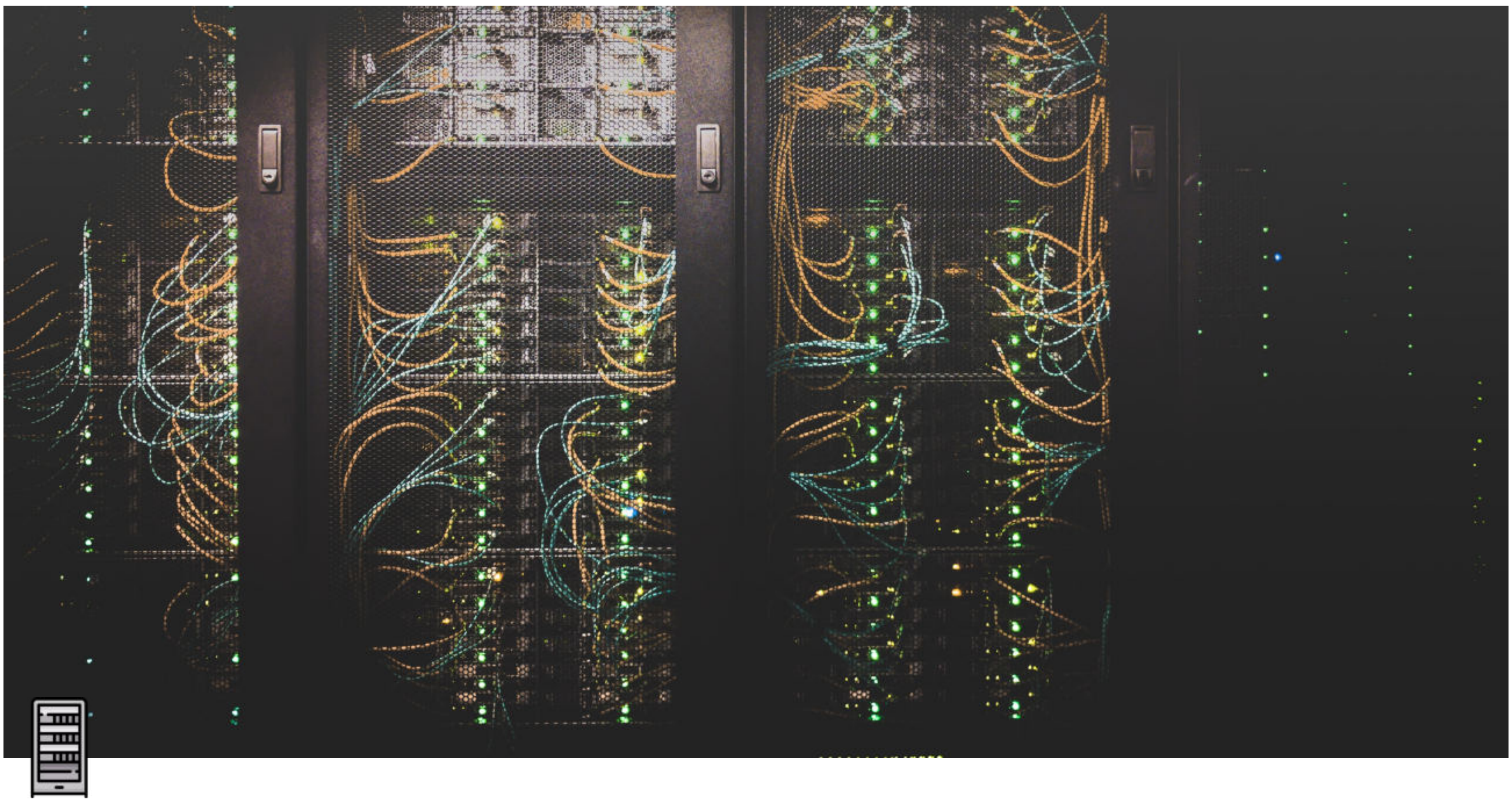


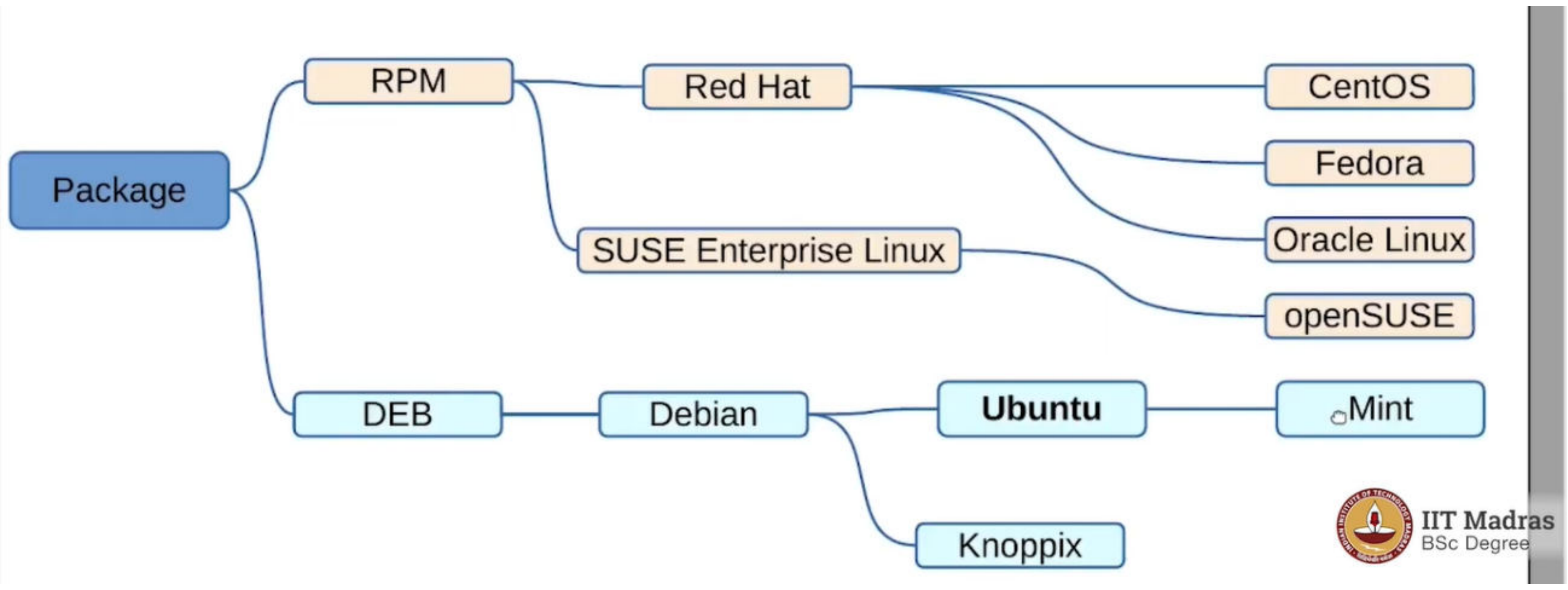
Software Management - pt. 1

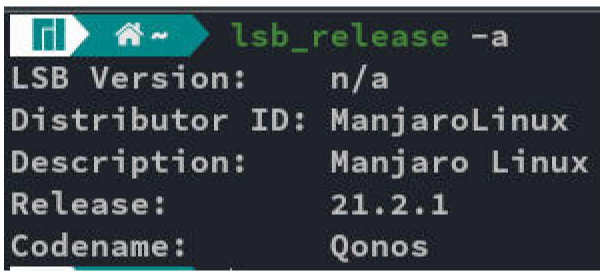
Type	📒 Lecture
Date	@January 16, 2022
Lecture #	1
Lecture URL	https://youtu.be/hG-bxCmfArc
Notion URL	https://21f1003586.notion.site/Shell-variables-3c228fce1aef41719f77bc4b8e786ff1
Week #	4

Need for a package manager

- Tools for installing, updating, removing, managing software
- Install new/updated software across network
- Package — File look up, both ways
- Database of packages on the system including versions
- Dependency checking
- Signature verification tools
- Tools for building packages

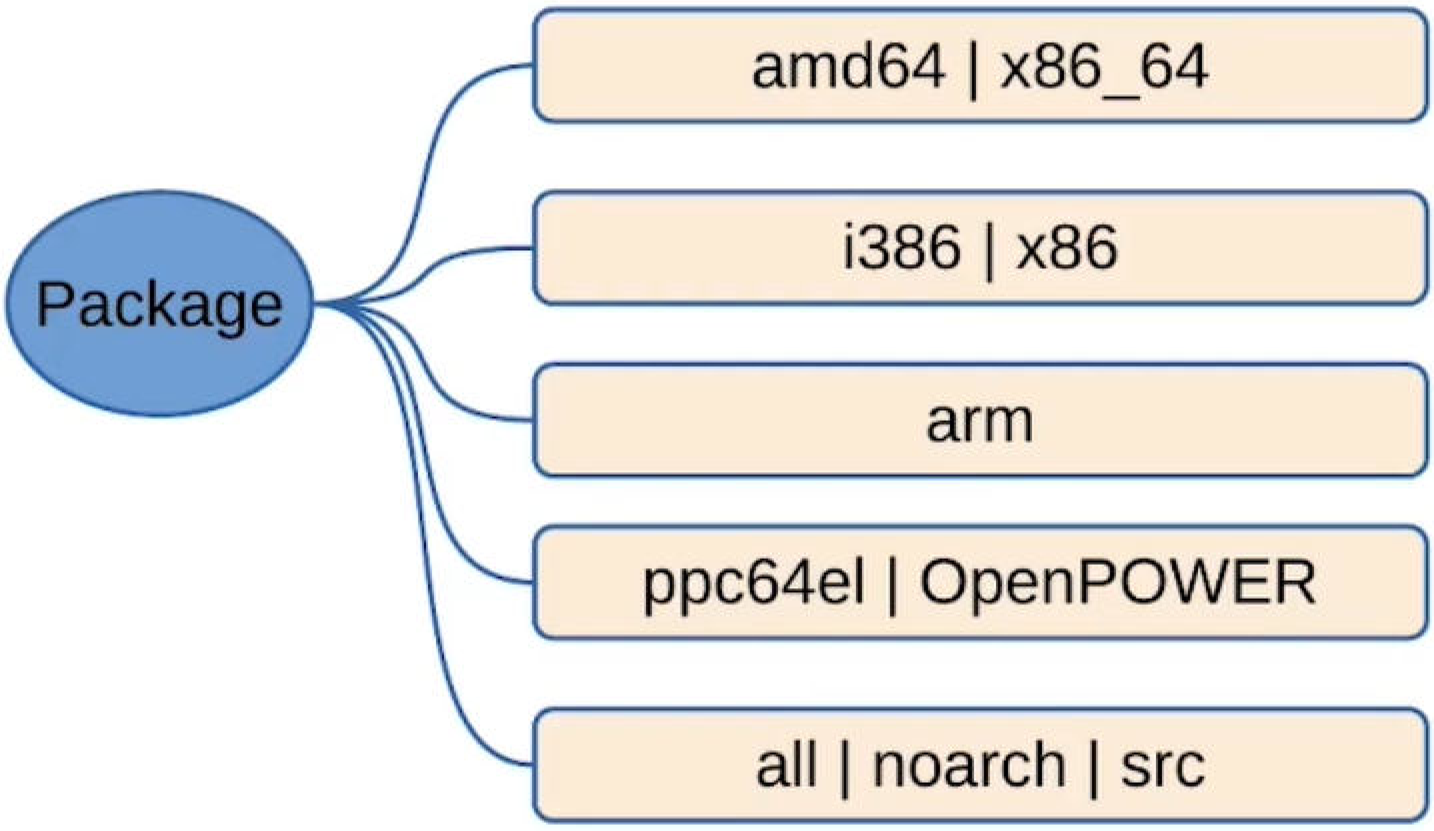
Package Types





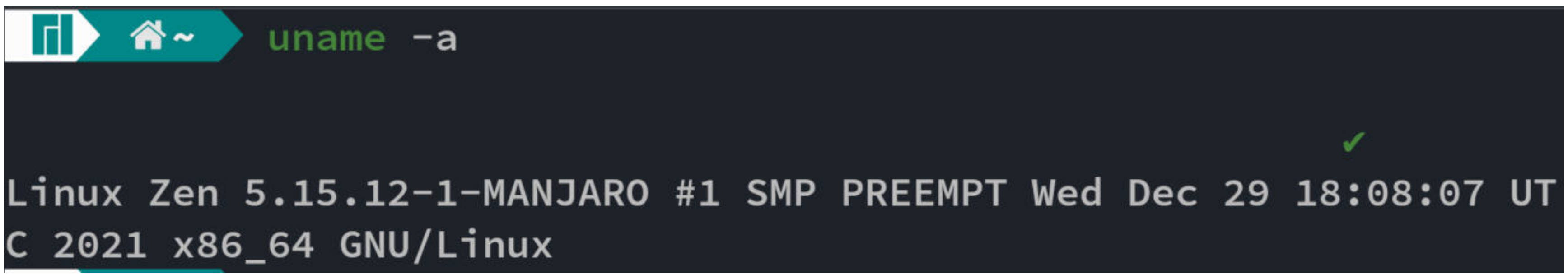
```
 ~  lsb_release -a
LSB Version:    n/a
Distributor ID: ManjaroLinux
Description:    Manjaro Linux
Release:        21.2.1
Codename:       Qonos
```

Architecture



To know your system architecture

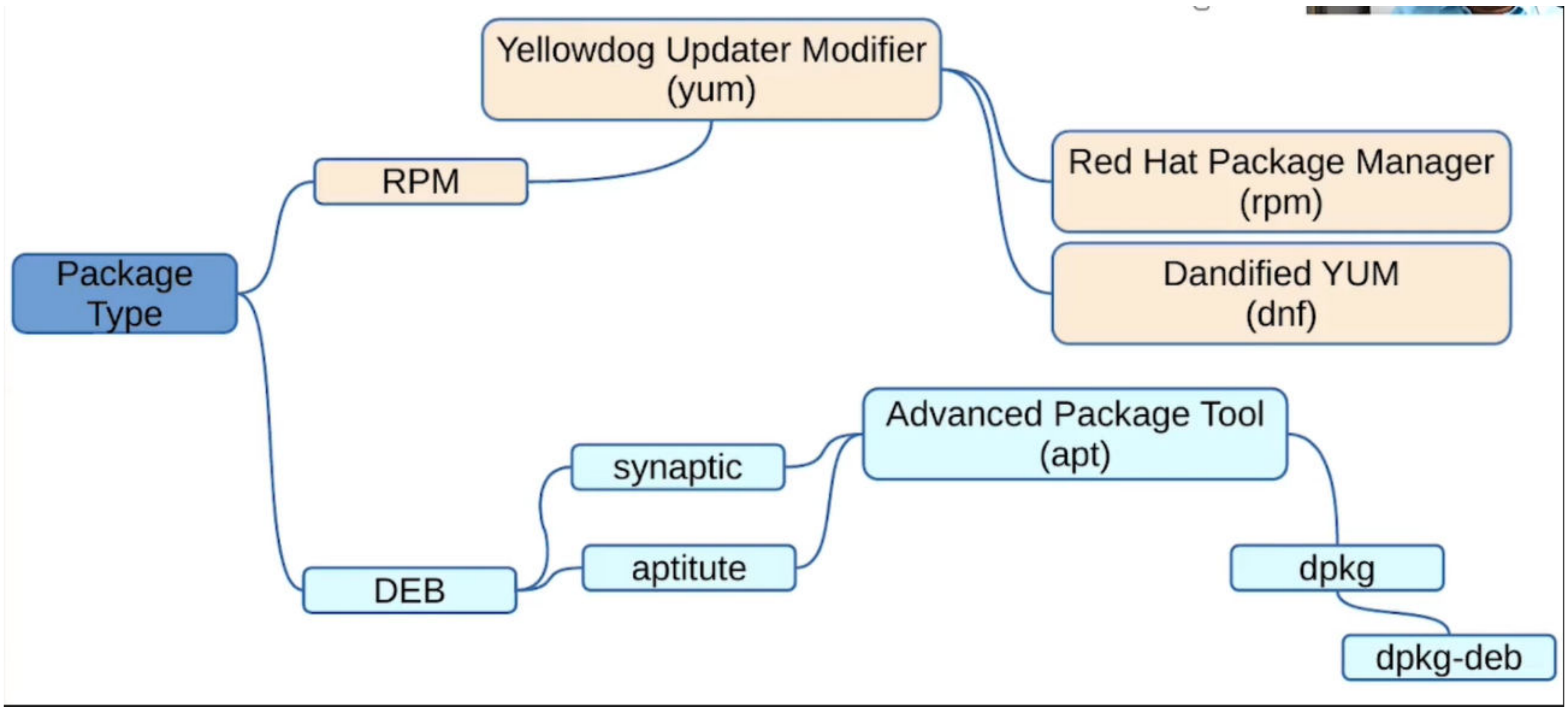
~~Can't spell architecture without arch btw~~



```
 ~  uname -a

                                                            ✔
Linux Zen 5.15.12-1-MANJARO #1 SMP PREEMPT Wed Dec 29 18:08:07 UT
C 2021 x86_64 GNU/Linux
```

Tools

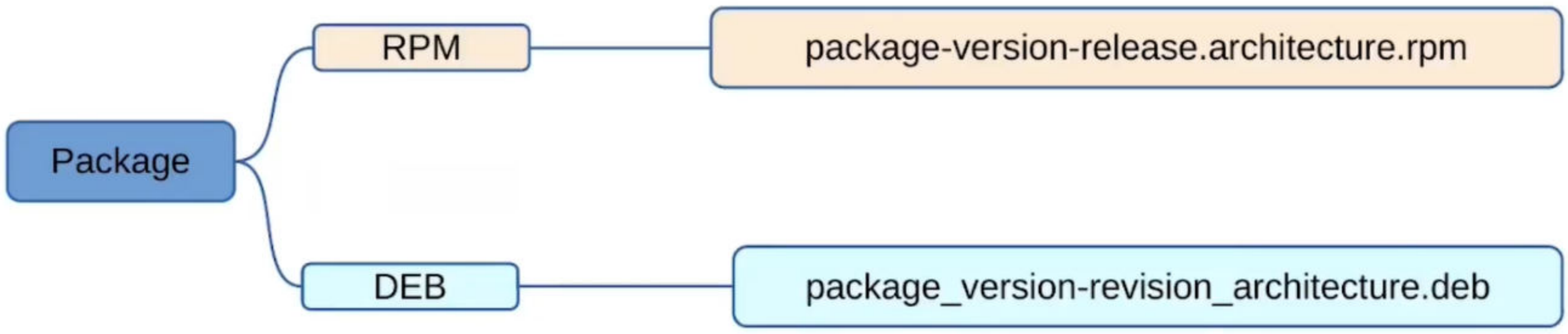

Package Type
RPM
Yellowdog Updater Modifier (yum)
Red Hat Package Manager (rpm)
Dandified YUM (dnf)
DEB
synaptic
aptitute
Advanced Package Tool (apt)
dpkg
dpkg-deb

Package Management in Ubuntu using `apt`

Inquiring package db

- Search packages for a keyword
  - `apt-cache search keyword`
- List all packages
  - `apt-cache pkgnames`
  - Try `apt-cache pkgnames <starting-char>` to search to packages with the few starting chars
- Display package records of a package
  - `apt-cache show -a package`

Package names

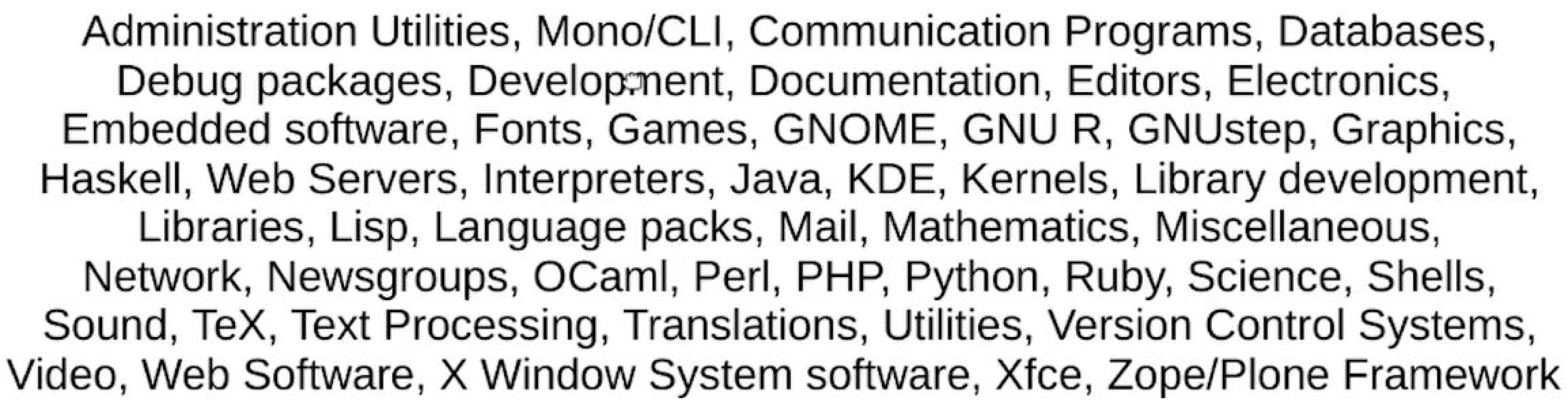

Package
RPM
package-version-release.architecture.rpm
DEB
package_version-revision_architecture.deb

Package priorities

- **required** → essential to the proper functioning of a system
- **important** → provides functionality that enables the system to run well
- **standard** → included in a standard system installation
- **optional** → can omit if you do not have enough storage
- **extra** → could conflict with packages with higher priority, has specialized requirements, install only if needed

Package sections

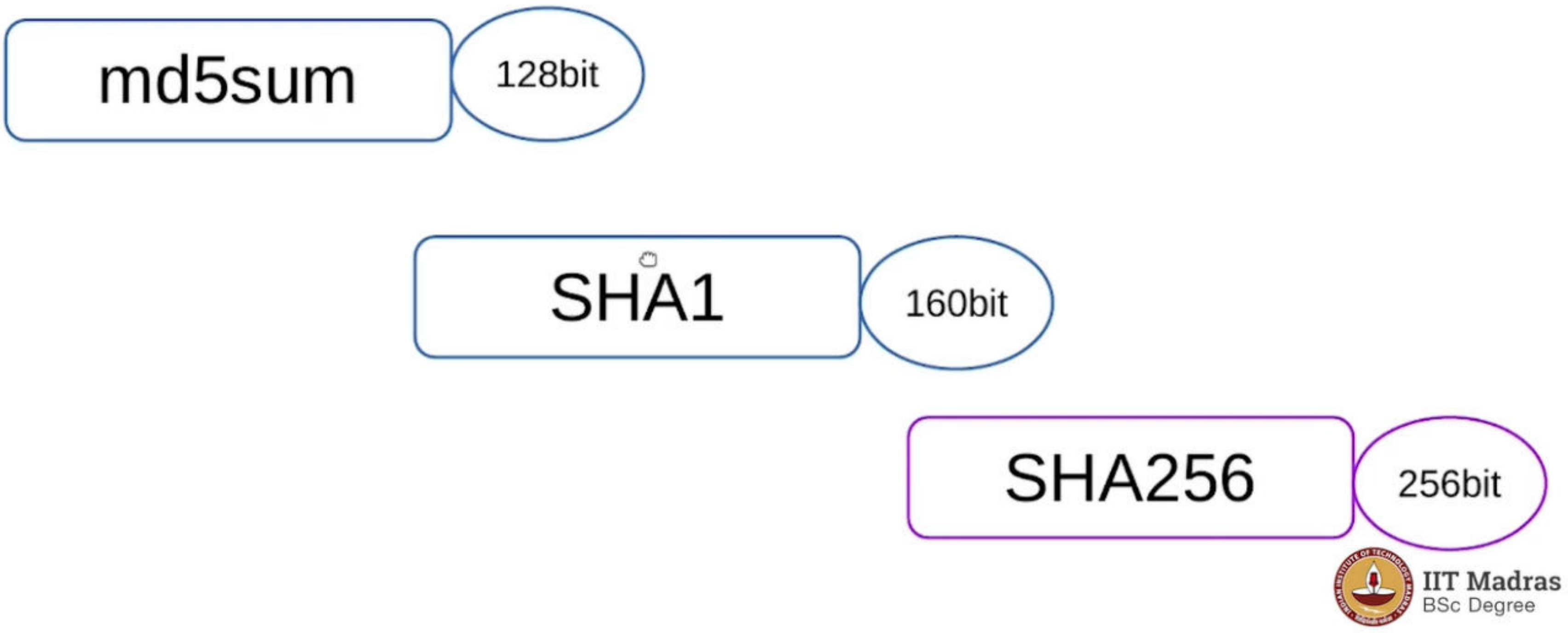
https://packages.ubuntu.com/focal



Administration Utilities, Mono/CLI, Communication Programs, Databases, Debug packages, Development, Documentation, Editors, Electronics, Embedded software, Fonts, Games, GNOME, GNU R, GNUstep, Graphics, Haskell, Web Servers, Interpreters, Java, KDE, Kernels, Library development, Libraries, Lisp, Language packs, Mail, Mathematics, Miscellaneous, Network, Newsgroups, OCaml, Perl, PHP, Python, Ruby, Science, Shells, Sound, TeX, Text Processing, Translations, Utilities, Version Control Systems, Video, Web Software, X Window System software, Xfce, Zope/Plone Framework

Checksums

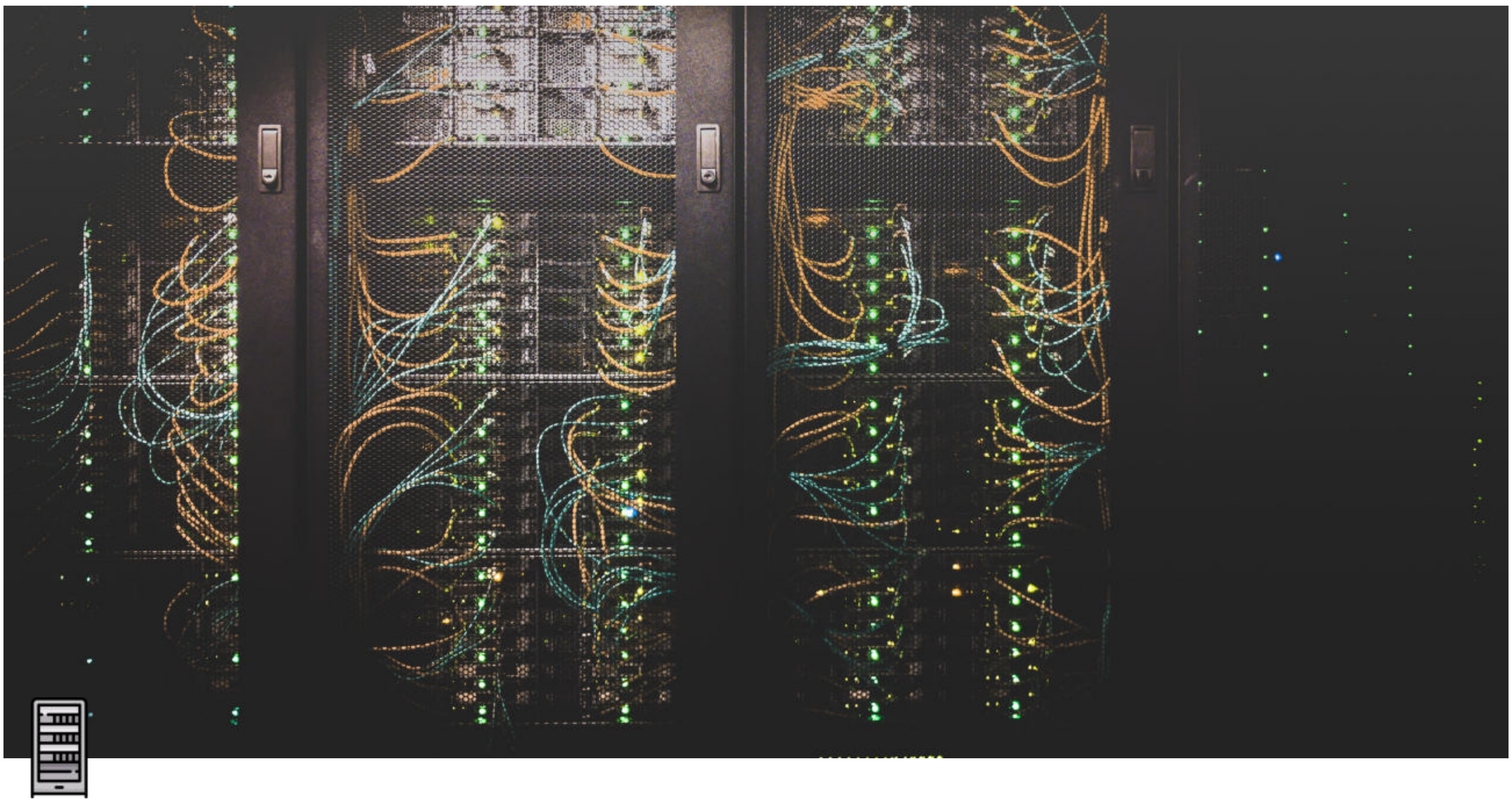
A way to verify that the packages that we are installing are legit



`md5sum <filename>` to get the MD5 checksum of a file, use it to compare it to other files

`sha1sum <filename>` to get the SHA1 checksum of a file, use it to compare it to other files

`sha256sum <filename>` to get the SHA256 checksum of a file, use it to compare it to other files



Software Management - pt. 2

Type	📒 Lecture
Date	@January 16, 2022
Lecture #	2
Lecture URL	https://youtu.be/ctn_s7SDTV0
Notion URL	https://21f1003586.notion.site/Software-Management-b0cd46d7790f479aacf4378fbe9611cf
Week #	4

Only sudoers can install/upgrade/remove packages → `/etc/sudoers`

We can view the `sudoers` file by typing `sudo cat /etc/sudoers`

This will ask for the user password and depending on the `su` status of the user, it will either display the file or show an error like `user is not in the sudoers file. This incident will be reported.`

How can a superuser find out the failed `sudo` attempt by non-su?

Go to `/var/log`

Look for `auth.log` file, these events are reported in this file

Sources for packages

Location: `/etc/apt`

File → `sources.list` → Contains repo URLs for Ubuntu pkgs

Folder → `sources.list.d` → Contains repo URLs for 3rd party pkgs

To get updates

`sudo apt-get update`

To upgrade the pkgs (if an upgrade is available)

`sudo apt-get upgrade`

You can pass the `-y` flag to not get bugged by the Y/N prompt

To remove older, usually obsolete pkgs

`sudo apt autoremove`

To remove a specific pkg

`sudo apt-get remove <pkg-name>`

Installing/Updating

- Install a package
  - `apt-get install <pkg-name>`
- Reinstall a package
  - `apt-get reinstall <pkg-name>`

Removing/Cleaning up

- Remove packages that were automatically installed to satisfy a dependency and not needed
  - `apt-get autoremove`
- Clean local repository of retrieved package files
  - `apt-get clean`
- Purge package files from the system
  - `apt-get purge package`

Package management in Ubuntu using `dpkg`

Text info regarding packages found at `/var/lib/dpkg`

Files: `arch, available, status`

Folder: `info`

Using `dpkg`

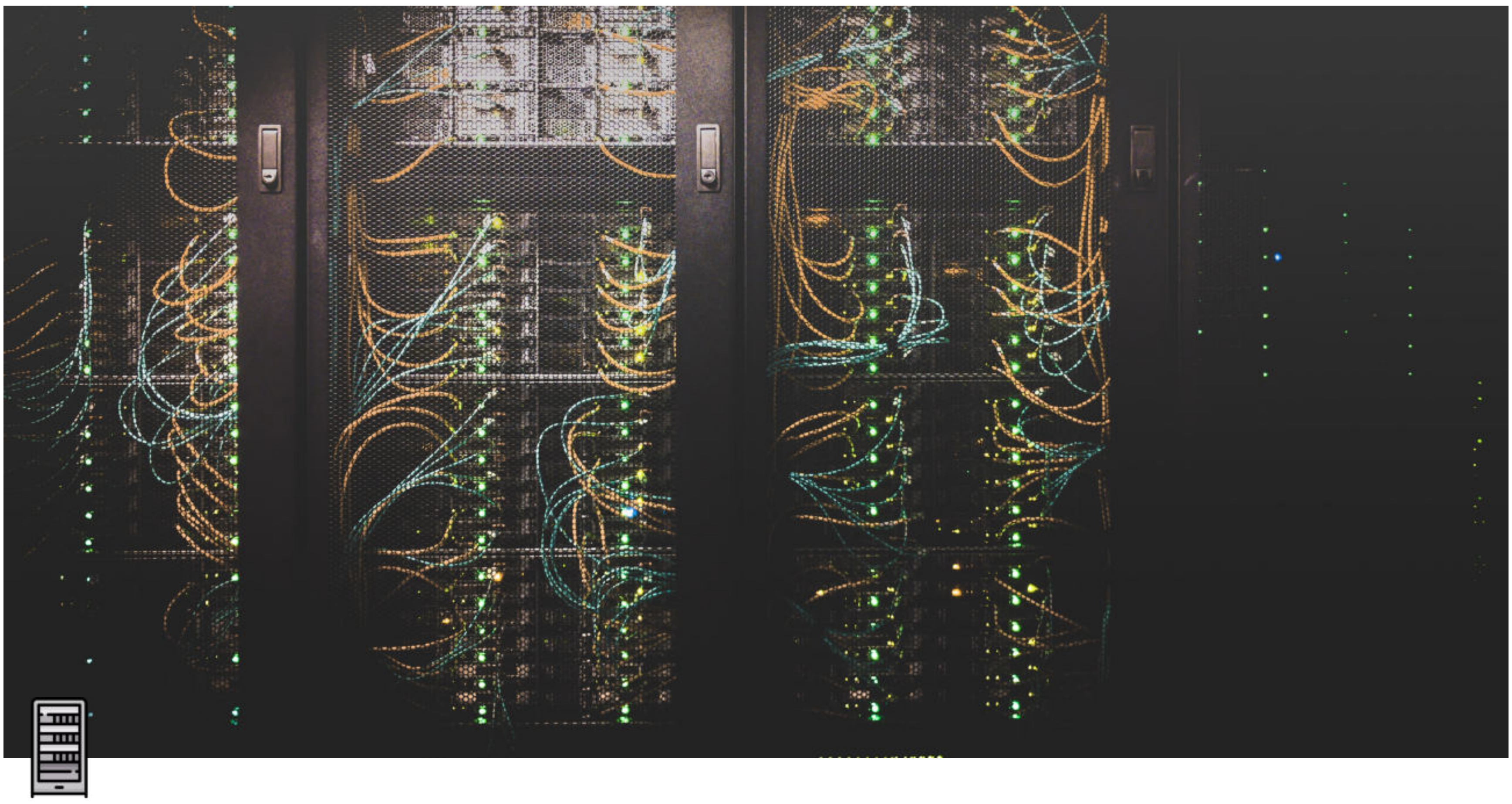
- List all the packages whose names match the pattern
  - `dpkg -l pattern`
- List installed files that came from packages
  - `dpkg -L package`
- Report the status of packages
  - `dpkg -s package`
- Search installed packages for a file
  - `dpkg -S pattern`
- A tool to query the `dpkg` database
  - `dpkg-query`
  - Example usage
    - `dpkg-query -W -f=’${Section} ${binary:Package}\n’`

Installing a `deb` package

`dpkg -i filename.deb`

By default, use package management pointing to a reliable repository

Uninstalling packages using `dpkg` is ***NOT*** recommended



Pattern Matching - pt. 1

Type	📒 Lecture
Date	@January 17, 2022
Lecture #	3
Lecture URL	https://youtu.be/1y85iTaqq8Y
Notion URL	https://21f1003586.notion.site/Pattern-Matching-pt-1-8cb1aa5c0e6c42b2a9b517b040006284
Week #	4

Pattern matching

`regex` & `grep`

Regex

- regex is a pattern template to filter text
- BRE: POSIX Basic Regular Expression engine
- ERE: POSIX Extended Regular Expression engine

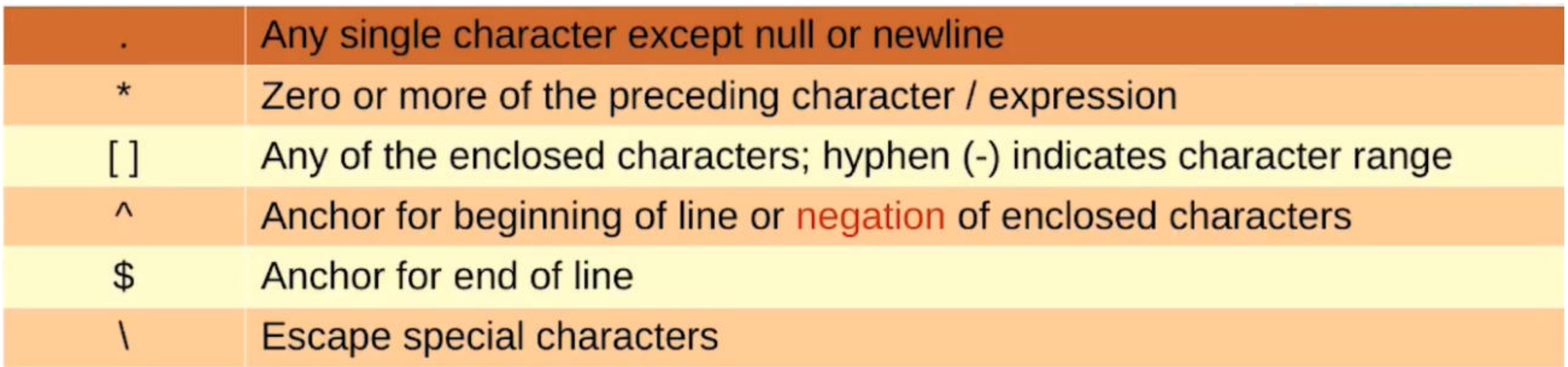
Why learn regex?

- Languages: Java, Perl, Python, Ruby, ...
    - To process input from the user
    - To perform string operations
- Tools: `grep`, `sed`, `awk`, ...
- Applications: MySQL, PostgreSQL, ...

Usage

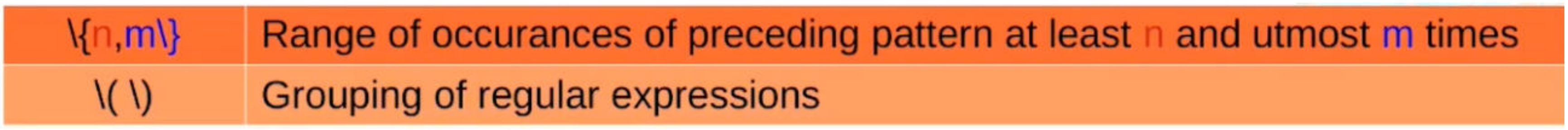
- `grep ‘pattern’ filename`
- `command | grep ‘pattern’`
- Default engine: BRE
- Switch to use ERE:
    - `egrep ‘pattern’ filename`
    - `grep -E ‘pattern’ filename`

Special characters (BRE & ERE)



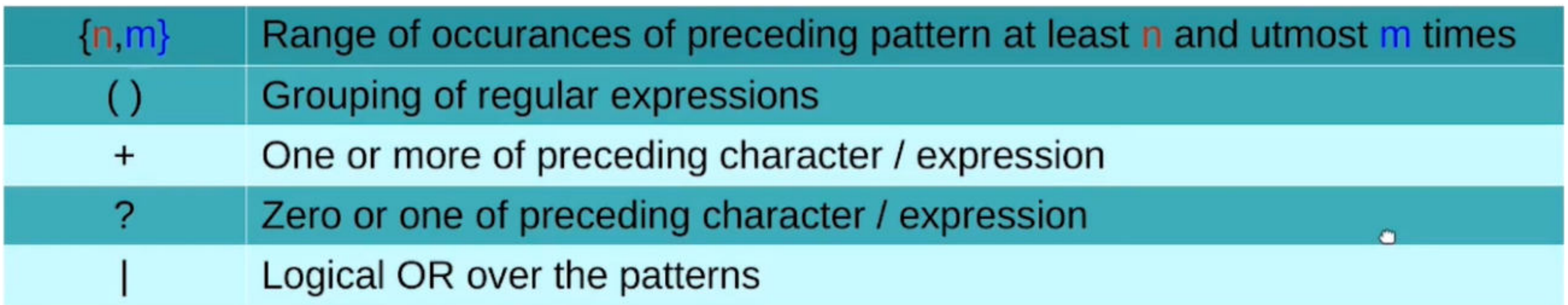
.	Any single character except null or newline
*	Zero or more of the preceding character / expression
[ ]	Any of the enclosed characters; hyphen (-) indicates character range
^	Anchor for beginning of line or negation of enclosed characters
$	Anchor for end of line
\	Escape special characters

Special characters (BRE)



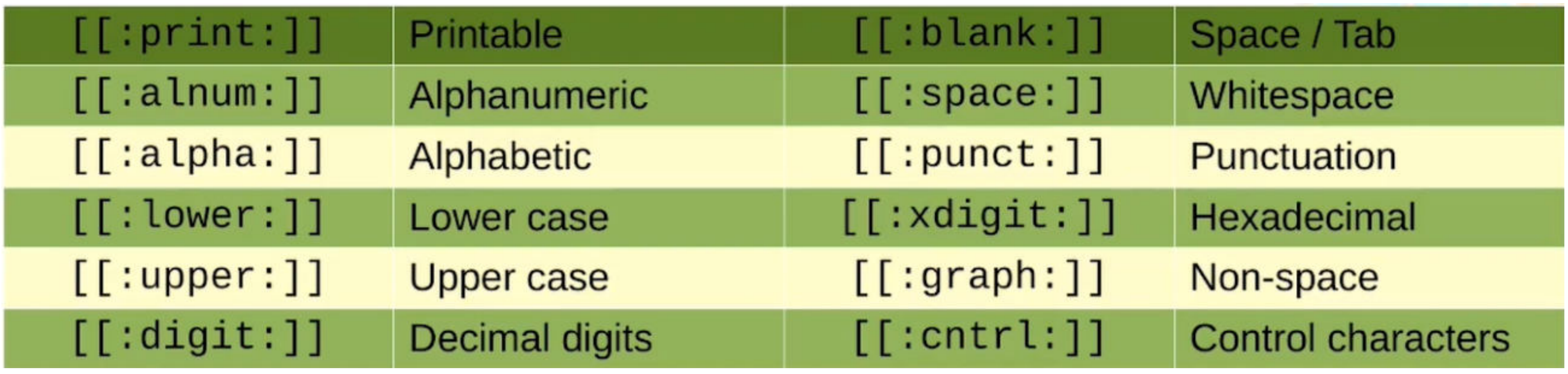
\{n,m\}	Range of occurances of preceding pattern at least n and utmost m times
\(\)	Grouping of regular expressions

Special characters (ERE)



{n,m}	Range of occurances of preceding pattern at least n and utmost m times
( )	Grouping of regular expressions
+	One or more of preceding character / expression
?	Zero or one of preceding character / expression
|	Logical OR over the patterns

Character classes



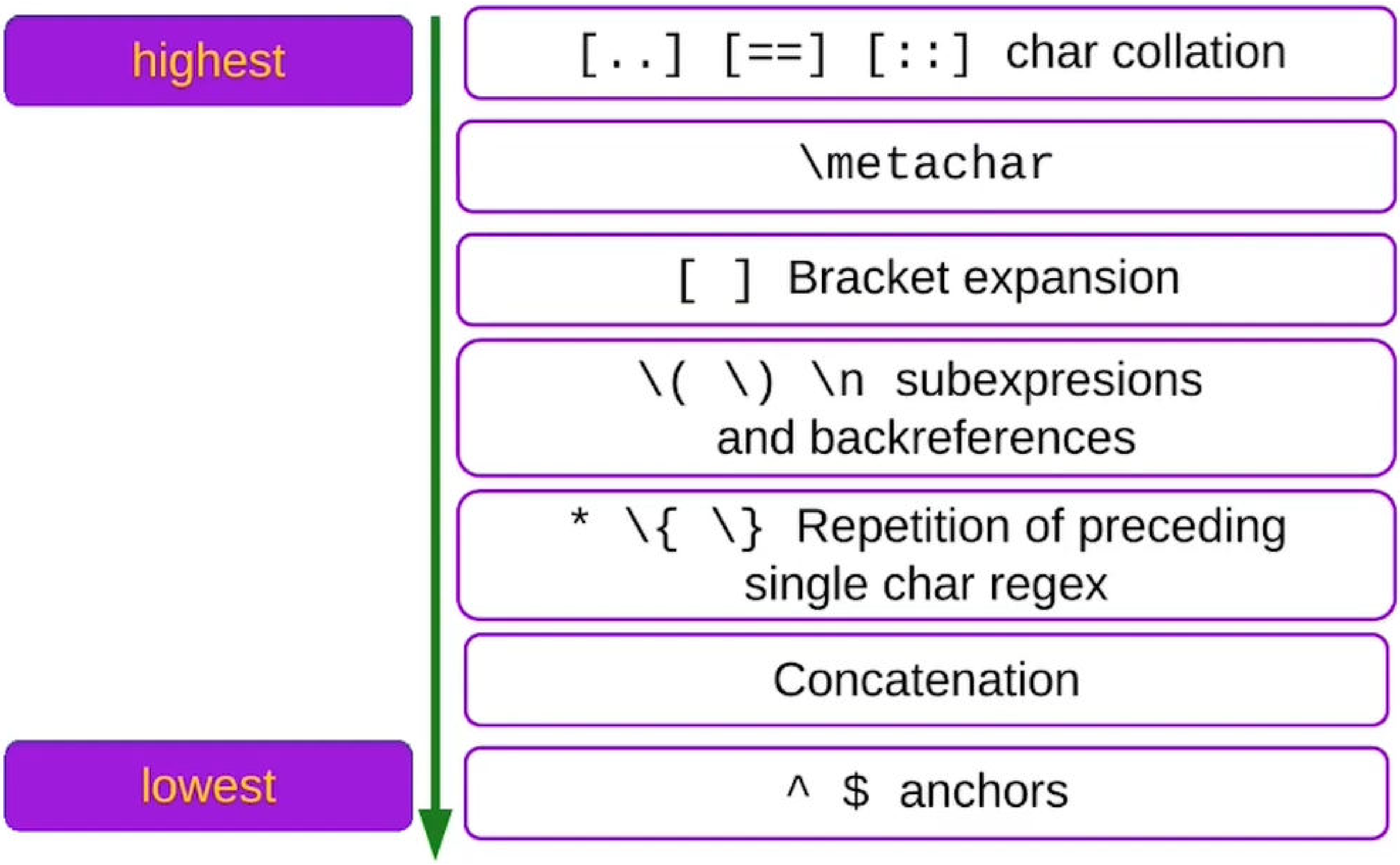
`[[:print:]]`	Printable	`[[:blank:]]`	Space / Tab
`[[:alnum:]]`	Alphanumeric	`[[:space:]]`	Whitespace
`[[:alpha:]]`	Alphabetic	`[[:punct:]]`	Punctuation
`[[:lower:]]`	Lower case	`[[:xdigit:]]`	Hexadecimal
`[[:upper:]]`	Upper case	`[[:graph:]]`	Non-space
`[[:digit:]]`	Decimal digits	`[[:cntrl:]]`	Control characters

To make it slightly easy to match pattern

Backreferences

- `\1` through `\9`
- `\n` matches whatever was matched by the `n` th earlier paranthesized sub-expression
- A line with two occurences of *“hello”* will be matched using: `\(hello\).*\1`

BRE operator precedence



highest

lowest

[..] [==] [::] char collation

\metachar

[ ] Bracket expansion

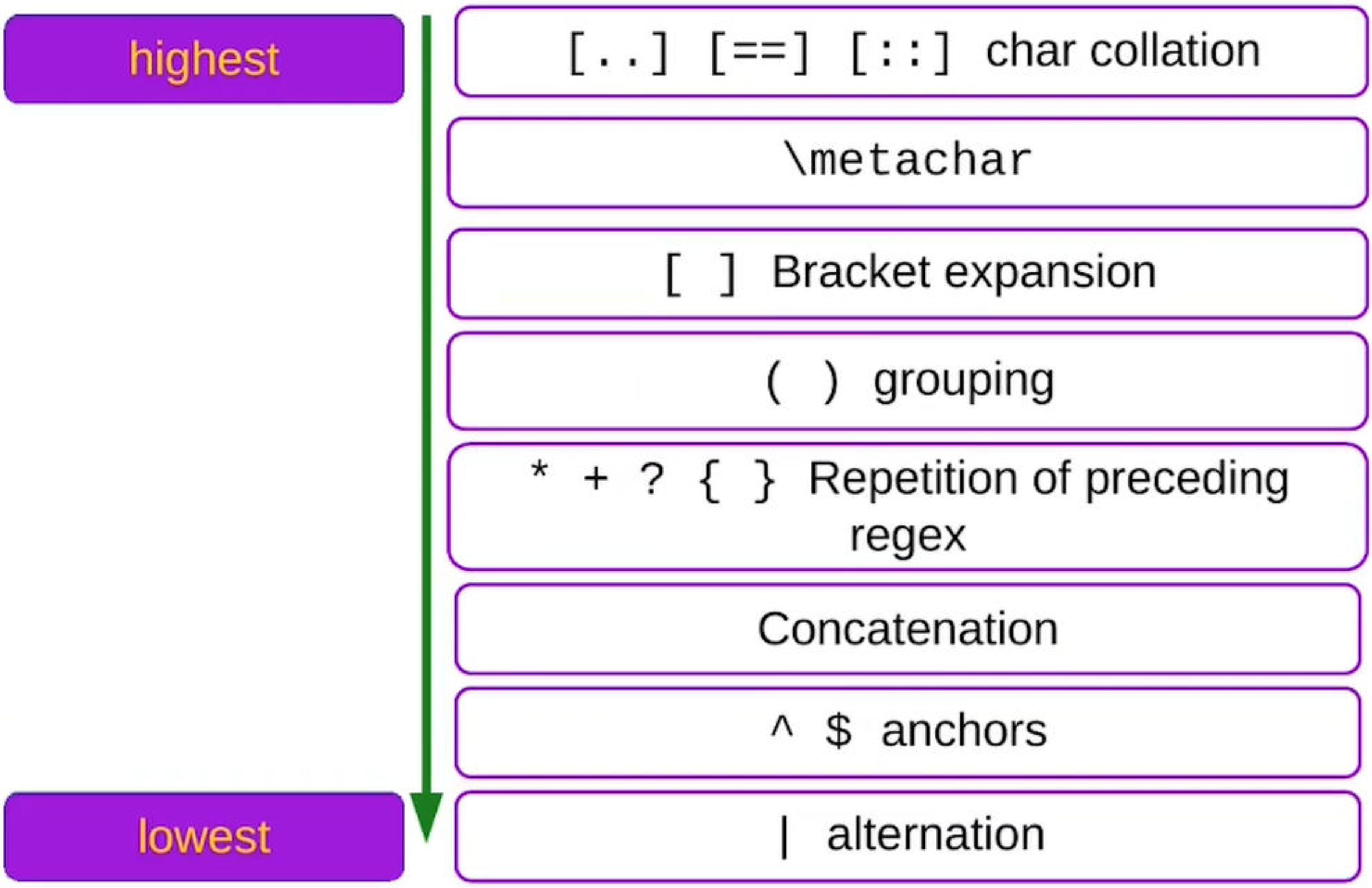
\(\) \n subexpresions and backreferences

* \{ \} Repetition of preceding single char regex

Concatenation

^ $ anchors

ERE operator precedence



highest

lowest

[..] [==] [::] char collation

\metachar

[ ] Bracket expansion

( ) grouping

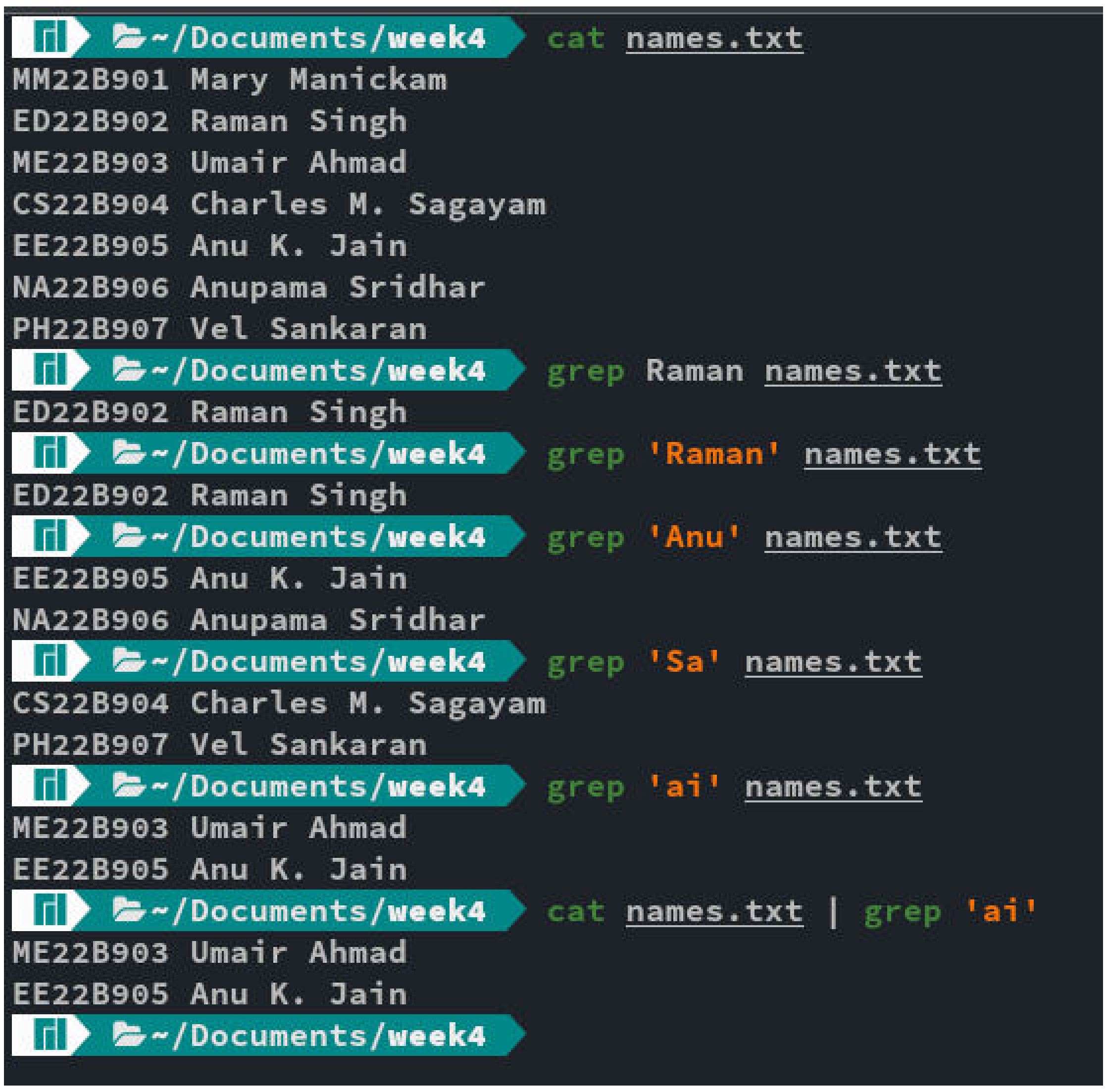
* + ? { } Repetition of preceding regex

Concatenation

^ $ anchors

| alternation

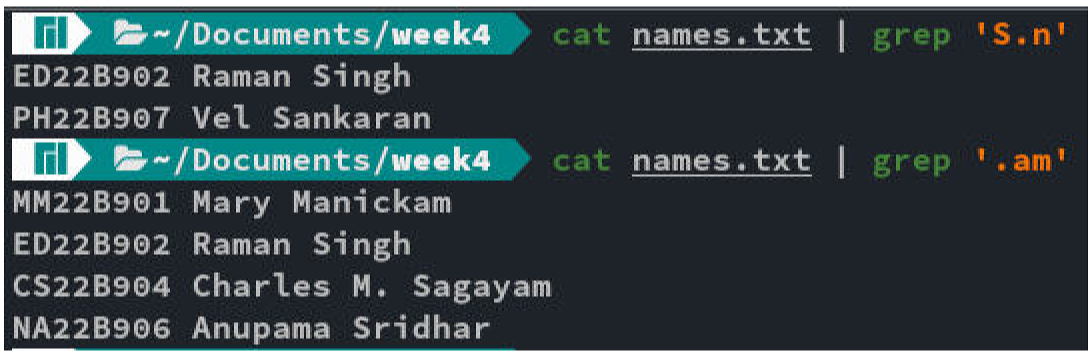
Uses of `grep`



```
~/Documents/week4  cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
~/Documents/week4  grep Raman names.txt
ED22B902 Raman Singh
~/Documents/week4  grep 'Raman' names.txt
ED22B902 Raman Singh
~/Documents/week4  grep 'Anu' names.txt
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
~/Documents/week4  grep 'Sa' names.txt
CS22B904 Charles M. Sagayam
PH22B907 Vel Sankaran
~/Documents/week4  grep 'ai' names.txt
ME22B903 Umair Ahmad
EE22B905 Anu K. Jain
~/Documents/week4  cat names.txt | grep 'ai'
ME22B903 Umair Ahmad
EE22B905 Anu K. Jain
~/Documents/week4
```

Usage of `.` in the `grep` command

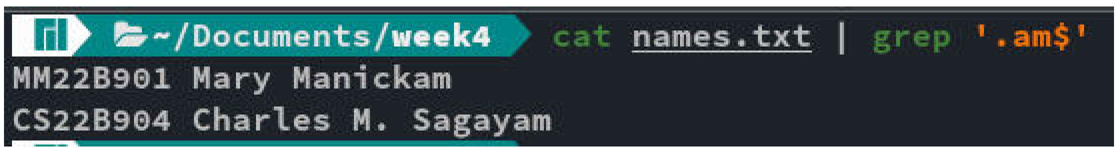
`.` is like a wildcard for a single character



```
~/Documents/week4  cat names.txt | grep 'S.n'
ED22B902 Raman Singh
PH22B907 Vel Sankaran
~/Documents/week4  cat names.txt | grep '.am'
MM22B901 Mary Manickam
ED22B902 Raman Singh
CS22B904 Charles M. Sagayam
NA22B906 Anupama Sridhar
```

`$` is used to denote an anchor

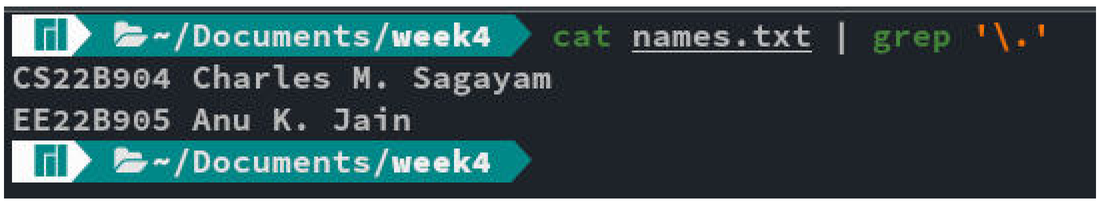
Like a pattern at the end of the line



```
~/Documents/week4  cat names.txt | grep '.am$'
MM22B901 Mary Manickam
CS22B904 Charles M. Sagayam
```

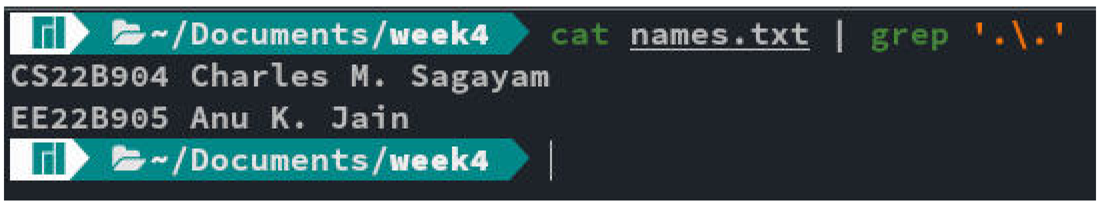
Well what if your name contains a `.`

Then escape it using the `\` character



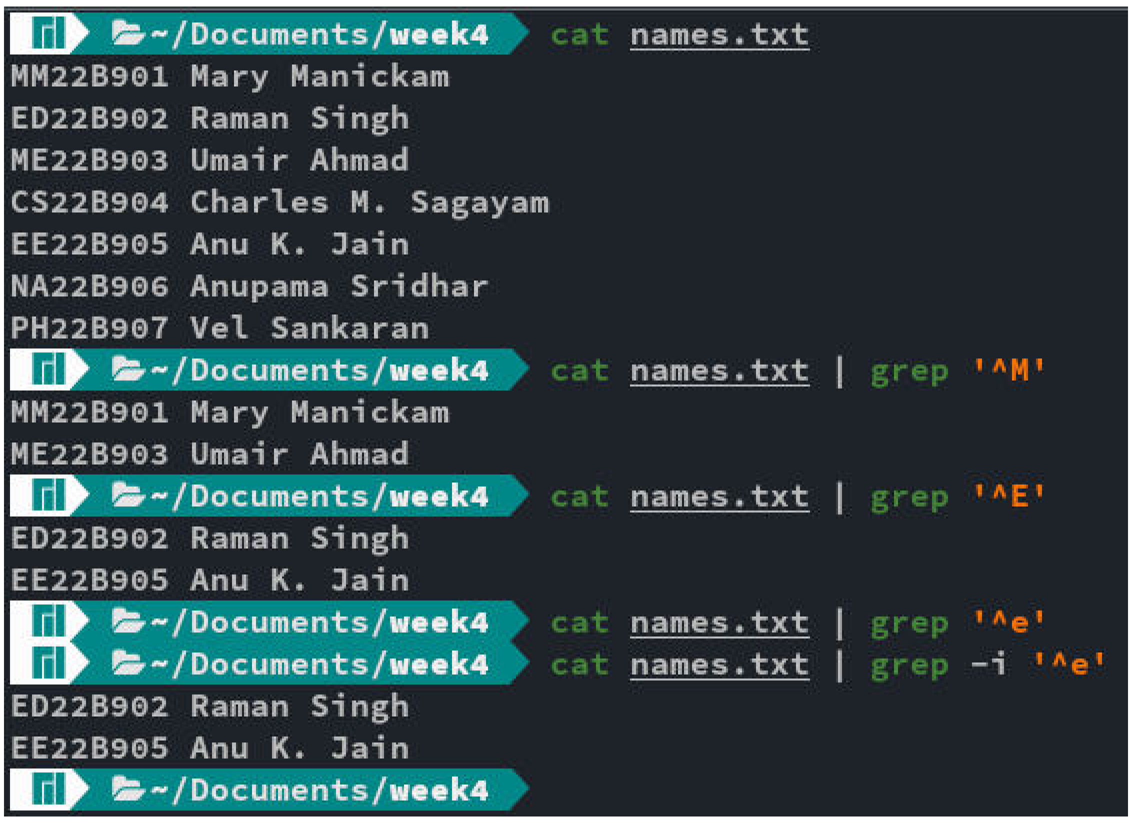
```
~/Documents/week4  cat names.txt | grep '\.'
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
~/Documents/week4
```

If we want the `.` to be necessarily after a character



```
~/Documents/week4  cat names.txt | grep '.\.'
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
~/Documents/week4
```

Match string using anchors at the beginning ask `grep` to ignore the case by passing in the `-i` flag

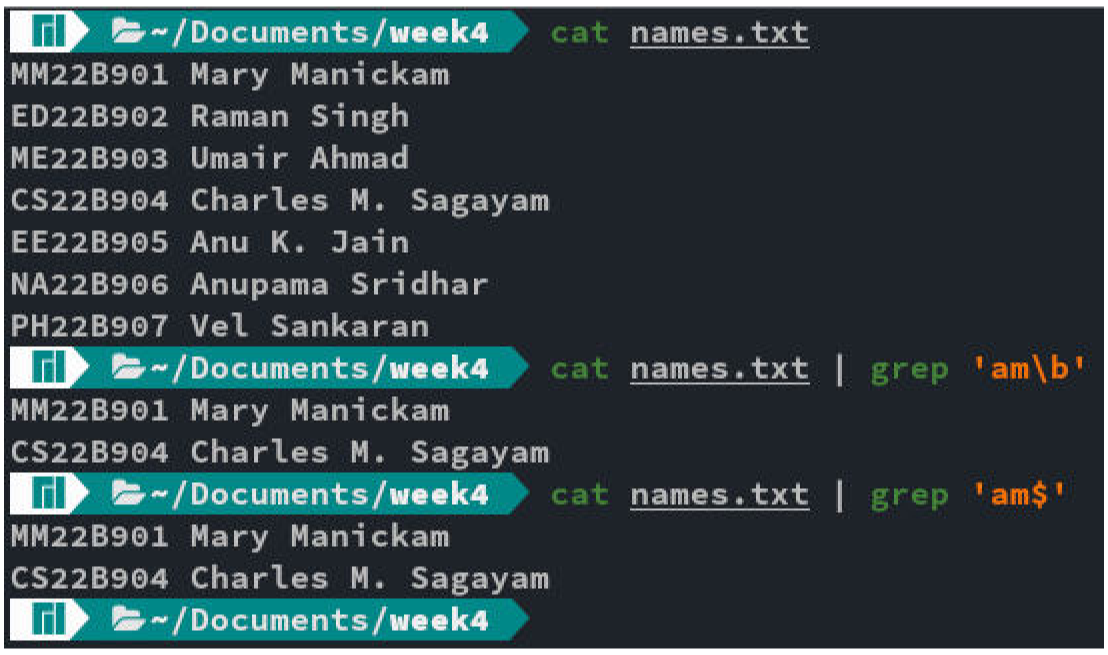


```
~/Documents/week4  cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
~/Documents/week4  cat names.txt | grep '^M'
MM22B901 Mary Manickam
ME22B903 Umair Ahmad
~/Documents/week4  cat names.txt | grep '^E'
ED22B902 Raman Singh
EE22B905 Anu K. Jain
~/Documents/week4  cat names.txt | grep '^e'
~/Documents/week4  cat names.txt | grep -i '^e'
ED22B902 Raman Singh
EE22B905 Anu K. Jain
~/Documents/week4
```

Match a pattern at the end of the line, a word boundary one might say

`\b` looks for a word boundary, so that pattern could also occur at the end of a word in the middle of a line

`$` looks for line boundary only, so the pattern occurs at the end of the line



```
~/Documents/week4  cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
~/Documents/week4  cat names.txt | grep 'am\b'
MM22B901 Mary Manickam
CS22B904 Charles M. Sagayam
~/Documents/week4  cat names.txt | grep 'am$'
MM22B901 Mary Manickam
CS22B904 Charles M. Sagayam
~/Documents/week4
```

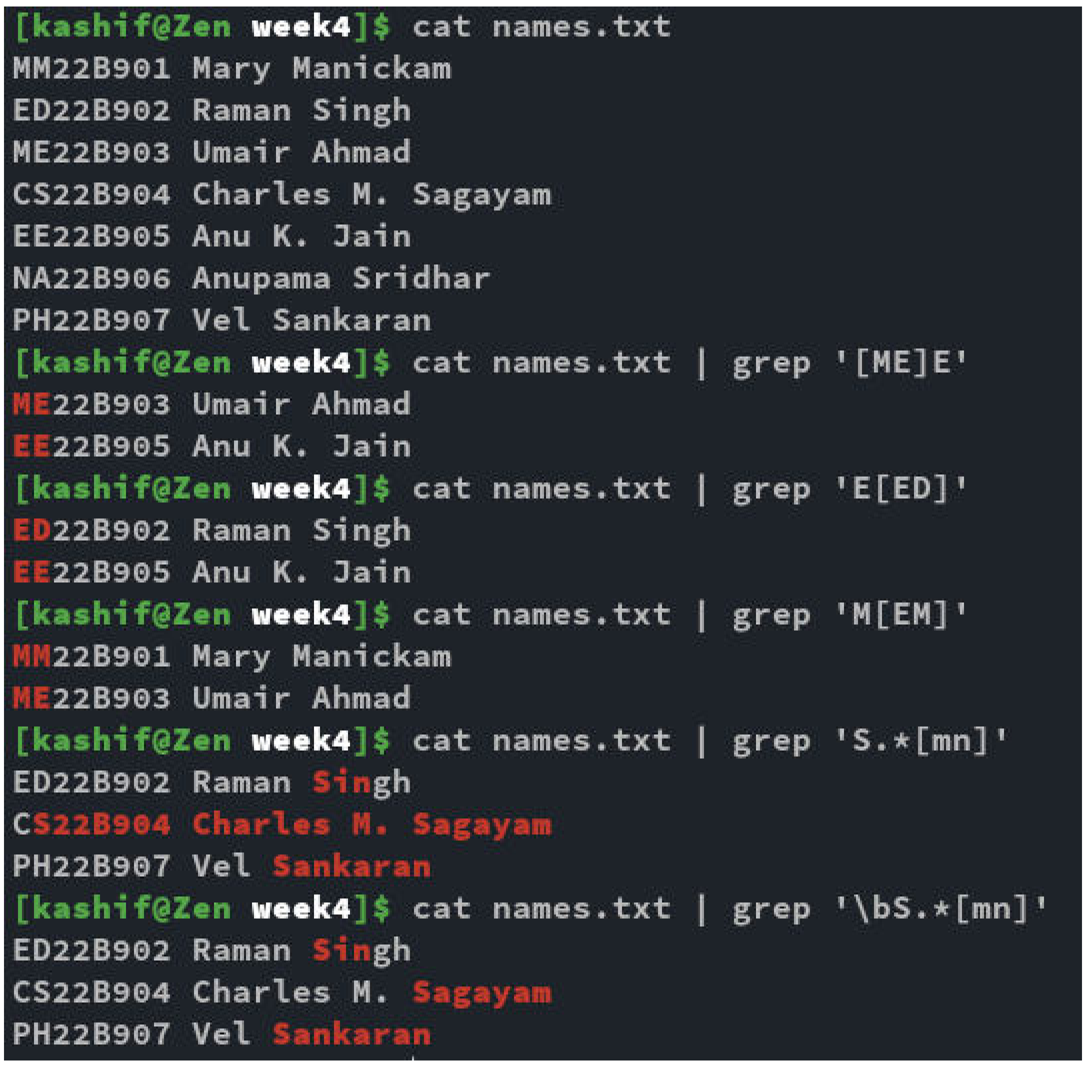
Usage of square brackets `[]`

Here, the first character in the pattern is followed by either of the 2 characters given in `[]`

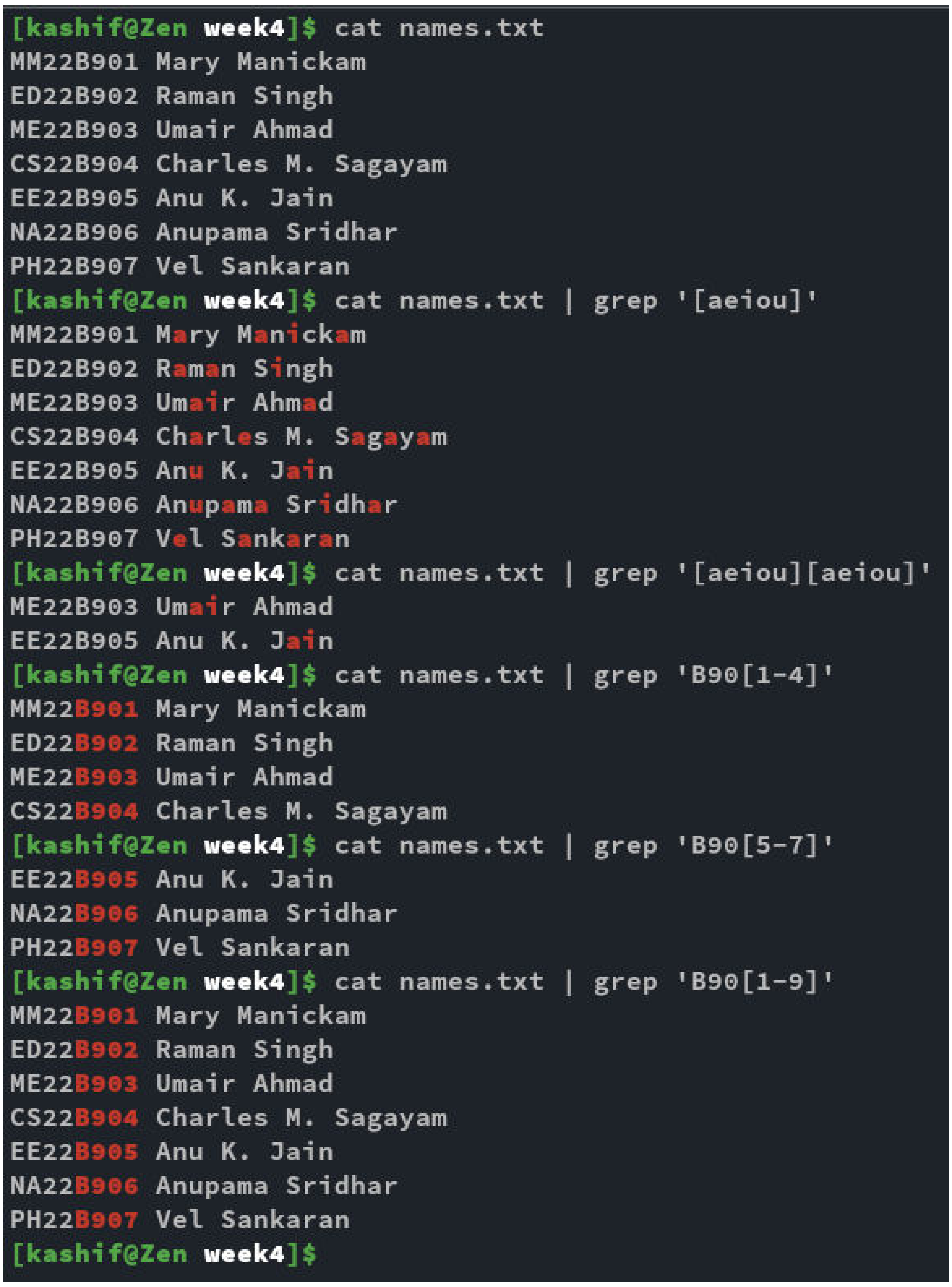
In the `grep 'S.*[mn]'` command, it matches from the start of the line

We add `\b` to mark a word boundary just to match it within a word

> *Had to switch to `bash` to show the formatting*

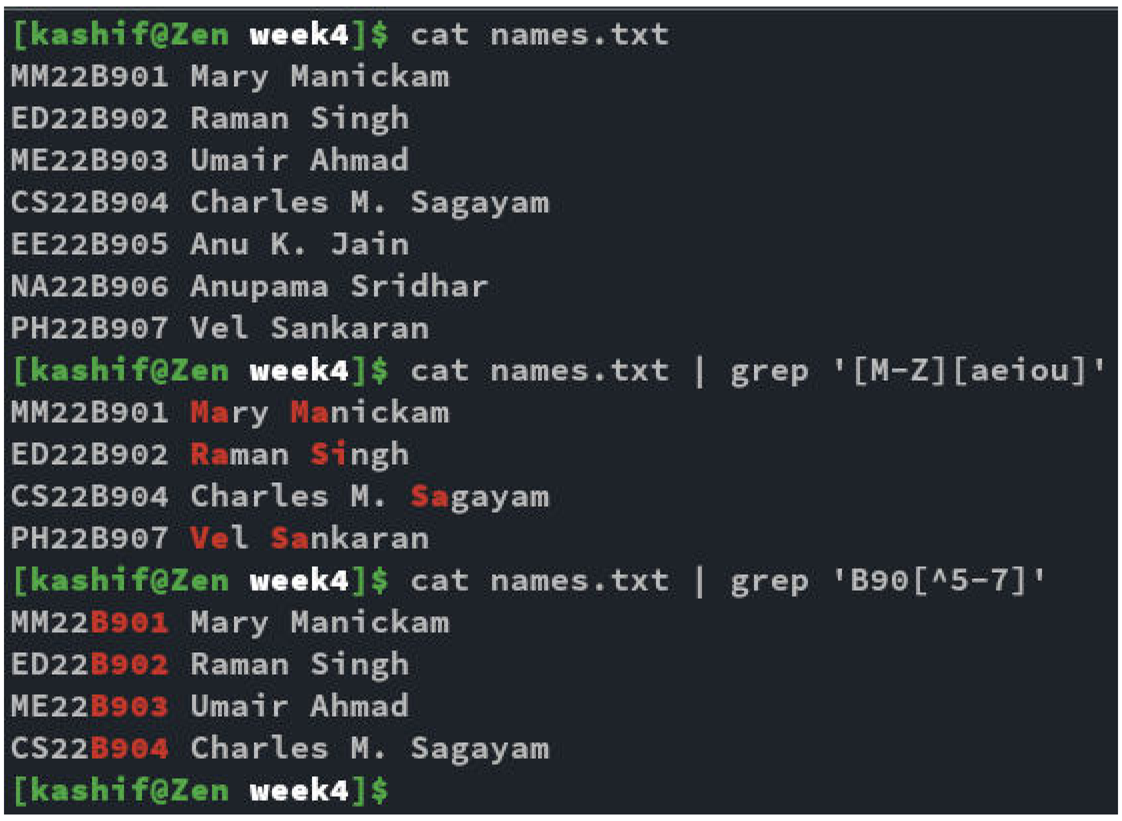


```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '[ME]E'
ME22B903 Umair Ahmad
EE22B905 Anu K. Jain
[kashif@Zen week4]$ cat names.txt | grep 'E[ED]'
ED22B902 Raman Singh
EE22B905 Anu K. Jain
[kashif@Zen week4]$ cat names.txt | grep 'M[EM]'
MM22B901 Mary Manickam
ME22B903 Umair Ahmad
[kashif@Zen week4]$ cat names.txt | grep 'S.*[mn]'
ED22B902 Raman Singh
CS22B904 Charles M. Sagayam
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '\bS.*[mn]'
ED22B902 Raman Singh
CS22B904 Charles M. Sagayam
PH22B907 Vel Sankaran
```



```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '[aeiou]'
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '[aeiou][aeiou]'
ME22B903 Umair Ahmad
EE22B905 Anu K. Jain
[kashif@Zen week4]$ cat names.txt | grep 'B90[1-4]'
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
[kashif@Zen week4]$ cat names.txt | grep 'B90[5-7]'
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep 'B90[1-9]'
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$
```

The last command in the following screenshot does negation

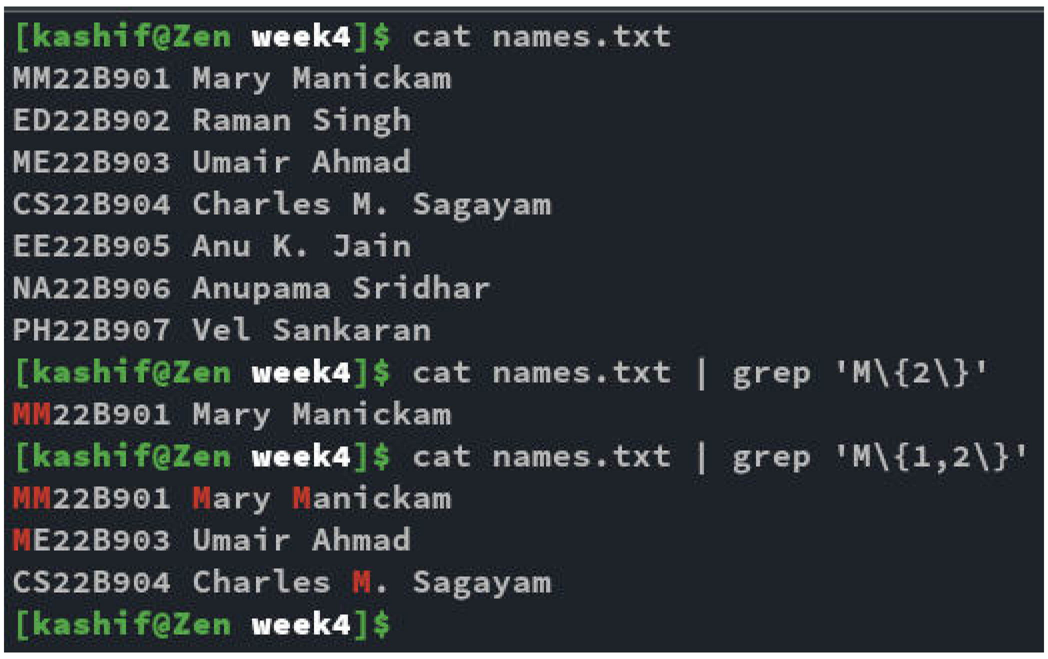


```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '[M-Z][aeiou]'
MM22B901 Mary Manickam
ED22B902 Raman Singh
CS22B904 Charles M. Sagayam
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep 'B90[^5-7]'
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
[kashif@Zen week4]$
```

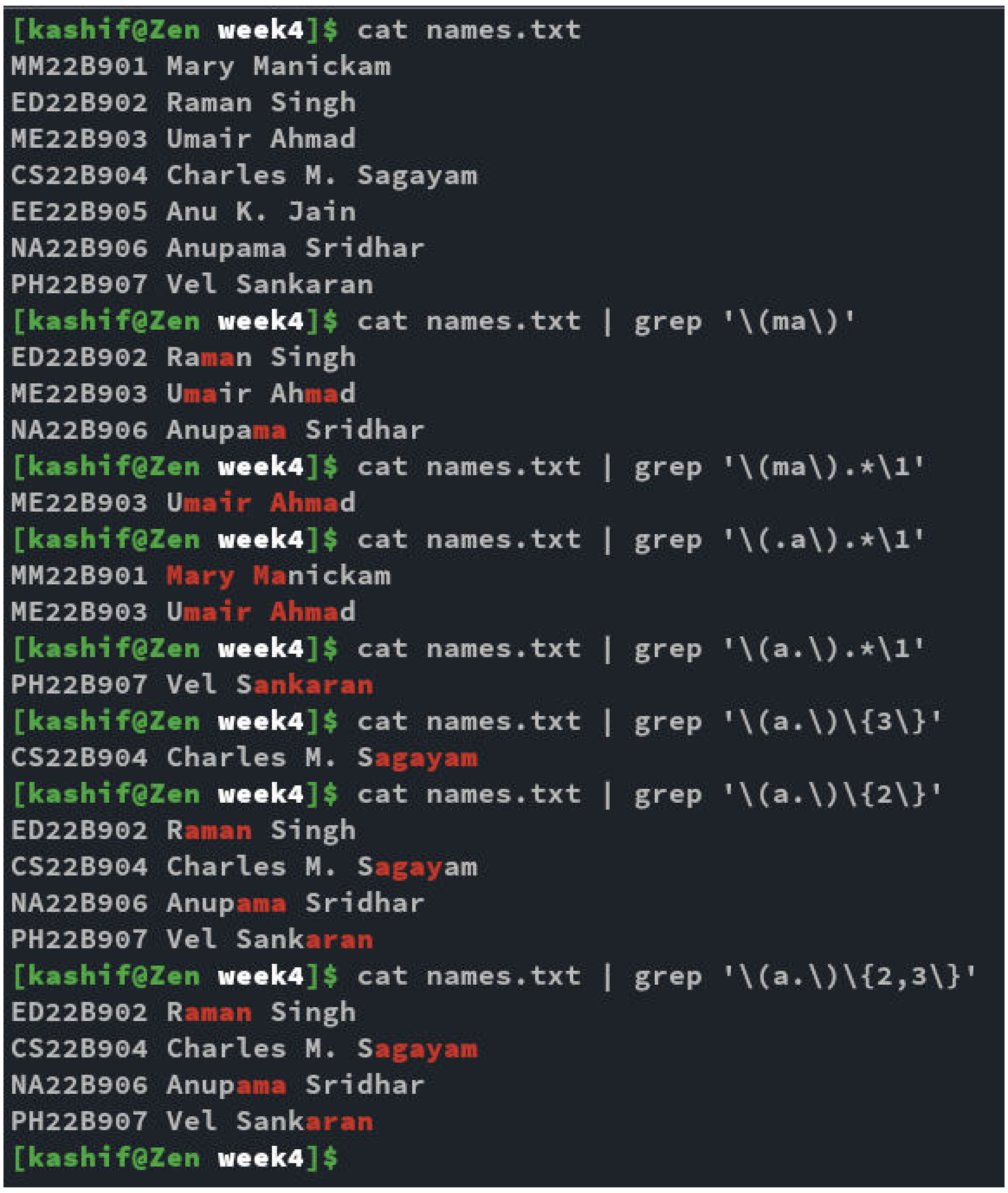
Number of times a character should occur

In the curly braces, we provide the # of times the preceding character should be matched

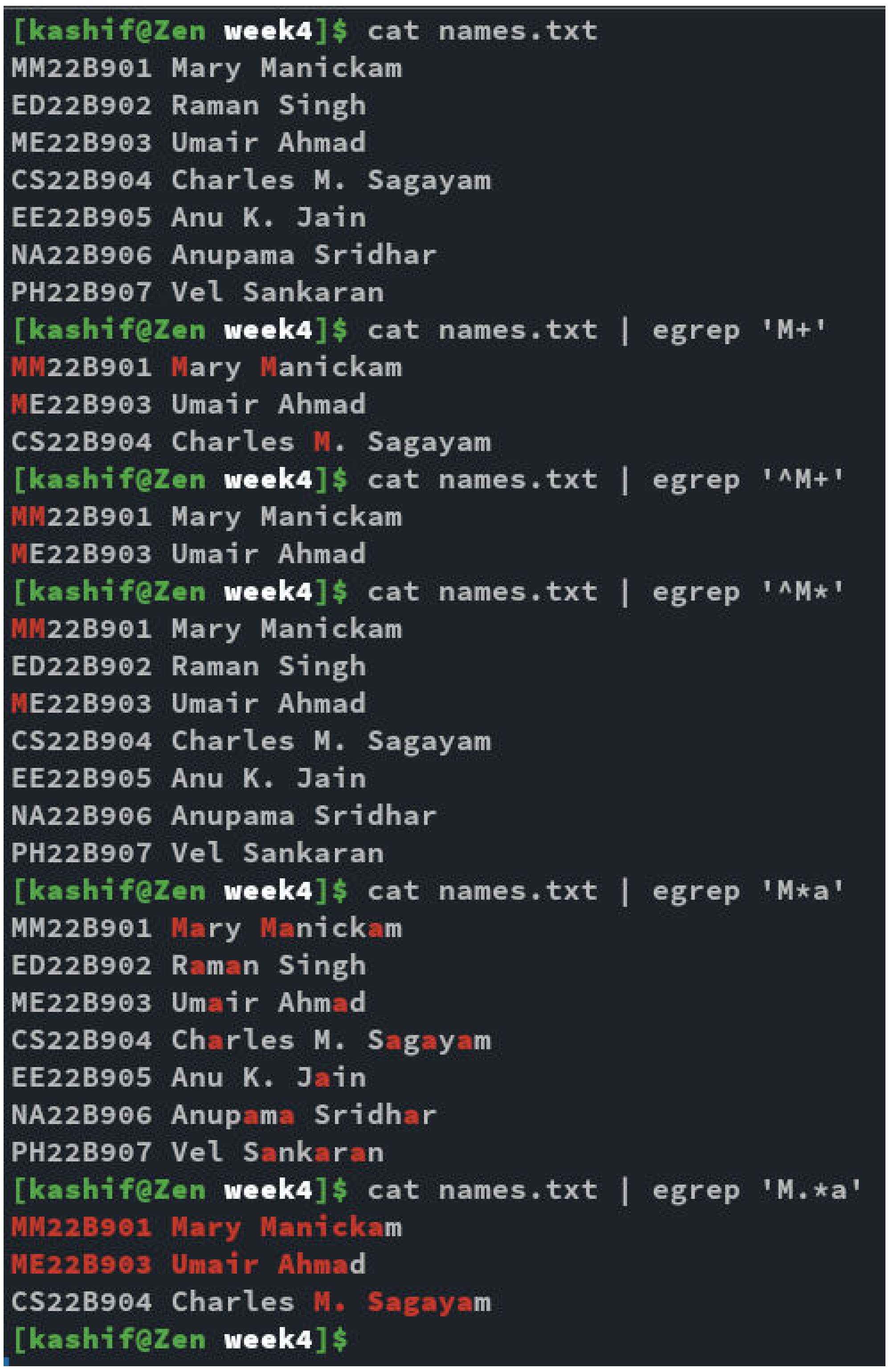
We can pass one number or a multiple numbers separated by comma for their matching



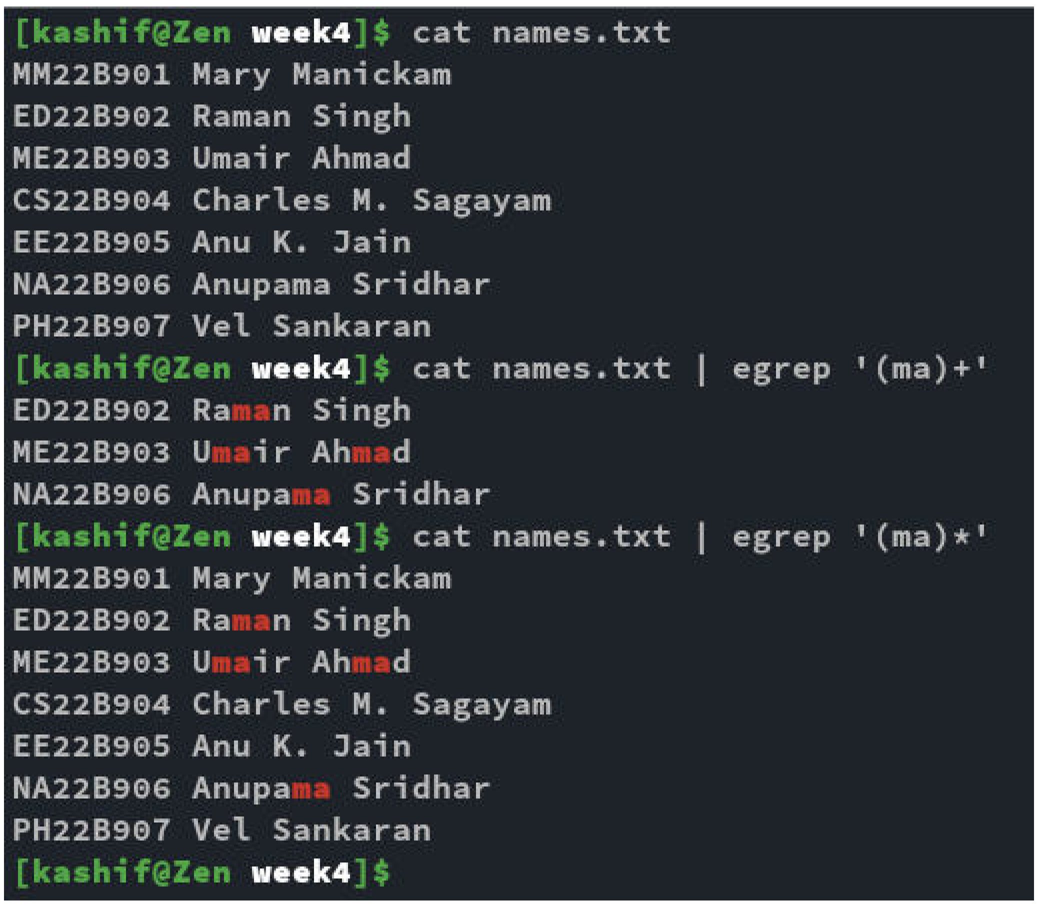
```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep 'M\{2\}'
MM22B901 Mary Manickam
[kashif@Zen week4]$ cat names.txt | grep 'M\{1,2\}'
MM22B901 Mary Manickam
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
[kashif@Zen week4]$
```



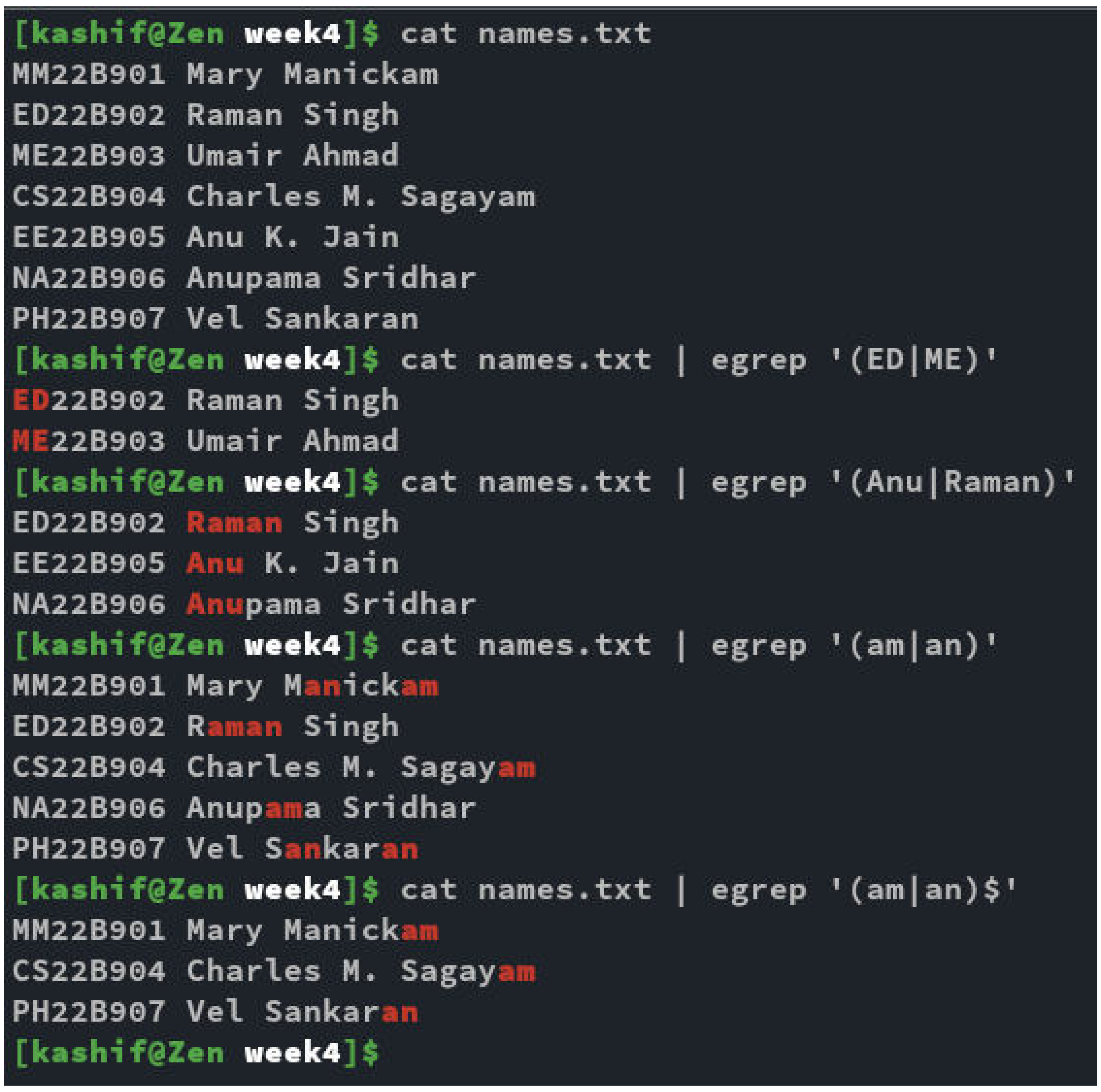
```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '\(ma\)'
ED22B902 Raman Singh
ME22B903 Umair Ahmad
NA22B906 Anupama Sridhar
[kashif@Zen week4]$ cat names.txt | grep '\(ma\).*\1'
ME22B903 Umair Ahmad
[kashif@Zen week4]$ cat names.txt | grep '\(.a\).*\1'
MM22B901 Mary Manickam
ME22B903 Umair Ahmad
[kashif@Zen week4]$ cat names.txt | grep '\(a.\).*\1'
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '\(a.\)\{3\}'
CS22B904 Charles M. Sagayam
[kashif@Zen week4]$ cat names.txt | grep '\(a.\)\{2\}'
ED22B902 Raman Singh
CS22B904 Charles M. Sagayam
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '\(a.\)\{2,3\}'
ED22B902 Raman Singh
CS22B904 Charles M. Sagayam
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$
```



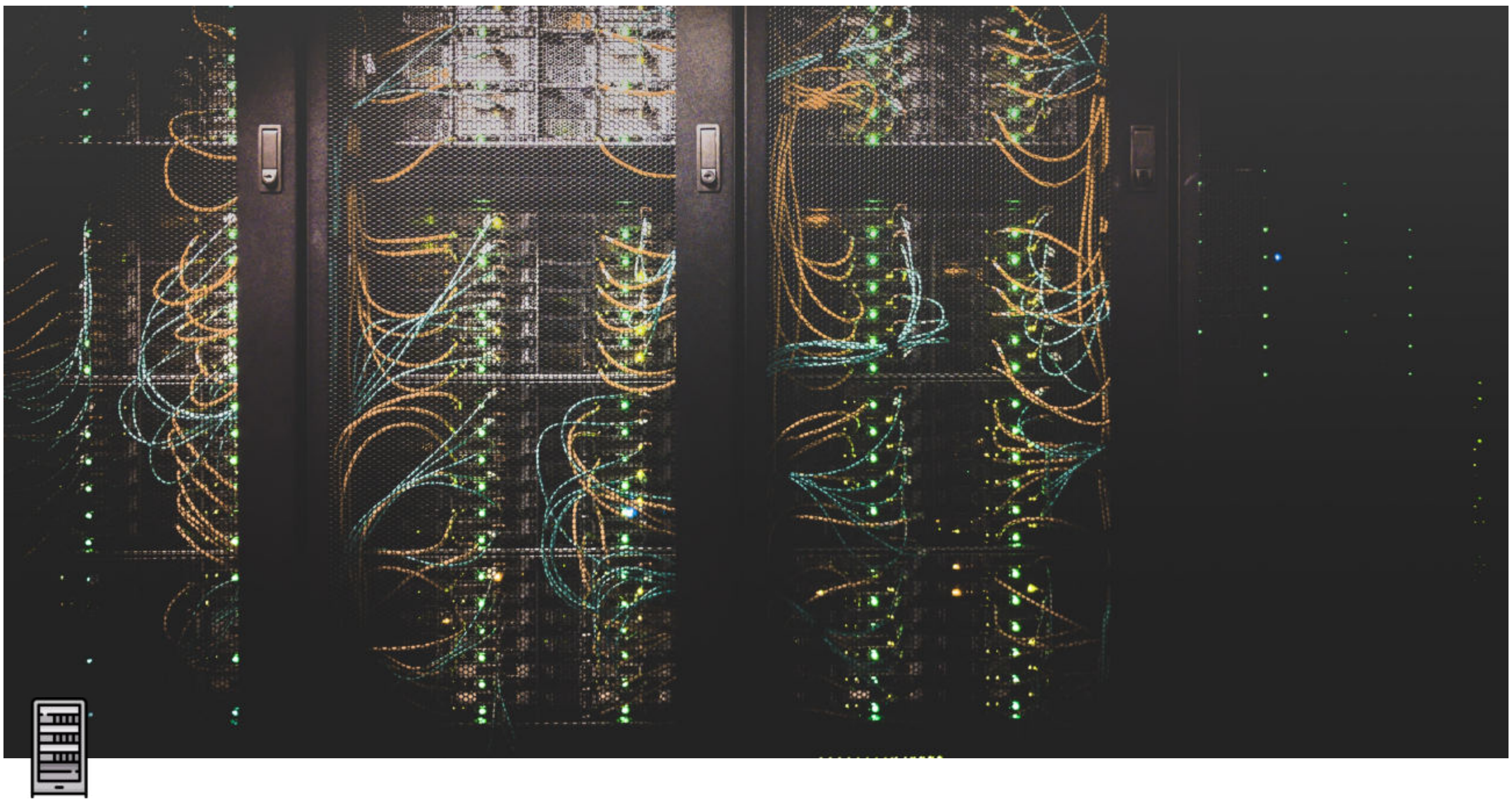
```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | egrep 'M+'
MM22B901 Mary Manickam
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
[kashif@Zen week4]$ cat names.txt | egrep '^M+'
MM22B901 Mary Manickam
ME22B903 Umair Ahmad
[kashif@Zen week4]$ cat names.txt | egrep '^M*'
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | egrep 'M*a'
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | egrep 'M.*a'
MM22B901 Mary Manickam
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
[kashif@Zen week4]$
```



```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | egrep '(ma)+'
ED22B902 Raman Singh
ME22B903 Umair Ahmad
NA22B906 Anupama Sridhar
[kashif@Zen week4]$ cat names.txt | egrep '(ma)*'
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$
```



```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | egrep '(ED|ME)'
ED22B902 Raman Singh
ME22B903 Umair Ahmad
[kashif@Zen week4]$ cat names.txt | egrep '(Anu|Raman)'
ED22B902 Raman Singh
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
[kashif@Zen week4]$ cat names.txt | egrep '(am|an)'
MM22B901 Mary Manickam
ED22B902 Raman Singh
CS22B904 Charles M. Sagayam
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | egrep '(am|an)$'
MM22B901 Mary Manickam
CS22B904 Charles M. Sagayam
PH22B907 Vel Sankaran
[kashif@Zen week4]$
```



Pattern Matching - pt. 2

Type	📒 Lecture
Date	@January 17, 2022
Lecture #	4
Lecture URL	https://youtu.be/XQUJPRc-7zA
Notion URL	https://21f1003586.notion.site/Pattern-Matching-pt-2-18da9c08be534558ad86e28d07cba0bd
Week #	4

Match package names that are 4 characters long

```
dpkg-query -W -f='${Section} ${binary:Package}\n' | egrep ' .{4}$'
```

Match package names that are 3 characters long and start with the letter `g`

```
dpkg-query -W -f='${Section} ${binary:Package}\n' | egrep ' g.{3}$'
```

Match package names that are between 1 to 5 characters long and start with the letter `g`

```
dpkg-query -W -f='${Section} ${binary:Package}\n' | egrep ' g.{1,5}$'
```

Match package names that are from the `math` category

```
dpkg-query -W -f='${Section} ${binary:Package}\n' | egrep '^math'
```

make sure to use the `^` *(hat) character in the front of the regex pattern to match the* `math` *category, otherwise it will match package category and the names*

Match package names that from KDE

```
dpkg-query -W -f='${Section} ${binary:Package}\n' | egrep ' kd.*$'
```

To skip empty lines from a file

```
cat filename.txt | egrep -v ‘^$’
```

- Pick any 12 digit or more number from a text file
    - `egrep ‘[[:digit:]]{12}’ filename.txt`
- Pick any 6 digit or more number from a text file
    - `egrep ‘[[:digit:]]{6}’ filename.txt`

But, there is one problem, if there is any number that is more than 12 digits or more than 6 digits respectively, it will pick that up too

- Pick an exactly 6 digit number from a text file
    - Add a word boundary `\b`

    - `egrep ‘\b[[:digit:]]{6}\b’ filename.txt`
- Pick a roll number (of the type MM22B001) from a text file
    - `egrep ‘\b[[:alpha:]]{2}[[:digit:]]{2}[[:alpha:]][[:digit:]]{3}\b’ filename.txt`
- Pick a URL from a text file (like github.com or https://www.iitm.ac.in)
    - `egrep ‘\b[[:alnum:]]+\.[[:alnum:]]+\b’ filename.txt`

`cut`

A command used to cut lines from files

does horizontal trimming

A sample file `fields.txt`

```
1234;hello world,line-1
234567;welcome cmdline,line-2
3456;parse text,line-3
```

- Cut first 4 characters from the beginning of the lines
    - `cut -c 1-4 fields.txt`
- Cut the next 4 characters from the previous
    - `cut -c 5-8 fields.txt`
- We can skip the beginning or the ending of the substring parameter, *it works like python*
    - Cut 4 chars from the beginning
        - `cut -c -4 fields.txt`
    - Cut from 8th char to the end
        - `cut -c 8- fields.txt`
- Use space as the delimiter and print the first field
    - `cat fields.txt | cut -d “ ” -f 1`
- Similarly, print the second field
    - `cat fields.txt | cut -d “ ” -f 2`
- If we want both fields
    - `cat fields.txt | cut -d “ ” -f 1-2`
- Delimit at a semi-colon `;` and get the first field
    - `cat fields.txt | cut -d “;” -f 1`
- Similarly, get the 2nd field
    - `cat fields.txt | cut -d “;” -f 2`
- We can pipe multiple commands
    - To get the part of the line between `;` and `,`
        - `cat fields.txt | cut -d “;” -f 2 | cut -d “,” -f 1`
    - To do the same thing using `grep` (*similar thing, not exactly the same*)
        - `cat fields.txt | egrep ‘;.*,’`
- To get the part `welcome cmdline` from the file `fields.txt`
    - `cat fields.txt | cut -d “;” -f 2 | cut -d “,” -f 1 | head -n 2 | tail -n 1`