Shell:-

* Pwd (print working directory)
* ls (listing)
* If the first character starts with ‘/’, it is absolute, if it does not begin with ‘/’, it is relative
* Cd (change directory)
* Two dots without spaces followed cd .. moves you up by one level in the path (space between cd and dots)
* A single dot “.”, means current directory
* ~ means home directory, ls ~ or cd ~
* Cp and mv will overwrite files
* Cp (copy) – cp source destination - cp original.txt, duplicate.txt
* If the last command to cp is a directory, then it copies all the files to that directory
* Cp seasonal/autumn.csv seasonal/winter.csv backup – copies the files to backup
* Mv (move)
* Mv can also be used to rename files, mv old\_name.txt new\_name.txt
* Rm (remove) – rm file1 file2 – removes both of the files
* Rmdir – To delete directories (when empty) rmdir -r (recursively)
* Mkdir – To make new directory
* Store temporary files in /tmp – This is under the root ‘/ ’
* Cat (To show the files on the screen) – cat filename
* less – for paginated view – Press space for moving to next page and q to quit, :n – to move to next file (in case of multiple files), :p to go back to the previous one, :q to quit
* Example - less file1 file2
* Head filename – To see the first 10 lines
* **Tab completion is available in shell**
* Head -n 100 (will display the number of lines after -n)
* Put flags before the file name (a good convention)
* Ls -R (shows all the files under the directory) -F (prints additional info)
* To get help about a command – man – man head
* To select columns from a file – cut:-
  + Cut -f 2-5, 8 -d , values.csv (f – fields, d – delimiter)
* Using up arrow key to cycle through the previous commands
* History will print the list of commands used
* !55 – This will run the 55th command in the history
* !head – This will re-run the most recent use of the command
* Grep – select lines according to what they contain
  + -c: print a count of matching lines rather than the lines themselves
  + -h: do *not* print the names of files when searching multiple files
  + -i: ignore case (e.g., treat "Regression" and "regression" as matches)
  + -l: print the names of files that contain matches, not the matches
  + -n: print line numbers for matching lines
  + -v: invert the match, i.e., only show lines that *don't* match
  + Grep bicuspid seasonal/winter.csv
* Paste – To combine multiple files
* ‘>’ tells shell to redirect a command’s output to a file
  + Head -n 5 seasonal/summer.csv > top.csv
* Combining commands – use ‘|’ to combine two commands. The output of first command will be used as input to the second command
* head -n 5 seasonal/summer.csv | tail -n 3
* The command wc (short for "word count") prints the number of **c**haracters, **w**ords, and **l**ines in a file. You can make it print only one of these using -c, -w, or -l respectively.
* Wild characters –
  + \* - match 0 or more characters
  + ? matches a single character, so 201?.txt will match 2017.txt or 2018.txt, but not 2017-01.txt.
  + [...] matches any one of the characters inside the square brackets, so 201[78].txt matches 2017.txt or 2018.txt, but not 2016.txt.
  + {...} matches any of the comma-separated patterns inside the curly brackets, so {\*.txt, \*.csv} matches any file whose name ends with .txt or .csv, but not files whose names end with .pdf
* Sort -n (numerically), -r (reverse), -b (ignore leading blanks), -f (fold case)
* Uniq, is to remove adjacent duplicated lines

**Example**

* get the second column from seasonal/winter.csv
* remove the word "Tooth" from the output so that only tooth names are displayed
* sort the output so that all occurrences of a particular tooth name are adjacent; and
* display each tooth name once along with a count of how often it occurs.
* Ans – cut -d , -f 2 seasonal/winter.csv | grep -v tooth | sort | uniq -C
* Shell stores information in variables. Variables are of two types. Environment variables and shell variable. Environment variables are available all the time
* HOME – user’s home directory
* PWD – present working directory
* Shell – which shell program is being used
* User – user’s ID
* Set – list of all environment variables
* To get the value of variable, use $ with echo – echo $user
* To create shell variable, assign a value to a name – training=seasonal/summer.csv (no spaces before or after =)
* Looping in shell – for filetype in gif jpg png; do echo $filetype; done
* Editing files:-
* Nano filename – To open a file in an editor
  + Ctrl + K: delete a line.
  + Ctrl + U: un-delete a line.
  + Ctrl + O: save the file ('O' stands for 'output'). *You will also need to press Enter to confirm the filename!*
  + Ctrl + X: exit the editor.
* To save all the commands executed to a file, do the following:- history > steps.txt
* The saved steps can be re-used again:-
  + You have been using the shell interactively so far. But since the commands you type in are just text, you can store them in files for the shell to run over and over again. To start exploring this powerful capability, put the following command in a file called headers.sh:
  + head -n 1 seasonal/\*.csv
  + This command selects the first row from each of the CSV files in the seasonal directory. Once you have created this file, you can run it by typing:
  + bash headers.sh
  + This tells the shell (which is just a program called bash) to run the commands contained in the file headers.sh, which produces the same output as running the commands directly.
  + We can redirect the output of shell script to a file bash headers.sh > output.csv
* A file full of shell commands are called – shell script. No need to end the files with .sh
* We can parametrize the bash script using $@ - Ex:- sort $@ | uniq
* Instead of $@, we can also use $1, $2 in shell script