SHELL SCRIPT

GENERAL

SHORTCUTS:

1. Ctrl + L – Clears the screen (But everything is stored above, if scrolled up)
2. Ctrl + R – Back Search for commands
3. Ctrl + D – Exits the current shell
4. Ctrl + Z – Suspends the current process
5. Ctrl + C – Kills the current process
6. Command + K – Clears the screen and deletes all previous workings
7. Command + D – Opens 2 parallel windows
8. History – Displays the history of commands used
    a. history – Displays the history of last few commands
    b. history 1 – Displays the history of all the commands (from the beginning)
9. clear – Clears the screen

SPECIAL COMMANDS:

1. !! – Entire last command, including arguments is executed
2. !<Command> – Runs the previous command
3. $? – Return code of the previous command (0 or 1)
4. $0 – Name of the script
5. $$ – Process Identification Number (PID) of the current script
6. $ – Prints the value stored in that variable
7. | – Combines/Joins 2 commands
8. / – Next directory
9. .. – Previous directory
10. . – Current directory
11. - – Home directory; Flag/ Option
    (A single dash is interpreted as a flag or an option by the system)
    (Here 'semicolon' is used to show that the above command has two uses)
12. -- – Not a Flag/ Option
    (Two separate dashes, with a space on either side, is interpreted as an argument/logic, and not as a flag or an option by the system)
13. ~ – Home directory
14. * – All files in the current directory
15. & – Runs a command in the background

BASICS:

1. date – Prints the date and time

2. cal – Displays the calendar
    a. cal -j – Displays Julian Calendar
    b. cal -y– Displays current year's Calendar
    c. cal -y <Year> – Displays the whole year's calendar (of the year mentioned)
    d. cal -3 – Displays the previous, present, and next months in the calendar
    e. cal -N – Displays the calendar vertically
    f. cal -d <year>-<month> – Displays the month and year mentioned
    g. cal -H <year>-<month>-<date> – Displays the month and year mentioned along with the date highlighted
3. watch -n <seconds> – Changes the time after 'n' seconds
4. say "<Text>" – Calls out the text
5. echo – Prints out the arguments
6. printf – Prints out the arguments
7. print – Prints out the arguments
8. curl http://wttr.in – Displays the weather of the next three days
9. $PATH – Prints all the paths present in the machine, separated by colons
10. which – Prints out the path of a function
11. pwd – Prints the Present working directory
12. cd/chdir – Changes the directory as instructed
13. pushd – Changes the directory and stores it in a stack
14. popd – Goes back to the previous directory and removes the current one from stack
15. whoami – Prints the username (Short for "Who am I")
16. chmod <Permission> <Filename> – Changes the permission of files
17. sudo – Helps to access a file as the root user
18. chown <Owner Name/Number> <File/Directory Name> – Changes the owner of the file
19. chgrp <Group Name/Number> <File/Directory Name> – Changes the group of the file
20. column <Filename> – Prints each line in a separate column
21. vi/vim – Opens the editor of a file
22. open – Opens a file
23. hostname – Prints the name of the host (of the computer being used)
24. less/more – Allows the user to view the contents in 2 or more files simultaneously
25. wc – Counts words, lines, characters, and bytes (-w, -l, -m and -c respectively)
26. alias – Stores command/word/sentence in a variable

a. alias <Variable name>="echo <Word/Sentence to be stored>" – Stores the word/sentence in the variable assigned and prints it when then the variable is called
b. alias <Variable name>="<Command to be executed>" – Stores the command in the variable assigned

(Typing alias **<variable name (where the command is stored)>** gives the command stored under that variable name)

27. unalias – Removes the command stored in alias
28. lsof – Lists open files
29. ps – Lists all the processes running
30. kill -<Signal name> <Process ID (PID)> – Performs the task as indicated by the signal name

    (**kill -l** displays all the signal names)

    Keywords for <Signal name>:
    SIGKILL (9) – Kills the process
    SIGSTOP (17) – Pauses the process
    SIGCONT (19) – Continues the process
31. NOHUP <Process Name> & – Runs a process in the background
32. ping <Process Name> – Continuously pings a process to get a reply

    (Here, the process name can be google.com)
33. PS1="<Symbol>" – Starts all the succeeding lines with the symbol
34. systemsetup -remotelogin – Shows whether the remote login option is on/off hello

    (By pressing **sudo systemsetup -remotelogin on/off**, the option can be changed)
35. bg – Resumes the suspended jobs (that are running in the background)
36. du <filename/directory name> – Displays the disk space of an individual file or all the files in a directory

    ('du -h' displays the disk space in human readable format)
37. wait – Waits until the current process is completed

    (This strategy will fail if a new shell window is opened)
38. caffeinate – Prevents the Computer from going into sleep mode
39. shutdown -h now – Shuts down the Computer
40. shutdown -r now – Restarts the Computer
41. /var/log/system.log – Shows general messages and information regarding the system

    (The logs will be visible if the above function is typed after 'cat')
42. logger <Text> – Provides an easy way to add messages to '/var/log/system.log'

43. log show [--last <seconds>s] – Displays all the processes running in the computer [in the last 's' seconds]
44. gdb – Displays minor system functions and workings **
45. dtrace – Displays system calls and all the processes running in the system
    Syntax:
    - ❖ sudo dtrace -l – Lists all the system calls and processes

    (Other important commands can be viewed by typing 'man dtrace' or 'sudo dtrace')
46. printf <Text> shasum – Encrypts the text using letters and numbers
    (The syntax can also be written as: shasum **<Text>**)
47. writegood – Checks and corrects the grammatical errors/observations in a sentence ##
48. cron – Performs a task repetitively (after a certain interval of time)
49. systemctl – Displays all the processes and informations of a system
    ('systemctl' stands for 'System Control')
50. ifconfig – Displays the network interface details of a machine along with ip address
51. hyperfine – Compares 2 command execution parameters
52. top – Displays sorted information about processes
    (htop – Newer version of 'top')
53. hexdump – Displays a file in the specified format, i.e., hexadecimal, octal, decimal, etc.
54. <any language name/program name> --version – Prints the version of a program/programming language
    (Instead of '--version', '–V' can also be used)
55. <any language name/program name> --verbose – Prints extra information about a program/ programming language
    (Instead of '--verbose, '--V' or '--v' can also be used)
56. rsync – Syncs local and remote files
57. --quiet – Quietly runs commands or installs packages, without displaying any of the processes apart from the errors
    (The above command can also be executed using '-q'. In either case, only the errors occurring in the installation process are displayed)
    ('silent' is an alternative for 'quiet')

LOGICAL OPERATORS:

1. || – OR operator (If the first command is true, returns/prints that else, returns or prints the second command)
2. && – AND operator (If both the commands are true only then returns/prints them)

ASSIGNMENT OPERATORS:

1. <Variable 1>+=<Variable 1> – <Variable 1> = <Variable 1> + <Variable 2>
2. <Variable 1>-=<Variable 1> – <Variable 1> = <Variable 1> - <Variable 2>
3. <Variable 1>*=<Variable 1> – <Variable 1> = <Variable 1> * <Variable 2>
4. <Variable 1>/=<Variable 1> – <Variable 1> = <Variable 1> / <Variable 2>
5. <Variable 1>^=<Variable 1> – <Variable 1> = <Variable 1> ^ <Variable 2>
6. <Variable 1>%=<Variable 1> – <Variable 1> = <Variable 1> % <Variable 2>

HELP:

1. man <Function Name> – Helps to find all available information about the function
2. $help <Function Name> – Lists only the synopsis of the function
3. <Function Name> --help – Helps to find a few available information about the function

FILES

LISTING FILES:

1. ls – Lists all files and directories under a directory
2. ls -l – Lists all the files/directories along with extra information
3. ls -l | tail -<Line Number> – Lists the last n lines only
4. ls -l | head -n<Line Number> – Lists the top n lines only
5. ls -S – Sorts files according to their size
6. ls -la – Lists hidden files
7. ls -la ~ – Lists all the hidden files along with those hidden in the home directory
8. ls -lt – Sorts files according to recency
9. ls -G – Colours certain files
10. ls -h – Lists the file sizes in human readable format (e.g., 13B, 51M, etc.)
11. ls *.<Extension> – Lists all the files having that extension
12. ls -R – Lists all the files under all the directories (in that directory) recursively
13. ls <Filename>? – Lists the filenames having only 1 number after its name

    Example:

    ls project?

    Output: project1 and project2, but not project11 or project12, etc.
14. exa – Lists all files and directories under a directory, in coloured format ('exa' is similar to 'ls')

CREATING FILES:

1. touch <Filename> – Creates a blank file
2. echo <Text to be written in the File> > <File Name> – Creates a file containing the text written before the filename
3. echo <Text to be written in the File> | tee <Filename> – Prints as well as creates a file with the content written

REMOVING FILES:

1. rm – Removes a file
2. rm -i – Requests for confirmation before removing a file in a directory

MOVING FILES/RENAMING FILES:

1. mv <Filename> <New Filename> – Renames/moves a file

COPYING FILES:

1. cp <Filename of the File to be Copied From> <Filename of the File to be Copied To> – Copies a file
2. cat <Filename of the File to be Copied From> > <Filename of the File to be Copied To> – Copies a file

COMPRESSING FILES:

1. gzip <Filename> – Compresses a file, without keeping a copy of the original file
2. gzip -k <Filename> – Compresses the copy of a file and keeps the original file intact
3. gzip -r <Folder Name/Directory> – Compresses all the files in a folder/directory (individually), without keeping a copy of the original file
4. gzip -d <Compressed Filename> – Decompresses a file
5. tar – Compresses & decompress a file

EXECUTING FILES:

1. cat ./<Filename> – Executes the file
2. cat *.<Extension> – Prints all the contents in all the files having the similar extension
3. echo *.<Extension> – Prints all the files having that extension

DIFFERNECE BETWEEN 2 FILES:

1. diff <Filename 1> <Filename 2> – Prints the difference between the contents of both the files

Keywords:

c – Change
a – Add
d – Delete
< – Refers to File 1 (filename 1)
> – Refers to file 2 (filename 2)

2. diff <(ls <Folder Name 1>) <(ls <Folder Name 2>) – Prints the names of the files (as well as its number from the top if saved according to alphabetical order) that are saved under different names in either of the folders
3. diff3 <Filename 1> <Filename 2> <Filename 3> – Prints the difference between the contents of 3 files
4. cmp <filename 1> <filename 2> – Compares both the files
   (**echo $?** Prints the output value, i.e., 0 for similar/identical files and 1 for non-similar/non-identical files)

NOTE: The keywords mentioned in point 1 are applicable in all the 'diff' functions.

DOTFILES

ZSH:

1. ~/.zshrc – Any command stored in this editor becomes permanent for zsh
   (' ~ ' means home directory)

BASH:

1. vi ~/.bashrc – Any command stored in this editor becomes permanent for bash

NOTE: **vi ~/.<filename>** gives access to any file in the home directory

TEXTS

APPENDING/CONCATING TEXTS (LINES/ELEMENTS):

1. cat <Filename of the File 1> >> <Filename of the File 2> – Appends the contents in both the files and prints them in file 2
2. <Statement 1> ; <Statement 2> – Both the statements are appended
3. paste number [optional] <Filename 1> <Filename 2> – Concatenates and prints the texts in both the files separated by space (along with serial numbers displayed at the extreme left side).
4. paste -d "<One or More Delimiters>" number [optional] <Filename 1> <Filename 2> – Concatenates and prints the texts in both the files separated by the delimiter (along with serial numbers displayed at the extreme left side).

5. paste -s Number [optional] <Filename 1> <Filename 2> – Concatenates the texts in both the files and prints them in separate columns (along with serial numbers displayed at the top).

REMOVING TEXT (LINES/WORDS):

1. uniq <Filename> – Removes duplicate lines in a line, only if they are adjacent to each other.

REPLACING TEXTS (LETTERS/WORDS):

1. sed 's/<Letter/Word to be Replaced>/<New Letter/Word>/' <Filename> – Replaces only the first word (to be replaced) in a line with the new letter/word
2. sed 's/<Letter/Word to be Replaced>/<New Word>/<Nth Letter/Word>' <Filename> – Replaces only the nth word (to be replaced) in a line with the new letter/word
3. sed 's/<Letter/Word to be Replaced>/<New Letter/Word>/g' <Filename> – Replaces all the words (to be replaced) in a line with the new letter/word
4. sed 's/<Letter/Word to be Replaced>/<New Word>/<Nth Letter/Word>' <Filename> – Replaces from the nth word (to be replaced) in a line with the new letter/word
5. sed '<Nth Line> s/<Letter/Word to be Replaced>/<New Letter/Word>/' <Filename> – Replaces all the words (to be replaced) in the nth line with the new letter/word
6. sed 's/[<Letters/Texts to be Replaced>]/<New Letter/Text>/' <Filename> – Replaces only the first letter/text (to be replaced) with the new letter/text.
7. sed 's/[<Letters/Text to be Replaced>]/<New Letter/Text>/g' <Filename> – Replaces all the letters/texts (to be replaced) with the new letter/text.
8. sed 's/(<Text to be Replaced>)*/<New text>/' <Filename> – Replaces only the first text (to be replaced) with the new letter/text.
   (Here, the text written inside the parenthesis is considered as a single text)
9. sed 's/(<Text to be Replaced>)*/<New Text>/g' <Filename> – Replaces all the texts (to be replaced) with the new text.
   (Here, the text written inside the parenthesis is considered as a single text)
10. sed 's/(<Text 1 to be Replaced>|<Text 2 to be Replaced>)*//g' <Filename> – Replaces all the texts (matching with text 1 and text 2 respectively) with nothing (or simply removes them).

(Here, the text written inside the parenthesis is considered as a single text and the pipe operator(|) concats the action of both the functions and searches for both the texts to be replaced)

NOTE: sed function works with double quotes (“ ”) too.

SORTING TEXTS (LINES/WORDS):

1. sort <Filename> – Sorts the texts inside the file according to their ASCII values
2. sort -n <Filename> – Numeric Sort (Only sorts numbers, not letters)
3. sort -k<Starting Column Number>,<Ending Column Number [optional]> – Sorts the columns mentioned, from the starting column to the ending column (first the ‘Starting’ column is sorted, followed by the ‘Ending’ column)
4. sort -r – Sorts in reverse order

SPLITTING TEXTS (LINES/WORDS) IN FILES:

1. split -l <Line Number to be Splitted From> <Filename> <New Filename> – Splits the file into 2 separate files (from the line mentioned)

SEARCING FOR WORDS:

1. <Document Name/History> | grep <Word to be Searched For> – Searches for the word in the document mentioned

PRINTING TEXTS:

1. cat <Filename> – Prints the contents of a file
2. cat ./<Filename> – Prints the contents of a file
3. tail -r <filename> – Prints the contents of a file in reverse format
   (‘tail -r’ is equivalent to ‘tac’, which is the reverse of ‘cat’)
4. source <Filename> – Execute a file
5. awk ‘{print}’ <Filename> – Prints all the text in a file
6. awk ‘/<Word to be Printed>/ {print}’ <Filename> – Prints all the lines containing the word
7. awk ‘{print $<Any Column Number> $<Any Column Cumber>}’ <Filename> – Prints only the columns mentioned
8. awk ‘$<Any Column Number> == <Letter/Number> && $<Any Column Number> ~ <Regular Expression> {print $0}’ <Filename> – Prints the lines in the column (mentioned) containing the letter/number and checks if that line matches with the regular expression given.
   ($0 – Prints the first text)

(Here, the 2 expressions are separated by AND (&&) operator. These 2 statements can be written separately too)

9. echo -e "\e[<Number 1>;<Any Number 2>m<Text 1>\e[<Number 3>m"<Text 2> – Prints coloured text
   (Changing 'Number 1' changes the colour of the text)
   (Changing 'Number 2' changes the pattern of the text, i.e., Underlined, Bold, Italics, Highlight, Blink, Vanish etc.)
   (Changing 'Number 3' changes the colour of Text 2, present at the end of the sentence)
   Number 1:
   30 – Black
   31 – Red
   32 – Green
   33 – Yellow
   34 – Purple
   35 – Pink
   36 – Blue
   31 – Grey
   (The above numbers indicate the basic colours. Other coloured patterns can be brought about by changing 'Number 1' accordingly)
   Number 2/Number 3:
   0 – Normal Black Colour
   1 – Bold
   2 – Fade
   3 – Italics
   4 – Underlined
   5 – Blinking
   6 – Normal Indicated Colour
   7 – Highlighted
   8 – Disappearing
   9 – Normal Indicated Colour
   (These are the only numbers valid for 'Number 2' and 'Number 3')
10. column – Prints each line in a separate column
11. column -t – Aligns all the white spaces in separate columns

TEXT EDITOR

EDITTING FILES (vi EDITOR):

NORMAL MODE:

1. r – Replaces a character

2. I – Inserts character/words/sentences
3. a – Append
4. o – Inserts a new line after a line (in which the cursor currently is) and changes the mode to insert mode
5. O – Inserts a new line before a line (in which the cursor currently is) and changes the mode to insert mode
6. ~ – Flips the cases of all the letters/ words/ sentences selected.
7. . – Repeats the last command
8. dw – Deletes all the letters in a word, including space, till before the next word
   (Here 'd' stands for delete)
9. de – Deletes all the letters in a word, excluding space, till before the next word
   (Here 'd' stands for delete)
10. dd – Deletes a line
    (Here 'd' stands for delete)
11. ce/ cw – Deletes all the letters in a word, excluding space, till before the next word and changes the mode to insert mode (to change characters)
    (Here 'c' stands for change)
12. cc – Deletes a line and changes the mode to insert mode
13. v – Visual Mode
14. <shift> + v – Visual Line
15. <ctrl> + v – Visual Block
16. / <Word/Character/Sentence> – Searches for the word/ sentence/ character
17. u – Undo
18. Ctrl + r – Redo
19. Ctrl + o – Moves the cursor to previous position (word/program)
    (The above command undoes the action of the cursor)
20. Ctrl + i – Moves the cursor to next position (word/program)
    (The above command redoes the action of the cursor)
21. yy – Copies a line
22. yw — Copies a word
23. p – Pastes a word/ line after a certain letter/ line (on which the cursor is)
24. : – Command Line
25. x – Deletes a character
26. :help :w – Helps to find all possible functions for 'w' command
27. :help w – Helps to find all the functions for 'w' key
28. :sp – Opens 2 parallel windows
29. w – Shifts the cursor to the first letter of the next word

30. b – Shifts the cursor to the first letter of the previous word/ current word (if the cursor is pointed to the last letter of a word)
31. e – Shifts the cursor to the last letter of the next word/ current word (if the cursor is pointed to the first letter of a word)
32. <Count n> w/ b/ e – Shifts the cursor n number of times to the specified word/ letter as mentioned by the user.
33. :q – Quits the current window/ last opened window
34. :w – Saves the current window/ last opened window
35. :wq – Saves and quits the current window/ last opened window
36. :qa – Quits all the windows
37. hjkl – Arrow keys to move the cursor left, down, up, and right
38. <Count n> hjkl – Arrow keys to move the cursor left, down, up, and right n number of times
39. 0 – Moves to the first character in a line
40. $ – Moves to the last character in a line
41. ^ – Moves to the first non-empty character in a line
42. Ctrl + U – Moves up in a buffer
43. Ctrl + D – Moves down in a buffer
44. G – Moves down the entire buffer
45. gg – Moves up the entire buffer
46. M – Moves the cursor to the middle of the screen
47. L – Moves the cursor to the lowest line on the screen
48. H – Moves the cursor to the highest line on the screen
49. f<Any Character> – Jumps to the first next character (as given by the user)
50. F<Any Character> – Jumps to the first previous character (as given by the user)
51. t<Any Character> – Jumps to the letter just before the first next character (as given by the user)
52. T<Any Character> – Jumps to the letter just after the first previous character (as given by the user)

NOTE 1: Usually a key pressed twice applies for a line.

NOTE 2: **<count n> <Any function>** – The function will be performed n number of times.

VISUAL MODE:

NOTE: Visual Mode uses all the normal mode commands for selecting texts. After an action is performed in Visual mode, the mode automatically changes back to Normal Mode.

INSERT MODE:

1. <esc> – Exit any mode
2. echo `<Function>` – Prints the output of the function
3. echo $<FUNCTION> – Prints the output of the function

MODIFIERS:

1. I – Inside
    a. ci (– Deletes all the text inside the parenthesis and changes the mode to insert mode
    b. di (– Deletes all the text inside the parenthesis
2. a – Around
    a. ca (– Deletes all the text inside the parenthesis, including the parenthesis itself and changes the mode to insert mode
    b. da (– Deletes all the text inside the parenthesis, including the parenthesis itself.

NOTE: Modifiers 'a' and 'i' can be combined with any command to perform a certain task.

FILES AND TEXTS

SEARCING FOR/FROM FILES:

1. grep <Word/Sentence to be Searched For> <Filename> – Searches for the word/sentence in a file
2. grep -R <Word/Sentence to be Searched For> . – Recursively searches for the word/sentence in the current file

REGULAR EXPRESSION (Used with grep):

1. . – Any one character (special)
2. * – Matches any number of previous (including 0) (special)
3. + – Matches any number of previous (excluding 0) (not special)
4. $ – End of the line (special)
5. ^ – Beginning of the line (special)
6. \S – Any non-whitespace character (special)
7. \s – Any whitespace character (special)
8. ? – Optional (special)
9. \d – Any digit
10. \D – Any non-digit character
11. \w – White space characters
12. \W – Non-white space characters
13. [0-9] – Any number
14. [a-z] – Any lowercase letter
15. [A-Z] – Any uppercase letter

16. [A-Za-z] – Any letter
17. [abc] – Only a, b, and c
18. [^abc] – Neither a, nor b, nor c
19. (…) – Capture Group (Treats the whole thing inside the bracket as a single text)
21. \ – Escape something (This character makes the action of special characters normal and turns normal characters special)

3. find . -name <Filename> -type <Filetype> – Searches for a file in the current directory
4. find . -path <Path name> -type <Filetype> – Searches for a file in the current directory

   Keywords for <Filetype>:

   f – file

   d – directory
5. find . -name "*.<Extension>" – Searches for files with a certain extension in the current directory and the sub-directories
6. find . -mtime -<Hours> – Prints the names of the files opened in 'h' hours

VALUES/VARIABLES

STORING VALUES:

1. <Variable Name>=<Value/Variable> – Stores the value in the variable assigned

PRINTING VALUES/VARIABLES:

1. Echo "<Variable name>" – Prints the value stored in the variable
2. Echo '<Variable name>' – Prints the variable name
3. Echo <Variable name> – Prints the value stored in the variable
4. $<Variable name> – Prints the contents stored in the variable

DIRECTORIES

MAKING DIRECTORIES:

1. mkdir – Makes a directory

REMOVING DIRECTORIES:

1. rmdir – Removes a blank directory
2. rm -r – Removes a directory along with all the file and directories under it

SEARCING FOR DIRECTORIES:

1. find . -name <Filename> -type <Filetype> – Searched for a directory in the current directory

   Keyword for <Filetype>:
   f – file
   d – directory

FILES AND DIRECTORIES

LINK TO FILES/DIRECTORIES:

1. ln -s <File/Directory name> <New File/Directory Name> – Stores the link of a file in the new file/folder

MIXED

TREATING OUTPUT AS INPUT:

1. <Expression> | xargs <Function to be Performed> – Takes the output of the expression and treats it as the input of the function to be performed

INPUT/OUTPUT

INPUT:

1. read <Variable Name> – Takes a word from the user and stores it

(There can be any number of variables and inputs)

OUTPUT:

1. $<Variable Name> – Prints the value of the variable

GIT

1. git help <Git Function Name> – Prints the activities of the git functions
2. git auth login – Authenticates and logs in to GitHub
   (On typing **git login**, all the options under **git login**, i.e., **alias**, **auth**, **browse**, **repo**, **ssh-key**, **gpg-key**, **config**, etc., will be displayed.)
3. git init – Initialises the git repository in the home folder
4. git status – Displays git history
   (New files and directories/Recently added files and directories/the files and directories not added to git status (using git add) will be displayed as 'untracked files')

5. git add <Filename/Directory Name> – Keeps a track of the files/directories in git status
6. git add -p <Filename> – Interactive method of adding files to git commit

   Keywords for interaction:

   s – Splits into smaller changes with one change showing at a time

   y – Yes

   n – No
7. git add :/ – Adds all the files from top to bottom in the git repository
8. git rm <Filename/Directory Name> – Removes the files/directories from git status
9. git commit – Opens the text editor and allows the user to make changes (like add/delete a git command)

   (This command creates a new snapshot in the git repository with the changes included)
10. git commit -m <Commit Message> – Commits the files added (by directly taking in a message) without opening the text editor
11. git commit --amend – Helps to alter the commit message
12. git stash – Stores the recent changes without committing then, and allows the user to work on a different commit simultaneously
13. git stash pop – Allows the user to edit the previous commit (that has not been added or committed)

    ('git stash pop' basically undo's 'git stash')
14. git log – Shows the history of git commit in a flattened way
15. git log --all --graph --decorate – Shows the history of git commit as a graph
16. git log --all –graph --decorate --oneline – Shows the history of git commit in a single line
17. git show – Displays the details and changes of the latest commit
18. git show <hash code> – Displays the details and changes of that commit
19. git show <Hash Code> – Displays the contents of a file
20. git cat-file -p <Hash Code> – Prints all hash codes of the sub-files or sub-folders. Moreover, if the hash code of any of the sub-directories or sub-files are searched for, then the hash codes of the following folders/files are printed (if any), else, the contents present in the respective file is printed
21. git diff <Filename> – Shows the changes recently made to a file with respect to the 'HEAD' file

    (Can also write 'git diff HEAD **<Filename>**')

    ('--a.text' represents the file commited in git '--b.text' file represents the changed file)

('-**<Starting line number in the original file>,<Number of lines originally present>**' indicates the number of lines that were originally present in the commited file and '+**<Starting line number in the edited/changed file>,<Number of lines finally present>**' indicates the number of lines that are present in the edited file (which has not been commited yet)

22. git diff <Hash Code of a File> <Filename> – Shows the changes made to a file since the code number (with respect to the 'HEAD' file)
23. git diff <Hash Code of File 1> <Hash Code of File 2/Branch Name/HEAD> <Filename> – Compares the changes in a file from code 1 to code 2
24. git diff --cached – Displays the changes that are staged for git commit
25. git blame <Filename> – Examines the contents of a file line by line and checks when each line was last modified and who the author of the modifications was
26. git branch – Lists all the branches present in the local repository
27. git branch <Name of Branch> – Creates a new branch (with the name designated) which points to the same commit
28. git branch -d <Branch name> – Deletes a branch name
29. git branch -f <Main/Master Branch/Branch Name> [HEAD~]<Number/Hash Code> – Shifts the main/master/any other branch to the 'nth' preceding branch (if a number is mentioned) or preceding/succeeding branch (if hash code is mentioned)
30. git merge <Branch Name 1/Hash Code 1> <Branch Name 2/Hash Code 2> – Merges the 2 branches into one
31. git merge --abort – Discards all the changes done in git merge
32. git merge --continue – Opens the vi editor for committing the changes in git merge (after adding the changed/edited files)
33. git mergetool – Merges 2 files using a tool (like opendiff, vimdiff, etc.)
34. git config --global merge.tool <Tool Name> – Merges 2 files using the merge tool mentioned
    (The instructions are stored in the **.config** file in the home directory)
    (The tool names are mentioned in the instruction that appears on typing 'mergetool'. For example: **opendiff**, **vimdiff**, etc.)
35. git switch <Hash Code/Branch Name> – Switches the 'HEAD' to the branch mentioned
36. git checkout <Filename> – Discards the changes that haven't been committed
37. git checkout <Hash Code of a File/Branch Name> – Shifts the 'HEAD' (file/text currently being viewed) to the code/branch mentioned, but 'master/main' still points to the recently created/modified file
38. git checkout HEAD^ – Points the 'HEAD' to the previous branch/code

39. git checkout HEAD~<Number/Hash Code> – Shifts the head to the 'nth' preceding branch (if number is mentioned) or preceding/succeeding branch (if hash code is mentioned)
40. git checkout -b <Branch Name> – Creates a new branch (with the given branch name) pointing at the 'HEAD'
    (Alternative Syntax: git branch **<Branch Name>**; git checkout **<Branch Name>**)
41. git rebase <Main/Any Branch Name> – Makes a copy of the branch (where the 'HEAD' currently points) and attaches it directly below the main/master/any other branch mentioned
    (The main/master/any other branch remains at the same place, and is not updated)
    (The branch name of the new branch shifts to the new position)
42. git rebase -i HEAD~<Number> – Allows the user to omit/rearrange the first 'n' commits starting from 'HEAD' and creates a copy of them parallel to those branches
43. git cherry-pick <Branch Name 1/Hash Code 1> – Creates a copy of branch/snapshot, just below the 'HEAD' pointer and shifts the branch name (where 'HEAD' points) to the copied snapshot
    (There can be multiple branch names/hash codes)
44. git revert <HEAD/<Branch Name>/<Hash Code>> – Makes a copy of the snapshot (where the 'HEAD' currently points) just below the current one and points the HEAD there
45. git clone <URL> <Folder Name> – Clones a folder to the remote user/folder mentioned
46. git clone –shallow [Date] – Clones only certain commands
47. git remote add <Name of Remote> <URL> – Connects the local server with the remote server
    (Name of remote is usually 'origin' by convention)
48. git push <Remote Name> <Local branch Name>:<Remote Branch Name>– Sends the changes from the local computer to the remote
    (**Local branch name** – As saved on the local computer)
    (**Remote branch name** – As to be saved on the remote server)
49. git pull – Pulls files from the remote user and downloads them
    (git pull = git fetch + git merge)
50. git fetch – Pulls the files from the remote user but does not download them
51. gitignore – Allows the user to select the files that he/she wants to upload to the remote user

(Open the vi editor of gitignore, i.e., **.gitignore**, and write down the files that you do not wish to upload)

52. git bisect – Finds and displays the commit which has a bug
    (The above process is performed using 'Binary Search')

NOTE: 'HEAD' denotes which file we are currently looking at and 'master' refers to the main/most up-to-date branch of the project (It is created by default at the time of creation/modification of the git repository).

NOTE: Tree – Folder, Blob – File, Commit – Snapshot (Edited/changed file)

NOTE: If there are a lot of changes, then the files cannot be merged automatically. They must be manually merged, by aborting the previous change, and then continuing the merge function after making the desired changes in the file.

MAKE

SPECIAL COMMANDS:

1. % – Any String
2. : – Input files are compiled only if changes are made to it
   (Usually 'colons' are placed after the **<Output Filename>**)
3. $* – Stores the same string as in the previous lines or as stored in %
4. $@ – Stores the output filename

GENERAL COMMANDS:

1. I – Input
2. o – Output
3. gcc -o <Output Filename> <Input Filename 1> <Input Filename 2> -I – The input files are compiled by the compiler (gcc) and the output is stored in the output file
4. make [<target name>] – Checks if the code of the files under a target is edited/changed or not
   (Typing 'make' without a specified target name will check and compile only the first set of dependencies without a fall through)
   (Typing 'make' with a specified target name will check and compile only the code of the files under the target mentioned, without a fall through)
   ('Targets' can have any name, i.e., 'test', 'prod', 'release' etc.)
   Syntax of 'makefile':
   - <Target name 1>: <Dependency 1> <Dependency 2>
     <Rules/Statements>
     <Target name 2>: <Dependency 1> <Dependency 2>
     <Rules/Statements>

NOTE: There can be infinite number of targets as well as dependencies.

CALCULATOR

CALCULATOR:

1. bc – Calculates the value of any expression

   Syntax:

   - echo "<Expression>" | bc
   - echo <Expression> | bc
   - <Variable> = `echo "<Expression>" | bc`
     echo $<Variable>
   - Echo "<Variable Name>=<Numeric Value>;<Variable Name><Arithmetic Operator><Numeric Value>;<Variable Name>" | bc

     Example: Input – "var=10;var^=2;var" | bc, Output – 100
2. dc -expression "<Expression in Postfix Notation> p" – Evaluates and prints the value of the expression

PYTHON

DE-BUGGING:

1. pyflakes <filename> – Prints the errors in the file along with the line number
2. mypy <filename> – Prints the syntax, logical and type casting (int, float, char, etc.) errors in the file along with the line number

WEB-SERVER

1. python -m http.server <port number> – Starts a web server with the current directory as the www root

   Example:

   Create a sample index.html file and open browser with the url http://127.0.0.1:<port number> or http://localhost:<port number>

DE-BUGGER (ipdb/ pdb):

1. l(ist) – Lists the whole program
2. s(tep) – Prints each line of the program
3. c(ontinue) – Continues execution until the next line in the current function is reached
4. q(uit) – Quits the de-bugger

EXTRA

SLEEP:

1. sleep <Seconds> – Suspends execution of an interval of time/ Puts the computer in sleep mode for a certain interval of time.

NOTE: Press Ctrl + C to bring the terminal to normal mode.
NOTE: Press Ctrl + Z to suspend the sleep function

BUGS/ERRORS IN A PROGRAM:

1. shellcheck <filename> – Prints all the errors in a particular program

CHANGING SHELL TYPE:

1. bash – Changes the shell type to bash (Bourne Again Shell)
    a. vi ~/.bashrc – Any command stored in this editor becomes permanent for bash
    (\w means the working directory)
2. csh – Changes the shell type to csh (C Shell)
3. tmux – Changes the shell type to tmux
    a. tmux new -t <Name> – Opens a new tmux window having 'name' as its name
    b. tmux ls – Lists all the tmux windows currently running

NOTE: Default shell of Mac is zsh

URL:

1. ssh <URL> – Allows the user to access the shell of another user, only if the password is known
    a. ssh-keygen (Enter the hostname after pressing the 'Enter' key) – Makes a private and a public key (with 256 characters each)
    (The private key can be used to enter any machine without repeatedly giving the password)
2. curl [option] <URL> – Opens the source code of the URL

GENERATING KEYS FOR ENCRYPTION/DECRYPTION:

1. ssh-keygen (Enter the hostname after pressing the 'Enter' key) – Makes a private and a public key (with 256 characters each)
2. openssl aes-256-cbc -salt -in <filename> -out <new filename> – Encrypts a file and stores it as a new file under the new filename
    (Here, 'in' stands for input, 'out' stands for output and 'salt' strengthens a password)

3. openssl aes-256-cbc -d -in <encrypted filename> -out <new filename> – Decrypts a file and stores it as a new file under the new filename
   (Here, 'd' stands for the decrypt)

EDITTING VIDEOS AND IMAGES:

1. ffmpeg <Original Video (with extension)> <Video File (with extension) to be Changed To> – Changes the video type (Have to be downloaded) ##
2. convert <Original Image (with extension)> <Image File (with extension) to be Changed To> – Converts the image type **

EXIT:

1. exit – Exits from the shell

FUN

FUN COMMANDS:

1. banner <Text> – Prints anything in the form of a banner
2. banner -w <Width of the Text> <Text> – Prints the text in the form of a banner having the mentioned width
3. rev – Reverses any input
4. yes <Text> – Recursively prints the text.
   (Press Ctrl + L to terminate the process)

IMPORTANT

** – Commands not available in zsh.
– Require package installation.
TEXT – Syntax / Important Instructions