




Further on Bash!

Day11_proBash.md



Recall

LAST TIME TOPIC

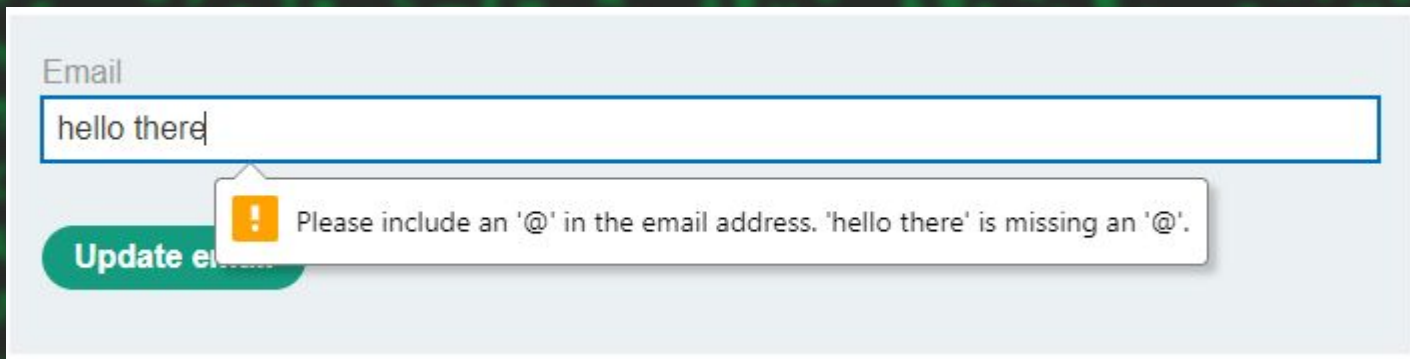


Today's Topics

- Regular expressions
- Else if
- loops
- functions
- bash and linux shell

Regular Expressions! /regex/

How do this site know that our input is not email?



The screenshot shows a web form for email registration. At the top, the label "Email" is positioned above a text input field. The input field contains the text "hello there". Below the input field, there is a green button with the text "Update email". A white error message box with a yellow exclamation mark icon is displayed below the input field. The message reads: "Please include an '@' in the email address. 'hello there' is missing an '@'." The entire form is set against a light blue background.

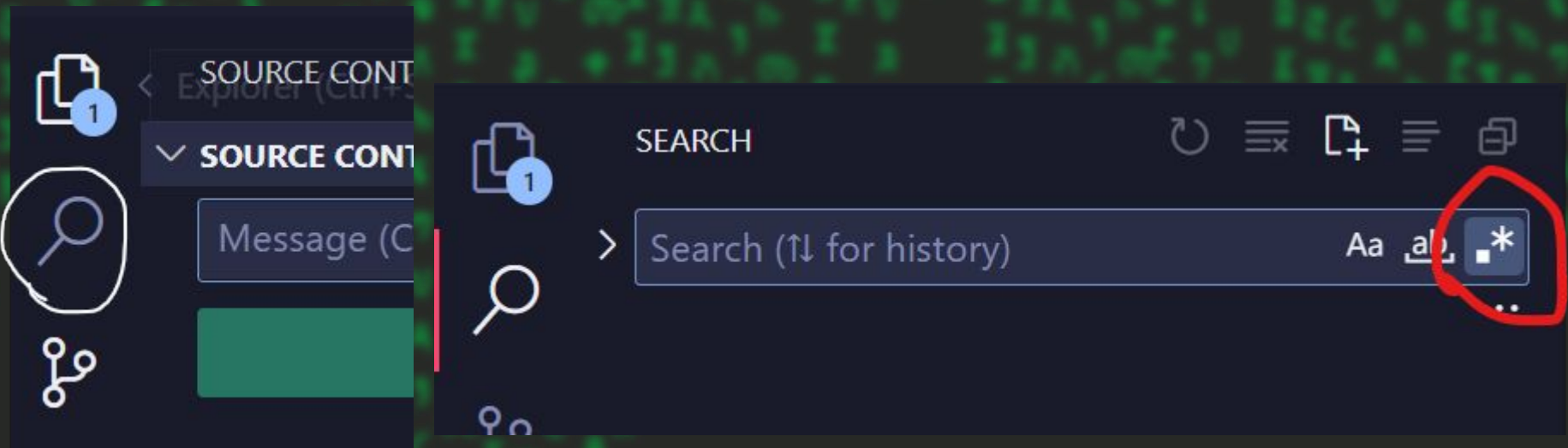


Cont...

- Most filter Validation on any platform done by Regular expression/regex/.
- They are patterns that helps to filter so texts,space,tabs & symbols.
- Like telegram or other platforms filtering links inside group, filtering some bad words.. All are regex.
- Regex is PATTERN!
- Regex are used on linux tools called grep,awk and sed

regex...

- To demonstrate this we can simply use vscode search tool.
- And don't forget to turn the regex button on





regex...

- The pattern is same but the implementation may differ on programming languages.
- On Python,

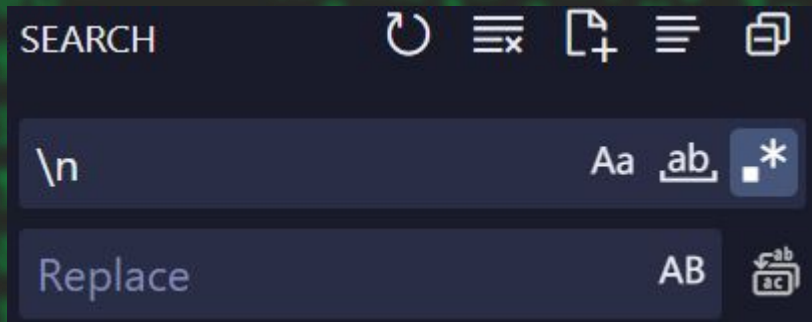
```
import re

a = re.search("PATTERN","SEARCHING FILE").group(0)

# Regex on Python
print(a)
```

Quick Test!

What do u think the output of this



```
1 hello world
2
3 natanhailu@gmail.com
4 geeztech@gmail.com
5
```

```
.....\n
.....\t ..... \n
.....
```

```
1 hello world
2
3 natanhailu@gmail.com
4 geeztech@gmail.com
5
```

SO, `\n` is a pattern, it is called metacharacter



Metacharacters

- Those are regex pattern symbols for filter.
- They are :
 - .
 - ^
 - \$
 - *
 - +
 - ?
 - {}
 - []
 - ()
 - |
 - \

Dot (.)

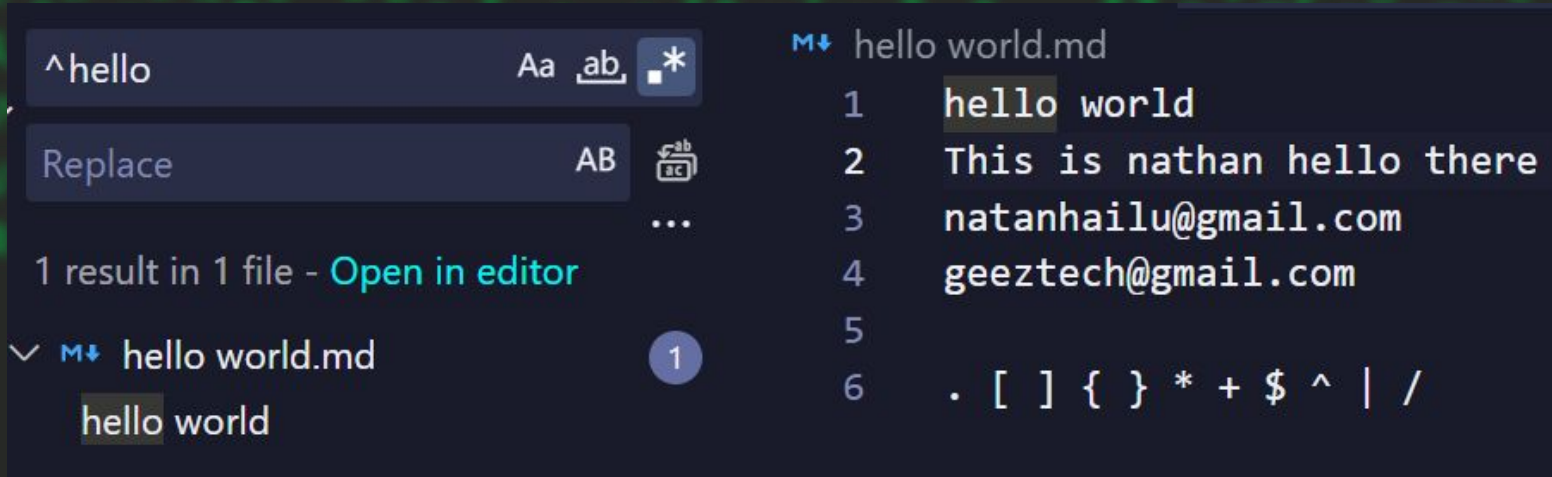
- Used to get All the line except new lines
- Syntax: .
 - This means give me all lines except the new lines



The screenshot shows the Visual Studio Code interface. On the left, the Search sidebar is open, displaying results for the search term `.ab`. It indicates '90 results in 1 file' and provides a link to 'Open in editor'. Below this, a file named 'hello world.md' is expanded, showing a single line of code: `hello world`. On the right, the editor window displays the content of 'hello world.md', which consists of five lines of text: `hello world`, `This is nathan hello`, `natanhailu@gmail.com`, `geeztech@gmail.com`, and a blank line. At the bottom of the editor, a status bar shows a list of regex characters: `. [] { } * + $ ^ | /`.

Caret (^) - Assertion

- Used to get line that start with pattern
- Syntax: ^hello
 - This means lines that start with hello

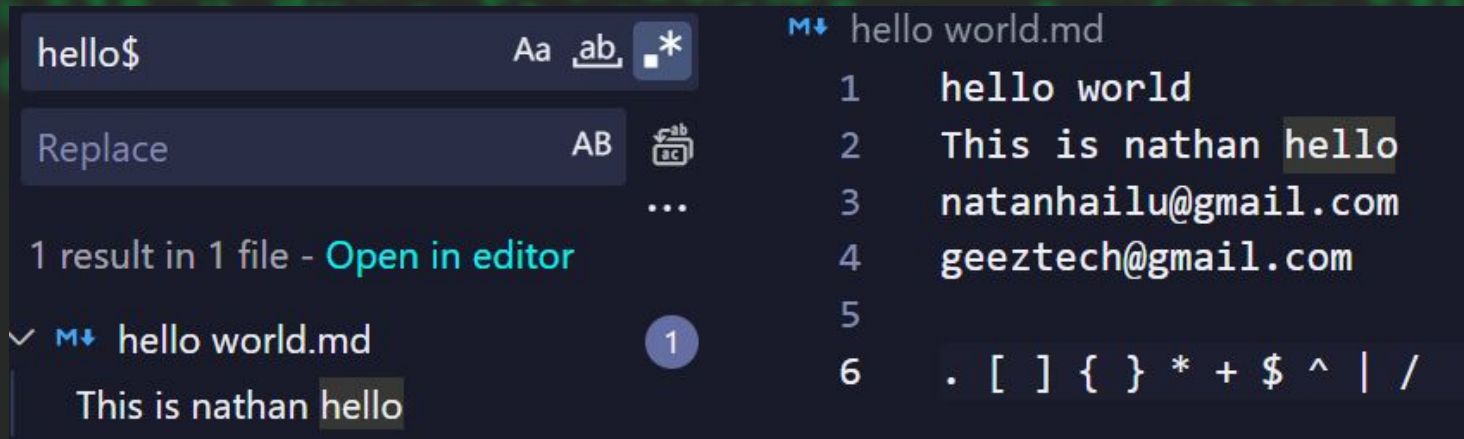


The screenshot shows the VS Code search interface. The search bar contains the pattern `^hello`. The results pane shows one match in the file `hello world.md` at line 1, with the text `hello world` highlighted. The editor pane shows the content of `hello world.md`, which is:

```
1 hello world
2 This is nathan hello there
3 natanhailu@gmail.com
4 geeztech@gmail.com
5
6 . [ ] { } * + $ ^ | /
```


Dollar sign(\$) - Assertion

- Used to get line that ends with some pattern.
- Syntax: hello\$
 - That ends with hello



The screenshot shows a code editor with a search bar at the top containing 'hello\$'. Below the search bar, a 'Replace' button is visible. The search results show '1 result in 1 file - Open in editor'. The file 'hello world.md' is selected, and the search results show 'This is nathan hello' with 'hello' highlighted. The code editor also shows the content of 'hello world.md' with line numbers 1 through 6. Line 2 contains 'This is nathan hello' with 'hello' highlighted. Line 6 contains a list of symbols: . [] { } * + \$ ^ | /.

```
hello$ Aa ab *
```

```
Replace AB ab
```

```
1 result in 1 file - Open in editor
```

```
✓ M↓ hello world.md 1
```

```
This is nathan hello
```

```
M↓ hello world.md
```

```
1 hello world
```

```
2 This is nathan hello
```

```
3 natanhailu@gmail.com
```

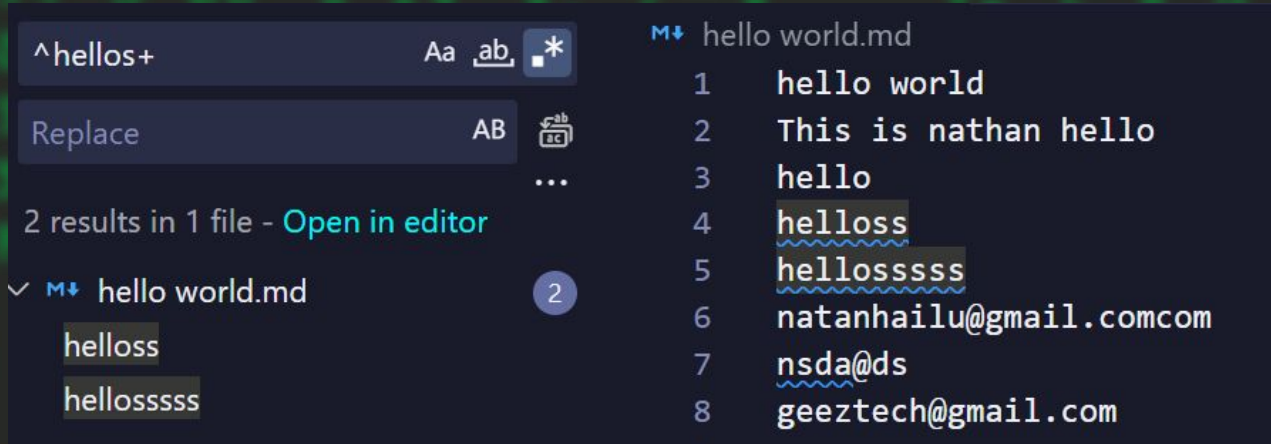
```
4 geeztech@gmail.com
```

```
5
```

```
6 . [ ] { } * + $ ^ | /
```


Plus (+) - Quantity

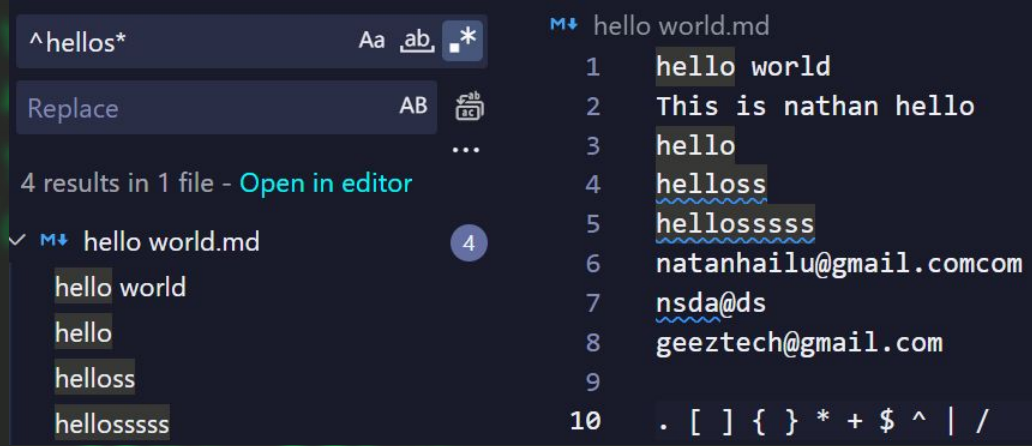
- Used to get line that have pattern that occurs 1 and more times.
- Syntax: `hellos+`
 - A text hello that have s at least 1 times and more.




The screenshot shows a code editor interface with a search bar at the top left containing the regex pattern `^hellos+`. Below the search bar, the text "2 results in 1 file - [Open in editor](#)" is displayed. The search results are shown in a list on the left, with the first result selected: `✓ M+ hello world.md` with a count of 2. The main editor area shows the content of `hello world.md` with line numbers 1 through 8. The lines are: 1 `hello world`, 2 `This is nathan hello`, 3 `hello`, 4 `helloss`, 5 `hellosssss`, 6 `natanhailu@gmail.comcom`, 7 `nsda@ds`, and 8 `geeztech@gmail.com`. The words `helloss` and `hellosssss` on lines 4 and 5 are highlighted with a blue background and a red wavy underline, indicating they are the matches for the search pattern.

Asteriks (*) - Quantity

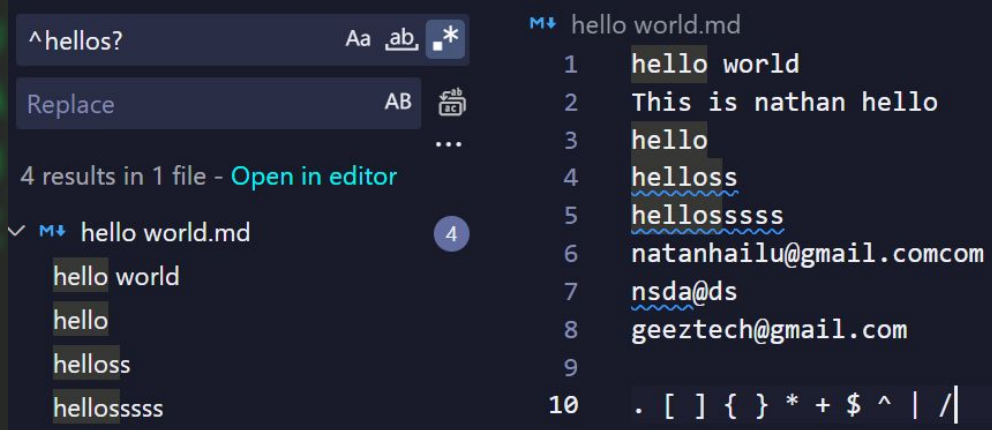
- Used to get line that have pattern that occurs 0 and more times.
- Syntax: `hellos*`
 - A text hello that have s at least 0 times and more.



```
^hellos*  Aa ab *  
Replace  AB    
4 results in 1 file - Open in editor  
✓ M+ hello world.md 4  
hello world  
hello  
helloss  
hellosssss  
M+ hello world.md  
1 hello world  
2 This is nathan hello  
3 hello  
4 helloss  
5 hellosssss  
6 natanhailu@gmail.comcom  
7 nsda@ds  
8 geeztech@gmail.com  
9  
10 . [ ] { } * + $ ^ | /
```

Question mark (?) - Quantity

- Used to get line that have pattern that occurs 0 and 1 times.
- Syntax: hellos?
 - A text hello that have s at least 0 time or 1 time.

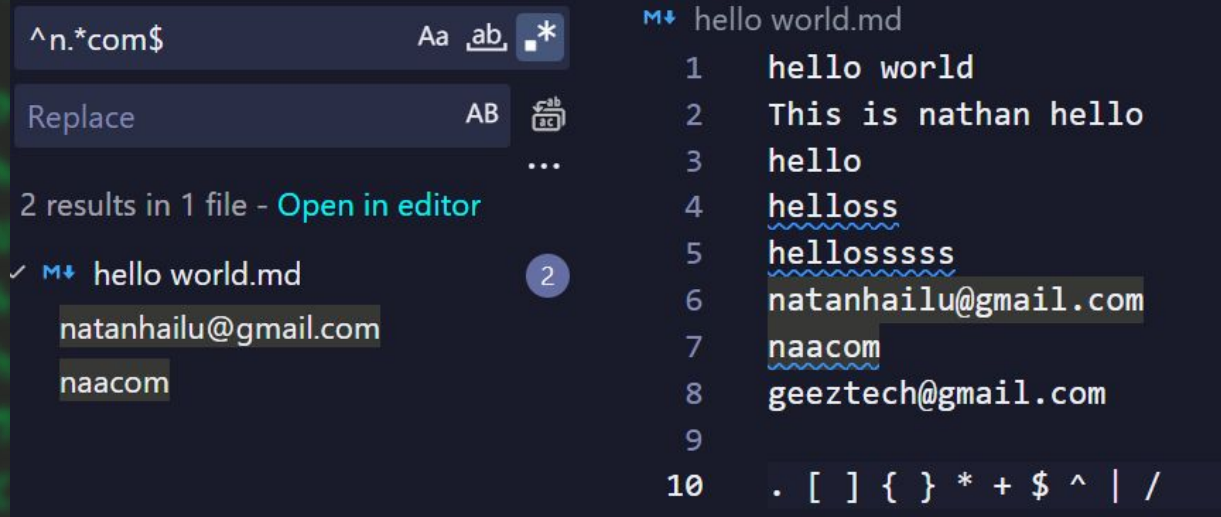


```
^hellos? Aa .ab.*  
Replace AB  
4 results in 1 file - Open in editor  
✓ M+ hello world.md 4  
hello world  
hello  
helloss  
hellosssss  
M+ hello world.md  
1 hello world  
2 This is nathan hello  
3 hello  
4 helloss  
5 hellosssss  
6 natanhailu@gmail.comcom  
7 nsda@ds  
8 geeztech@gmail.com  
9  
10 . [ ] { } * + $ ^ | /|
```


What if...

We need to get texts that starts with n and ends with com including all the texts between those 2 expressions...

- Start with n = ^n
- End with com = com\$
- All Test between them = .*,.+
- We can do it in 1 line
- ^n.*com\$



```
^n.*com$ Aa ab *
```

```
Replace AB
```

```
2 results in 1 file - Open in editor
```

```
✓ M+ hello world.md 2
```

```
natanhailu@gmail.com
```

```
naacom
```

```
M+ hello world.md
```

```
1 hello world
```

```
2 This is nathan hello
```

```
3 hello
```

```
4 helloss
```

```
5 hellosssss
```

```
6 natanhailu@gmail.com
```

```
7 naacom
```

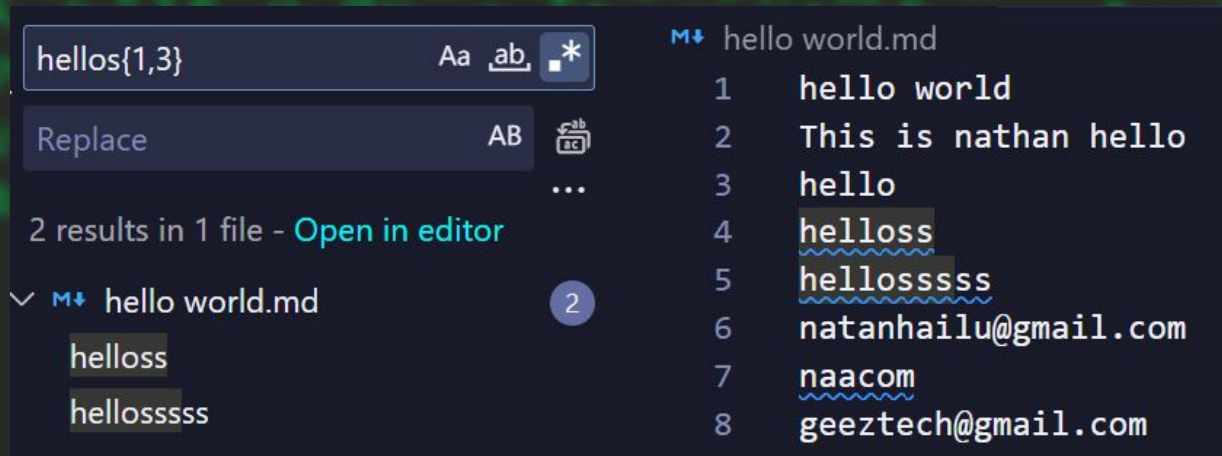
```
8 geeztech@gmail.com
```

```
9
```

```
10 . [ ] { } * + $ ^ | /
```


Curly Bracket ({ min , max }) - Quantity

- Used to get line that have pattern that occurs min and max times.it is custom
- Syntax: `hellos{1,3}`
 - A text hello that have s at least 0 times and more.



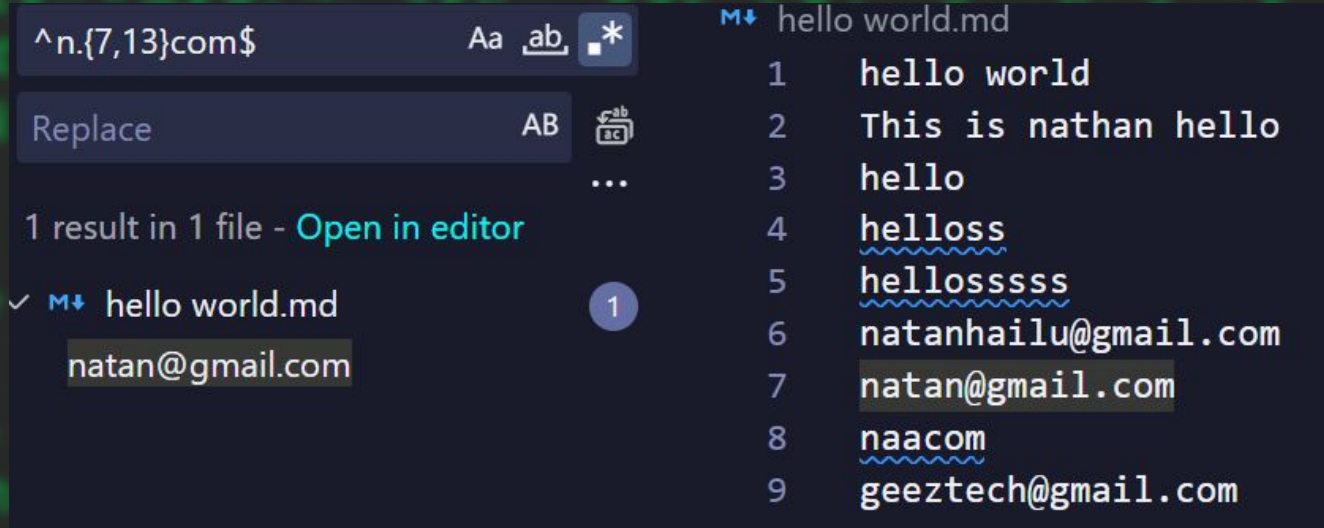
The screenshot shows a code editor with a search bar at the top containing the regex `hellos{1,3}`. Below the search bar, there are options for case sensitivity (Aa), whole word matching (ab), and regex mode (*). The search results show 2 results in 1 file, with a link to "Open in editor". The file "hello world.md" is open, and the search results are displayed as a list of matches: "helloss" and "hellosssss".

{1,} = plus sign
{0,} = asterisk
{0,1} = what..

What if...

We need to get texts that starts with n and ends with com including the texts that have 7-13 character between those 2 expressions...

- Start with n = ^n
- End with com = com\$
- 7-10 character = {7,13}
-
- We can do it in 1 line
- ^n.{7,13}com\$



```
^n.{7,13}com$ Aa ab *
```

```
Replace AB ab
```

```
1 result in 1 file - Open in editor
```

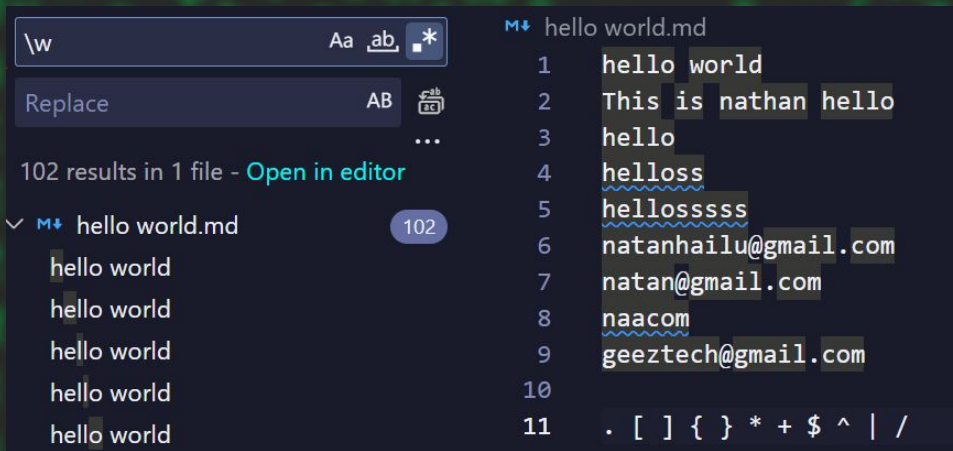
```
✓ M+ hello world.md 1  
natan@gmail.com
```

```
M+ hello world.md  
1 hello world  
2 This is nathan hello  
3 hello  
4 helloss  
5 hellosssss  
6 natanhailu@gmail.com  
7 natan@gmail.com  
8 naacom  
9 geeztech@gmail.com
```



\w

- Used to get Alphanumeric
- Syntax: \w
 - All texts except newlines and symbols



The screenshot shows the VS Code interface. On the left, the search panel is open with the search term `\w`. It shows 102 results in 1 file, `hello world.md`. The search results list five instances of the text `hello world`. On the right, the editor shows the content of `hello world.md`. The file contains 11 lines of text. Lines 1-5 are `hello world`, `This is nathan hello`, `hello`, `helloss`, and `hellosssss` respectively. Lines 6-9 are email addresses: `natanhailu@gmail.com`, `natan@gmail.com`, `naacom`, and `geeztech@gmail.com`. Line 10 is empty, and line 11 contains a list of symbols: `. [] { } * + $ ^ | /`. The search results in the left panel are highlighted in the editor, showing that the search term `\w` matches alphanumeric characters and underscores.

```
\w
```

Replace AB ...

102 results in 1 file - [Open in editor](#)

✓ M+ hello world.md 102

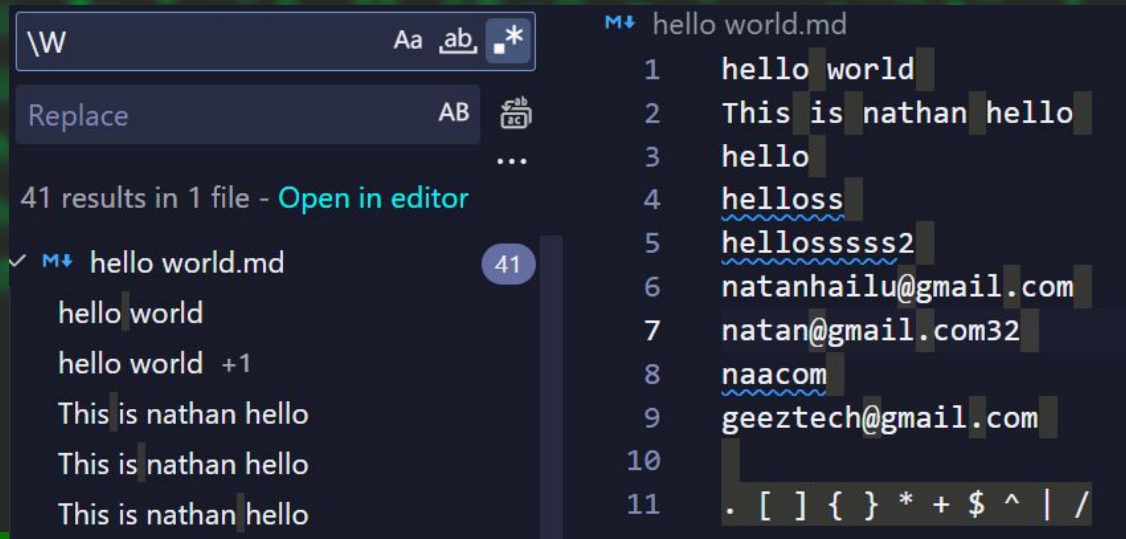
- hello world
- hello world
- hello world
- hello world
- hello world

```
M+ hello world.md
1 hello world
2 This is nathan hello
3 hello
4 helloss
5 hellosssss
6 natanhailu@gmail.com
7 natan@gmail.com
8 naacom
9 geeztech@gmail.com
10
11 . [ ] { } * + $ ^ | /
```




\W

- Used to get All except Alphanumeric
- Syntax: \W



```
\W
```

Replace AB

41 results in 1 file - [Open in editor](#)

✓ M+ hello world.md 41

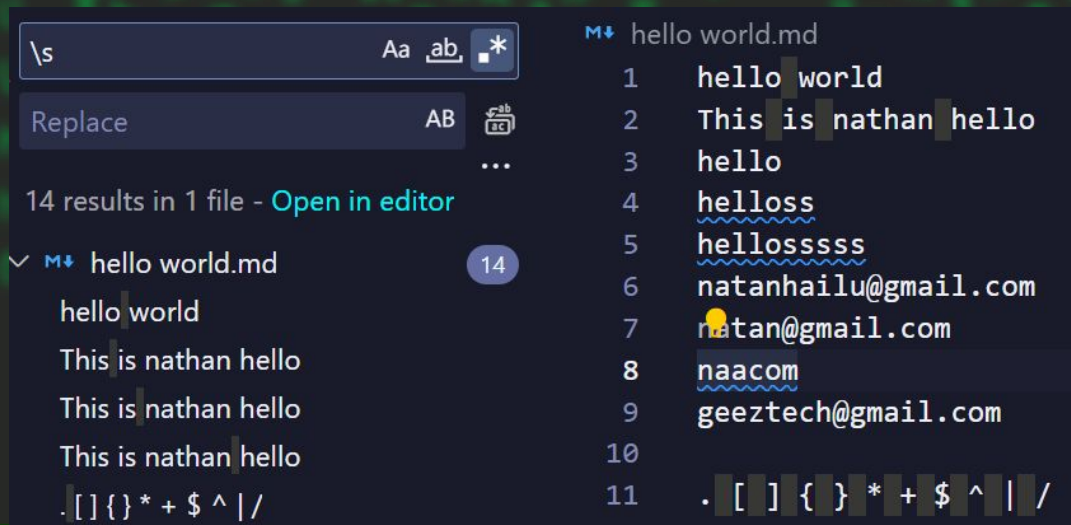
- hello world
- hello world +1
- This is nathan hello
- This is nathan hello
- This is nathan hello

```
M+ hello world.md
1 hello world
2 This is nathan hello
3 hello
4 helloss
5 hellosssss2
6 natanhailu@gmail.com
7 natan@gmail.com32
8 naacom
9 geeztech@gmail.com
10
11 . [ ] { } * + $ ^ | /
```




\s

- Used to get whitespace.
- Syntax: \s



```
\s
```

Replace AB

14 results in 1 file - [Open in editor](#)

✓ M+ hello world.md 14

```
hello world
This is nathan hello
This is nathan hello
This is nathan hello
.[[]{}]*+${^|/}
```

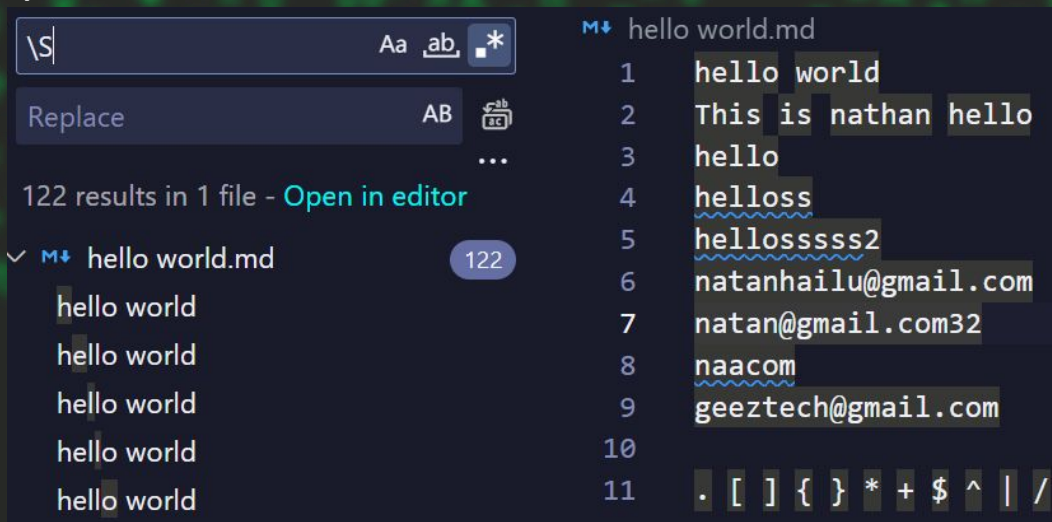
hello world.md

```
1 hello world
2 This is nathan hello
3 hello
4 helloss
5 hellosssss
6 natanhailu@gmail.com
7 natan@gmail.com
8 naacom
9 geeztech@gmail.com
10
11 .[[]{}]*+${^|/}
```



\S

- Used to get all except whitespace.
- Syntax: \S

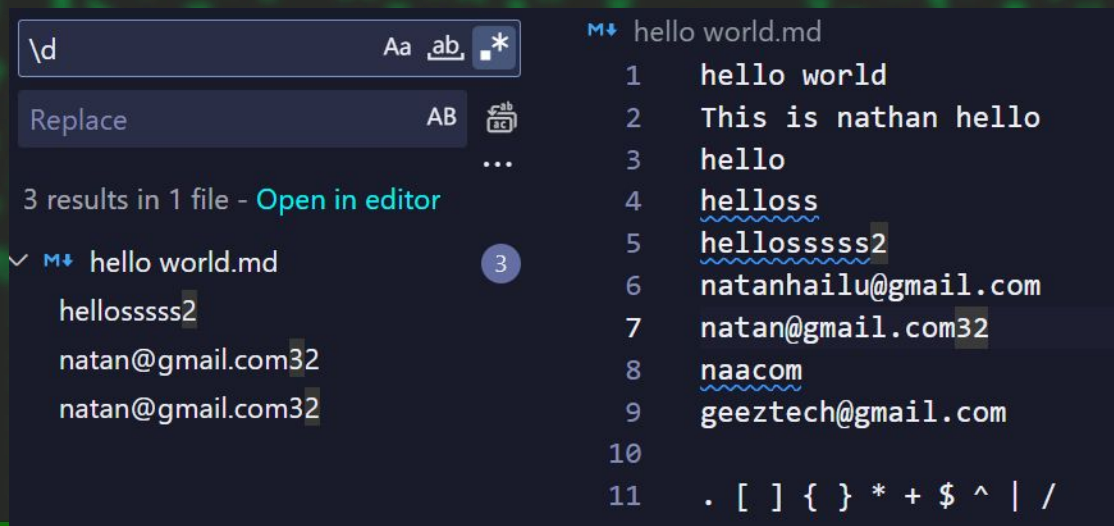


The screenshot shows a code editor interface. On the left, a search panel displays the search term `\S` with options for case sensitivity (Aa), whole words (ab), and regex (a star icon). Below the search bar, it says '122 results in 1 file - Open in editor'. A list of results shows the file 'hello world.md' with 122 matches. The main editor area shows the content of 'hello world.md' with line numbers 1 through 11. The text in the editor is:
1 hello world
2 This is nathan hello
3 hello
4 helloss
5 hellosssss2
6 natanhailu@gmail.com
7 natan@gmail.com32
8 naacom
9 geeztech@gmail.com
10
11 . [] { } * + \$ ^ | /



\d

- Used to get Digits/numbers/
- Syntax: \d



```
\d Aa ab *
```

Replace AB

3 results in 1 file - [Open in editor](#)

✓ M+ hello world.md 3

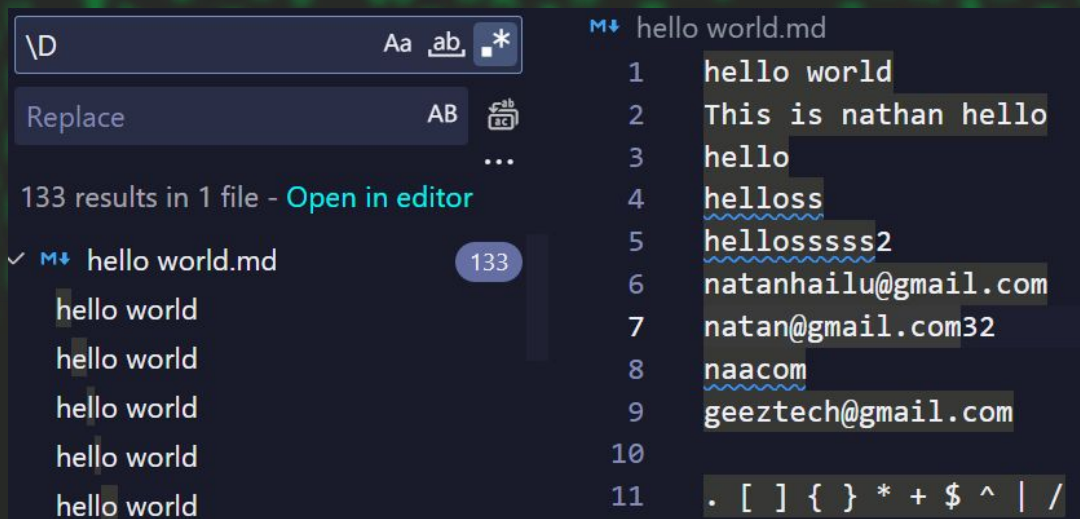
- hellossss2
- natan@gmail.com32
- natan@gmail.com32

```
M+ hello world.md
1 hello world
2 This is nathan hello
3 hello
4 helloss
5 hellossss2
6 natanhailu@gmail.com
7 natan@gmail.com32
8 naacom
9 geeztech@gmail.com
10
11 . [ ] { } * + $ ^ | /
```




\D

- Used to get all except Digits/numbers/
- Syntax: \D

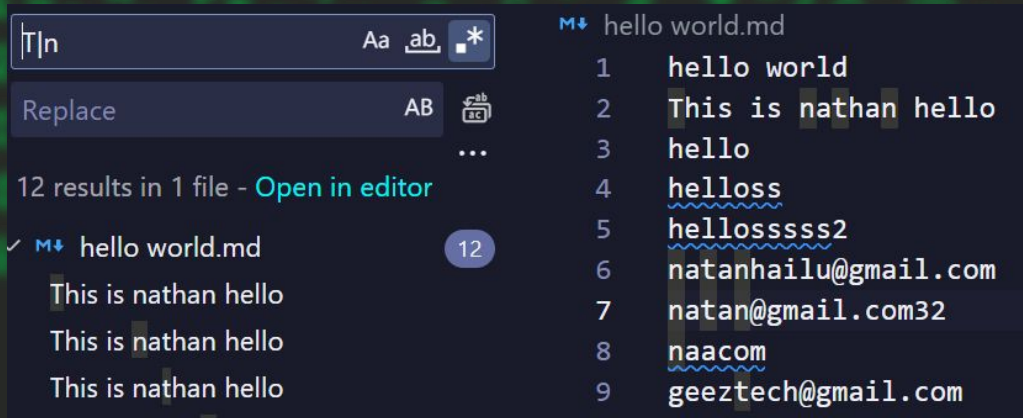


The screenshot shows a text editor interface. On the left, a search bar contains the regex `\D`. Below it, a 'Replace' button is visible. A notification indicates '133 results in 1 file - Open in editor'. A list of results shows the file 'hello world.md' with 133 matches. On the right, the content of 'hello world.md' is displayed, showing a list of lines where non-digit characters are highlighted. The lines are:

```
1 hello world
2 This is nathan hello
3 hello
4 helloss
5 hellosssss2
6 natanhailu@gmail.com
7 natan@gmail.com32
8 naacom
9 geeztech@gmail.com
10
11 . [ ] { } * + $ ^ | /
```


Pipe (|) - OR

- Used to search 2 different things.
- Syntax: a|b



The screenshot shows a code editor interface with a search bar at the top left containing the text `n|n`. Below the search bar, there are buttons for 'Replace', 'AB', and a search icon. The search results are displayed in a list on the left, showing 12 results in 1 file. The file is named 'hello world.md' and contains the following text:

```
1 hello world
2 This is nathan hello
3 hello
4 helloss
5 hellosssss2
6 natanhailu@gmail.com
7 natan@gmail.com32
8 naacom
9 geeztech@gmail.com
```

Cont...

^T Aa ab *

Replace AB

1 result in 1 file - [Open in editor](#)

✓ M+ hello world.md 1

This is nathan hello

hello world.md

```
1 hello world
2 This is nathan hello
3 hello
4 helloss
5 hellosssss2
6 natanhailu@gmail.com
7 natan@gmail.com32
```

^T.*|^n.* Aa ab *

Replace AB

4 results in 1 file - [Open in editor](#)

✓ M+ hello world.md 4

This is nathan hello
natanhailu@gmail.com
natan@gmail.com32
naacom

hello world.md

```
1 hello world
2 This is nathan hello
3 hello
4 helloss
5 hellosssss2
6 natanhailu@gmail.com
7 natan@gmail.com32
8 naacom
9 geeztech@gmail.com
10
```

^T.* Aa ab *

Replace AB

1 result in 1 file - [Open in editor](#)

✓ M+ hello world.md 1

hello world.md

```
1 hello world
2 This is nathan hello
3 hello
4 helloss
5 hellosssss2
```

-

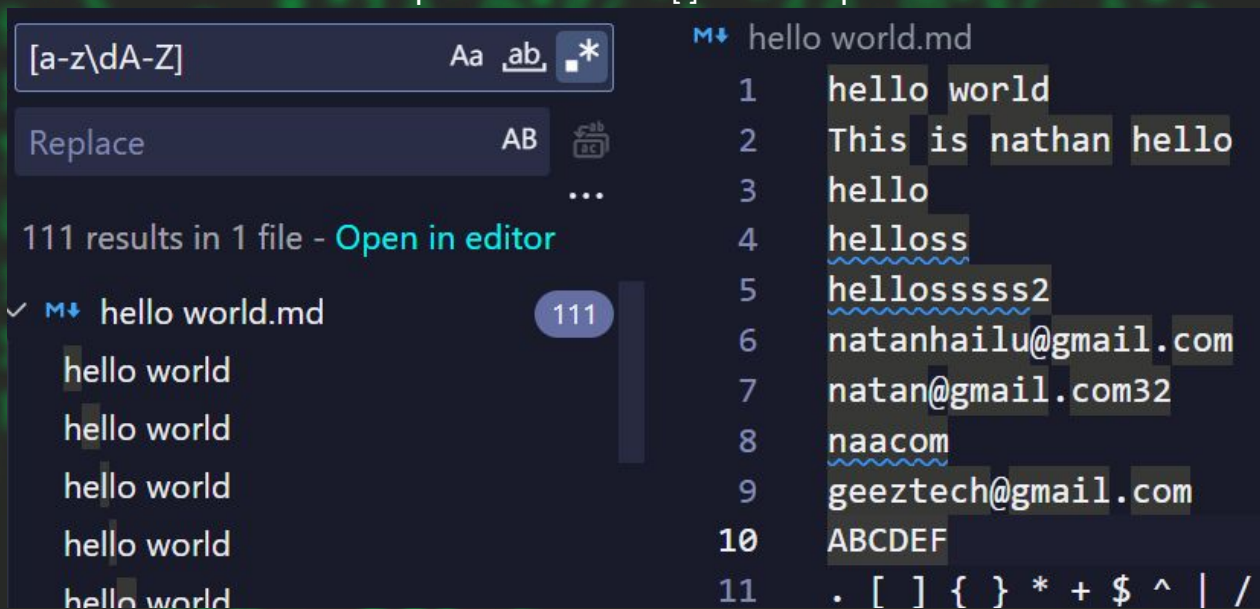
- Used to Create your own patterns
- Syntax: [pattern]



You cant get
small letters
with the `\w` or
built in
patterns.

Cont...

- You can add other patterns in same [] with no space between them.



The screenshot displays a code editor interface with a search and replace panel on the left and the file content on the right.

Search and Replace Panel:

- Search pattern: `[a-z\dA-Z]*`
- Replace with: `AB`
- Results: 111 results in 1 file - [Open in editor](#)

File Content (hello world.md):

```
1 hello world
2 This is nathan hello
3 hello
4 helloss
5 hellosssss2
6 natanhailu@gmail.com
7 natan@gmail.com32
8 naacom
9 geeztech@gmail.com
10 ABCDEF
11 . [ ] { } * + $ ^ | /
```



Exercise 1

10min

1. Demlachewu is A software engineer and wanted to make a telegram bot that can delete links like https and ends with .com from the group, can u help him by writing the regex?
2. Write a Regex That Filters the emails only from the following list

john@gmail.com

natanhailu@yahoo.com

micky43@geeztecg.net

rexder@gmail.com

Hello there this is me

My phone number 0987654321

Bash regex

- You can use it on awk, sed, but for today i will show you using grep.

```
GNU nano 6.2          testingREGEX.txt
hello world
This is nathan hello
hello
helloss
hellosssss2
natanhailu@gmail.com
natan@gmail.com32
s@
naacom
geeztech@gmail.com
ABCDEF
. [ ] { } * + $ ^ | /
```

On bash terminal these metacharacters have meaning so, we have to escape them by adding '\ ' in front of the metacharacters

```
# Some Times you have to add -Eo for Good result on Regex on Grep
grep -Eo "YourPattern"
```

```
rexder@HunterMachine ~> cat testingREGEX.txt | grep "^.*@[a-z]\$"
natanhailu@gmail.com
geeztech@gmail.com
```

BASH FOR REGEX

- We use `=~` operator for regex check with if condition statements
- Here we use double Brackets for our conditional Statements.
- SYNTAX:
-

```
pattern="YourRegex"  
if [[ $input =~ ${pattern} ]]
```

```
/bin/bash bashregex.sh  
Enter your number:23  
Good Number
```







```
/bin/bash bashregex.sh  
Enter your number:das  
Please Enter Number only
```

```
#!/bin/bash  
read -p "Enter your number:" num  
  
pattern="[0-9]"  
  
if [[ $num =~ ${pattern} ]]  
then  
echo "Good Number"  
else  
echo "Please Enter Number only"  
fi
```

Task

10min

Write A bash code that can accept emails from user name and check if it is Valid email or not, using regex

```
   /bin/bash bashregex.sh  
Enter your email:natan  
Invalid Email, Try again later!  
   /bin/bash bashregex.sh  
Enter your email:natan@gmail.com  
It is A Valid Email!
```




Bash else if

- To do more than 1 comparing.
- It is same with python is "elif"
- Syntax:

```
if condition
then
    body
elif condition
then
    body
else
    body
fi
```

```
a=23
if [ $a -gt 23 ]
then
echo "he"
elif [ $a -lt 20 ]
then
echo "how"
elif [ $a -eq 23 ]
then
echo "You are Correct!"
else
echo "bye"
fi
```

```
rexder@HunterMachine ~> bash hello.sh
You are Correct!
```



Loops

- On Bash, there are 3 types of loops
 - a. For loop
 - b. While loop
 - c. Until loop

For loops

- The iteration may be different here. We can use arrays like list on python but we dont have range in bash but we can do {a..b} this iterates from a to b
- Syntax:

```
for condition
do
    body
done
```

```
GNU nano 6.2
for num in {1..10}
do
echo $num
done
```

```
rexder@HunterMachine ~> bash hello.sh
1
2
3
4
5
6
7
8
9
10
```


Cont...

Works with words in strings too.

```
GNU nano 6.2
arrays=("element1" "element2" "elementN")

for i in ${arrays[@]}
do
echo $i
done
```

```
rexder@HunterMachine ~> bash hello.sh
element1
element2
elementN
```

```
GNU nano 6.2
text="Hello GTST this is Day11."

for i in $text
do
echo $i
done
```

```
rexder@HunterMachine ~> bash hello.sh
Hello
GTST
this
is
Day11.
```

For loop without sequence data

```
for ((i=1; i<=10; i++))  
    #START #END  #INCREMENT  
do  
echo $i  
done
```

```
rexder@HunterMachine ~> bash hello.sh  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

On bash

$i++ = i += 1$

$i-- = i -= 1$

For Loop with increment/decrement

Same syntax with index slicing but here it is for looping. - {start..stop..step}

```
for num in {1..10..1}
do
echo $num
done
```

```
for num in {1..10..2}
do
echo $num
done
```

```
for num in {10..0..-1}
do
echo $num
done
```

```
rexder@HunterMachine ~ [2]> bash hello.sh
10
9
8
7
6
5
4
3
2
1
0
```


While loop

```
while [ expression ]  
do  
    body  
done
```

```
rexder@HunterMachine ~> bash hello.sh  
Enter starting number: 10  
Enter ending number: 15  
10  
11  
12  
13  
14  
15  
This is the sequence that you wanted.
```

```
#!/bin/bash
```

```
#Script to get specified numbers
```

```
read -p "Enter starting number: " snum
```

```
read -p "Enter ending number: " enum
```

```
while [[ $snum -le $enum ]]
```

```
do
```

```
echo $snum
```

```
((snum++))
```

```
done
```

```
echo "This is the sequence that you wanted."
```

You do ((count++))

```
while :
do
echo "Welcome to GeezTech."
done
```

```
while [ expression ]
do
    body
done
```

```
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
Welcome to GeezTech.  
^C  
rexder@HunterMachine ~ [SIGINT]>
```



Break Statement

```
#!/bin/bash

echo "Countdown for Website Launching..."
i=10
while [ $i -ge 1 ]
do
    if [ $i == 2 ]
    then
        echo "Mission Aborted, Some Technical Error Found."
        break
    fi
    echo "$i"
    (( i-- ))
done
```




...Output

```
rexder@HunterMachine ~ [SIGINT]> bash hello.sh  
Countdown for Website Launching...  
10  
9  
8  
7  
6  
5  
4  
3  
Mission Aborted, Some Technical Error Found.
```

Continue Statement

- Is A function that helps to break the loop there and start if from the up again. - its python equivalent is called 'pass'

```
#!/bin/bash
#While Loop Example with a Continue Statement

i=0
while [ $i -le 10 ]
do
  ((i++))
  if [[ "$i" == 5 ]]
  then
    continue
  fi
  echo "Current Number : $i"
done

echo "Skipped number 5 using Continue Statement."
```

```
rexder@HunterMachine ~> bash hello.sh
Current Number : 1
Current Number : 2
Current Number : 3
Current Number : 4
Current Number : 6
Current Number : 7
Current Number : 8
Current Number : 9
Current Number : 10
Current Number : 11
Skipped number 5 using Continue Statement.
```



Exercise 2

- 1) Create a loop that prints your name 10 times
- 2) Accept 2 input from user 1 his name and 2 amount of repeat he needs and display it for the size he wanted to repeat.
- 3) Create a infinite loop that displays, "You are BASH MASTER!!!"

Until loop

- It is same but this one exits the loop when the expression is true.
- The name until means እስከ

```
until [ Expression ]  
do  
    body  
done
```

```
i=1  
until [ $i -gt 10 ]  
do  
    echo $i  
    ((i++))  
done
```

```
(rexder@HunterMachine)-[~]  
$ bash test.sh  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
  
i=0  
while [ $i -lt 10 ]  
do  
    echo $i  
    ((i++))  
done
```



Function

```
# Creating Function  
function_name() {  
    body  
}  
  
# Calling the function  
function_name
```

```
print_it () {  
    sum=$(( $1+$2 ))  
    echo "the sum: $sum"  
}  
  
print_it 5 6
```

```
rexder@HunterMachine ~> bash hello.sh  
the sum: 11
```

Cont...

- When you have function and the arguments will be \$1, \$2
- If we use \$? it will call the return value.

```
print_it () {  
    echo Your name is $1 and your fathername is $2  
    return 20  
}
```

```
print_it Nathan Hailu  
echo "You are $? years old."  
rexder@HunterMachine ~> bash hello.sh  
Your name is Nathan and your fathername is Hailu  
You are 20 years old.
```

```
print_it () {  
    return $((($1+$2))  
}
```

```
print_it 5 6  
echo "the sum: $?"
```

```
rexder@HunterMachine ~> bash hello.sh  
the sum: 11
```




Exercise 3

Write a code that can replace this `pow()` keyword. It accepts 2 arguments a number and its power. If you do `pow(4,2)` => it gives 16, means the power.

```
print(pow(5,2))
```

```
# Output: 25
```

- 1) Create A function
- 2) Accept 2 arguments
- 3) Do a calculation to power
- 4) Return the value
- 5) Call the function and print it.



BASH and Linux

- We can run linux commands inside our bash

```
#!/bin/bash
```

```
echo "I can run apt commands in my bash"  
apt update
```

```
rexder@HunterMachine ~> bash hello.sh  
I can run apt commands in my bash  
[sudo] password for rexder:  
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease  
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease  
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease  
Get:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease  
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64  
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64  
Get:4 http://kali.download/kali kali-rolling InRelease [30.6 kB]  
0% [4 InRelease 3955 B/30.6 kB 13%] [7 Packages 134 kB/774 kB 17%]
```



Cont...

- You can run bash codes in 1 line.

```
for i in {1..5}
do
echo "Hello $i"
done |
```

```
for i in {1..5} do echo "Hello $i" done
```

```
rexder@HunterMachine ~> bash hello.sh
Hello 1
Hello 2
Hello 3
Hello 4
Hello 5
```




Cont...

- You can access files in current directory using * sign

```
#!/bin/bash
```

```
echo *
```

```
(rexder@HunterMachine)-[~]
```

```
$ bash hello.sh
```

```
class.sh Day4_MoreLinux.md Desktop Documents Downloads exe3.sh gtst hello.sh  
InitProgramming intro-linux.pdf linux Music Perm.txt Pictures Public requirem  
ents.txt stderr.txt stdout.txt takeme1.txt Templates test.sh Videos
```



Interaction with linux

- You can interact with linux terminal using bash.
- For this our codes are bash so, your terminal have to be on bash
- Command: `/bin/bash`

```
rexder@HunterMachine ~/t> /bin/bash
sessions should be nested with care, unset $TMUX to force
rexder@HunterMachine:~/t$
```

For loop on terminal

Syntax: **for command; do body; done**

```
(rexder@HunterMachine)-[~]  
$ echo *  
class.sh Day4_MoreLinux.md Desktop Documents Downloads exe3.sh gtst hello.sh
```

```
(rexder@HunterMachine)-[~]  
$ for i in *; do echo $i; done  
class.sh  
Day4_MoreLinux.md  
Desktop  
Documents  
Downloads  
exe3.sh  
gtst  
hello.sh  
InitProgramming  
intro-linux.pdf  
linux  
Music  
Perm.txt  
Pictures  
Public  
requirements.txt  
stderr.txt  
stdout.txt  
takeme1.txt  
Templates  
test.sh  
videos
```




Tasks #1

Do the pwd command by your own.

```
(rexder@HunterMachine)-[~/t]  
$ pwd  
/home/rexder/t
```

```
(rexder@HunterMachine)-[~/t]  
$ echo $PWD  
/home/rexder/t
```

Tasks #2

Adding “old” text before filenames.

```
(rexder@HunterMachine)-[~/t]  
$ ls  
homefile  namefile  officefile  rexderfile  schoolfile
```

```
(rexder@HunterMachine)-[~/t]  
$ for i in *; do mv $i old_$i; done
```

```
(rexder@HunterMachine)-[~/t]  
$ ls  
old_homefile  old_namefile  old_officefile  old_rexderfile  old_schoolfile
```



Assignment.

- 1) Create 5 files
- 2) Replace Their name with sequences before them
 - a) 00_file
 - b) 01_filedsad
 - c) 02_sdas



Assignment

- 1) Mikiyas is a GTST company owner, and wanted to create a program that validates users so tried to make a login page can u help him?
 - a) If the login is failed it have to ask again 5 times then display “Sorry u are limited!”
 - b) Accept 2 inputs from user (username and password)
 - c) Validates it by just giving some usern and passwd variables
 - d) Display “Welcome to GTST Company!” if success else “Incorrect Login!”
 - e) HINT: use nested things only!

```
u="Nathan"  
p="2123"
```



Class is over

- 1) DO the note
- 2) Practice
- 3) Ask