CS 45, Lecture 3
Data Wrangling

Winter 2023
Akshay Srivatsan, Ayelet Drazen, Jonathan Kula

Administrivia

- Assignment 1 is live! It covers regular expressions and data wrangling. It is due on Monday, January 23rd.

- Office hours are listed on the course website. Come if you need help or want to chat!

What we know

In Lecture 1, we learned how to:

- What the shell is
- What the UNIX philosophy is
- How to run basic commands such as `ls`, `cd`, `cat`, `man`, `wc`
- How to pipe commands together using the `|` operator
- How to redirect output using `<` and `>`
- How to append to the end of a fine using `>>`

What we will learn today

In today's lecture, we will learn how to combine these commands in powerful ways:

- How to use shell commands to manipulate and analyze data
- How to write regular expressions
- How to run more complex shell commands such as `grep`, `sort`, `uniq`, `xargs`

What is Data Wrangling?

Data Wrangling Definition

The basic idea of data wrangling is that you take some raw data and convert or transform it into another form that is more useful.

Ideally, you do this in the most efficient way with the use of a tool 😁

More sources of data and larger amounts of data have made data wrangling increasingly important.
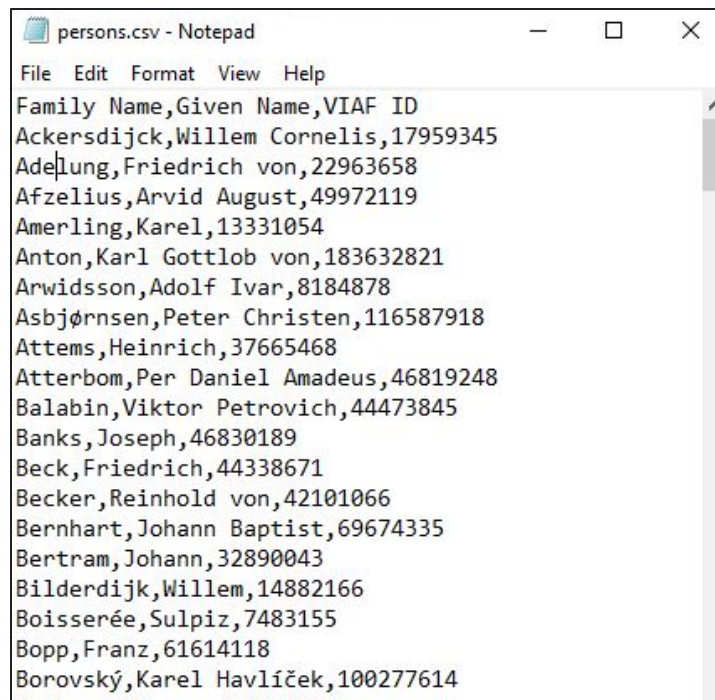
Source: XKCD

Data Formats

Data wrangling techniques are typically dictated by the data format you are using. Here are five common file formats for storing data:

- CSV
- XML
- HTML
- JSON
- TXT

Data Formats

A **CSV** file is a comma-separated values file where information is separated by commas.

- CSV are plain text files

- Data can be saved in tabular format (meaning a table of rows and columns)

- CSV files are often used to analyze data with spreadsheets

```
persons.csv - Notepad                    —    □    ✕
File  Edit  Format  View  Help
Family Name,Given Name,VIAF ID
Ackersdijck,Willem Cornelis,17959345
Adelung,Friedrich von,22963658
Afzelius,Arvid August,49972119
Amerling,Karel,13331054
Anton,Karl Gottlob von,183632821
Arwidsson,Adolf Ivar,8184878
Asbjørnsen,Peter Christen,116587918
Attems,Heinrich,37665468
Atterbom,Per Daniel Amadeus,46819248
Balabin,Viktor Petrovich,44473845
Banks,Joseph,46830189
Beck,Friedrich,44338671
Becker,Reinhold von,42101066
Bernhart,Johann Baptist,69674335
Bertram,Johann,32890043
Bilderdijk,Willem,14882166
Boisserée,Sulpiz,7483155
Bopp,Franz,61614118
Borovský,Karel Havlíček,100277614
```

Data Formats

A **XML** file is an Extensible Markup Language (XML) file that is used to store data in a hierarchical format.
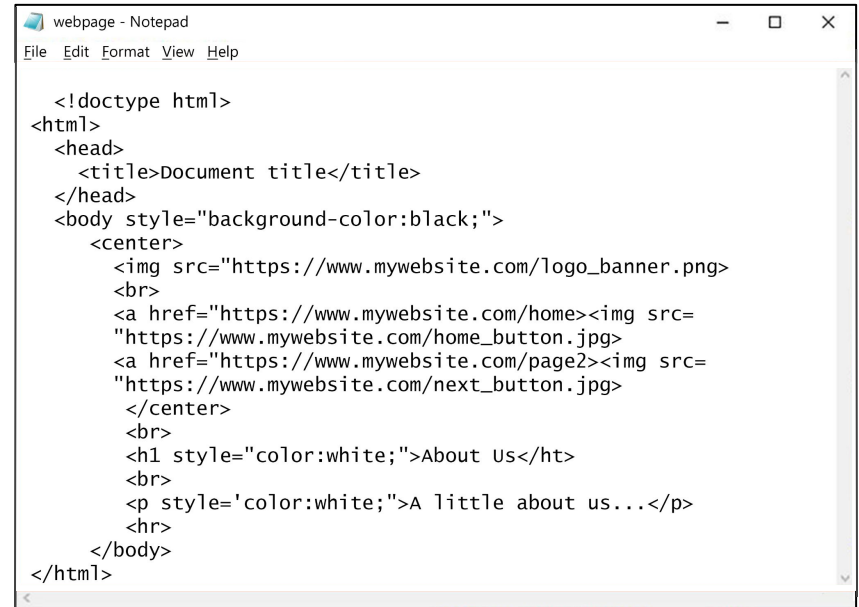
- XML files were created for storing documents in a way that both humans and machines could read.

- XML files consist of tags that define the hierarchy within the document.

Data Formats

An **HTML** file is an Hypertext Markup Language file that is used to store data in a hierarchical format, specifically webpages.

- HTML files are similar to XML files

- Key difference between the two is that HTML files must use a predefined set of tags to define hierarchical structure

```
webpage - Notepad

File   Edit   Format   View   Help

 <!doctype html>
<html>
   <head>
     <title>Document title</title>
   </head>
   <body style="background-color:black;">
      <center>
         <img src="https://www.mywebsite.com/logo_banner.png>
         <br>
         <a href="https://www.mywebsite.com/home><img src=
         "https://www.mywebsite.com/home_button.jpg>
         <a href="https://www.mywebsite.com/page2><img src=
         "https://www.mywebsite.com/next_button.jpg>
          </center>
          <br>
          <h1 style="color:white;">About Us</ht>
          <br>
          <p style='color:white;">A little about us...</p>
          <hr>
      </body>
</html>
```

Data Formats

A **JSON** file is a JavaScript Object Notation file that stores structured data in the form of JavaScript objects.

- JSON files are often used for transmitting data in web applications (e.g. sending some data from the server to the client)

Data Formats

A **TXT** file is a plaintext file that stores data in the form of lines.

- TXT files have no special formatting.
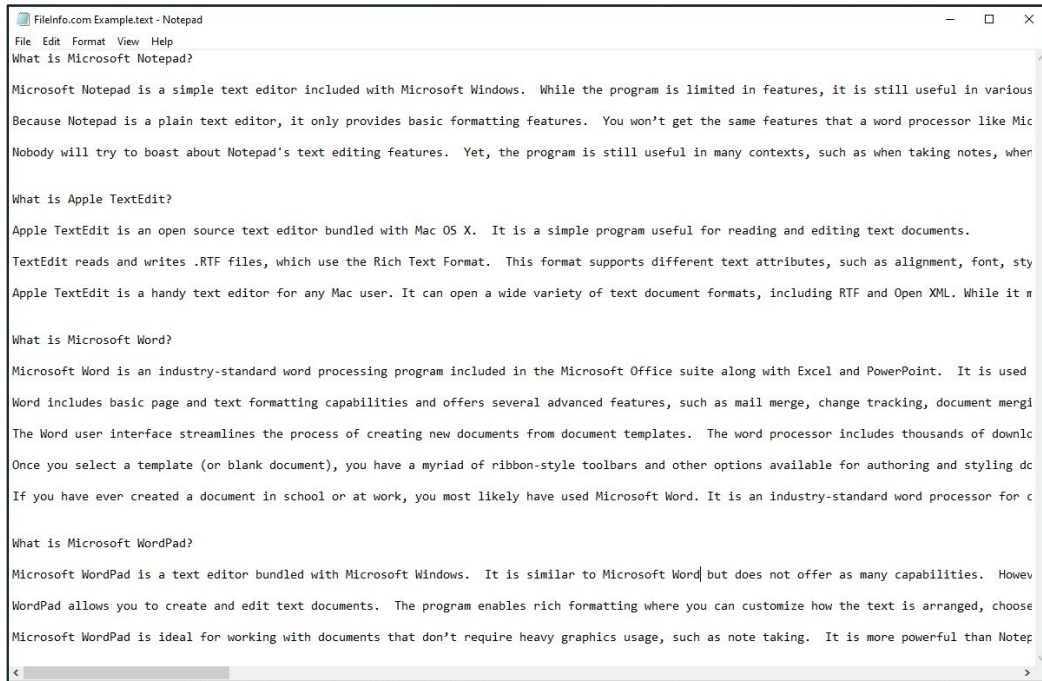
Basic Data Wrangling: grep

We've already seen a basic form of data wrangling with the | operator.

```
ls ~/Documents | grep -i transcript
```

Basic Data Wrangling: grep

Another example: system logs!

Basic Data Wrangling: grep

Another example: system logs!

System logs keep a record of operating system events on a machine, thereby producing a *lot* of data.
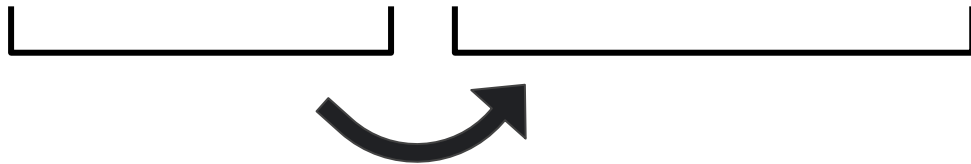
```
2023-01-04 11:06:43.580182-0500 0x128c185   Activity      0xecbfd4f           586    0   sharedfilelistd: (Security) SecTrustEvaluateIfNecessary
2023-01-04 11:06:44.873419-0500 0x128c27a   Activity      0xed943df           785    0   Code Helper (Renderer): (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:44.875961-0500 0x128c27a   Activity      0xed943cd           787    0   Code Helper (Renderer): (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:44.876058-0500 0x128c274   Activity      0xedbbe60           785    0   Code Helper (Renderer): (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:44.876397-0500 0x128c69b   Activity      0xed9448b           727    0   Code Helper (GPU): (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:44.877730-0500 0x128c27a   Activity      0xed943ce           787    0   Code Helper (Renderer): (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:44.983654-0500 0x128c27d   Activity      0xed9b9f6          1086    0   Siri: (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.066788-0500 0x1289a3f   Activity      0xed82d9b         82145    0   symptomsd-diag: (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.173469-0500 0x1289a3f   Activity      0xed82d9c         82145    0   symptomsd-diag: (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.176829-0500 0x1289a3f   Activity      0xed82d9d         82145    0   symptomsd-diag: (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.234107-0500 0x128c284   Activity      0xed9448c           727    0   Code Helper (GPU): (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.234144-0500 0x128c281   Activity      0xed7812a         82666    0   Slack Helper: (CoreFoundation) Loading Preferences From User CFPrefsD
2023-01-04 11:06:45.240382-0500 0x128c283   Activity      0xed9f104         82665    0   Slack Helper (GPU): (CoreFoundation) Loading Preferences From User CFPrefsD
2023-01-04 11:06:45.287416-0500 0x1289a3f   Activity      0xed82d9e         82145    0   symptomsd-diag: (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.292823-0500 0x128c69b   Activity      0xed9448d           727    0   Code Helper (GPU): (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.295226-0500 0x128c69b   Activity      0xed9448e           727    0   Code Helper (GPU): (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.323530-0500 0x1289a3f   Activity      0xed82d9f         82145    0   symptomsd-diag: (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.354035-0500 0x128c27d   Activity      0xed9b9f7          1086    0   Siri: (CoreFoundation) Loading Preferences From User CFPrefsD
2023-01-04 11:06:45.359223-0500 0x128c693   Activity      0xed7812b         82666    0   Slack Helper: (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.359355-0500 0x128c692   Activity      0xed9f105         82665    0   Slack Helper (GPU): (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.359575-0500 0x128c69b   Activity      0xed9448f           727    0   Code Helper (GPU): (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.365191-0500 0x128c69b   Activity      0xedbbf50           727    0   Code Helper (GPU): (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.367074-0500 0x128c693   Activity      0xed7812c         82666    0   Slack Helper: (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.369463-0500 0x128c69c   Activity      0xed943cf           787    0   Code Helper (Renderer): (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.369697-0500 0x128c692   Activity      0xed9f106         82665    0   Slack Helper (GPU): (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.370153-0500 0x128c69c   Activity      0xedbbf70           787    0   Code Helper (Renderer): (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.371021-0500 0x128c286   Activity      0xeda2b75           731    0   Code Helper: (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.371250-0500 0x128c27d   Activity      0xed9b9f8          1086    0   Siri: (CoreFoundation) Loading Preferences From System CFPrefsD
2023-01-04 11:06:45.371748-0500 0x128c286   Activity      0xeda2b76           731    0   Code Helper: (CoreFoundation) Loading Preferences From System CFPrefsD
```

Basic Data Wrangling: grep

Another example: system logs!

System logs keep a record of operating system events on a machine, thereby producing a *lot* of data.

```
log show | grep -i Chrome
```

(macOS)

```
journalctl | grep -i Chrome
```

(Linux)

```
wevutil gp Microsoft-Windows-Eventlog /ge:true | grep -i Chrome
```

(Windows)

Basic Data Wrangling: grep

We can even search for system log events on a remote server. Let's look for everything related to `ssh` on a CS45 honeypot:

```
ssh adrazen@192.9.152.85 journalctl | grep sshd
```

Basic Data Wrangling: grep

We can even search for system log events on a remote server. Let's look for everything related to `ssh` on a CS45 honeypot.

Let's look for times when we were disconnected.

```
ssh adrazen@192.9.152.85 journalctl | grep sshd | grep
"Disconnected from"
```

Basic Data Wrangling: grep

We should make sure to avoid sending unnecessary data across the machines. Let's run the pipeline on the remote machine by adding quotes:

```
ssh adrazen@192.9.152.85 'journalctl | grep sshd | grep "Disconnected from"'
```

Basic Data Wrangling: sed

Now maybe we are interested in extracting the usernames for the users who were disconnected.

Basic Data Wrangling: sed

Let's say we are interested in extracting the usernames for the users who were disconnected.

```
Jan 15 08:16:53 honeypot sshd[90474]: Disconnected from invalid user woshinidie 156.255.111.137 port 48758 [preauth]
Jan 15 08:16:59 honeypot sshd[90478]: Disconnected from invalid user zhangxiufang 218.255.245.10 port 61385 [preauth]
Jan 15 08:17:21 honeypot sshd[90483]: Disconnected from invalid user dandan 198.46.215.219 port 51776 [preauth]
Jan 15 08:17:26 honeypot sshd[90485]: Disconnected from invalid user liumin 128.199.111.126 port 36526 [preauth]
Jan 15 08:17:40 honeypot sshd[90487]: Disconnected from invalid user kevin 87.255.193.50 port 57162 [preauth]
Jan 15 08:17:50 honeypot sshd[90489]: Disconnected from invalid user shiny 198.46.215.219 port 49818 [preauth]
Jan 15 08:18:20 honeypot sshd[90491]: Disconnected from invalid user liumin 198.46.215.219 port 59334 [preauth]
Jan 15 08:18:33 honeypot sshd[90494]: Disconnected from invalid user hcarballo 156.255.111.137 port 48048 [preauth]
Jan 15 08:18:39 honeypot sshd[90496]: Disconnected from invalid user wangyi 128.199.111.126 port 51934 [preauth]
Jan 15 08:18:46 honeypot sshd[90498]: Disconnected from invalid user adnan 218.255.245.10 port 46769 [preauth]
Jan 15 08:18:54 honeypot sshd[90500]: Disconnected from invalid user woshinidie 198.46.215.219 port 45922 [preauth]
Jan 15 08:18:58 honeypot sshd[90502]: Disconnected from invalid user natalie 87.255.193.50 port 51706 [preauth]
Jan 15 08:19:23 honeypot sshd[90504]: Disconnected from invalid user carol 198.46.215.219 port 57490 [preauth]
Jan 15 08:19:46 honeypot sshd[90507]: Disconnected from invalid user dandan 128.199.111.126 port 56996 [preauth]
Jan 15 08:19:54 honeypot sshd[90509]: Disconnected from invalid user lqyi 198.46.215.219 port 54736 [preauth]
Jan 15 08:20:07 honeypot sshd[90511]: Disconnected from invalid user huangjun 156.255.111.137 port 47050 [preauth]
Jan 15 08:20:16 honeypot sshd[90513]: Disconnected from invalid user zjlang 87.255.193.50 port 46238 [preauth]
Jan 15 08:20:22 honeypot sshd[90515]: Disconnected from invalid user zhanghaomima 198.46.215.219 port 47014 [preauth]
Jan 15 08:20:35 honeypot sshd[90517]: Disconnected from invalid user wobuzhidao 218.255.245.10 port 60388 [preauth]
Jan 15 08:20:50 honeypot sshd[90519]: Disconnected from invalid user cals_oit_ssh 198.46.215.219 port 50912 [preauth]
```

Basic Data Wrangling: sed

Now maybe we are interested in extracting the usernames for the users who were disconnected.

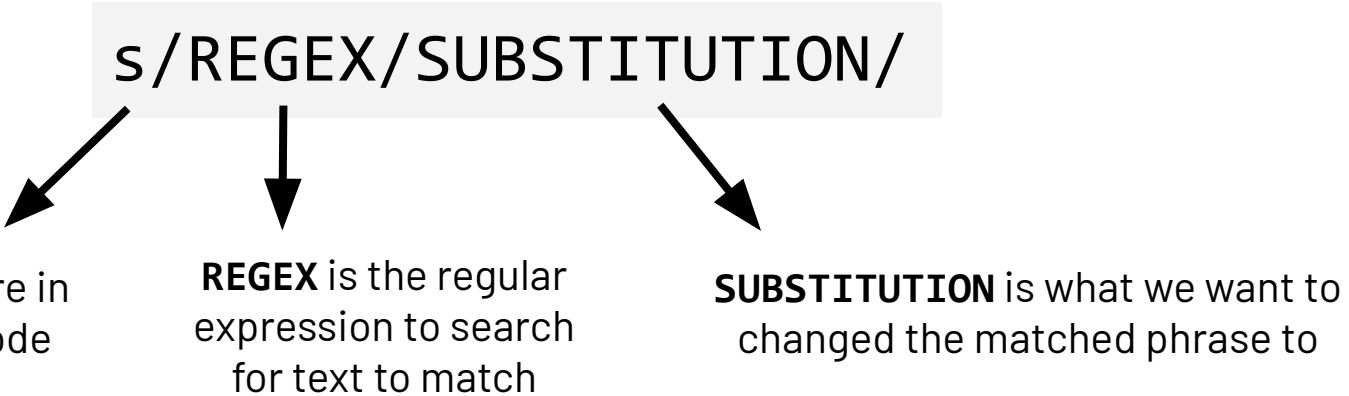We can use a tool called **sed** to help sift through our data.

sed is a stream editor that is built into Unix.

It can be used for searching a file, adding lines to a file, or substituting text in a file.

Basic Data Wrangling: sed

Let's use `sed` for substitution.

s/REGEX/SUBSTITUTION/

s indicates we are in substitution mode

REGEX is the regular expression to search for text to match

SUBSTITUTION is what we want to changed the matched phrase to

Basic Data Wrangling: sed

Original File:

```
Courses taken in AY21-22 will be counted towards your major.
You must take the course in AY21-22
```

Basic Data Wrangling: sed

Original File:

```
Courses taken in AY21-22 will be counted towards your major.
You must take the course in AY21-22
```

Command:

```
sed 's/AY21-22/AY22-23/' file.txt
```

Text to match   Replacement text

Basic Data Wrangling: sed

Original File:

```
Courses taken in AY21-22 will be counted towards your major.
You must take the course in AY21-22
```

Command:

```
sed 's/AY21-22/AY22-23/' file.txt
```

Text to match   Replacement text

After sed:

```
Courses taken in AY22-23 will be counted towards your major.
You must take the course in AY22-23
```

Basic Data Wrangling: sed

Let's say we interested in extracting the usernames for the users who were disconnected.

```
Jan 15 08:16:53 honeypot sshd[90474]: Disconnected from invalid user woshinidie 156.255.111.137 port 48758 [preauth]
Jan 15 08:16:59 honeypot sshd[90478]: Disconnected from invalid user zhangxiufang 218.255.245.10 port 61385 [preauth]
Jan 15 08:17:21 honeypot sshd[90483]: Disconnected from invalid user dandan 198.46.215.219 port 51776 [preauth]
Jan 15 08:17:26 honeypot sshd[90485]: Disconnected from invalid user liumin 128.199.111.126 port 36526 [preauth]
Jan 15 08:17:40 honeypot sshd[90487]: Disconnected from invalid user kevin 87.255.193.50 port 57162 [preauth]
Jan 15 08:17:50 honeypot sshd[90489]: Disconnected from invalid user shiny 198.46.215.219 port 49818 [preauth]
Jan 15 08:18:20 honeypot sshd[90491]: Disconnected from invalid user liumin 198.46.215.219 port 59334 [preauth]
Jan 15 08:18:33 honeypot sshd[90494]: Disconnected from invalid user hcarballo 156.255.111.137 port 48048 [preauth]
Jan 15 08:18:39 honeypot sshd[90496]: Disconnected from invalid user wangyi 128.199.111.126 port 51934 [preauth]
Jan 15 08:18:46 honeypot sshd[90498]: Disconnected from invalid user adnan 218.255.245.10 port 46769 [preauth]
Jan 15 08:18:54 honeypot sshd[90500]: Disconnected from invalid user woshinidie 198.46.215.219 port 45922 [preauth]
Jan 15 08:18:58 honeypot sshd[90502]: Disconnected from invalid user natalie 87.255.193.50 port 51706 [preauth]
Jan 15 08:19:23 honeypot sshd[90504]: Disconnected from invalid user carol 198.46.215.219 port 57490 [preauth]
Jan 15 08:19:46 honeypot sshd[90507]: Disconnected from invalid user dandan 128.199.111.126 port 56996 [preauth]
Jan 15 08:19:54 honeypot sshd[90509]: Disconnected from invalid user lqyi 198.46.215.219 port 54736 [preauth]
Jan 15 08:20:07 honeypot sshd[90511]: Disconnected from invalid user huangjun 156.255.111.137 port 47050 [preauth]
Jan 15 08:20:16 honeypot sshd[90513]: Disconnected from invalid user zjlang 87.255.193.50 port 46238 [preauth]
Jan 15 08:20:22 honeypot sshd[90515]: Disconnected from invalid user zhanghaomima 198.46.215.219 port 47014 [preauth]
Jan 15 08:20:35 honeypot sshd[90517]: Disconnected from invalid user wobuzhidao 218.255.245.10 port 60388 [preauth]
Jan 15 08:20:50 honeypot sshd[90519]: Disconnected from invalid user cals_oit_ssh 198.46.215.219 port 50912 [preauth]
```

Regular Expressions

What is a regular expression?

> A *regular expression* (also called a regex) is a set of characters that specifies a search pattern.

Most ASCII characters carry their normal meaning but some characters have special matching behavior.

There is some variation between different implementations of regular expressions.

Regular Expressions

First, there are **groups** of characters. These specify *which* characters we are interested in:

Regular Expressions

First, there are **groups** of characters. These specify *which* characters we are interested in:

- means any single character (except the newline character)

Regular Expressions

First, there are **groups** of characters. These specify *which* characters we are interested in:

- `.` means any single character (except the newline character)

- `[abc]` means any of the characters included inside the square brackets (in this case `a`, `b` or `c`)

Regular Expressions

First, there are **groups** of characters. These specify *which* characters we are interested in:

`.` means any single character (except the newline character)

`[abc]` means any of the characters included inside the square brackets (in this case a, b or c)

`[a-z]` means any character in the range a-z

Regular Expressions

First, there are **groups** of characters. These specify *which* characters we are interested in:

`.` means any single character (except the newline character)

`[abc]` means any of the characters included inside the square brackets (in this case a, b or c)

`[a-z]` means any character in the range a-z

`(a|b)` means either a or b

Regular Expressions

Next, we have **quantifiers**. These specify *how many* characters we are interested in:

Regular Expressions

Next, we have **quantifiers**. These specify *how many* characters we are interested in:

* means we want 0 or more characters of the specified kind

Regular Expressions

Next, we have **quantifiers**. These specify *how many* characters we are interested in:

*   means we want 0 or more characters of the specified kind

+   means we want 1 or more characters of the specified kind

Regular Expressions

Next, we have **quantifiers**. These specify *how many* characters we are interested in:

* * means we want 0 or more characters of the specified kind

* \+ means we want 1 or more characters of the specified kind

* ? means we want exactly 0 or 1 characters of the specified kind

Regular Expressions

Next, we have **quantifiers**. These specify *how many* characters we are interested in:

* means we want 0 or more characters of the specified kind

+ means we want 1 or more characters of the specified kind

? means we want exactly 0 or 1 characters of the specified kind

{X} means we want exactly X characters of the specified kind

Regular Expressions

Finally, we have **anchors**. These specify *specific starting or stopping* conditions we are interested in:

Regular Expressions

Finally, we have **anchors**. These specify *specific starting or stopping* conditions we are interested in:

`^` specifies the start of the line

Regular Expressions

Finally, we have **anchors**. These specify *specific starting or stopping* conditions we are interested in:

^ specifies the start of the line

$ specifies the end of the line

Source: @garabatokid

RegEx Applications

Regular expressions are really useful in all kinds of applications!

You can use regular expressions inside of applications such as Excel and Google Sheets that support data processing.

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

Which characters?

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

Which characters? Any character A-Z, a-z, 0-9, ., _, %, +, -

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

Which characters?   Any character A-Z, a-z,0-9, ., _, %, +, -

`[A-Za-z0-9._%+-]`

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

Which characters?   Any character A-Z, a-z,0-9, ., _, %, +, -

`[A-Za-z0-9._%+-]`

How many of them?

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

Which characters? Any character A-Z, a-z,0-9, ., _, %, +, -

`[A-Za-z0-9._%+-]`

How many of them? As many as you want... (at least 1!)

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

Which characters? Any character A-Z, a-z,0-9, ., _, %, +, -

`[A-Za-z0-9._%+-]`

How many of them? As many as you want... (at least 1!)

`[A-Za-z0-9._%+-]+`

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`

Which characters?

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen**@stanford**.edu

`[A-Za-z0-9._%+-]+`

Which characters?   Any character A-Z, a-z,0-9, **.**, -

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`

Which characters?   Any character A-Z, a-z,0-9, **.**, -

`[A-Za-z0-9.-]`

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`

Which characters?　　Any character A-Z, a-z,0-9, **.**, -

`[A-Za-z0-9.-]`

How many of them?

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`

Which characters?　　Any character A-Z, a-z,0-9, **.**, -

`[A-Za-z0-9.-]`

How many of them?　　As many as you want... (at least 1!)

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen**@stanford**.edu

`[A-Za-z0-9._%+-]+`

Which characters? Any character A-Z, a-z, 0-9, **.**, -

`[A-Za-z0-9.-]`

How many of them? As many as you want... (at least 1!)

`[A-Za-z0-9.-]+`

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`   `[A-Za-z0-9.-]+`

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`   `[A-Za-z0-9.-]+`

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`   `[A-Za-z0-9.-]+`

Which characters?

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`   `[A-Za-z0-9.-]+`

Which characters?   Any character A-Z, a-z

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`   `[A-Za-z0-9.-]+`

Which characters?   Any character A-Z, a-z

`[A-Za-z]`

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`   `[A-Za-z0-9.-]+`

Which characters?   Any character A-Z, a-z

`[A-Za-z]`

How many of them?

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`   `[A-Za-z0-9.-]+`

Which characters?   Any character A-Z, a-z

`[A-Za-z]`

How many of them?   At least 2

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`   `[A-Za-z0-9.-]+`

Which characters?   Any character A-Z, a-z

`[A-Za-z]`

How many of them?   At least 2

`[A-Za-z]{2,}`

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

[A-Za-z0-9._%+-]+   [A-Za-z0-9.-]+   [A-Za-z]{2,}

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

[A-Za-z0-9._%+-]+   [A-Za-z0-9.-]+   [A-Za-z]{2,}

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

[A-Za-z0-9._%+-]+   @   [A-Za-z0-9.-]+   .   [A-Za-z]{2,}

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`   `@`   `[A-Za-z0-9.-]+`   `\.`   `[A-Za-z]{2,}`

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

`[A-Za-z0-9._%+-]+`   `@`   `[A-Za-z0-9.-]+`   `\.`   `[A-Za-z]{2,}`

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

adrazen@stanford.edu

```
[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}
```

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

```
[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}
```

Regular Expressions

Time for examples!

1. Write a regular expression to match all email addresses

```
[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}
```

Technically, this RegEx only matches some 99% of email addresses. Here is the fully RFC compliant RegEx for all emails... 😱

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user mongodb 13.87.204.143 port 50660 [preauth]
```

```
Jan 13 15:25:02 honeypot sshd[68939]: Disconnected from authenticating user root 205.185.126.149 port 44302 [preauth]
```

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user mongodb 13.87.204.143 port 50660 [preauth]
```

```
Jan 13 15:25:02 honeypot sshd[68939]: Disconnected from authenticating user root 205.185.126.149 port 44302 [preauth]
```

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user mongodb 13.87.204.143 port 50660 [preauth]
```

Regular Expressions

2. Write a regular expression to parse username from log line

`Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user mongodb 13.87.204.143 port 50660 [preauth]`

| `Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user` | `mongodb` | `13.87.204.143 port 50660 [preauth]` |

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user mongodb 13.87.204.143 port 50660 [preauth]
```

| Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user | mongodb | 13.87.204.143 port 50660 [preauth] |

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user
```

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user
```

```
Jan 13 15:25:02 honeypot sshd[68939]: Disconnected from authenticating user
```
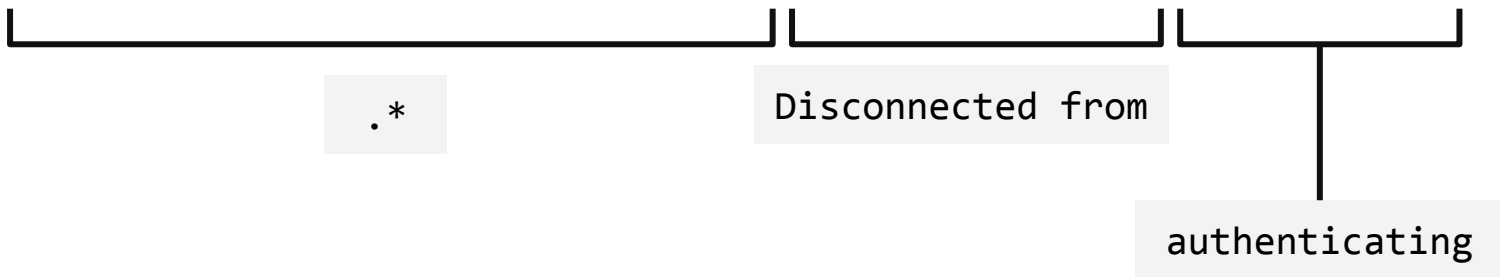
Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user
```

```
Jan 13 15:25:02 honeypot sshd[68939]: Disconnected from authenticating user
```
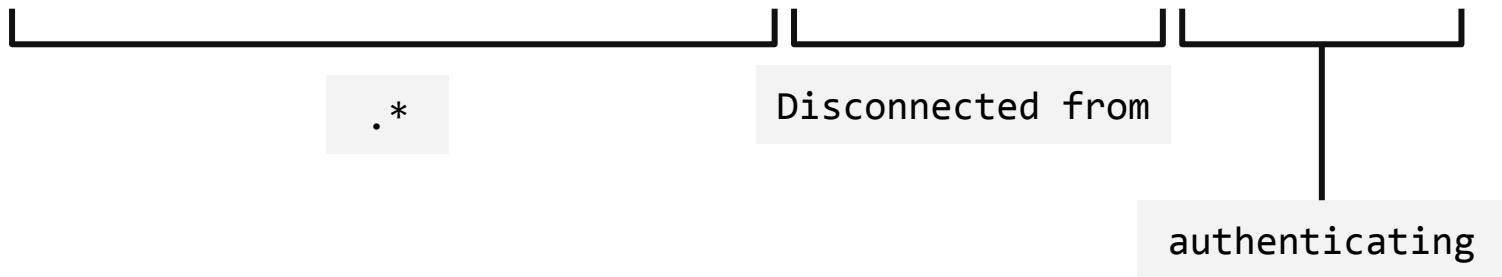
```
.*
```

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user
```

```
Jan 13 15:25:02 honeypot sshd[68939]: Disconnected from authenticating user
```

```
.*
```

```
Disconnected from
```

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user
```

```
Jan 13 15:25:02 honeypot sshd[68939]: Disconnected from authenticating user
```

`.*`

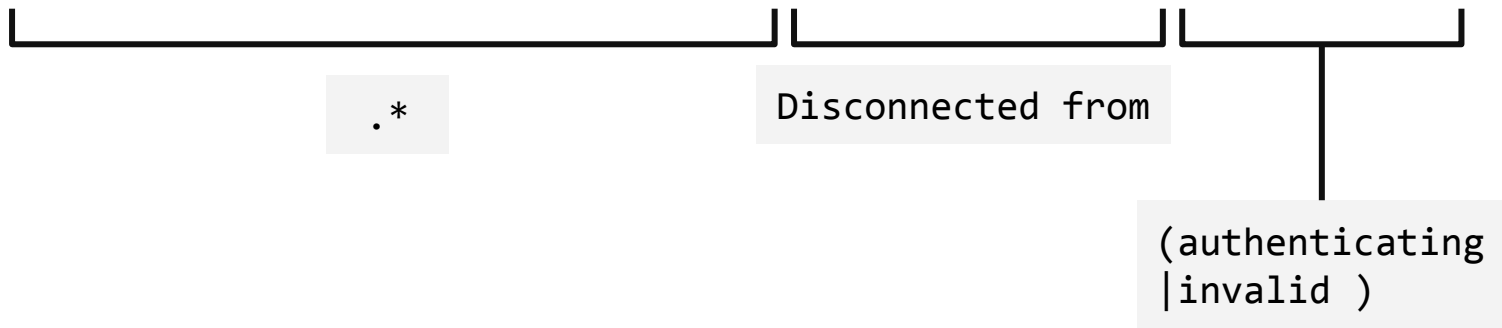`Disconnected from`

`authenticating`

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user
```

```
Jan 13 15:25:02 honeypot sshd[68939]: Disconnected from authenticating user
```

`.*`

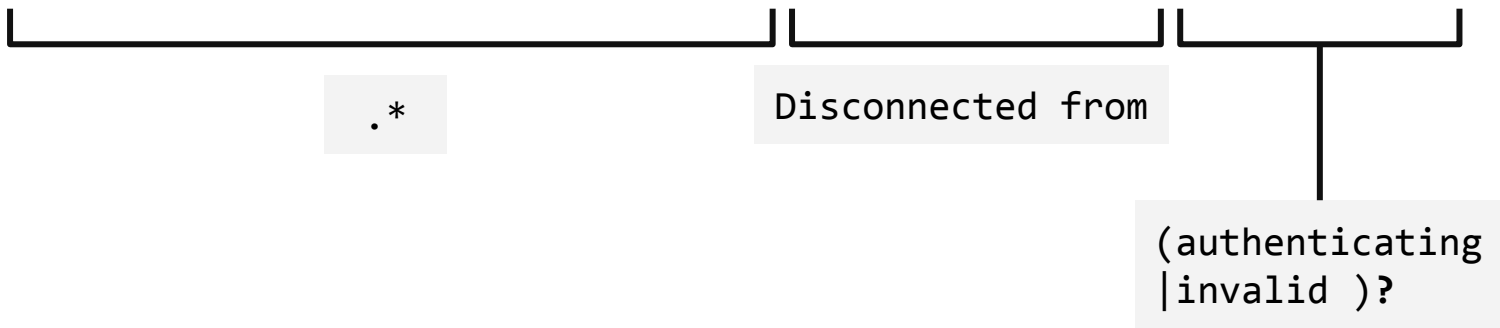`Disconnected from`

`authenticating`

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user
```

```
Jan 13 15:25:02 honeypot sshd[68939]: Disconnected from authenticating user
```

`.*`

`Disconnected from`
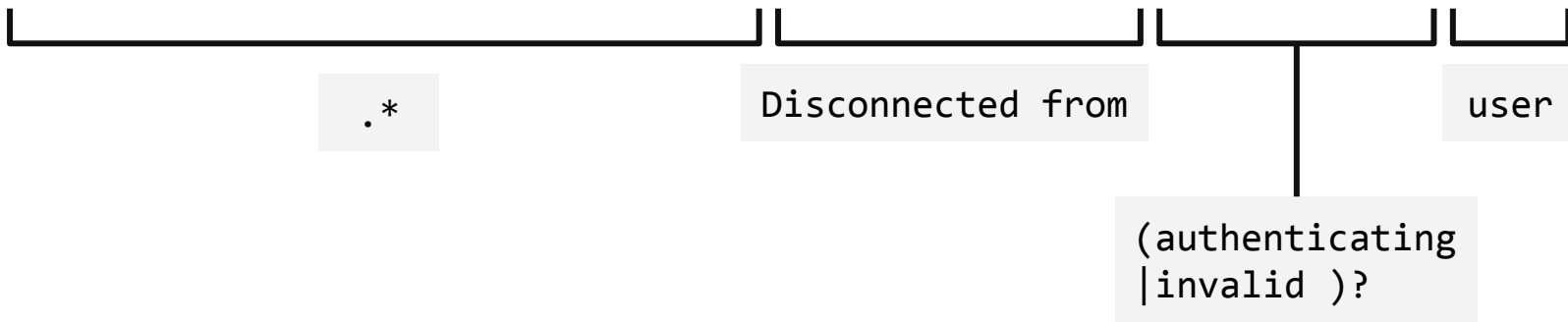
`(authenticating|invalid )`

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user
```

```
Jan 13 15:25:02 honeypot sshd[68939]: Disconnected from authenticating user
```

`.*`

`Disconnected from`

`(authenticating |invalid )?`

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user
```

```
Jan 13 15:25:02 honeypot sshd[68939]: Disconnected from authenticating user
```

`.*` `Disconnected from` `user`

`(authenticating|invalid )?`

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user
```

```
Jan 13 15:25:02 honeypot sshd[68939]: Disconnected from authenticating user
```

```
.* Disconnected from (authenticating |invalid )?user
```

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user mongodb 13.87.204.143 port 50660 [preauth]
```

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user   mongodb   13.87.204.143 port 50660 [preauth]
```

Regular Expressions

2. Write a regular expression to parse username from log line

mongodb

root

Regular Expressions

2. Write a regular expression to parse username from log line

```
mongodb
```

```
root
```

⎣_____⎦

```
.*
```

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user mongodb 13.87.204.143 port 50660 [preauth]
```

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user          mongodb          13.87.204.143 port 50660 [preauth]
```

Regular Expressions

2. Write a regular expression to parse username from log line

```
13.87.204.143 port 50660 [preauth]
```

```
114.5.119.116 port 55342 [preauth]
```

Regular Expressions

2. Write a regular expression to parse username from log line

```
13.87.204.143 port 50660 [preauth]
```

```
114.5.119.116 port 55342 [preauth]
```

```
[0-9.]+
```

Regular Expressions

2. Write a regular expression to parse username from log line

```
13.87.204.143 port 50660 [preauth]
```

```
114.5.119.116 port 55342 [preauth]
```

```
[0-9.]+
```

Regular Expressions

2. Write a regular expression to parse username from log line

```
13.87.204.143 port 50660 [preauth]
```

```
114.5.119.116 port 55342 [preauth]
```

`[0-9.]+`   `port`

Regular Expressions

2. Write a regular expression to parse username from log line

```
13.87.204.143 port 50660 [preauth]
```

```
114.5.119.116 port 55342 [preauth]
```

```
[0-9.]+       port
```

Regular Expressions

2. Write a regular expression to parse username from log line

13.87.204.143 port 50660 [preauth]

114.5.119.116 port 55342 [preauth]

[0-9.]+   port   [0-9]+

Regular Expressions

2. Write a regular expression to parse username from log line

```
13.87.204.143 port 50660 [preauth]
```

```
114.5.119.116 port 55342 [preauth]
```

[0-9.]+   port   [0-9]+

Regular Expressions

2. Write a regular expression to parse username from log line

```
13.87.204.143 port 50660 [preauth]
```

```
114.5.119.116 port 55342 [preauth]
```

| [0-9.]+ | port | [0-9]+ | (\[preauth\])? |

Regular Expressions

2. Write a regular expression to parse username from log line

```
13.87.204.143 port 50660 [preauth]
```

```
114.5.119.116 port 55342 [preauth]
```

```
[0-9.]+ port [0-9]+( \[preauth\])?
```

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user mongodb 13.87.204.143 port 50660 [preauth]
```

| Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user | mongodb | 13.87.204.143 port 50660 [preauth] |

| .* Disconnected from (authenticating \|invalid)?user | .* | [0-9.]+ port [0-9]+(\[preauth\])? |

Regular Expressions

2. Write a regular expression to parse username from log line

```
Jan 13 15:24:43 honeypot sshd[68935]: Disconnected from invalid user mongodb 13.87.204.143 port 50660 [preauth]
```

```
Jan 13 15:25:02 honeypot sshd[68939]: Disconnected from authenticating user root 205.185.126.149 port 44302 [preauth]
```

```
.* Disconnected from (authenticating |invalid )?user .* [0-9.]+ port [0-9]+( \[preauth\])?
```

Regular Expressions

Let's use `sed` for substitution.

s/REGEX/SUBSTITUTION/

s indicates we are in substitution mode

REGEX is the regular expression to search for text to match

SUBSTITUTION is what we want to changed the matched phrase to

Regular Expressions

Let's use `sed` for substitution.

s/REGEX/SUBSTITUTION/

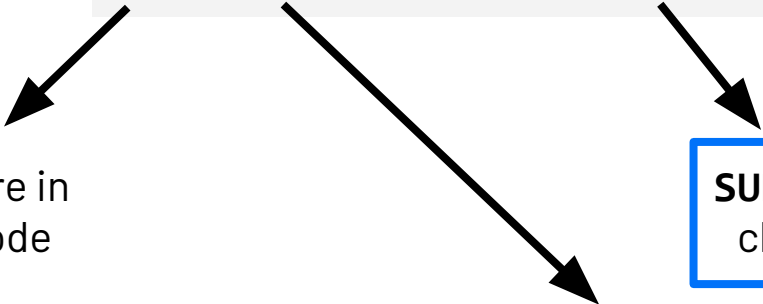**s** indicates we are in substitution mode

SUBSTITUTION is what we want to changed the matched phrase to

```
.* Disconnected from (authenticating |invalid )?user .* [0-9.]+ port [0-9]+( \[preauth\])?
```

Regular Expressions

Let's use `sed` for substitution.

s/REGEX/SUBSTITUTION/

s indicates we are in substitution mode

SUBSTITUTION is what we want to changed the matched phrase to

```
.* Disconnected from (authenticating |invalid )?user .* [0-9.]+ port [0-9]+( \[preauth\])?
```

Regular Expressions

Let's use `sed` for substitution.

s/REGEX/SUBSTITUTION/

s indicates we are in substitution mode

SUBSTITUTION is what we want to changed the matched phrase to

.* Disconnected from (authenticating |invalid)?user .* [0-9.]+ port [0-9]+(\[preauth\])?

Regular Expressions

Let's use `sed` for substitution.

s/REGEX/SUBSTITUTION/

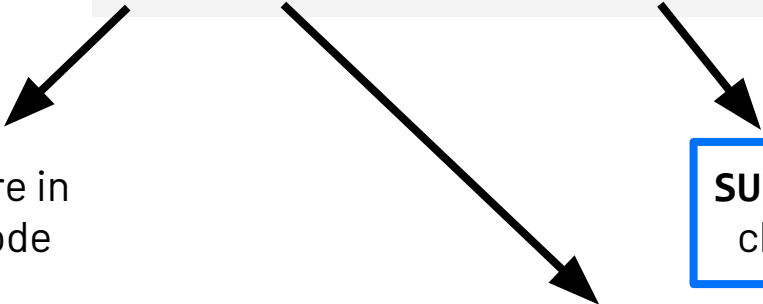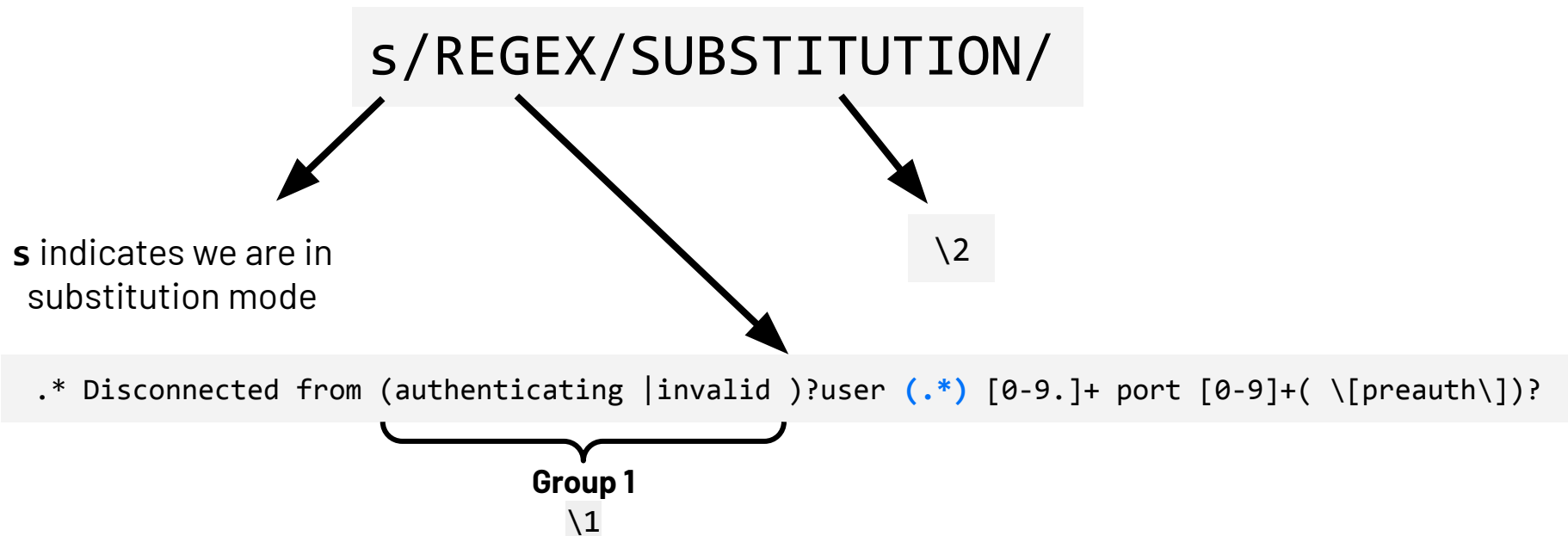**s** indicates we are in substitution mode

SUBSTITUTION is what we want to changed the matched phrase to

```
.* Disconnected from (authenticating |invalid )?user (.*) [0-9.]+ port [0-9]+( \[preauth\])?
```

Regular Expressions

Let's use `sed` for substitution.

$$s/REGEX/SUBSTITUTION/$$

s indicates we are in substitution mode

\2

`.* Disconnected from (authenticating |invalid )?user (.*) [0-9.]+ port [0-9]+( \[preauth\])?`

Group 1
\1

Regular Expressions

Let's use `sed` for substitution.

s/REGEX/SUBSTITUTION/

s indicates we are in substitution mode

\2

.* Disconnected from (authenticating |invalid)?user (.*) [0-9.]+ port [0-9]+(\[preauth\])?

Group 1
\1

Group 2
\2

Regular Expressions

Let's use `sed` for substitution.

s/REGEX/SUBSTITUTION/

\2

s indicates we are in substitution mode

.* Disconnected from (authenticating |invalid)?user (.*) [0-9.]+ port [0-9]+(\[preauth\])?

Group 1
\1

Group 2
\2

Group 3
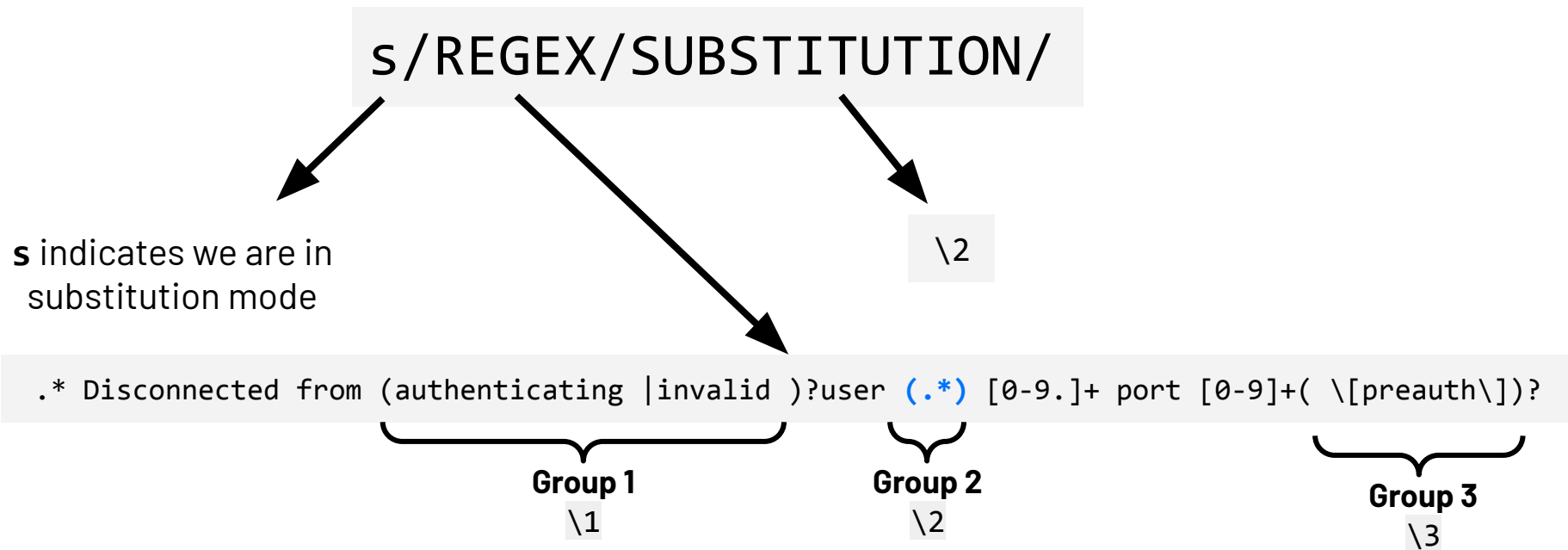\3

Regular Expressions

Let's use `sed` for substitution.

```
s/.* Disconnected from (authenticating |invalid )?user (.*) [0-9.]+ port [0-9]+( \[preauth\])?/\2/
```

Basic Data Wrangling: sed

Common Pitfalls and Usage Notes:

Basic Data Wrangling: sed

Common Pitfalls and Usage Notes:

- sed assumes data instances makes one substitution *per line*. If you want sed to keep repeating the substitution process for all instances on that line, use /g:

```
sed 's/AY21-22/AY22-23/g' file.txt
```

Basic Data Wrangling: sed

Common Pitfalls and Usage Notes:

- `sed` assumes data instances makes one substitution *per line*. If you want `sed` to keep repeating the substitution process for all instances on that line, use /g:

  ```
  sed 's/AY21-22/AY22-23/g' file.txt
  ```

- If you want to use a regex with sed, make sure to include the -E flag:

  ```
  sed -E 's/AY[0-9]{2}-[0-9]{2}/AY22-23' file.txt
  ```

Useful Commands

Now that we have all of the usernames, we can run some analysis on the data.

`sort` is a command that will arrange (i.e. sort) the data alphabetically or numerically.

The `-n` flag indicates to `sort` the data numerically.

Useful Commands

`uniq` is a command that reports or filters out the repeated lines in a file.

The `-c` flag is especially useful as it reports unique lines in the file and counts the number of occurrences for each line.

Useful Commands

`uniq` is a command that reports or filters out the repeated lines in a file.

The `-c` flag is especially useful as it reports unique lines in the file and counts the number of occurrences for each line.

`tail` is a command that prints the last lines X lines of a files

The `-nX` flag allows you to specify the number of lines you are interested in printing.

Useful Commands

`xargs` is a command that allows you to use the output of one command as the *arguments* to another command.

Let's look at an example!

Useful Commands

`xargs` is a command that allows you to use the output of one command as the *arguments* to another command.

```
adrazen@ayelet-computer ~ % cat filenames.txt
homework.txt
program.py
todo-list.txt
random.txt
```

Useful Commands

`xargs` is a command that allows you to use the output of one command as the *arguments* to another command.

```
touch homework.txt
```

```
touch program.py
```

```
touch todo-list.txt
```

```
touch random.txt
```

Useful Commands

`xargs` is a command that allows you to use the output of one command as the *arguments* to another command.

```
cat filenames.txt | xargs touch
```

Useful Commands

`xargs` is a command that allows you to use the output of one command as the *arguments* to another command.

```
cat filenames.txt | xargs touch
```

```
homework.txt
program.py
todo-list.txt
random.txt
```

Useful Commands

`xargs` is a command that allows you to use the output of one command as the *arguments* to another command.

```
cat filenames.txt | xargs touch homework.txt program.py
todo-list.txt random.txt
```

Useful Commands

`xargs` is a command that allows you to use the output of one command as the *arguments* to another command.

```
cat filenames.txt | x̶a̶r̶g̶s̶ touch homework.txt program.py
todo-list.txt random.txt
```

Useful Commands

`xargs` is a command that allows you to use the output of one command as the *arguments* to another command.

```
cat filenames.txt | xargs touch homework.txt program.py
todo-list.txt random.txt
```

Other Commands and Tools

There are *many* useful commands, tools and languages out there for data wrangling. Here are a few to check out if you are interested:

`awk` is a scripting language for manipulating data and generating reports

`R` is another programming language that is great at data analysis and plotting.

`perl` is a programming language for text manipulation