Control	
CTRL-c	Cancel running command
CTRL-d	Destroy running command
CTRL-r	Back-search previous commands (CTRL-r again to cycle backwards through matches)
CTRL-u	Clear current line
CTRL-l	Clear current window
CTRL-a	Start of current line
CTRL-e	End of current line
META-f	Move forward a word
META-b	Move backwards a word
META-.	Last element of previous command
META-?	Show possible completions
META-h	View man page of current command *ZSH only*
META-q	Clear current line, but return it after running next command *ZSH only*

Processes	
CMD &	Send cmd to background
jobs	Show all running processes
fg [%INT]	Bring last used [number] process to the foreground

Networking	
127.0.0.1	localhost
ssh [-p PORT] [-i KEYFILE] USER@SERVER	ssh into server
nmap -p RANGE HOST	Scan ports. -p- to scan all ports.
nc HOST PORT	Connect to host on port
nc -l -p PORT	Listen for connections on port
openssl s_client -connect HOST:PORT [-ign-eof]	Connect via ssl [non-interactively]
curl [-s] [-o FILE] [-w FORMAT] “HOST” OPTIONS	Transfer a url [without showing progress] [outputting to file] [writing formatted output to stdout]
scp [-i KEYFILE] FILE USER@HOST:PATH	Securely copy file to path on host

Basic utility	
man CMD	View manual for cmd
whoami	Print current user
pwd	Print working directory
cd [PATH]	Change directory to home [directory specified]
ls [-hal]	List contents of current directory [inc. hidden, in human-readable list format]
file FILE	Describe file contents
which NAME	Show alias or location
mkdir [-p PATH]	Makes directory [including any intermediates required]
mktemp [-d]	Make a temporary file [directory] and print name
cp FILE1 FILE2	Make a copy of file1 called file2
mv FILE1 FILE2	Move (rename) file1 to file2
yes	Print “yes” forever

Standard files	
0	stdin
1	stdout
2	stderr

Dev files	
/dev/null	Ignores all input (sink)
/dev/random	Stream of random bytes (can block)
/dev/urandom	Stream of random bytes (non-blocking)

Redirects	
CMD < FILE	Send file to stdin of cmd
CMD << HERE-DOC	Send here-doc to stdin of cmd
CMD <<< HERE-STRING	Send here-string to stdin of cmd
CMD > FILE	Send stdout of cmd to file
CMD &> FILE	Send both stdout and stderr of cmd to file
CMD >> FILE	Append stdout of cmd to file
CMD1 | CMD2	Pipe stdout of cmd1 to stdin of cmd2

grep	
grep REGEX [FILE]	Search for pattern [in file] and print matching lines
-f FILE	Get patterns from file
-i	Case-insensitve
-v	Invert match
-c	Return count of matches
-o	Print only the matched parts of a matching line

jq	
jq FLAGS “COMMANDS”	Work with JSON
setpath(KEY:VALUE)	Add value at key
FIELD=VALUE	Update value of field
keys[_unsorted]	Get keys [unsorted]
map()	Get only values
@csv	Convert to csv (must be values-only)
CMD1 | CMD2	Chain commands
-M	Monochrome output
-c	Concise output (one line per object)
-r	Raw output

Other text	
cat FILE	Print contents of file
less FILE	View contents of file
sort FILE	Sort lines
uniq [-c]	Filter unique lines [count how many of each] (must be sorted)
sed -e ’s/PATTERN/REPLACEMENT/g’	Replace all occurrences of pattern with replacement
ent -c FILE	Print the number of occurrences of each byte in, and entropy of file *NON-STANDARD*

Numbers	
base64 [-D] FILE	Encode to [decode from] base64
xxd [-r] FILE	Create [reverse] hexdump
xxd -b FILE	Create binary dump

Scripts	
#!/bin/bash	Shebang. Ensures script runs in bash. Should be first line.
while BOOLEAN; do … done	While loop
if BOOLEAN; then … elif BOOLEAN; then … else … fi	If statement
[. . .]	Evaluates statement as a boolean expression. Alias for test
{A..Z}	Expands to run command with each character/number in range. A and Z can be anything
exit INT	End the script. 0 for success
$0	The script
$#	The number of arguments
$?	Exit code of previous command
${VAR}	Parameter substitution. Same as $VAR, but unambiguous. Can take a variety of special characters inside the brackets to manipulate or expand variables
${#VAR}	Length of VAR (example of parameter substitution)
set -e	Exit immediately is a command returns a non-zero exit code
set -x [+x]	Print commands and their output [stop printing]
shellcheck [-x] SCRIPT	Check script for syntax errors etc [following external sources] *NON-STANDARD*
trap CMD SIGNAL	Run cmd when signal is received
command -v CMD > /dev/null	Check if cmd exists *take care to check that running command w/o arguments won’t have an effect