Digital Evidence Search with A Pattern Matching Game

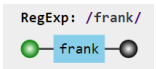
The Power of Regular Expression

What types of evidence to match?

- Personal information
  - name, phone number, email address, date of birth, zip code, SSN
- User account validation
  - username, password
- Source code
  - HTML, Java
- Network
  - website visited (URL), IP address, Hex, MAC address, timestamp
- File
  - file names, file attributes

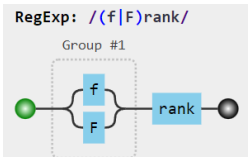
What is regular expression (regex)?

- A special text string for describing a search pattern
  - string is written in an expression language
- Extremely useful in extracting information from
  - text file: source code, log files,
  - documents: spreadsheets, PowerPoint, Word (need to unzip them)
  - binary strings in files
- Often use with *grep* command




```
┌──(student㉿kalit150)-[~/regex]
└─$ echo "This is Frank Xu" | grep -Ei "frank"
This is Frank Xu

┌──(student㉿kalit150)-[~/regex]
└─$ echo "This is Frank Xu" | grep -E "(f|F)rank"
This is Frank Xu
```

f, r,a,n,k Literal Characters
| Logic: or
(|) or relation in a group



https://jex.im/regulex/#!flags=&re=(f%7CF)rank

https://staff.washington.edu/weller/grep.html
https://www.regular-expressions.info/
echo "This is Frank Xu" | grep -Ei "frank“
echo "This is Frank Xu" | grep -E "(f|F)rank"
https://jex.im/regulex/#!flags=&re=frank
https://jex.im/regulex/#!flags=&re=(f%7CF)rank

https://staff.washington.edu/weller/grep.html

Verify some files

```
┌──(student㉿kalit150)-[~/regex]
└─$ ls CYFI727_Roster_2028.* -l
-rw-r--r-- 1 student student  2623 Apr 22 23:49 CYFI727_Roster_2028.csv
-rw-r--r-- 1 student student  2623 Apr 22 23:50 CYFI727_Roster_2028.txt
-rw-r--r-- 1 student student 13202 Apr 22 21:34 CYFI727_Roster_2028.xlsx
```

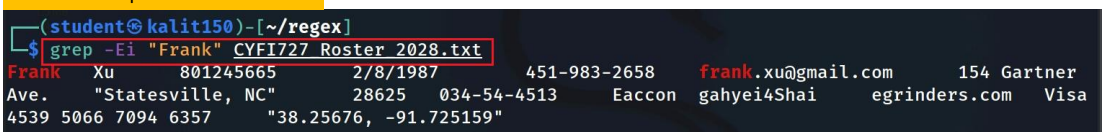
Verify the txt file. We will search the content of the file using regular expression.

A	B	C	D	E	F	G	H	I	J
Course Roster for CYFI 727 (Instrumentation)									
Term=Spring 2028-2029 Instructor=hoover Credits=4 Number of Students=9									
Last	First	ID #	Date of Birth	Phone	Email	Address	City	Zip	SSN
Clark	Jonathan	801242170	1/1/1985	443-154-5135	cjonathan@gmail.com	803 West Fairground Avenue	Muscatine, IA	52761	767-16-5642
Collins	Richard	8014025393	11/3/1986	451-552-5425	crichard@hotmail.com	741 Pennington St.	Allison Park, PA	15101	222-26-7753
Davis	Cody	801306930	5/5/1996	451-451-2546	davis.richard@ubalt.edu	53 Glen Creek Dr.	Glenside, PA	19038	533-10-8329
German	Paul	801229248	10/9/1984	785-213-0254	german_pau001@gmail.com	676 Valley Farms St.	Springfield Gardens, NY	11413	471-64-0361
Hargis	David	801063452	8/10/1986	789-351-9520	hargis-david@md.gov	913 Blackburn Street	Littleton, CO	80123	504-82-4609
Hengehold	Kevin	801292801	8/22/1985	210-897-6321	kevin001@mit.edu	980 Creekside Street	Madisonville, KY	42431	516-25-5740
Jacobson	Mark	801229266	5/21/1986	215-985-9652	mark.jacobson@gannon.edu	441 Myrtle Road	Smithtown, NY	11787	660-24-1002
McKinney	Kyle	801310462	3/11/1987	369-874-8521	mckyle_cool@example.com	720 E. Galvin Street	Pasadena, MD	21122	519-42-3715
Frank	Xu	801245665	2/8/1987	451-983-2658	frank.xu@gmail.com	154 Gartner Ave.	Statesville, NC	28625	034-54-4513
Ziebell	Jonathan	801297030	12/21/1988	521-854-3652	ziebell.jonathan@rsa.org	8 Big Rock Cove Avenue	Park Forest, IL	60466	512-36-9198

```
┌──(student㉿kalit150)-[~/regex]
└─$ cat CYFI727_Roster_2028.txt | head -n5
Course Roster for CYFI 727 (Instrumentation)
Term=Spring 2028-2029  Instructor=hoover  Credits=4  Number of Students=9

Last    First   ID #    Date of Birth   Phone   Email   Address City    Zip     SSN     Username   P
assword web site        Credit Card     Card Number     Geo coordinates
Clark   Jonathan        801242170       1/1/1985        443-154-5135    cjonathan@gmail.com     803
West Fairground Avenue  "Muscatine, IA" 52761   767-16-5642     snowstorm       miGhoh7aem3oh   furf
ley.com Master  5121 9811 1926 4947     "38.953112, -81.839352"
```

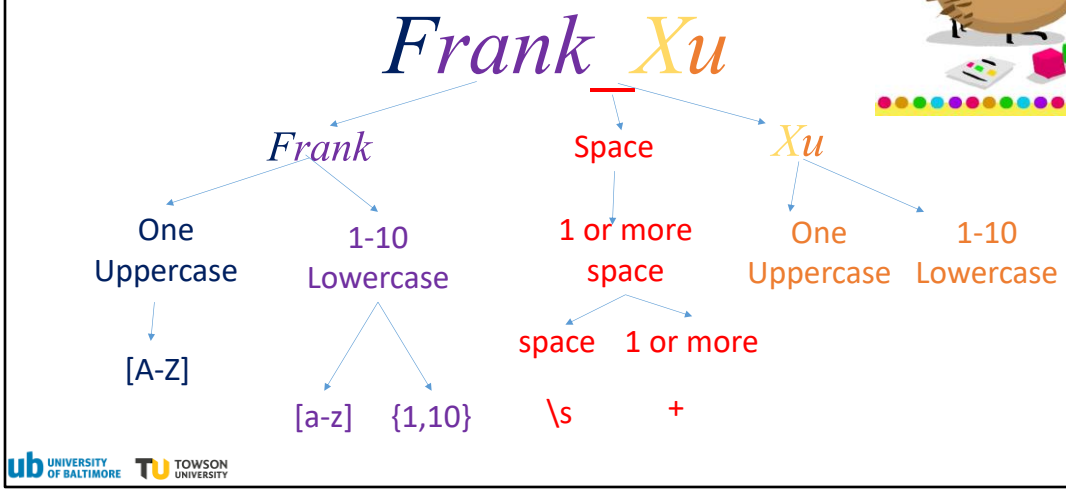
cat CYFI727_Roster_2028.txt | head -n5
grep -Ei "Frank" CYFI727_Roster_2028.txt

Search for a specific name “Frank”



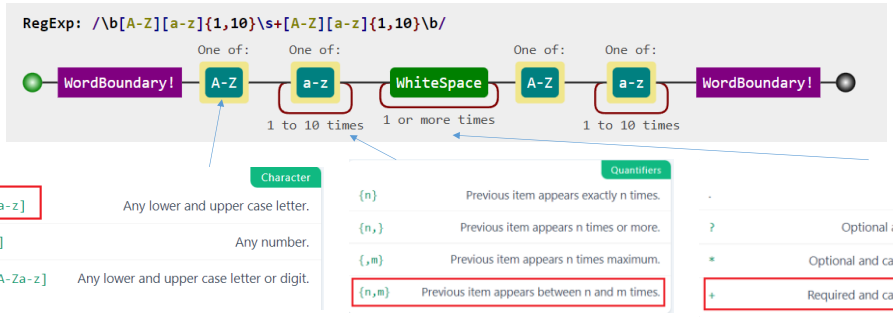
cat CYFI727_Roster_2028.txt | head -n5
grep -Ei "Frank" CYFI727_Roster_2028.txt

A simple pattern of all names



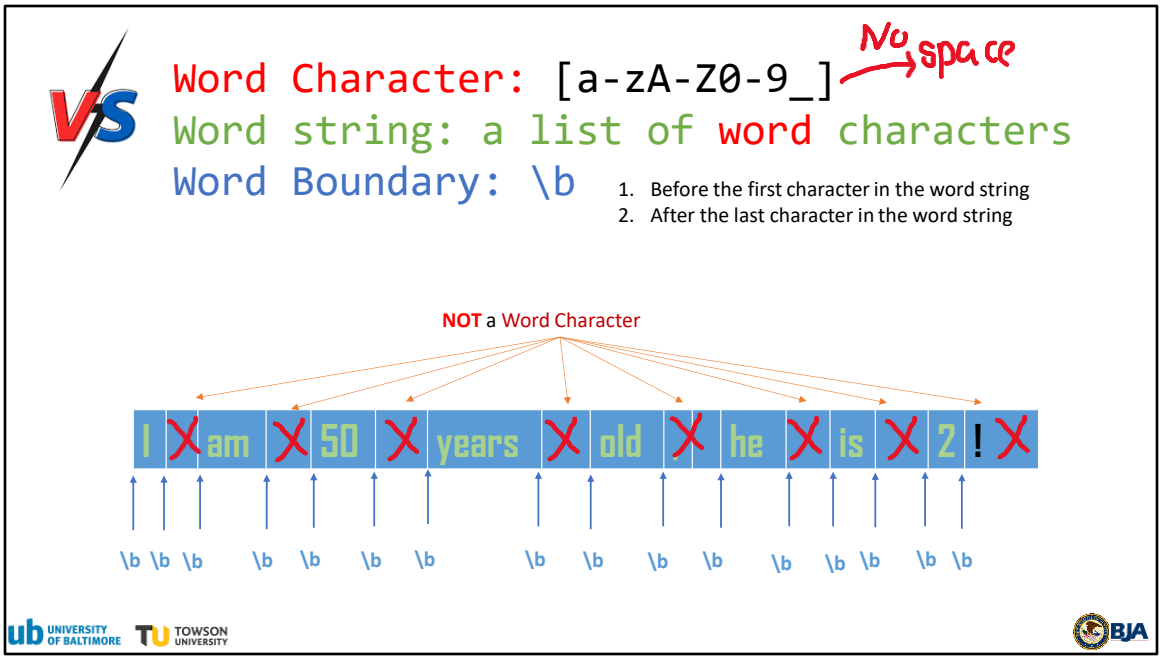
Match all names in a text file

```
┌──(student㉿kalit150)-[~/regex]
└─$ echo "I am Frank Xu" | grep -E -o "\b[A-Z][a-z]{1,10}\s+[A-Z][a-z]{1,10}\b"
Frank Xu
```



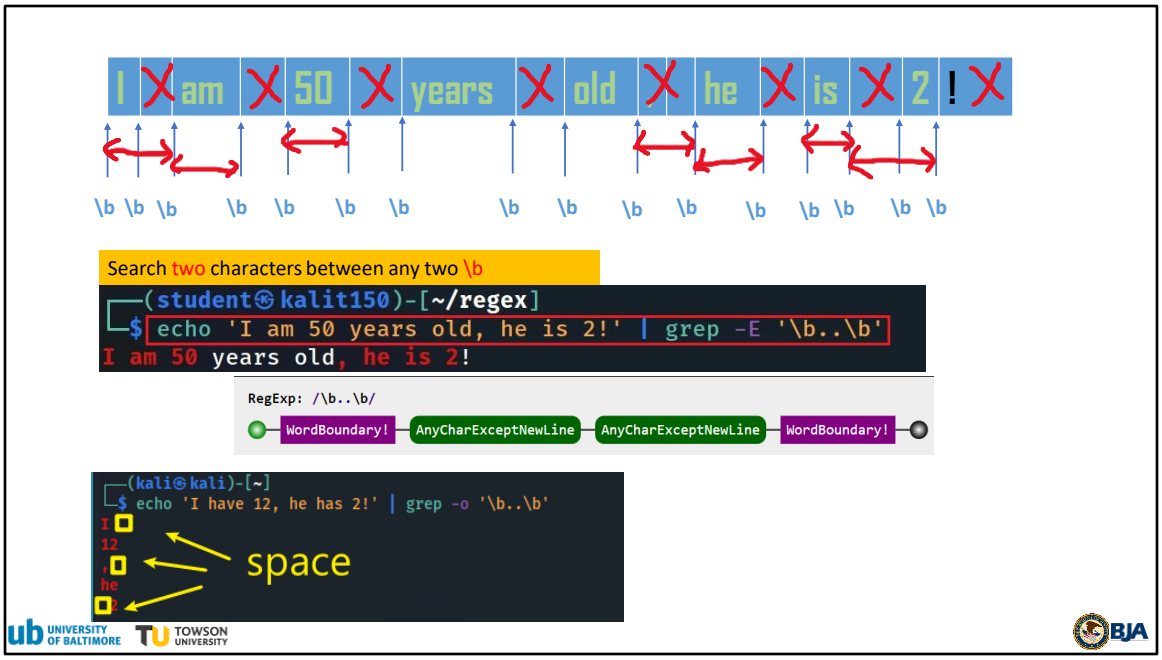
UNIVERSITY OF BALTIMORE TOWSON UNIVERSITY BJA

echo "I am Frank Xu" | grep -E -o "\b[A-Z][a-z]{1,10}\s+[A-Z][a-z]{1,10}\b“
https://jex.im/regulex/#!flags=&re=%5Cb%5BA-Z%5D%5Ba-z%5D%7B1%2C10%7D%5Cs%2B%5BA-Z%5D%5Ba-z%5D%7B1%2C10%7D%5Cb



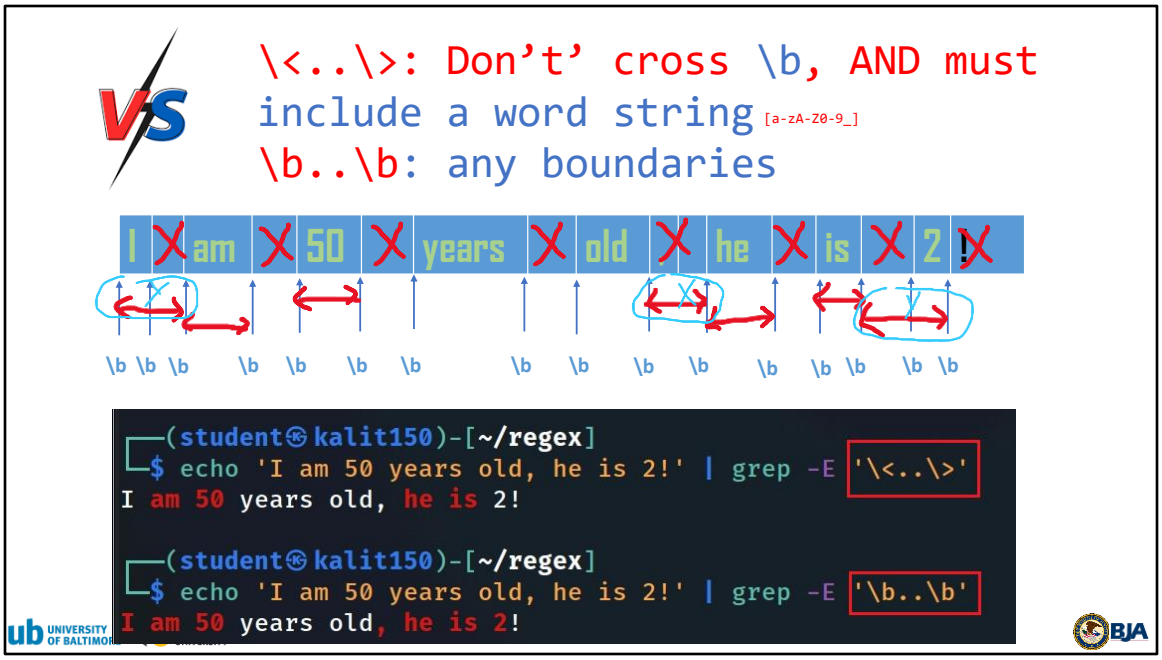
echo 'I am 50 years old, he is 2!' | grep -E '\b..\b'

https://learnbyexample.github.io/learn_gnugrep_ripgrep/gotchas-and-tricks.html#:~:text=Word%20boundary%20differences,-The%20%2Dw%20option&text=The%20test%20is%20that%20the,a%20non%2Dword%20constituent%20character.
https://www.regular-expressions.info/wordboundaries.html



echo 'I am 50 years old, he is 2!' | grep -E '\b..\b'

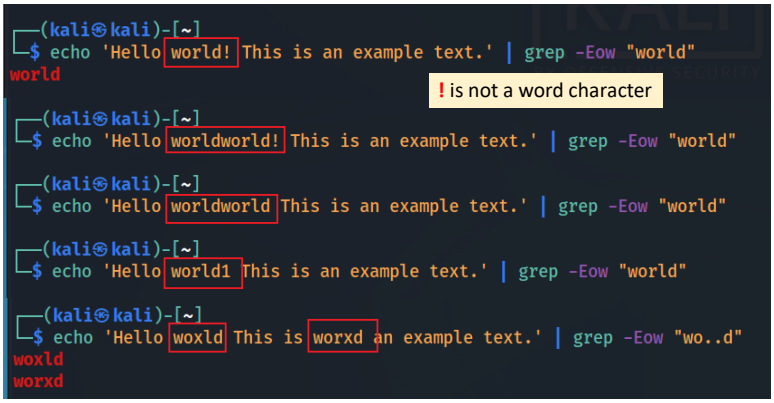
https://learnbyexample.github.io/learn_gnugrep_ripgrep/gotchas-and-tricks.html#:~:text=Word%20boundary%20differences,-The%20%2Dw%20option&text=The%20test%20is%20that%20the,a%20non%2Dword%20constituent%20character.



echo 'I am 50 years old, he is 2!' | grep -E '\<..\>'

-w : match only whole words

`[a-zA-Z0-9_]+`



`\<world\>`

Test the patten in a text file

```
└─$ grep -E -o "\b[A-Z][a-z]{1,10}\s+[A-Z][a-z]{1,10}\b" CYFI727 Roster 2028.txt
Course Roster
Last    First
Birth   Phone
Email   Address
City    Zip
Username        Password
Credit Card
Card Number
Clark   Jonathan
West Fairground
Collins Richard
Pennington St
Allison Park
Davis   Cody
Glen Creek
German  Paul
Valley Farms
Springfield Gardens
Hargis  David
Blackburn Street
Hengehold       Kevi
Creekside Street
Jacobson        Mark
Myrtle Road
Galvin Street
Frank   Xu
Gartner Ave
Ziebell Jonathan
Big Rock
Cove Avenue
Park Forest
```

head

address

Course Roster for CYFI 727 (Instrumentation)
Term=Spring 2028-2029 Instructor=hoover Credits=4 Number of Students=9

Last	First	ID #	Date of Birth	Phone	Email	Address	City	Zip	SSN
Clark	Jonathan	801242170	1/1/1985	443-154-5135	cjonathan@gmail.com	803 West Fairground Avenue	Muscatine, IA	52761	767-16-5642
Collins	Richard	3014025393	11/3/1986	451-552-5425	crichard@hotmail.com	741 Pennington St.	Allison Park, PA	15101	222-26-7753
Davis	Cody	801306930	5/5/1996	451-451-2546	davis.richard@ubalt.edu	53 Glen Creek Dr.	Glenside, PA	19038	533-10-8329
German	Paul	801229248	10/9/1984	785-213-0254	german_pau001@gmail.com	676 Valley Farms St.	Springfield Gardens, NY	11413	471-64-0361
Hargis	David	801063452	8/10/1986	789-351-9520	hargis-david@md.gov	913 Blackburn Street	Littleton, CO	80123	504-82-4609
Hengehold	Kevin	801292801	8/22/1985	210-897-6321	kevin001@mit.edu	980 Creekside Street	Madisonville, KY	42431	516-25-5740
Jacobson	Mark	801229266	5/21/1986	215-985-9652	mark.jacobson@gannon.edu	441 Myrtle Road	Smithtown, NY	11787	660-24-1002
McKinney	Kyle	801310462	3/11/1987	369-874-8521	mckyle_cool@example.com	720 E. Galvin Street	Pasadena, MD	21122	519-42-3715
Frank	Xu	801245665	2/8/1987	451-983-2658	frank.xu@gmail.com	154 Gartner Ave.	Statesville, NC	28625	034-54-4513
Ziebell	Jonathan	801297030	12/21/1988	521-854-3652	ziebell.jonathan@rsa.org	8 Big Rock Cove Avenue	Park Forest, IL	60466	512-36-9198

ub UNIVERSITY OF BALTIMORE TU TOWSON UNIVERSITY

Conditional match ?

look behind
(?<=*foo*)**xxx**

Match xxx with a preceding string *foo*

look ahead
xxx(?=*foo*)

Match xxx with a following string *foo*

has to be Perl-compatible

```
┌──(student㉿kalit150)-[~/regex]
└─$ echo "pretend prefix present" | grep -P  "(?<=pre)\w+"
pretend prefix present

┌──(student㉿kalit150)-[~/regex]
└─$ echo "fishing working enjoying" | grep -P  "\w+(?=ing)"
fishing working enjoying
```

shorthand classes
\w "word" character (letter, digit, or underscore) [a-zA-Z0-9_]
\d digit
\s whitespace (space, tab, vtab, newline)

-Po

UNIVERSITY OF BALTIMORE TOWSON UNIVERSITY BJA

echo "pretend prefix present" | grep -P "(?<=pre)\w+“
echo "fishing working enjoying" | grep -P "\w+(?=ing)“

Negative lookup

look behind

(?<!*foo*)**xxx**

Match xxx without a preceding string *foo*

look ahead

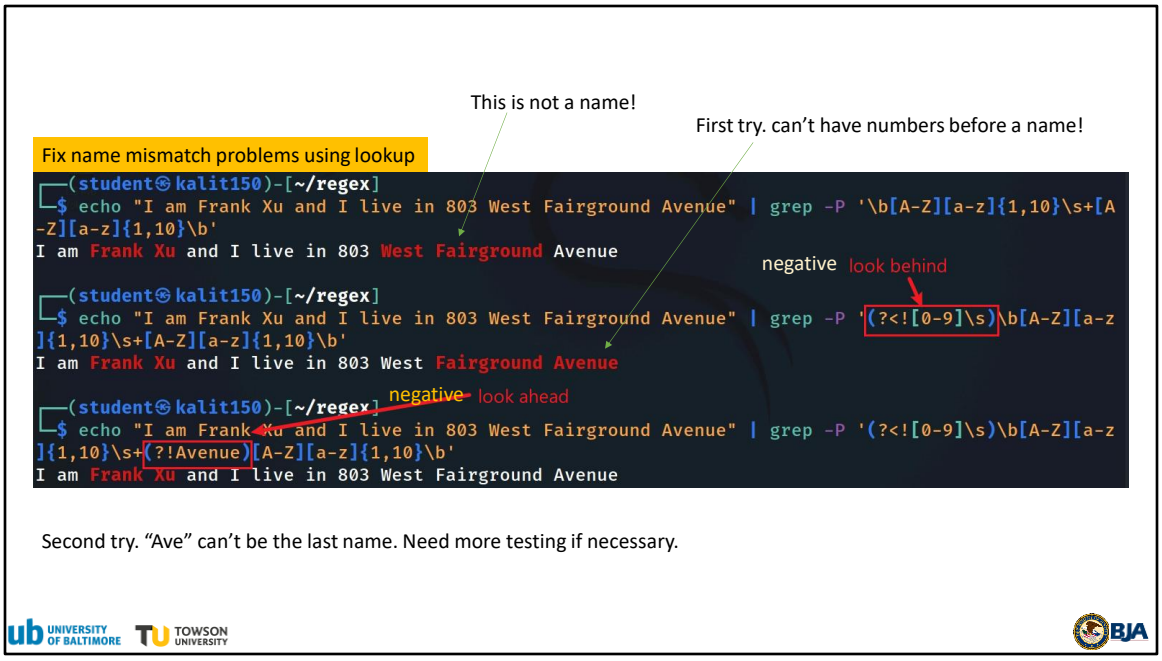
xxx(?!*foo*)

Match xxx without a following string *foo*

```
┌──(student㉿kalit150)-[~/regex]
└─$ echo "prefix postfix present" | grep -P '(?<!pre)fix'
prefix postfix present

┌──(student㉿kalit150)-[~/regex]
└─$ echo "prefix postfix present" | grep -P 'pre(?!fix)'
prefix postfix present
```

- Negative lookup pattern **MUST be** single quote ‘ ’
- a lookup string length must be fixed

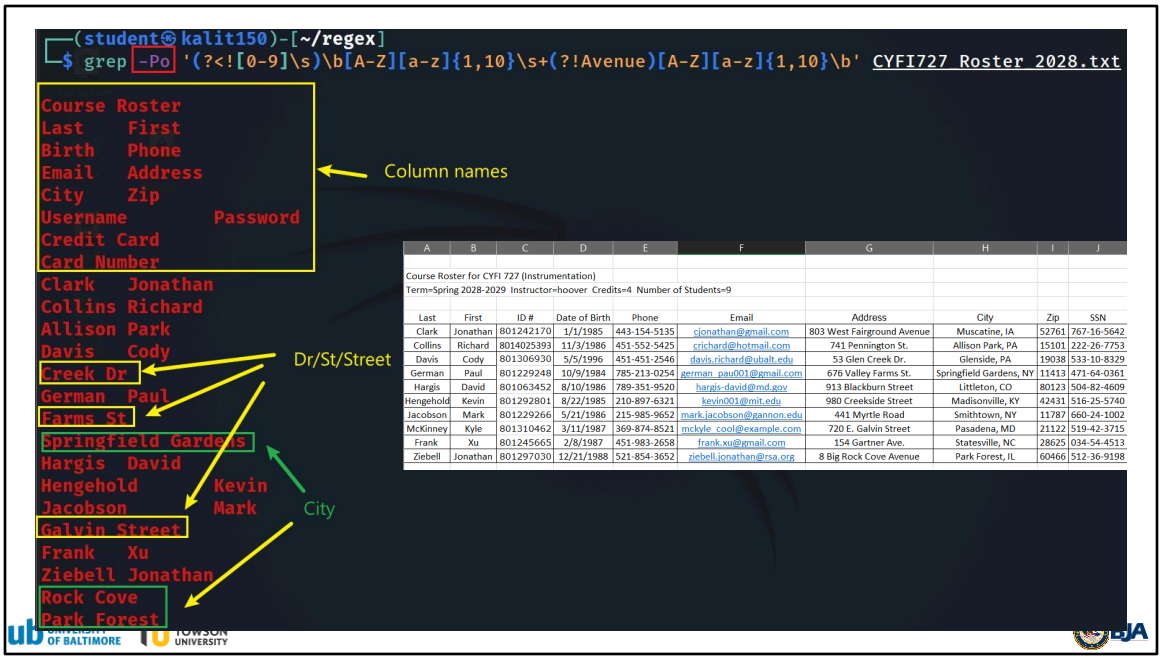
echo “prefix postfix present" | grep -P ‘?<!pre)fix’
echo "prefix postfix present" | grep -P 'pre(?<!fix)'



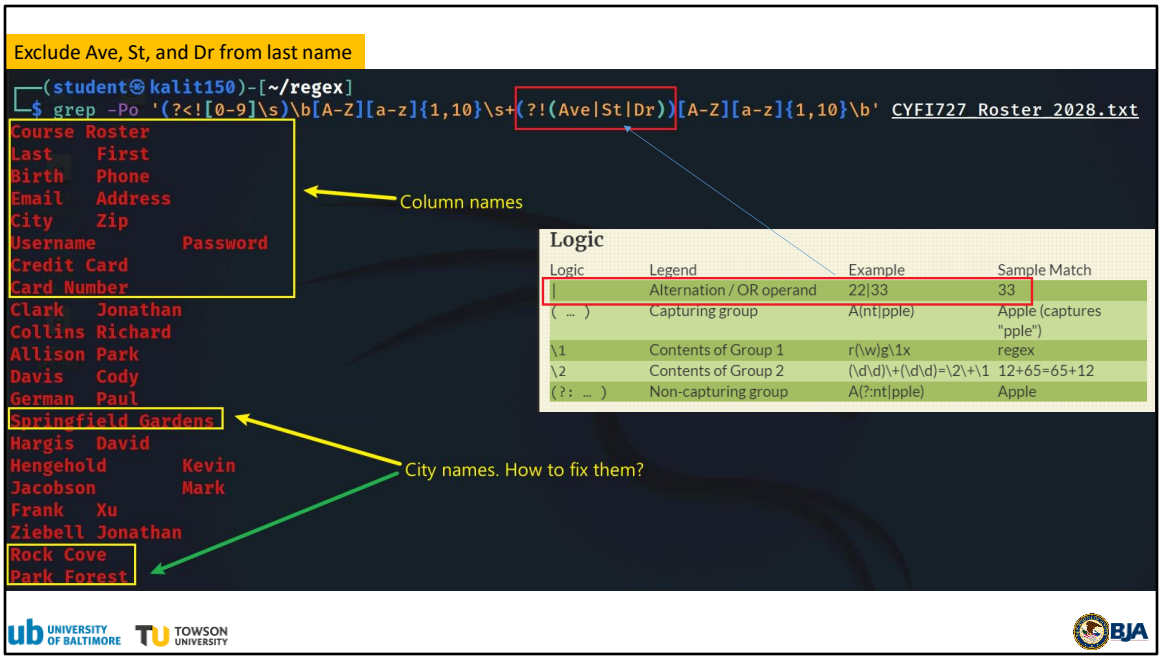
Second try. "Ave" can't be the last name. Need more testing if necessary.

UNIVERSITY OF BALTIMORE TOWSON UNIVERSITY BJA

echo "I am Frank Xu and I live in 803 West Fairground Avenue" | grep -P '\b[A-Z][a-z]{1,10}\s+[A-Z][a-z]{1,10}\b'
echo "I am Frank Xu and I live in 803 West Fairground Avenue" | grep -P '(?<![0-9]\s)\b[A-Z][a-z]{1,10}\s+[A-Z][a-z]{1,10}\b
echo "I am Frank Xu and I live in 803 West Fairground Avenue" | grep -P '(?<![0-9]\s)\b[A-Z][a-z]{1,10}\s+(?!Avenue)[A-Z][a-z]{1,10}\b'



grep -Po '(?<![0-9]\s)\b[A-Z][a-z]{1,10}\s+(?!Avenue)[A-Z][a-z]{1,10}\b' CYFI727_Roster_2028.txt



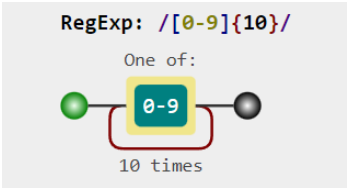
grep -Po '(?<![0-9]\s)\b[A-Z][a-z]{1,10}\s+(?![Ave|St|Dr])[A-Z][a-z]{1,10}\b' CYFI727_Roster_2028.txt

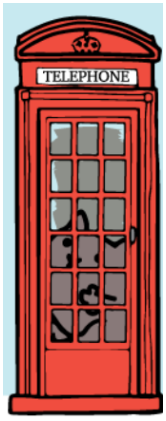
Match phone numbers

- 1234567890
- 123.456.7890
- 123-456-7890
- 123 456 7890
- (123)456-7890
- +11234567890

1234567890

[0-9]{10}

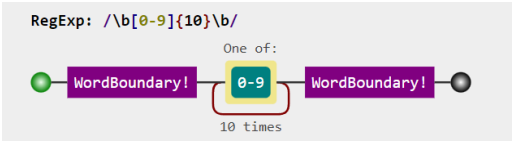




https://jex.im/regulex/#!flags=&re=%5B0-9%5D%7B10%7D

Match any 10 digitals phone numbers (xxxxxxxxxx and x is a digital)

```
┌──(student㉿kalit150)-[~/regex]
└─$ echo  "My phone number is 1234567890" | grep -E "[0-9]{10}"
My phone number is 1234567890

┌──(student㉿kalit150)-[~/regex]
└─$ echo  "The serial number  is 1234567890123" | grep -E "[0-9]{10}"
The serial number  is 1234567890123
```




```
┌──(student㉿kalit150)-[~/regex]
└─$ echo  "My phone number is 1234567890" | grep -E "\b[0-9]{10}\b"
My phone number is 1234567890

┌──(student㉿kalit150)-[~/regex]
└─$ echo  "The serial number  is 1234567890123" | grep -E "\b[0-9]{10}\b"
            no results
```

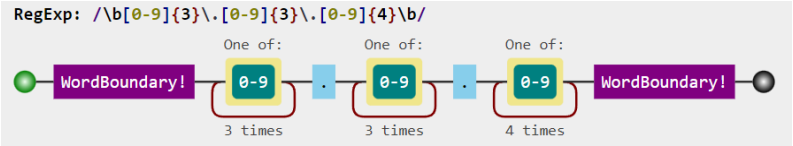
UNIVERSITY OF BALTIMORE TOWSON UNIVERSITY BJA

echo "My phone number is 1234567890" | grep -E "[0-9]{10}“
echo "The serial number is 1234567890123" | grep -E "[0-9]{10}“
echo "My phone number is 1234567890" | grep -E "\b[0-9]{10}\b“
echo "The serial number is 1234567890123" | grep -E "\b[0-9]{10}\b“
https://jex.im/regulex/#!flags=&re=%5Cb%5B0-9%5D%7B10%7D%5Cb

Match any 10 digitals phone numbers with the patten xxx.xxxx.xxx and x is a digital

123.456.7890

\b[0-9]{3} \.[0-9]{3}\.[0-9]{4}\b



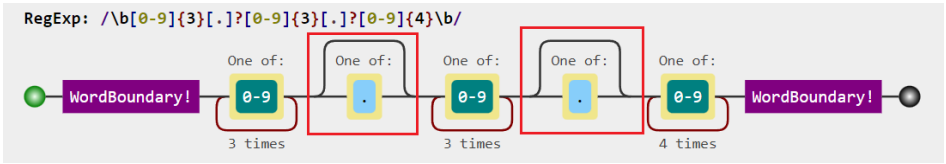

```
┌──(student㉿kalit150)-[~/regex]
└─$ echo  "My phone number is 123.456.7890" | grep -E "\b[0-9]{3}\.[0-9]{3}\.[0-9]{4}\b"
My phone number is 123.456.7890
```


echo "My phone number is 123.456.7890" | grep -E "\b[0-9]{3}\.[0-9]{3}\.[0-9]{4}\b"
https://jex.im/regulex/#!flags=&re=%5Cb%5B0-9%5D%7B3%7D%5C.%5B0-9%5D%7B3%7D%5C.%5B0-9%5D%7B4%7D%5Cb

Match both?

1234567890

123.456.7890



Wildcards	
.	Any character.
?	Optional and can only occur once.
*	Optional and can occur more than once.
+	Required and can occur more than once.

{0,1}

Test both phone number types

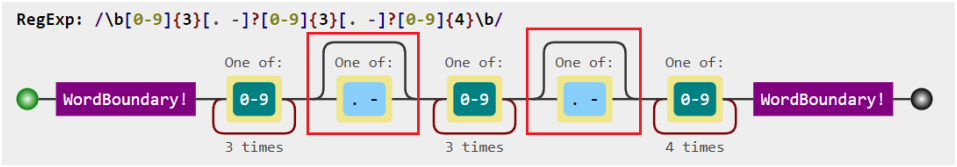
```
┌──(student㉿kalit150)-[~/regex]
└─$ echo  "My phone number is 123.456.7890" | grep -E "\b[0-9]{3}[\.]?[0-9]{3}[\.]?[0-9]{4}\b"
My phone number is 123.456.7890

┌──(student㉿kalit150)-[~/regex]
└─$ echo  "My phone number is 1234567890" | grep -E "\b[0-9]{3}[\.]?[0-9]{3}[\.]?[0-9]{4}\b"
My phone number is 1234567890
```


echo "My phone number is 123.456.7890" | grep -E "\b[0-9]{3}[\.]?[0-9]{3}[\.]?[0-9]{4}\b“

echo "My phone number is 1234567890" | grep -E "\b[0-9]{3}[\.]?[0-9]{3}[\.]?[0-9]{4}\b"

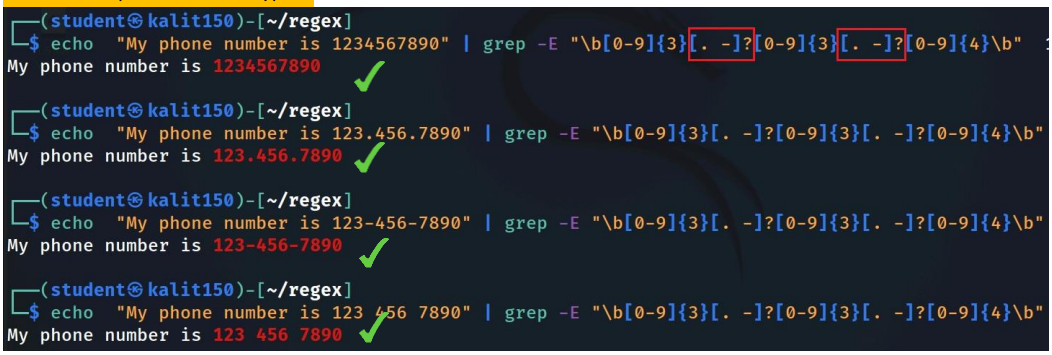
Match all four?

1234567890
123.456.7890
123-456-7890
123 456 7890



https://jex.im/regulex/#!flags=&re=%5Cb%5B0-9%5D%7B3%7D%5B.%20-%5D%3F%5B0-9%5D%7B3%7D%5B.%20-%5D%3F%5B0-9%5D%7B4%7D%5Cb

Test all four phone number types



UNIVERSITY OF BALTIMORE TOWSON UNIVERSITY BJA

echo "My phone number is 1234567890" | grep -E "\b[0-9]{3}[. -]?[0-9]{3}[. -]?[0-9]{4}\b“
echo "My phone number is 123.456.7890" | grep -E "\b[0-9]{3}[. -]?[0-9]{3}[. -]?[0-9]{4}\b“
echo "My phone number is 123-456-7890" | grep -E "\b[0-9]{3}[. -]?[0-9]{3}[. -]?[0-9]{4}\b“
echo "My phone number is 123 456 7890" | grep -E "\b[0-9]{3}[. -]?[0-9]{3}[. -]?[0-9]{4}\b"

Match all five?

1234567890
123.456.7890
123-456-7890
123 456 7890
(123)456-7890

syntax: (?if then | else)

(\()? \b[0-9]{3} (?(1) \) | [. -]?) [0-9]{3}[. -]?[0-9]{4}\b

```
┌──(student㉿kalit150)-[~/regex]
└─$ echo  "My phone number is (123)456-7890" | grep -P "(\()?\b[0-9]{3}(?(1)\)?|[. -]?)[0-9]
{3}[. -]?[0-9]{4}\b"
My phone number is (123)456-7890
```

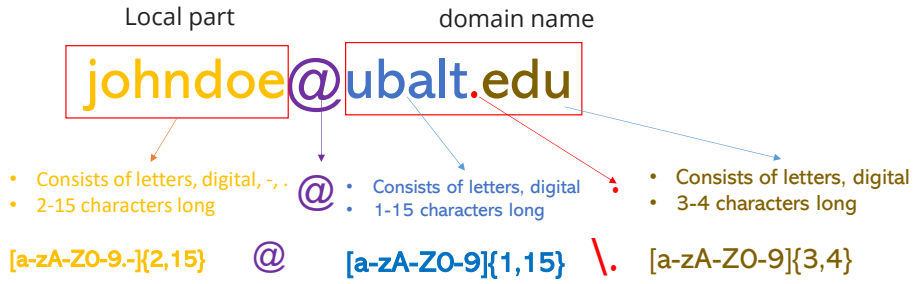
ub UNIVERSITY OF BALTIMORE TU TOWSON UNIVERSITY BJA

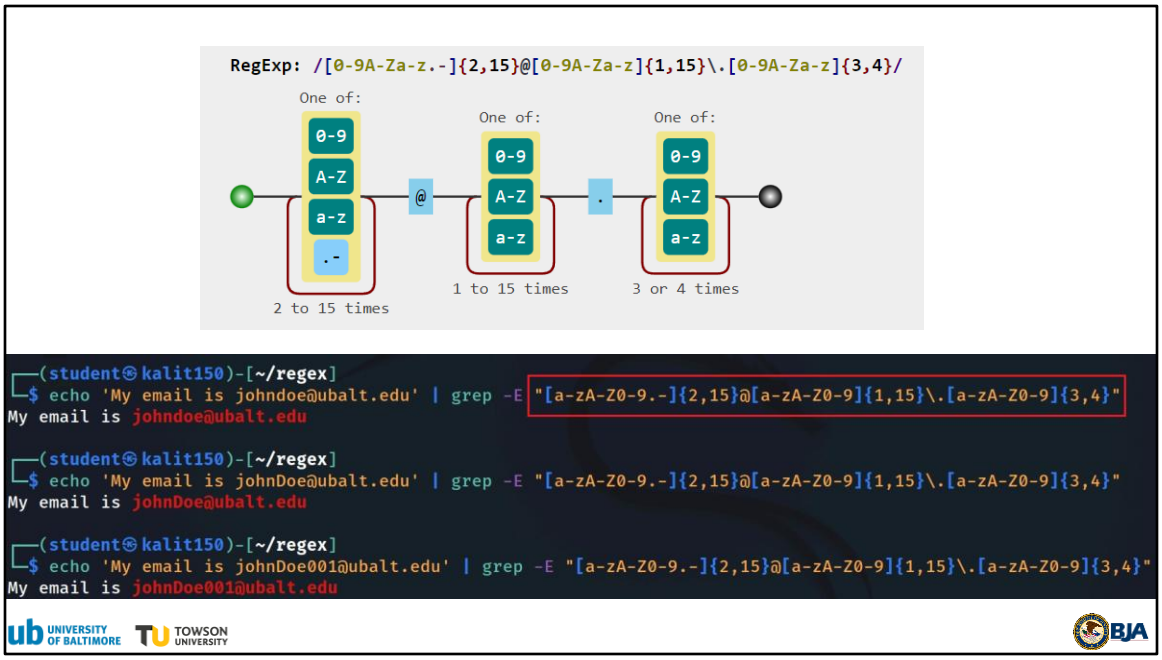
https://jex.im/regulex/#!flags=&re=%5Cb%5B0-9%5D%7B3%7D%5B.%20-%5D%3F%5B0-9%5D%7B3%7D%5B.%20-%5D%3F%5B0-9%5D%7B4%7D%5Cb

echo "My phone number is (123)456-7890" | grep -P "(\()?\b[0-9]{3}(?(1)\)?|[. -]?)[0-9]{3}[. -]?[0-9]{4}\b"

Match email addresses

john.doe@gmail.com
john-doe@md.gov
johnDoe001@mozilla.org





https://jex.im/regulex/#!flags=&re=%5Ba-zA-Z0-9.-%5D%7B2%2C15%7D%40%5Ba-zA-Z0-9%5D%7B1%2C15%7D%5C.%5Ba-zA-Z0-9%5D%7B3%2C4%7D

echo 'My email is johndoe@ubalt.edu' | grep -E "[a-zA-Z0-9.-]{2,15}@[a-zA-Z0-9]{1,15}\.[a-zA-Z0-9]{3,4}“

echo 'My email is johnDoe@ubalt.edu' | grep -E "[a-zA-Z0-9.-]{2,15}@[a-zA-Z0-9]{1,15}\.[a-zA-Z0-9]{3,4}“

echo 'My email is johnDoe001@ubalt.edu' | grep -E "[a-zA-Z0-9.-]{2,15}@[a-zA-Z0-9]{1,15}\.[a-zA-Z0-9]{3,4}"

A	B	C	D	E	F	G	H	I	J
Course Roster for CYFI 727 (Instrumentation)									
Term=Spring 2028-2029 Instructor=hoover Credits=4 Number of Students=9									
Last	First	ID #	Date of Birth	Phone	Email	Address	City	Zip	SSN
Clark	Jonathan	801242170	1/1/1985	443-154-5135	cjonathan@gmail.com	803 West Fairground Avenue	Muscatine, IA	52761	767-16-5642
Collins	Richard	8014025393	11/3/1986	451-552-5425	crichard@hotmail.com	741 Pennington St.	Allison Park, PA	15101	222-26-7753
Davis	Cody	801306930	5/5/1996	451-451-2546	davis.richard@ubalt.edu	53 Glen Creek Dr.	Glenside, PA	19038	533-10-8329
German	Paul	801229248	10/9/1984	785-213-0254	german_pau001@gmail.com	676 Valley Farms St.	Springfield Gardens, NY	11413	471-64-0361
Hargis	David	801063452	8/10/1986	789-351-9520	hargis-david@md.gov	913 Blackburn Street	Littleton, CO	80123	504-82-4609
Hengehold	Kevin	801292801	8/22/1985	210-897-6321	kevin001@mit.edu	980 Creekside Street	Madisonville, KY	42431	516-25-5740
Jacobson	Mark	801229266	5/21/1986	215-985-9652	mark.jacobson@gannon.edu	441 Myrtle Road	Smithtown, NY	11787	660-24-1002
McKinney	Kyle	801310462	3/11/1987	369-874-8521	mckyle_cool@example.com	720 E. Galvin Street	Pasadena, MD	21122	519-42-3715
Frank	Xu	801245665	2/8/1987	451-983-2658	frank.xu@gmail.com	154 Gartner Ave.	Statesville, NC	28625	034-54-4513
Ziebell	Jonathan	801297030	12/21/1988	521-854-3652	ziebell.jonathan@rsa.org	8 Big Rock Cove Avenue	Park Forest, IL	60466	512-36-9198

```
┌──(student㉿kalit150)-[~/regex]
└─$ grep -Eo "[a-zA-Z0-9.-]{2,15}@[a-zA-Z0-9]{1,15}\.[a-zA-Z0-9]{3,4}" CYFI727_Roster_2028.txt
cjonathan@gmail.com
crichard@hotmail.com
davis.richard@ubalt.edu
pau001@gmail.com
hargis-david@md.gov
kevin001@mit.edu
mark.jacobson@gannon.edu
cool@example.com
frank.xu@gmail.com
iebell.jonathan@rsa.org
```

UNIVERSITY OF BALTIMORE TOWSON UNIVERSITY BJA

grep -Eo "[a-zA-Z0-9.-]{2,15}@[a-zA-Z0-9]{1,15}\.[a-zA-Z0-9]{3,4}" CYFI727_Roster_2028.txt

Check if a password is valid

Pattern definition:

- Minimum length of 3, maximum length of 18
- Composed by letters, numbers or dashes or @

```
┌──(student㉿kalit150)-[~/regex]
└─$ echo "1234@PwdsUer-" |  grep -E "[a-zA-Z0-9@-]{3,18}"
1234@PwdsUer-
```


echo "1234@PwdsUer-" | grep -E "[a-zA-Z0-9@-]{3,18}"

Match Java source code

Search string in Java and show line numbers

```
┌──(student㉿kalit150)-[~/regex]
└─$ grep -En "instanceof"  ScriptablePointer.java
61:         if (obj instanceof Scriptable) {
63:             if (node instanceof NativeArray) {
68:                 if (val instanceof Number) {
86:         if (value instanceof Wrapper) {
```

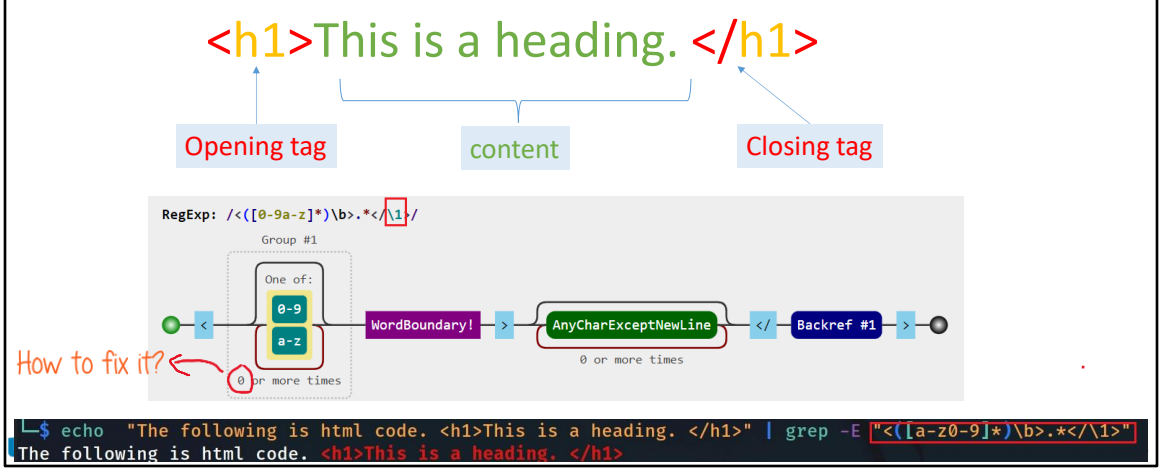
Show how many time the key words appears in Java source code

```
┌──(student㉿kalit150)-[~/regex]
└─$ grep -En "instanceof"  ScriptablePointer.java | wc -l
4
```

ub UNIVERSITY OF BALTIMORE TU TOWSON UNIVERSITY BJA

grep -En "instanceof" ScriptablePointer.java

Match HTML code (including tags and content) using Backreference \n

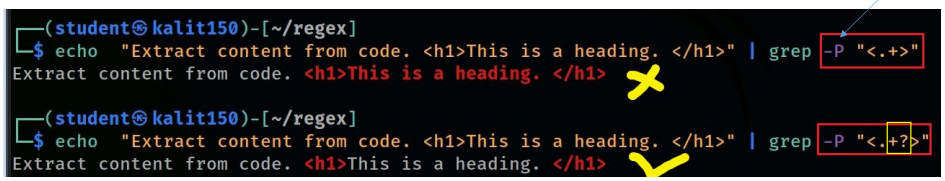


echo "The following is html code. <h1>This is a heading. </h1>" | grep -E "<([a-z0-9]*)\b>.*</\1>"

Match HTML tags using Lazy quantifier

<h1>This is a heading.</h1>

MUST use -P



Greedy quantifier	Lazy quantifier	Description
*	* ?	Star Quantifier: 0 or more
+	+ ?	Plus Quantifier: 1 or more
?	? ?	Optional Quantifier: 0 or 1
{n}	{n} ?	Quantifier: exactly n
{n,}	{n,} ?	Quantifier: n or more
{n,m}	{n,m} ?	Quantifier: between n and m

UNIVERSITY OF BALTIMORE TOWSON UNIVERSITY BJA

echo "Extract content from code. <h1>This is a heading. </h1>" | grep -P "<.+>“
echo "Extract content from code. <h1>This is a heading. </h1>" | grep -P "<.+?>“
https://stackoverflow.com/questions/3027518/how-to-do-a-non-greedy-match-in-grep

Match content of HTML code

<h1>This is a heading. </h1>

(?<=<([a-z0-9]{2}) >).* (?=</ \1>)

look behind

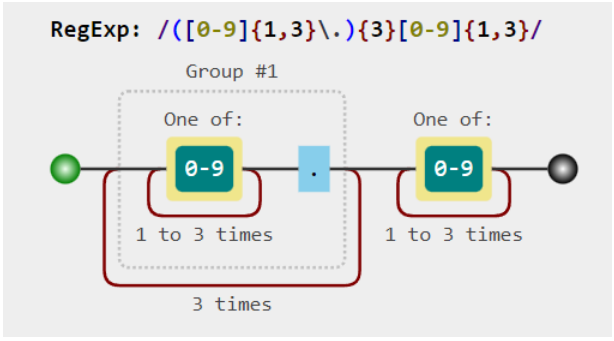
look ahead

```
┌──(student㉿kalit150)-[~/regex]
└─$ echo  "Extract content from code. <h1>This is a heading. </h1>" | grep -P "(?<=<([a-z0-9]{2})>).*(?=</\1>)"
Extract content from code. <h1>This is a heading. </h1>
```


echo "Extract content from code. <h1>This is a heading. </h1>" | grep -P "(?<=<([a-z0-9]{2})>).*(?=</\1>)"

Match IP4 address with group ()

192.168.0.1
443.125.121.1
255.255.255.0
89.25.23.0



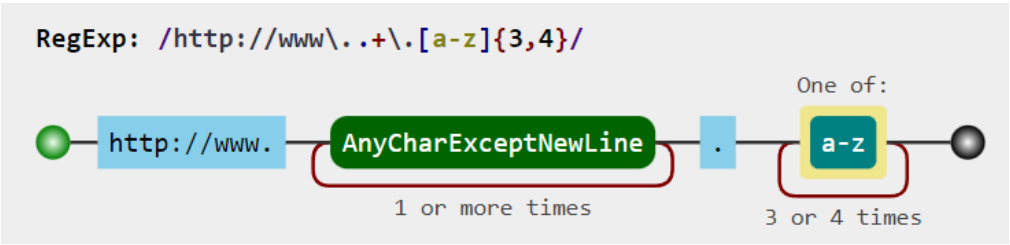

```
┌──(student㉿kalit150)-[~/regex]
└─$ echo  "The IP address is 443.125.121.1" | grep -E "([0-9]{1,3}\.){3}[0-9]{1,3}"
The IP address is 443.125.121.1
```

https://jex.im/regulex/#!flags=&re=(%5B0-9%5D%7B1%2C3%7D%5C.)%7B3%7D%5B0-9%5D%7B1%2C3%7D

echo "The IP address is 443.125.121.1" | grep -E "([0-9]{1,3}\.){3}[0-9]{1,3}"

Match HTTP requests

http://www.ubalt.edu

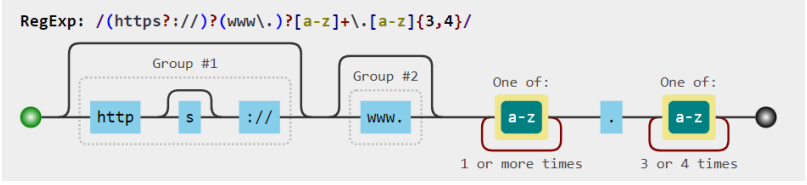



```
┌──(student㉿kalit150)-[~/regex]
└─$ echo  "It continas a valid url http://www.ubalt.edu" | grep -E "http://www\..+\.[a-z]{3,4}"
It continas a valid url http://www.ubalt.edu
```


echo "It continas a valid url http://www.ubalt.edu" | grep -E "http://www\..+\.[a-z]{3,4}"

Match variations of website

- http://www.ubalt.edu
- https://www.ubalt.edu
- www.ubalt.edu
- ubalt.edu

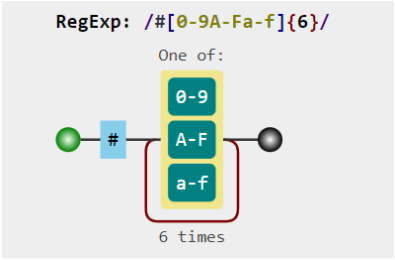



```
┌──(student㉿kalit150)-[~/regex]
└─$ echo "It continas a valid url www.ubalt.edu" | grep -E "(https?://)?(www\.)?[a-z]+\.[a-z]{3,4}"
It continas a valid url www.ubalt.edu
```

https://jex.im/regulex/#!flags=&re=(https%3F%3A%2F%2F)%3F(www%5C.)%3F%5Ba-z%5D%2B%5C.%5Ba-z%5D%7B3%2C4%7D

echo "It continas a valid url www.ubalt.edu" | grep -E "(https?://)?(www\.)?[a-z]+\.[a-z]{3,4}"

Match Hexadecimal number



IndianRed	#CD5C5C	rgb(205, 92, 92)
LightCoral	#F08080	rgb(240, 128, 128)
Salmon	#FA8072	rgb(250, 128, 114)
DarkSalmon	#E9967A	rgb(233, 150, 122)
LightSalmon	#FFA07A	rgb(255, 160, 122)

Match Hex of colors

```
┌──(student㉿kalit150)-[~/regex]
└─$ echo "The hex of a color #CD5C5C" | grep -E "#[a-fA-F0-9]{6}"
The hex of a color #CD5C5C
```

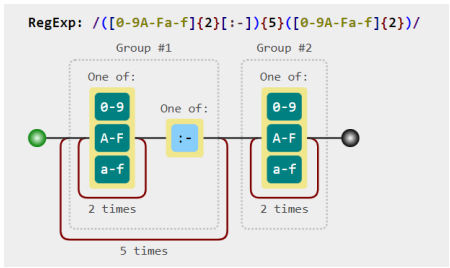
ub UNIVERSITY OF BALTIMORE TU TOWSON UNIVERSITY BJA

echo "The hex of a color #CD5C5C" | grep -E "#[a-fA-F0-9]{6}"

Match MAC address

The standard (IEEE 802) format for printing MAC-48 addresses in human-friendly form is six groups of two hexadecimal digits, separated by hyphens - or colons :.

01-23-45-67-89-AB
01:23:45:67:89:AB



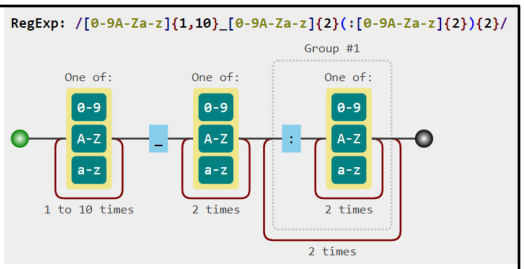

```
┌──(student㉿kalit150)-[~/regex]
└─$ echo "My MAC address is 01-23-45-67-89-AB" | grep -E "([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f
]{2})"
My MAC address is 01-23-45-67-89-AB

┌──(student㉿kalit150)-[~/regex]
└─$ echo "My MAC address is 01:23:45:67:89:AB" | grep -E "([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f
]{2})"
My MAC address is 01:23:45:67:89:AB
```

https://jex.im/regulex/#!flags=&re=(%5B0-9A-Fa-f%5D%7B2%7D%5B%3A-%5D)%7B5%7D(%5B0-9A-Fa-f%5D%7B2%7D)

echo "My MAC address is 01-23-45-67-89-AB" | grep -E "([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f]{2})“

echo "My MAC address is 01:23:45:67:89:AB" | grep -E "([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f]{2})"

PaloAlto_00:0a:30
VMware_86:44:c3



Match MAC patterns

```
┌──(student㉿kalit150)-[~/regex]
└─$ echo "My MAC address is PaloAlto_00:0a:30" | grep -E "[a-zA-Z0-9]{1,10}_[0-9a-zA-Z]{2}(:
[0-9a-zA-Z]{2}){2}"
My MAC address is PaloAlto_00:0a:30
```

https://jex.im/regulex/#!flags=&re=%5Ba-zA-Z%5D%7B1%2C10%7D_(%5B%3A-%5D%5B0-9a-zA-Z%5D%7B2%7D)%7B3%7D

echo "My MAC address is PaloAlto_00:0a:30" | grep -E "[a-zA-Z0-9]{1,10}_[0-9a-zA-Z]{2}(:[0-9a-zA-Z]{2}){2}"

Match MAC from a pcap file

tshark help

```
┌──(student㉿kalit150)-[~/regex]
└─$ tshark --help
TShark (Wireshark) 3.6.0 (Git v3.6.0 packaged as 3.6.0-1)
Dump and analyze network traffic.
See https://www.wireshark.org for more information.

Usage: tshark [options] ...
```

- -r <infile>, --read-file <infile>:
  - set the filename to read from (or '-' for stdin)
- -e <field>
  - field to print if -Tfields selected (e.g. tcp.port, _ws.col.Info)

Convert pcap to text

```
┌──(student㉿kalit150)-[~/regex]
└─$ tshark -r vm_to_ub_traffic.log -T fields -E header=y -E separator=, -E quote=d -E occurr
ence=f -e ip.version -e ip.hdr_len -e ip.tos -e ip.id -e ip.flags -e ip.flags.rb -e ip.flags
.df  -e ip.flags.mf -e ip.frag_offset -e ip.ttl -e ip.proto -e ip.checksum -e ip.src -e ip.d
st -e ip.len -e ip.dsfield -e tcp.srcport -e tcp.dstport -e tcp.seq -e tcp.ack -e tcp.len  -
e tcp.hdr_len -e tcp.flags -e tcp.flags.fin -e tcp.flags.syn -e tcp.flags.reset -e tcp.flags
.push -e tcp.flags.ack -e tcp.flags.urg -e tcp.flags.cwr -e tcp.window_size -e tcp.checksum
-e tcp.urgent_pointer -e tcp.options.mss_val -e eth.src_resolved -e eth.dst_resolved> output
.csv
```

tshark -r vm_to_ub_traffic.log -T fields -E header=y -E separator=, -E quote=d -E occurrence=f -e ip.version -e ip.hdr_len -e ip.tos -e ip.id -e ip.flags -e ip.flags.rb -e ip.flags.df -e ip.flags.mf -e ip.frag_offset -e ip.ttl -e ip.proto -e ip.checksum -e ip.src -e ip.dst -e ip.len -e ip.dsfield -e tcp.srcport -e tcp.dstport -e tcp.seq -e tcp.ack -e tcp.len -e tcp.hdr_len -e tcp.flags -e tcp.flags.fin -e tcp.flags.syn -e tcp.flags.reset -e tcp.flags.push -e tcp.flags.ack -e tcp.flags.urg -e tcp.flags.cwr -e tcp.window_size -e tcp.checksum -e tcp.urgent_pointer -e tcp.options.mss_val -e eth.src_resolved -e eth.dst_resolved> output.csv

https://www.linkedin.com/pulse/build-machine-learning-model-network-flow-tao-liu/
https://shantoroy.com/networking/convert-pcap-to-csv-using-tshark/

Match the first pattern in a pcap file

```
┌──(student㉿kalit150)-[~/regex]
└─$ grep -Eo "([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f]{2})" output.csv | sort | uniq -c| sort -n
      1 55:55:55:55:55:55
```

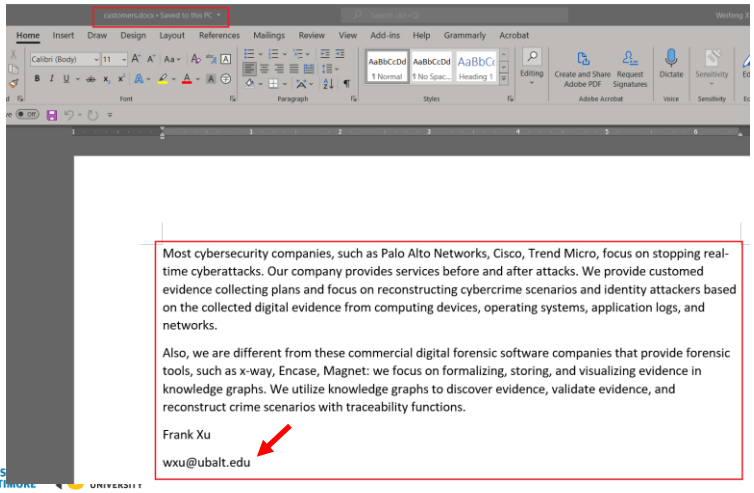
Match the second pattern in a pcap file

```
┌──(student㉿kalit150)-[~/regex]
└─$ grep -Eo "[a-zA-Z0-9]{1,10}_[0-9a-zA-Z]{2}(:[0-9a-zA-Z]{2}){2}" output.csv | sort | uniq
-c| sort -n
  21746 VMware_86:44:c3
  21747 PaloAlto_00:0a:30
```

UNIVERSITY OF BALTIMORE TOWSON UNIVERSITY BJA

grep -Eo "([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f]{2})" output.csv | sort | uniq -c| sort -n
grep -Eo "[a-zA-Z0-9]{1,10}_[0-9a-zA-Z]{2}(:[0-9a-zA-Z]{2}){2}" output.csv | sort | uniq -c| sort -n

Grep email from *customers.docx*



Most cybersecurity companies, such as Palo Alto Networks, Cisco, Trend Micro, focus on stopping real-time cyberattacks. Our company provides services before and after attacks. We provide customed evidence collecting plans and focus on reconstructing cybercrime scenarios and identity attackers based on the collected digital evidence from computing devices, operating systems, application logs, and networks.

Also, we are different from these commercial digital forensic software companies that provide forensic tools, such as x-way, Encase, Magnet: we focus on formalizing, storing, and visualizing evidence in knowledge graphs. We utilize knowledge graphs to discover evidence, validate evidence, and reconstruct crime scenarios with traceability functions.

Frank Xu

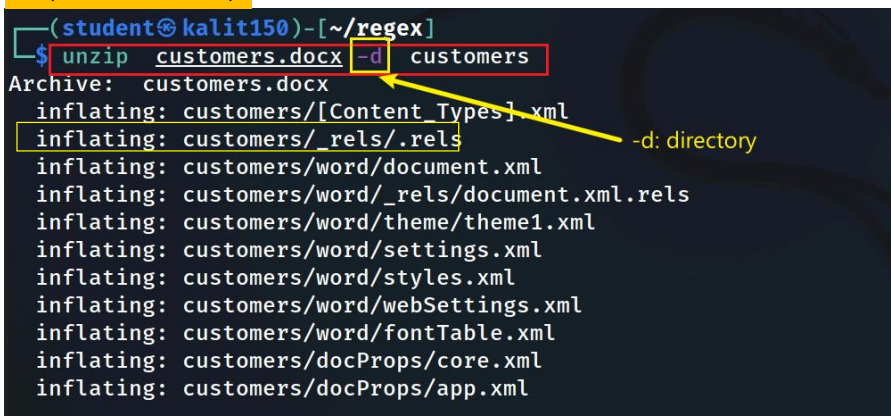
wxu@ubalt.edu

.docx is a compressed file

```
┌──(student㉿kalit150)-[~/regex]
└─$ unzip -l customers.docx
Archive:  customers.docx
  Length      Date    Time    Name
---------  ---------- -----   ----
     1312  1980-01-01 00:00   [Content_Types].xml
      590  1980-01-01 00:00   _rels/.rels
     8456  1980-01-01 00:00   word/document.xml
      817  1980-01-01 00:00   word/_rels/document.xml.rels
     8393  1980-01-01 00:00   word/theme/theme1.xml
     5081  1980-01-01 00:00   word/settings.xml
    29458  1980-01-01 00:00   word/styles.xml
      894  1980-01-01 00:00   word/webSettings.xml
     2215  1980-01-01 00:00   word/fontTable.xml
      748  1980-01-01 00:00   docProps/core.xml
      716  1980-01-01 00:00   docProps/app.xml
---------                     -------
    58680                     11 files
```

content

unzip .docx to a directory



The content of .docx is saved in a xml file.

```
┌──(student㉿kalit150)-[~/regex]
└─$ cat customers/word/document.xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<w:document xmlns:wpc="http://schemas.microsoft.com/office/word/2010/wordprocessi
s:cx="http://schemas.microsoft.com/office/drawing/2014/chartex" xmlns:cx1="http:/
soft.com/office/drawing/2015/9/8/chartex" xmlns:cx2="http://schemas.microsoft.com
g/2015/10/21/chartex" xmlns:cx3="http://schemas.microsoft.com/office/drawing/2016
xmlns:cx4="http://schemas.microsoft.com/office/drawing/2016/5/10/chartex" xmlns:c
emas.microsoft.com/office/drawing/2016/5/11/chartex" xmlns:cx6="http://schemas.mi
fice/drawing/2016/5/12/chartex" xmlns:cx7="http://schemas.microsoft.com/office/dr
3/chartex" xmlns:cx8="http://schemas.microsoft.com/office/drawing/2016/5/14/chart
http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:aink="http://s
ft.com/office/drawing/2016/ink" xmlns:am3d="http://schemas.microsoft.com/office/d
```

grep “ubalt.edu” but results show many Word format information

```
┌──(student㉿kalit150)-[~/regex]
└─$ grep -Ei "ubalt.edu" customers/word/document.xml
><w:t>w</w:t></w:r><w:r w:rsidR="00C44103"><w:t>xu</w:t></w:r><w:r w:rsidR="005A2AFF"><
:t></w:r><w:r w:rsidR="00C44103"><w:t>ubalt.edu</w:t></w:r></w:p><w:sectPr w:rsidR="004
w:pgSz w:w="12240" w:h="15840"/><w:pgMar w:top="1440" w:right="1440" w:bottom="1440" w:
40" w:header="720" w:footer="720" w:gutter="0"/><w:cols w:space="720"/><w:docGrid w:lin
360"/></w:sectPr></w:body></w:document>
```

grep -Ei "ubalt.edu" customers/word/document.xml

Remove xml tag using *sed*

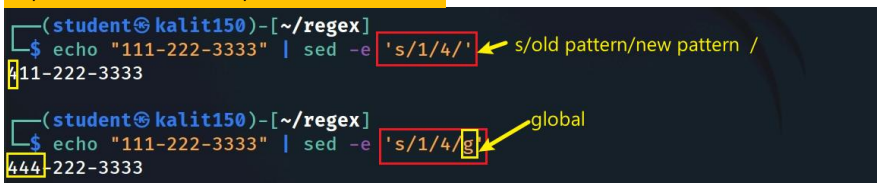
sed commands

```
┌──(student㉿kalit150)-[~/regex]
└─$ sed --help
Usage: sed [OPTION]... {script-only-if-no-other-script} [input-file]...

  -n, --quiet, --silent
                 suppress automatic printing of pattern space
      --debug
                 annotate program execution
  -e script, --expression=script
                 add the script to the commands to be executed
  -f script-file, --file=script-file
                 add the contents of script-file to the commands to be executed
  --follow-symlinks
                 follow symlinks when processing in place
  -i[SUFFIX], --in-place[=SUFFIX]
                 edit files in place (makes backup if SUFFIX supplied)
  -l N, --line-length=N
                 specify the desired line-wrap length for the `l' command
  --posix
                 disable all GNU extensions.
  -E, -r, --regexp-extended
                 use extended regular expressions in the script
                 (for portability use POSIX -E).
  -s, --separate
                 consider files as separate rather than as a single,
                 continuous long stream.
      --sandbox
                 operate in sandbox mode (disable e/r/w commands).
```

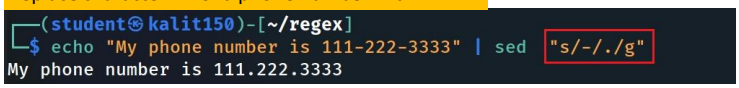
UNIVERSITY OF BALTIMORE TOWSON UNIVERSITY BJA

echo "111-222-3333" | sed -e 's/1/4/’
echo "111-222-3333" | sed -e 's/1/4/g'

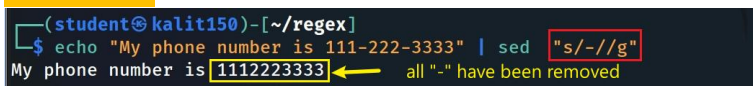
replace character “1” of a phone number with 4



replace character “-” of a phone number with “.”



Remove all “-”



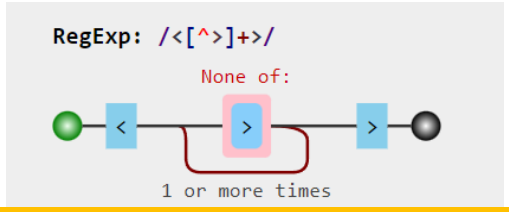
echo "111-222-3333" | sed -e 's/1/4/’
echo "111-222-3333" | sed -e 's/1/4/g'

<h1>This is a heading.</h1>

Remove (use lazy match) all html tags first failed attempt due to sed doesn't support -P

```
┌──(student㉿kalit150)-[~/regex]
└─$ echo  "Extract content from code. <h1>This is a heading. </h1>" | sed -P "s/<+?>//g"
```

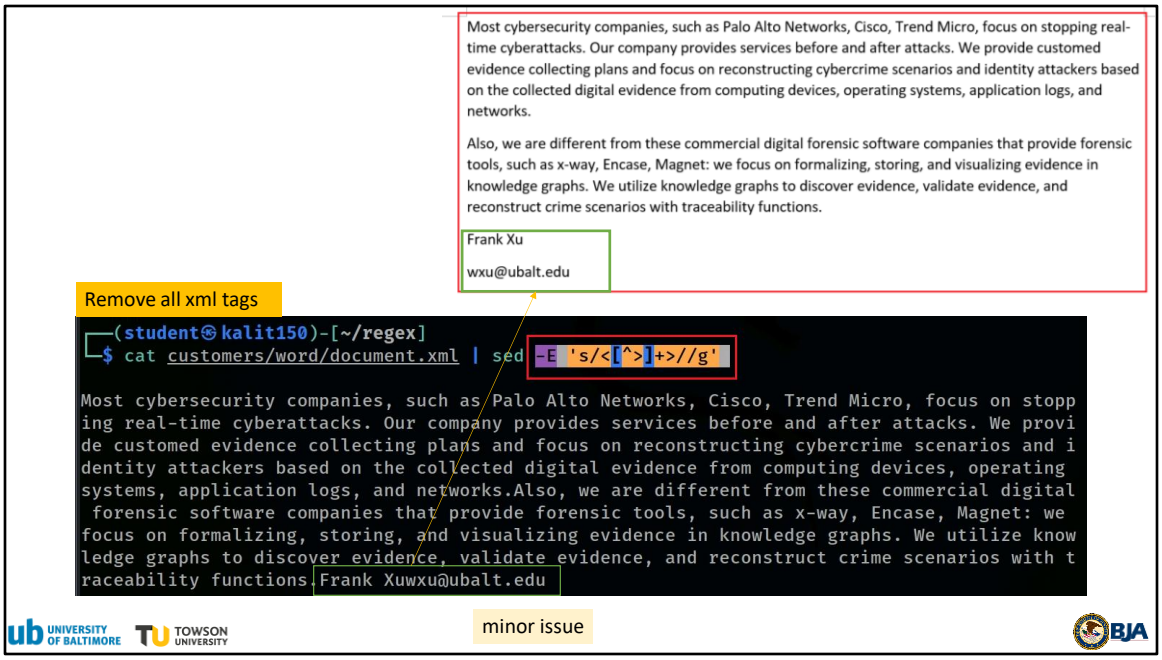
sed doesn't support -P



Remove all html tags using [^ not allowed character set]

```
┌──(student㉿kalit150)-[~/regex]
└─$ echo  "Extract content from code. <h1>This is a heading. </h1>" | sed -E "s/<[^>]+>//g"
Extract content from code. This is a heading.
```

echo "Extract content from code. <h1>This is a heading. </h1>" | sed -E "s/<[^>]+>//g“
https://jex.im/regulex/#!flags=&re=%3C%5B%5E%3E%5D%2B%3E



cat customers/word/document.xml | sed -E 's/<[^>]+>//g'

Replace paragraph (paraID) tag with spaces

```
┌──(student㉿kalit150)-[~/regex]
└─$ cat customers/word/document.xml | sed -E 's/<[^<]*paraId[^>]*>/ /g ; s/<[^>]+>//g'

 Most cybersecurity companies, such as Palo Alto Networks, Cisco, Trend Micro, focus on stop
ping real-time cyberattacks. Our company provides services before and after attacks. We prov
ide customed evidence collecting plans and focus on reconstructing cybercrime scenarios and
identity attackers based on the collected digital evidence from computing devices, operating
 systems, application logs, and networks. Also, we are different from these commercial digit
al forensic software companies that provide forensic tools, such as x-way, Encase, Magnet: w
e focus on formalizing, storing, and visualizing evidence in knowledge graphs. We utilize kn
owledge graphs to discover evidence, validate evidence, and reconstruct crime scenarios with
 traceability functions. Frank Xu wxu@ubalt.edu
```

grep emails

```
┌──(student㉿kalit150)-[~/regex]
└─$ sed -E 's/<[^<]*paraId[^>]*>/ /g ; s/<[^>]+>//g' customers/word/document.xml | grep -Eo
"[a-zA-Z0-9.-]{2,15}@[a-zA-Z0-9]{1,15}\.[a-zA-Z0-9]{3,4}"
wxu@ubalt.edu
```

cat customers/word/document.xml | sed -E 's/<[^<]*paraId[^>]*>/ /g ; s/<[^>]+>//g’
sed -E 's/<[^<]*paraId[^>]*>/ /g ; s/<[^>]+>//g' customers/word/document.xml
sed -E 's/<[^<]*paraId[^>]*>/ /g ; s/<[^>]+>//g' customers/word/document.xml |
grep -Eo "[a-zA-Z0-9.-]{2,15}@[a-zA-Z0-9]{1,15}\.[a-zA-Z0-9]{3,4}"

Show .docx content without unzip to disk

-p extract files to pipe, no messages

```
┌──(student㉿kalit150)-[~/regex]
└─$ unzip -p "*.docx" word/document.xml| sed -E 's/<[^<]*paraId[^>]*>/ /g ; s/<[^>]+>//g'

 Most cybersecurity companies, such as Palo Alto Networks, Cisco, Trend Micro, focus on stop
ping real-time cyberattacks. Our company provides services before and after attacks. We prov
ide customed evidence collecting plans and focus on reconstructing cybercrime scenarios and
identity attackers based on the collected digital evidence from computing devices, operating
 systems, application logs, and networks. Also, we are different from these commercial digit
al forensic software companies that provide forensic tools, such as x-way, Encase, Magnet: w
e focus on formalizing, storing, and visualizing evidence in knowledge graphs. We utilize kn
owledge graphs to discover evidence, validate evidence, and reconstruct crime scenarios with
 traceability functions. Frank Xu wxu@ubalt.edu
```

Show .docx props without unzip to disk

```
┌──(student㉿kalit150)-[~/regex]
└─$ unzip -p "*.docx" docProps/core.xml|  sed -E 's/<[^>]+>/ /g'

     Weifeng Xu      Weifeng Xu  53  2022-04-22T14:47:00Z  2022-04-23T01:42:00Z
```

unzip -p "*.docx" word/document.xml| sed -E 's/<[^<]*paraId[^>]*>/ /g ; s/<[^>]+>//g’

unzip -p "*.docx" docProps/core.xml| sed -E 's/<[^>]+>/ /g'

Summary

- *grep* is a powerful tool to extract digital forensic evidence
- *sed* is a stream editor
- *grep/sed* use regular expression (regex/pattern) to match text
- Key regex operations
  - literal string: cat, character classes: [], [^], or: a|b, group: ()
  - quantification: ?, *,+, {}
  - scope: \b, \< \>, \w
  - greedy vs. lazy: +?, *?, {}?
  - back reference: \1, \2, …,\n
  - lookahead and lookbehind: (?=), (?<=)
- Need both positive tests and negative tests

UNIVERSITY OF BALTIMORE TOWSON UNIVERSITY BJA

address: \\d+[ ](?:[A-Za-z0-9.-]+[
]?)+(?:Avenue|Lane|Road|Boulevard|Drive|Street|Ave|Dr|Rd|Blvd|Ln|St)\\.?

https://staff.washington.edu/weller/grep.html

GREP cheat sheet

characters — what to seek

ring matches **ring**, sp**ring**board, **ring**tone, etc.
. matches almost any character
h.o matches **hoo**, **h2o**, **h/o**, etc.
Use **** to search for these special characters:
[\ ^ $. | ? * + ( ) { }
ring**\?** matches **ring?**
\(quiet**\)** matches **(quiet)**
c:****windows matches **c:\windows**

alternatives — | (OR)

cat|dog match **cat** or **dog**
order matters if short alternative is part of longer
id|identity matches **id** or **id**entity
regex engine is "eager", stops comparing
as soon as 1st alternative matches
identity|id matches **id** or **identity**
order longer to shorter when alternatives overlap
(To match whole words, see scope and groups.)

character classes — [allowed] or [^NOT allowed]

[aeiou] match any vowel
[^aeiou] match a NON vowel
r**[iau]**ng match **ring**, wr**ang**le, sp**rung**, etc.
gr**[ae]**y match **gray** or **grey**
[a-zA-Z0-9] match any letter or digit
(In **[ ]** always escape **. \ ]** and sometimes **^ -** .)

shorthand classes

\w "word" character (letter, digit, or underscore)
\d digit
\s whitespace (space, tab, vtab, newline)
\W, **\D**, or **\S**, (NOT word, digit, or whitespace)
[\D\S] means not digit OR whitespace, both match
[^\d\s] disallow digit AND whitespace

occurrences — ? * + {n} {n,} {n,n}

? 0 or 1
colou?r match **color** or **colour**
***** 0 or more
[BW]ill**[ieamy's]*** match **Bill**, **Willy**, **William's** etc.
+ 1 or more
[a-zA-Z]+ match 1 or more letters
{n} require n occurrences
\d{3}-\d{2}-\d{4} match a SSN
{n,} require n or more
[a-zA-Z]{2,} 2 or more letters
{n,m} require n - m
[a-z]\w{1,7} match a UW NetID

*** greedy versus *? lazy**

*** +** and **{n,}** are greedy — match as much as possible
<.+> finds 1 big match in **bold**
***? +?** and **{n,}?** are lazy — match as little as possible
<.+?> finds 2 matches in ****bold****

https://staff.washington.edu/weller/grep.html

scope — \b \B ^ $

\b "word" edge (next to non "word" character)
\bring word starts with "ring", ex **ringtone**
ring\b word ends with "ring", ex **spring**
\b9\b match single digit **9**, not 19, 91, 99, etc..
\b[a-zA-Z]{6}\b match 6-letter words
\B NOT word edge
\Bring\B match **springs** and **wringer**
^ start of string $ end of string
^\d*$ entire string must be digits
^[a-zA-Z]{4,20}$ string must have 4-20 letters
^[A-Z] string must begin with capital letter
[\.!?"')]$ string must end with terminal puncutation

groups — ( )

(in|out)put match **input** or **output**
\d{5}(-\d{4})? US zip code ("+ 4" optional)
Locate all PHP input variables:
\$_(GET|POST|REQUEST|COOKIE|SESSION|SERVER)\[.+\]
NB: parser tries EACH alternative if match fails after group.
Can lead to catastrophic backtracking.

back references — \n

each ( ) creates a numbered "back reference"
(to) (be) or not \1 \2 match **to be or not to be**
([^\s])\1{2} match non-space, then same twice more **aaa**, **...**
\b(\w+)\s+\1\b match doubled words

back references — \n

each ( ) creates a numbered "back reference"
(to) (be) or not \1 \2 match **to be or not to be**
([^\s])\1{2} match non-space, then same twice more **aaa**, **...**
\b(\w+)\s+\1\b match doubled words

non-capturing group — (?:) prevent back reference

on(?:click|load) is faster than on(click|load)
use non-capturing or atomic groups when possible

atomic groups — (?>a|b) (no capture, no backtrack)

(?>red|green|blue)
faster than non-capturing
alternatives parsed left to right without return
(?>id|identity)\b matches **id**, but not **id**entity
"id" matches, but "\b" fails after atomic group,
parser doesn't backtrack into group to retry 'identity'
If alternatives overlap, order longer to shorter.

lookahead — (?=) (?!) lookbehind — (?<=) (?<!)

\b\w+?(?=ing\b) match **warbl**ing, **str**ing, **fish**ing, ...
\b(?!\w+ing\b)\w+\b words NOT ending in "ing"
(?<=\bpre).*?\b match pre**tend**, pre**sent**, pre**fix**, ...
\b\w{3}(?<!pre)\w*?\b words NOT starting with "pre"
(lookbehind needs 3 chars, \w{3}, to compare w/"pre")
\b\w+(?<!ing)\b match words NOT ending in "ing"
(see LOOKAROUND notes below)