scale=10; 4*a(1)" | bc -l > pi.txt
[Machine$ more pi.txt
3.1415926532
pi.txt (END)
```

with re-direction (>) into a file `pi.txt`, spacebar to get out of `more/less`

# command **bc**

- scale variable in bc:

There are four special variables, **scale**, **ibase**, **obase**, and **last**. **scale** defines how some operations use digits after the decimal point. The default value of **scale** is 0. **ibase** and **obase** define the conversion base for input and output numbers. The default for both input and output is base 10. **last** (an extension) is a variable that has the value of the last printed number. These will be discussed in further detail where

# command **bc**

- `scale = 1000`

```
[^DMachine$ bc -l
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
[scale = 100
[a(1)*4
3.141592653589793238462643383279502884197169399375105820974944592307\
8164062862089986280348253421170676
[scale = 1000
[a(1)*4
3.141592653589793238462643383279502884197169399375105820974944592307\
81640628620899862803482534211706798214808651328230664709384460955058\
22317253594081284811174502841027019385211055596446229489549303819644\
28810975665933446128475648233786783165271201909145648566923460348610\
45432664821339360726024914127372458700660631558817488152092096282925\
40917153643678925903600113305305488204665213841469519415116094330572\
70365759591953092186117381932611793105118548074462379962749567351885\
75272489122793818301194912983367336244065664308602139494639522473719\
07021798609437027705392171762931767523846748184676694051320005681271\
45263560827785771342757789609173637178721468440901224953430146549585\
37105079227968925892354201995611212902196086403441815981362977477130\
99605187072113499999983729780499510597317328160963185950244594553469\
08302642522308253344685035261931188171010003137838752886587533208381\
42061717766914730359825349042875546873115956286388235378759375195778\
18577805321712268066130019278766111959092164201988
```

# Math constant $e$

```
scale=50
e(1)
2.71828182845904523536028747135266249775724709369995
scale=51
e(1)
2.718281828459045235360287471352662497757247093699959
scale=52
e(1)
2.7182818284590452353602874713526624977572470936999595
□
```

## 100 Decimal Digits

<https://www.mathsisfun.com/numbers/e-eulers-number.html>

Here is  $e$  to 100 decimal digits:

**2.71828182845904523536028747135266249775724709369995957**  
**49669676277240766303535475945713821785251664274...**

# command **bc**

```
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
obase=2
7
111
254
11111110
obase=16
255
FF
15
F
13
D
█
```

Output base (obase) variable:

- obase=2 (*binary*)
- obase=16 (*hexadecimal*)  
0..9,A..F
- **Recall:** control-D to exit

# Positional Parameters

- Inside a Bash shell script, these \$ variables have values:
  - \$1: first parameter
  - \$2: 2<sup>nd</sup> parameter and so on...
  - \$#: # of parameters

- Program on webpage test.sh:

```
#!/bin/bash
echo "Number of parameters: $#"
```

```
if [ $# -eq 1 ]; then
    echo "1st parameter: $1"
```

```
fi
```

- Output:
  - bash test.sh  
Number of parameters: 0
  - bash test.sh 45  
Number of parameters: 1  
1st parameter: 45
  - bash test.sh 45 56  
Number of parameters: 2



```
or do chmod u+x test.sh
Run using: ./test.sh
```

# If-test

Program on webpage test2.sh

```
1#!/bin/bash
2echo "Number of parameters: $#"/>
3if [ $# -eq 1 ]; then
4    echo "1st parameter: $1"/>
5fi
```

```
$ bash test2.sh 1 2
Number of parameters: 2
test2.sh: line 3: [2: command not found
$
```

- Spaces around [ and ] are important!

# If-test

- Note the spaces between the [ ... ] !

| Primary     | Meaning                                                           |
|-------------|-------------------------------------------------------------------|
| [ -a FILE ] | True if FILE exists.                                              |
| [ -b FILE ] | True if FILE exists and is a block-special file.                  |
| [ -c FILE ] | True if FILE exists and is a character-special file.              |
| [ -d FILE ] | True if FILE exists and is a directory.                           |
| [ -e FILE ] | True if FILE exists.                                              |
| [ -f FILE ] | True if FILE exists and is a regular file.                        |
| [ -g FILE ] | True if FILE exists and its SGID bit is set.                      |
| [ -h FILE ] | True if FILE exists and is a symbolic link.                       |
| [ -k FILE ] | True if FILE exists and its sticky bit is set.                    |
| [ -p FILE ] | True if FILE exists and is a named pipe (FIFO).                   |
| [ -r FILE ] | True if FILE exists and is readable.                              |
| [ -s FILE ] | True if FILE exists and has a size greater than zero.             |
| [ -t FD ]   | True if file descriptor FD is open and refers to a terminal.      |
| [ -u FILE ] | True if FILE exists and its SUID (set user ID) bit is set.        |
| [ -w FILE ] | True if FILE exists and is writable.                              |
| [ -x FILE ] | True if FILE exists and is executable.                            |
| [ -O FILE ] | True if FILE exists and is owned by the effective user ID.        |
| [ -G FILE ] | True if FILE exists and is owned by the effective group ID.       |
| [ -L FILE ] | True if FILE exists and is a symbolic link.                       |
| [ -N FILE ] | True if FILE exists and has been modified since it was last read. |

|                             |                                                                                                                                                           |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| [ FILE1 -nt FILE2 ]         | True if FILE1 has been changed more recently than FILE2, or if FILE1 exists and FILE2 does not.                                                           |
| [ FILE1 -ot FILE2 ]         | True if FILE1 is older than FILE2, or if FILE2 exists and FILE1 does not.                                                                                 |
| [ FILE1 -ef FILE2 ]         | True if FILE1 and FILE2 refer to the same device and inode numbers.                                                                                       |
| [ -o OPTIONNAME ]           | True if shell option "OPTIONNAME" is enabled.                                                                                                             |
| [ -z STRING ]               | True if the length of "STRING" is zero.                                                                                                                   |
| [ -n STRING ] or [ STRING ] | True if the length of "STRING" is non-zero.                                                                                                               |
| [ STRING1 == STRING2 ]      | True if the strings are equal. "=" may be used instead of "==" for strict POSIX compliance.                                                               |
| [ STRING1 != STRING2 ]      | True if the strings are not equal.                                                                                                                        |
| [ STRING1 < STRING2 ]       | True if "STRING1" sorts before "STRING2" lexicographically in the current locale.                                                                         |
| [ STRING1 > STRING2 ]       | True if "STRING1" sorts after "STRING2" lexicographically in the current locale.                                                                          |
| [ ARG1 OP ARG2 ]            | "OP" is one of -eq, -ne, -lt, -le, -gt or -ge. These arithmetic binary operators return true if the expressions are true. "ARG1" and "ARG2" are integers. |

# If-test

- <https://www.gnu.org/software/bash/manual/bash.html>
- if `[[ condition ]]` (*newer test: older test [...] also supported*)

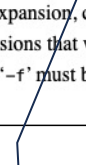
`[[...]]`

`[[ expression ]]`

Return a status of 0 or 1 depending on the evaluation of the conditional expression *expression*. Expressions are composed of the primaries described below in [Bash Conditional Expressions](#). The words between the `[[` and `]]` do not undergo word splitting and filename expansion. The shell performs tilde expansion, parameter and variable expansion, arithmetic expansion, command substitution, process substitution, and quote removal on those words (the expansions that would occur if the words were enclosed in double quotes). Conditional operators such as `'-f'` must be unquoted to be recognized as primaries.

see previous slide, plus pattern matching, e.g. `regex =~`  
`[[ $# -eq 1 && $1 -lt 10 ]]` vs. `[ $# -eq 1 ] && [ $1 -lt 10 ]`

```
1#!/bin/bash
2echo "Number of parameters: $#"
```



```
3if [[ $# -eq 1 && $1 -lt 10 ]]; then
4    echo "1st parameter: $1"
5fi
```

```
$ bash test3.sh 1
Number of parameters: 1
1st parameter: 1
$ bash test3.sh 10
Number of parameters: 1
$
```

# Homework 4

- Let's write a simple shell-script BMI calculator
  - solicit input from the terminal (using read) or from the command line (\$1 \$2)

| Measurement Units                            | Formula and Calculation                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Kilograms and meters (or centimeters)</b> | <p>Formula: <math>\text{weight (kg)} / [\text{height (m)}]^2</math></p> <p>With the metric system, the formula for BMI is weight in kilograms divided by height in meters squared. Since height is commonly measured in centimeters, divide height in centimeters by 100 to obtain height in meters.</p> <p>Example: Weight = 68 kg, Height = 165 cm (1.65 m)<br/>Calculation: <math>68 \div (1.65)^2 = 24.98</math></p> |
| <b>Pounds and inches</b>                     | <p>Formula: <math>\text{weight (lb)} / [\text{height (in)}]^2 \times 703</math></p> <p>Calculate BMI by dividing weight in pounds (lbs) by height in inches (in) squared and multiplying by a conversion factor of 703.</p> <p>Example: Weight = 150 lbs, Height = 5'5" (65")<br/>Calculation: <math>[150 \div (65)^2] \times 703 = 24.96</math></p>                                                                     |



try the metric one first

# Homework 4

- You can use if-test, bc (*from this lecture*), and shell scripting to build your program
- Submit your shell script and screenshots of your runs
- To get you started, let's play on the command line first:
  - *your instructor weights 75kg and is 1.72 meters tall*
  - `((bmi = 75 / (1.72 * 1.72)))`
  - `echo $bmi`
  - *won't work: why?*
  - *syntax error: invalid arithmetic operator (error token is ".72 \* 1.72")*

## Homework 4

- One approach is to scale height in cm instead of meters to make numbers be whole ones (100cm = 1m):
  - `((bmi = 75 / (172 * 172)))`
  - `echo $bmi`
  - *how to scale it properly if we use cm instead of m?*
  - *100 cm in a meter, multiply by what?*
  - *e.g. `((bmi = 75 * 100 * 100 / (172 * 172)))`*

## Homework 4

- Instead of scaling to integer, we could pipe floating point numbers to bc directly:

```
[(base) ling508-22$ echo "72 / (1.72 * 1.72)" | bc  
24
```

- or use variables:

```
[(base) ling508-22$ weight=72  
[(base) ling508-22$ echo $weight  
72  
[(base) ling508-22$ height=1.72  
[(base) ling508-22$ echo $height  
1.72  
[(base) ling508-22$ echo "$weight / ($height * $height)" | bc  
24  
(base) ling508-22$
```

## Homework 4

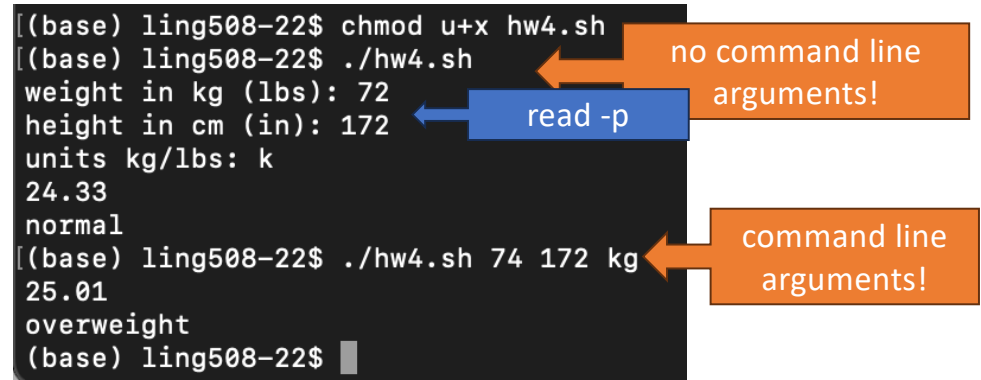
- After you figure out how to do on the command line, put it in a shell script, and try to add the following three embellishments:
  1. accept either command line arguments or read from the terminal if they're missing
  2. recall `read -p "Enter : " variablename`
    - `if [ $# -ne N ]; then`
    - *N* = number of command line arguments.

# Homework 4

2. print the weight status message according to the following table:
  3. modify the calculator to accept input in both metric and traditional units
- make sure you supply examples of your program working!

| BMI            | Weight Status |
|----------------|---------------|
| Below 18.5     | Underweight   |
| 18.5 – 24.9    | Normal        |
| 25.0 – 29.9    | Overweight    |
| 30.0 and Above | Obese         |

```
[(base) ling508-22$ chmod u+x hw4.sh
[(base) ling508-22$ ./hw4.sh
weight in kg (lbs): 72
height in cm (in): 172
units kg/lbs: k
24.33
normal
[(base) ling508-22$ ./hw4.sh 74 172 kg
25.01
overweight
(base) ling508-22$
```



# Homework 4

- Instructions:

- email sandiway@arizona.edu
- submit everything in one PDF file!
- subject of email: 408/508 Homework 4 *your name*
- cite any discussion or source
- due date: next Sunday by midnight