Text manipulation and scripting: some basics

`regex`	regular expressions
`grep`	global regex print: `[-i]` ignore case `[-l]` list names of matching files `[-n]` include line numbers `[-w]` whole word only `[-r]` recursively `[-h]` suppress output of filenames
`|`	pipe: connects two commands
`sed`	stream editor example: global search and replace (replaces all occurrences of old with new and saves a new file.

```
$sed 's/old/new/g' <old_file >new_file
```

`[s]` substitute, `[g]` global

`awk`	transforms text like sed, but is also its own programming language oriented toward delimited fields separated by spaces or tabs and is excellent for loops and arrays example: prints only the third column

```
$ cat <file> | awk  '{print $3}'
```

`vi`	visual editor (try VI tutor to get familiar with vi).
`sort`	sort lines of text files
`uniq`	report or omit repeated lines
`wc`	word count: `[-l]` line count `[-c]` byte count `[-m]` character count `[-L]` length of longest line `[-w]` word count
`pushd`	saves current working directory in memory
`popd`	returns the path at the top of the directory stack
`dirs`	print the directory stack, showing current directory first
`echo`	write to stdout (important tool to test a script before potentially making a mistake)
`./`	run a script
`control+c, control+d`	to escape a command
`control+z`	to escape a command (suspends, then you will have to kill)

regex special characters

`[...]`	set of possible matches
`\`	gives special meaning to a character or escapes special meaning of a special character
`^`	match beginning of line
`$`	match end of line
`.`	match any character
`|`	separates alternate possibilities
`?`	match the preceding element 0 or 1 time
`*`	match the preceding element >0 times
`+`	match the preceding element >1 times
`(...)`	group a series of elements
`\r`	carriage return
`\t`	tab
`\n`	line feed character
`\r\n`	line separator in Windows