Further on Bash!

Day11_proBash.md

Recall

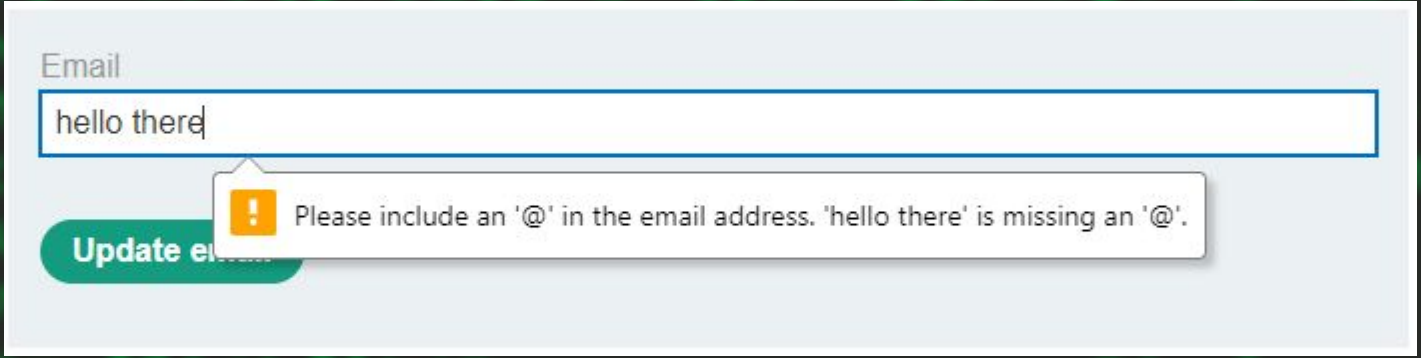
LAST TIME TOPIC

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?


Email

hello there

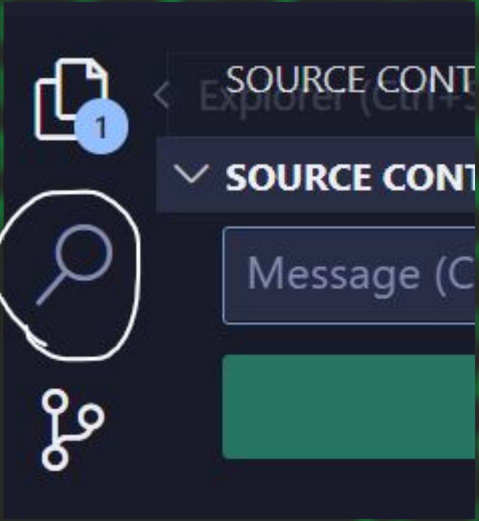
Please include an '@' in the email address. 'hello there' is missing an '@'.

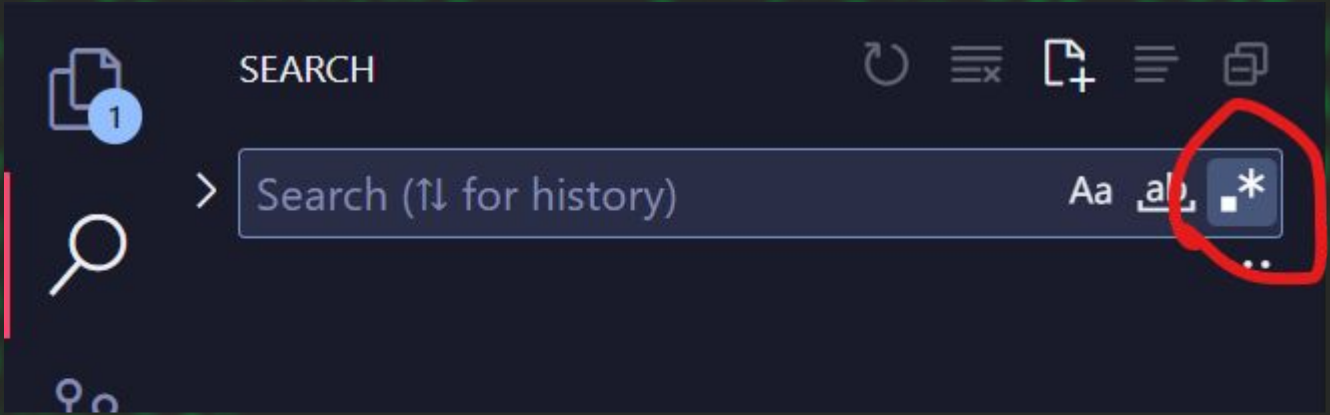
Cont...

- 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...

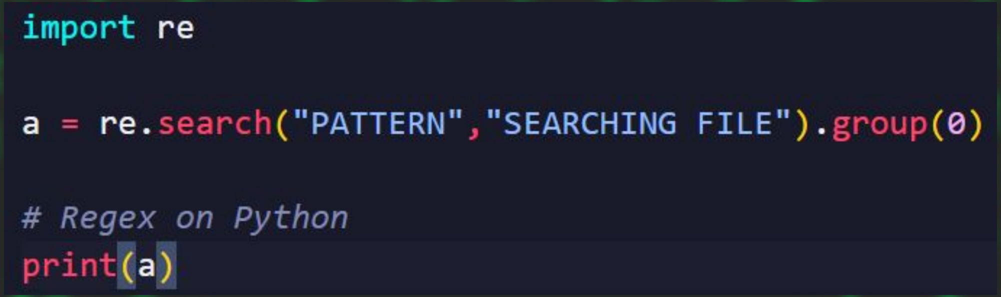
- To demonstrate this we can simply use vscode search tool.
- And don't forget to turn the regex button on





regex...

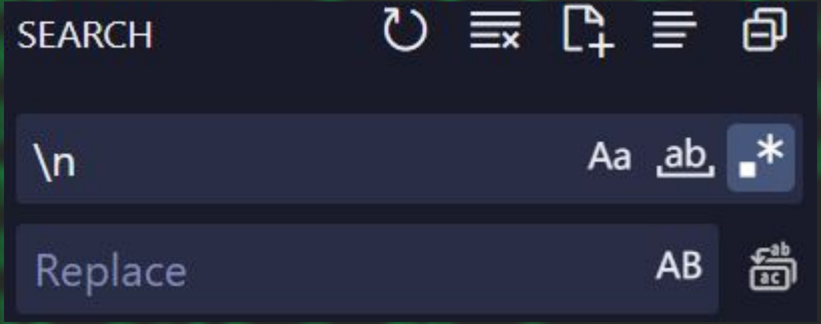
- The pattern is same but the implementation may differ on programming languages.
- On Python,

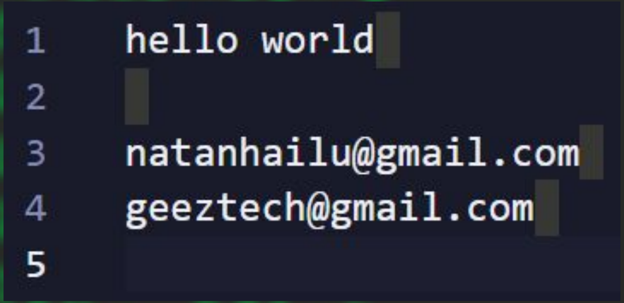


```python
import re

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

# Regex on Python
print(a)
```

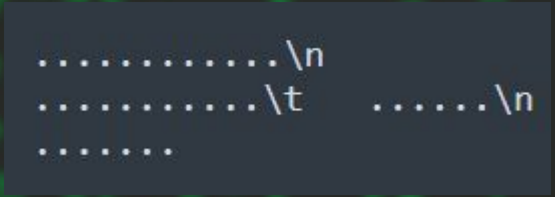
Quick Test!

What do u think the output of this

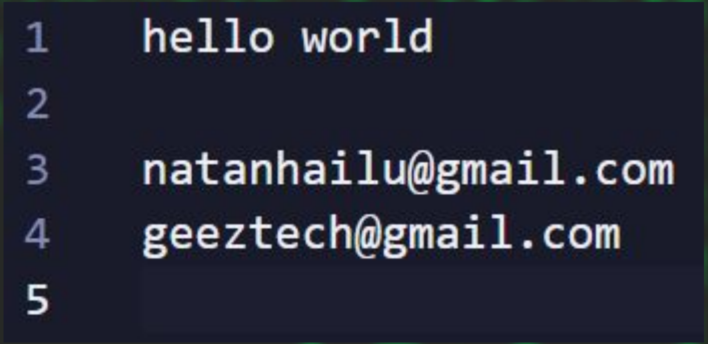




```
hello world

natanhailu@gmail.com
geeztech@gmail.com

```



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



```
hello world

natanhailu@gmail.com
geeztech@gmail.com

```

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

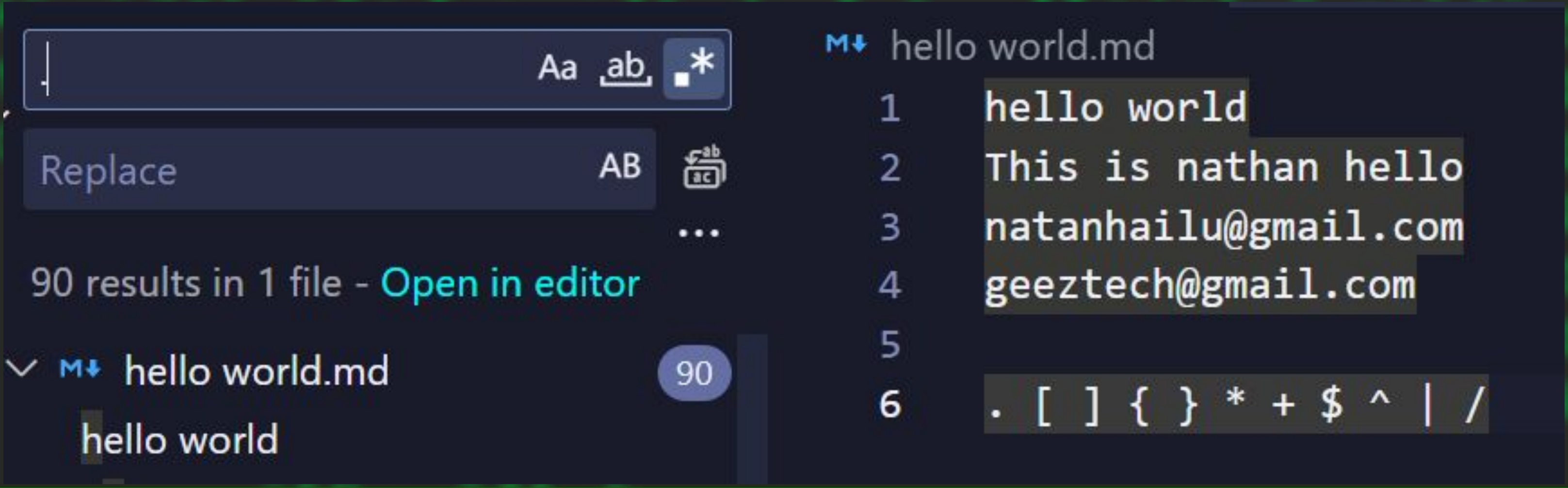
Metacharacters

- 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



90 results in 1 file - Open in editor

hello world.md 90

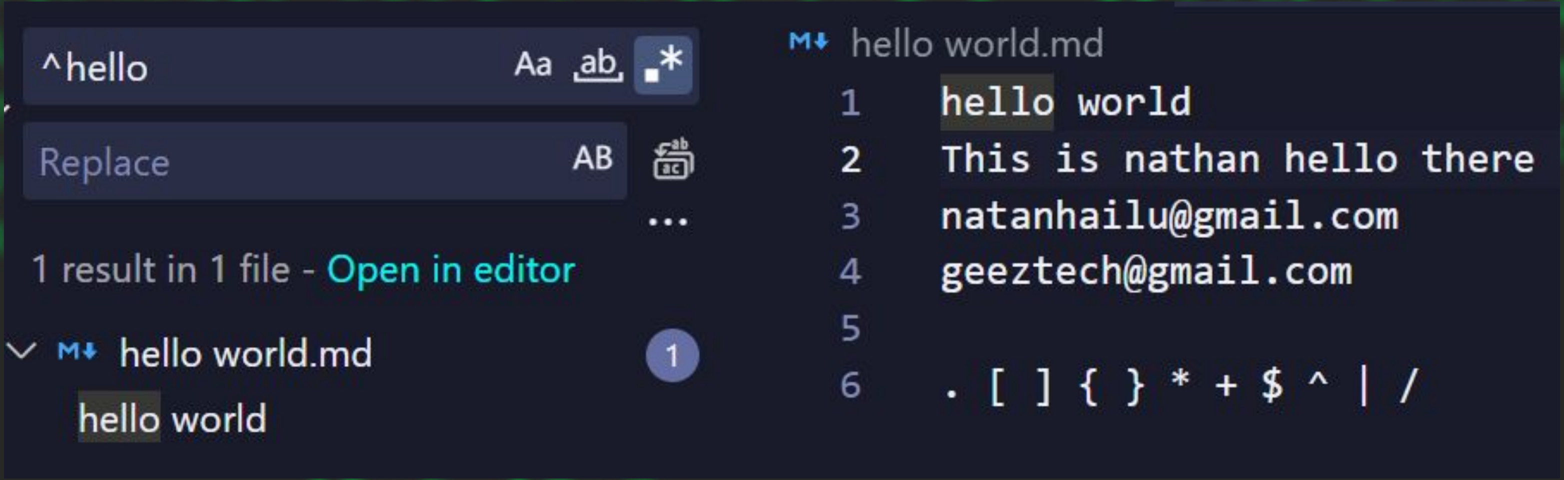
hello world

hello world.md

```
hello world
This is nathan hello
natanhailu@gmail.com
geeztech@gmail.com

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

Caret (^) - Assertion

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



^hello Aa ab .*

Replace AB

1 result in 1 file - Open in editor

hello world.md 1

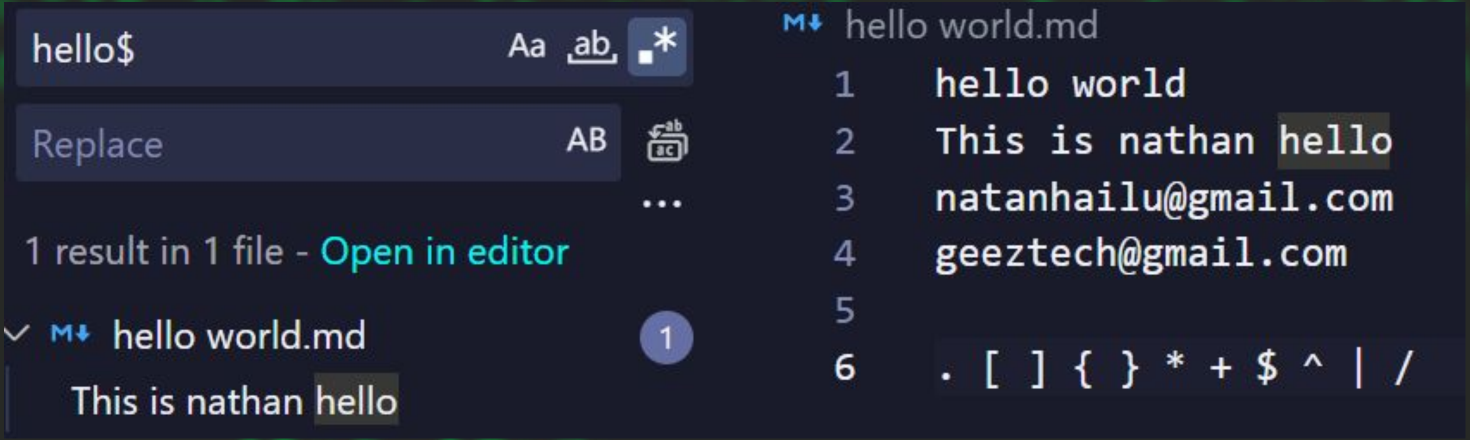
hello world

hello world.md

```
hello world
This is nathan hello there
natanhailu@gmail.com
geeztech@gmail.com

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

Dollar sign($) - Assertion

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



hello$

Replace

1 result in 1 file - Open in editor

hello world.md

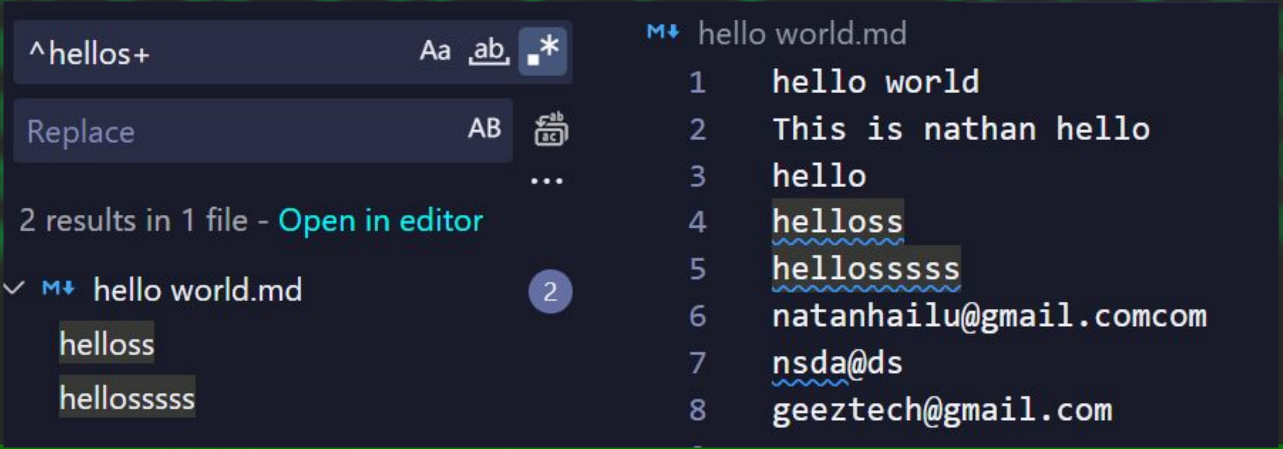
This is nathan hello

hello world.md

```
hello world
This is nathan hello
natanhailu@gmail.com
geeztech@gmail.com

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

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.



^hellos+

Replace

2 results in 1 file - Open in editor

hello world.md 2

helloss

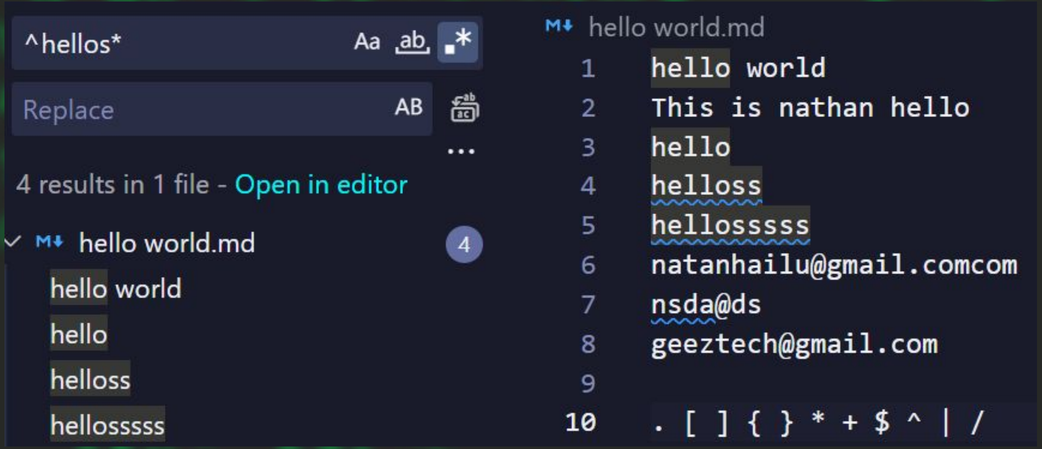
hellossss

hello world.md

```
hello world
This is nathan hello
hello
helloss
hellossss
natanhailu@gmail.comcom
nsda@ds
geeztech@gmail.com
```

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.

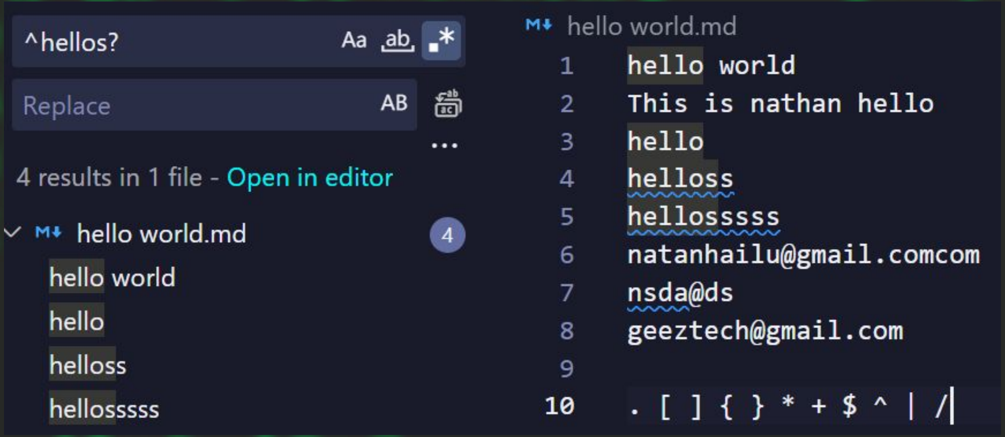


```
^hellos*                Aa  ab  .*
Replace                 AB

4 results in 1 file - Open in editor

hello world.md   4
  hello world
  hello
  helloss
  hellosssss
```

```
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.



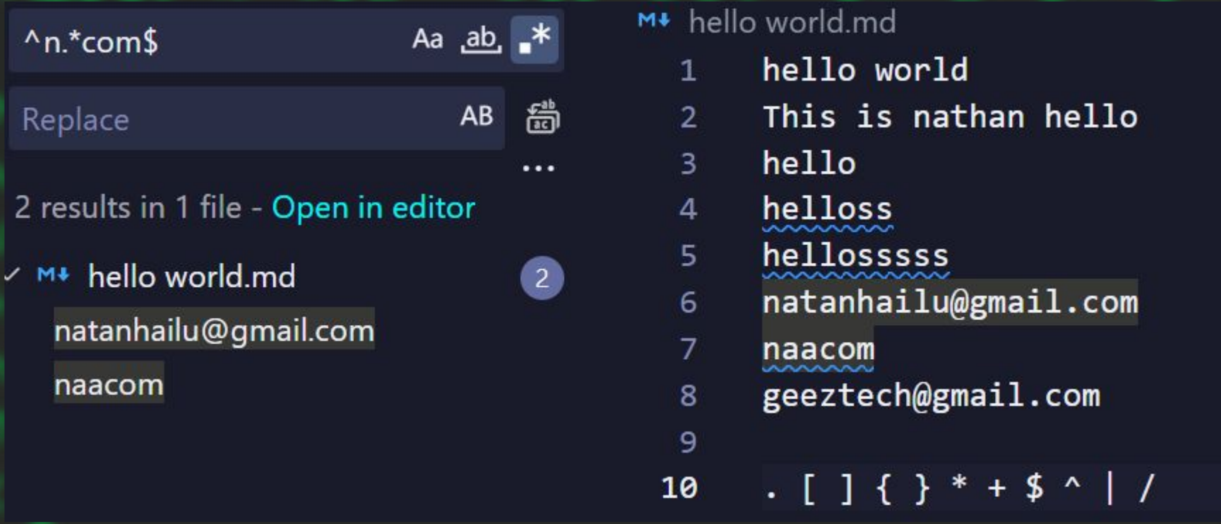
```
^hellos?                    Aa  ab  .*
Replace                     AB
4 results in 1 file - Open in editor
hello world.md   4
  hello world
  hello
  helloss
  hellosssss
```

```
hello world.md
hello world
This is nathan hello
hello
helloss
hellosssss
natanhailu@gmail.comcom
nsda@ds
geeztech@gmail.com

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

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$



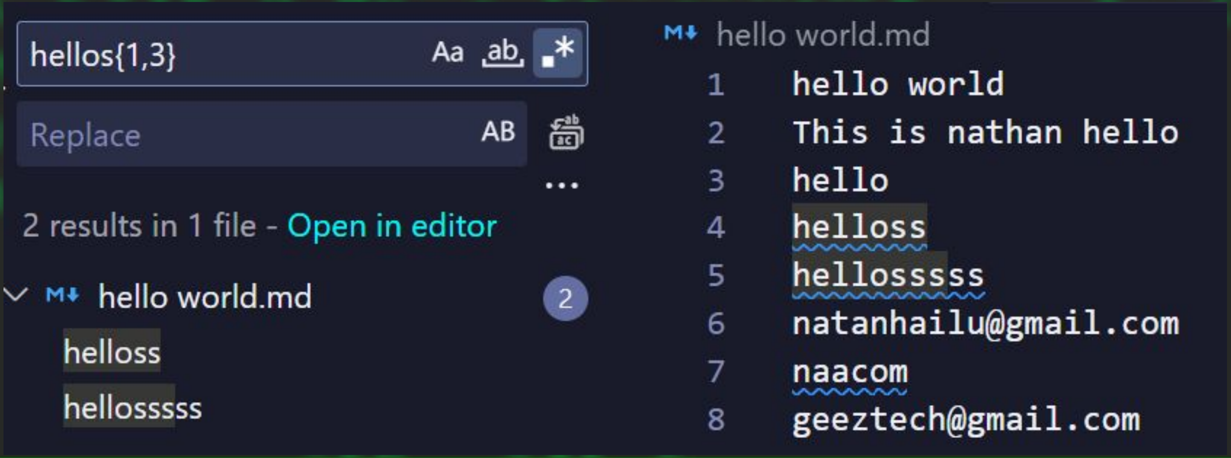
```
natanhailu@gmail.com
naacom
```

hello world.md

```
hello world
This is nathan hello
hello
helloss
hellosssss
natanhailu@gmail.com
naacom
geeztech@gmail.com

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

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.



```
hellos{1,3}
Replace

2 results in 1 file - Open in editor

hello world.md  2
helloss
hellosssss
```

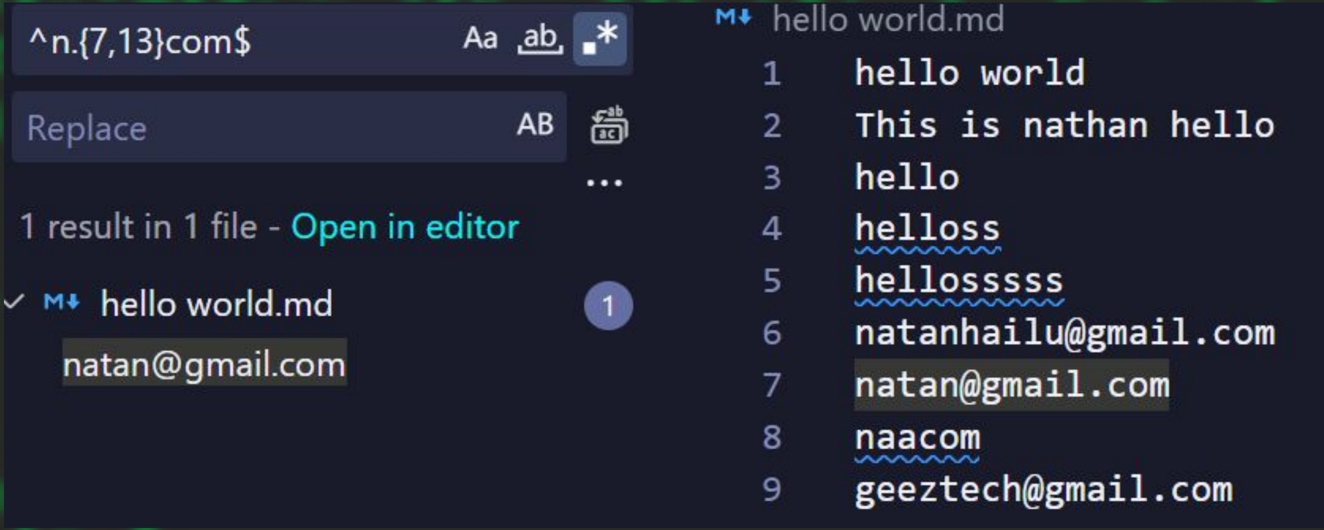
```
hello world.md
hello world
This is nathan hello
hello
helloss
hellosssss
natanhailu@gmail.com
naacom
geeztech@gmail.com
```

{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$

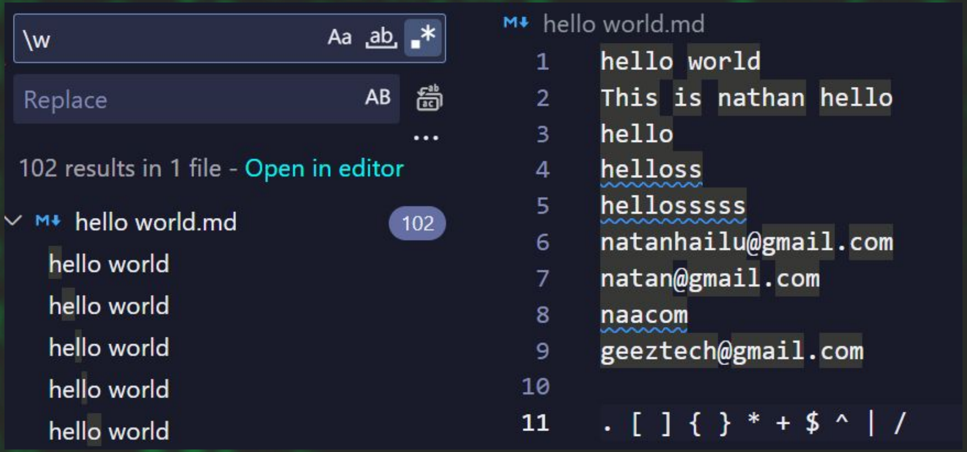


```
^n.{7,13}com$
1 result in 1 file - Open in editor
hello world.md  1
natan@gmail.com
```

```
hello world.md
hello world
This is nathan hello
hello
helloss
hellosssss
natanhailu@gmail.com
natan@gmail.com
naacom
geeztech@gmail.com
```

\w

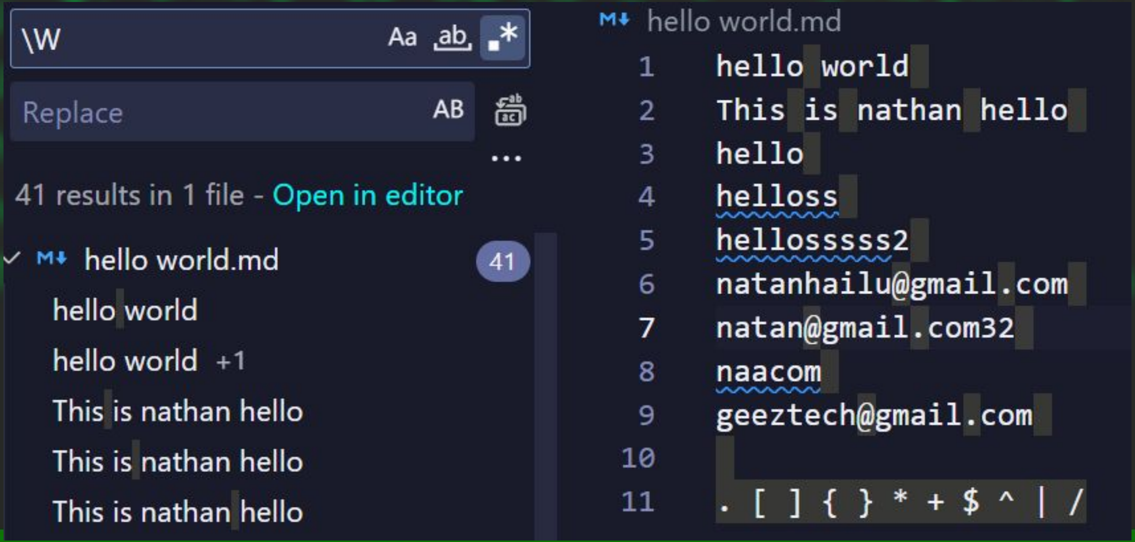
- Used to get Alphanumeric
- Syntax: \w
    - All texts except newlines and symbols



```
hello world
This is nathan hello
hello
helloss
hellosssss
natanhailu@gmail.com
natan@gmail.com
naacom
geeztech@gmail.com

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

\W

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

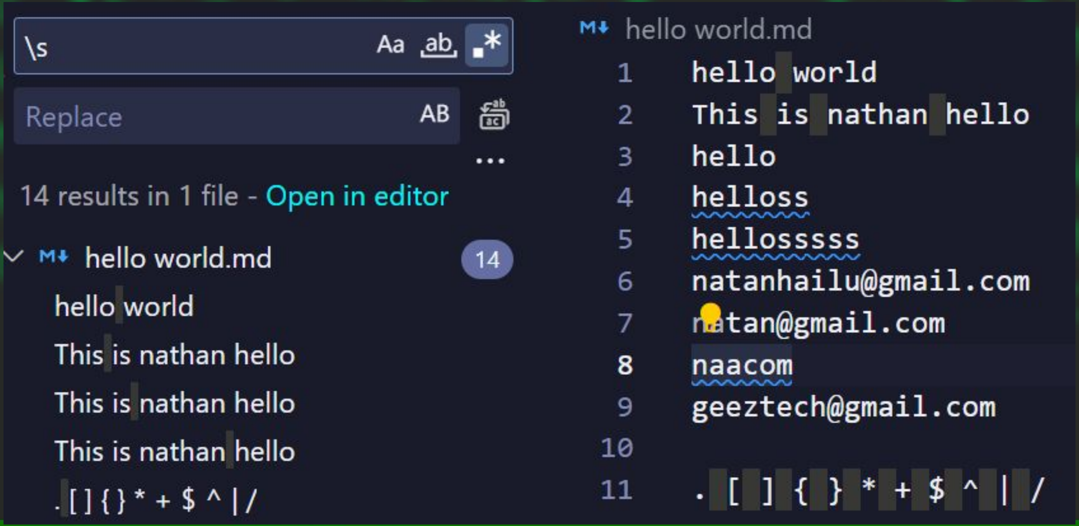


```
\W
Replace
41 results in 1 file - Open in editor
hello world.md  41
hello world
hello world +1
This is nathan hello
This is nathan hello
This is nathan hello
```

```
hello world.md
hello world
This is nathan hello
hello
helloss
hellosssss2
natanhailu@gmail.com
natan@gmail.com32
naacom
geeztech@gmail.com

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

\s

- Used to get whitespace.
- Syntax: \s



14 results in 1 file - Open in editor

hello world.md 14

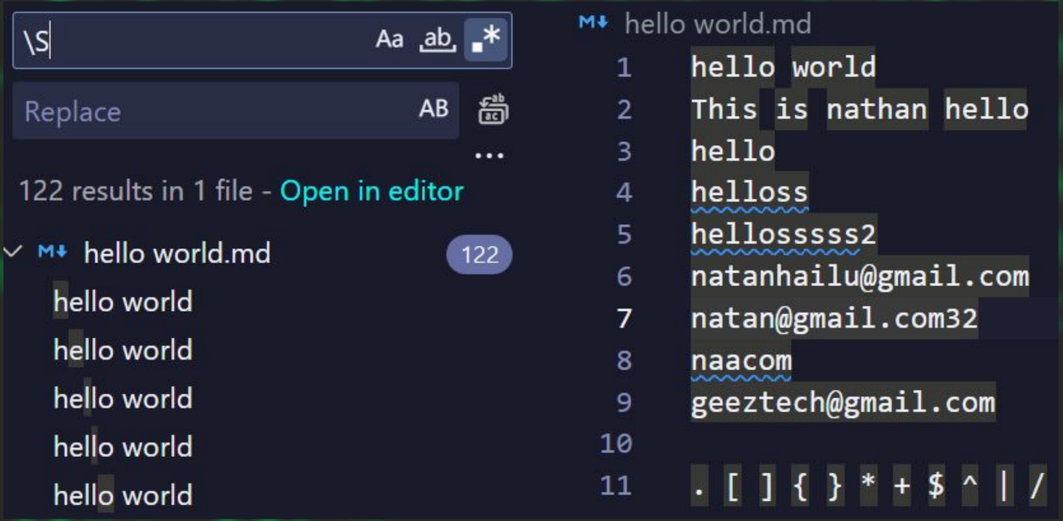
hello world
This is nathan hello
This is nathan hello
This is nathan hello
. [ ] { } * + $ ^ | /

hello world.md

```
hello world
This is nathan hello
hello
helloss
hellosssss
natanhailu@gmail.com
natan@gmail.com
naacom
geeztech@gmail.com

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

\S

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

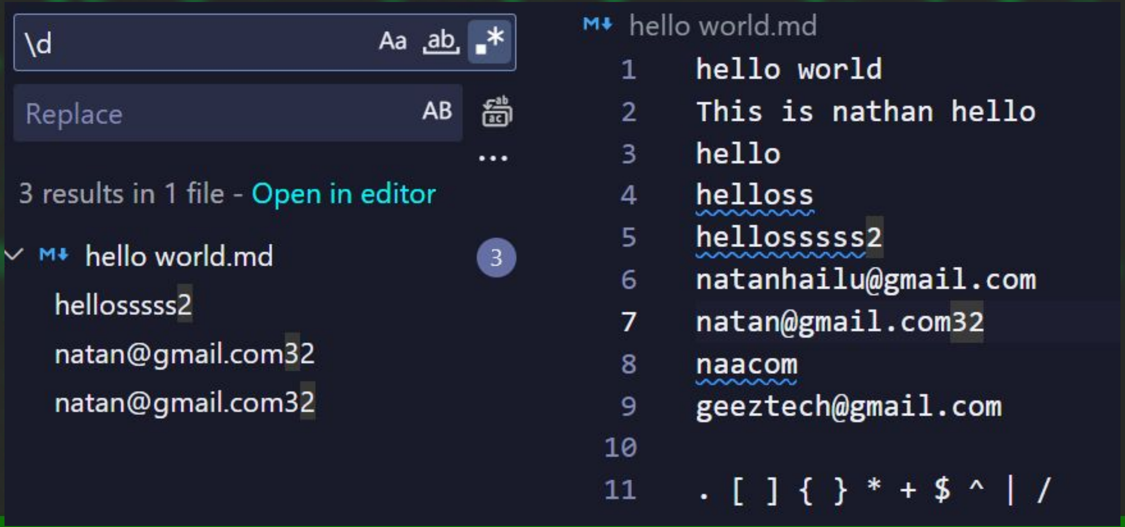


```
\S
Replace
122 results in 1 file - Open in editor
hello world.md 122
hello world
hello world
hello world
hello world
hello world
```

```
hello world.md
hello world
This is nathan hello
hello
helloss
hellosssss2
natanhailu@gmail.com
natan@gmail.com32
naacom
geeztech@gmail.com

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

\d

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



3 results in 1 file - Open in editor

hello world.md 3

hellosssss2

natan@gmail.com32

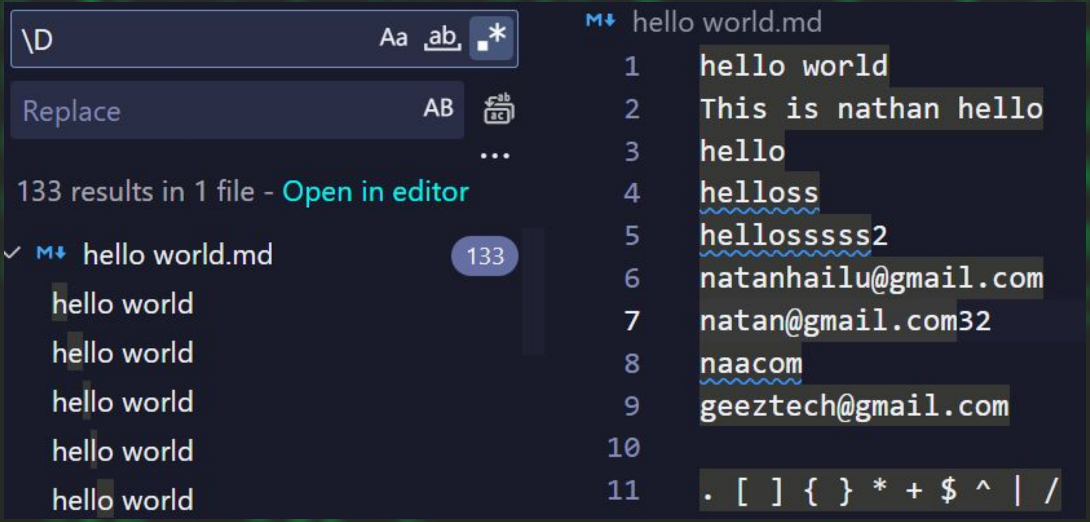
natan@gmail.com32

hello world.md

```
hello world
This is nathan hello
hello
helloss
hellosssss2
natanhailu@gmail.com
natan@gmail.com32
naacom
geeztech@gmail.com

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

\D

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

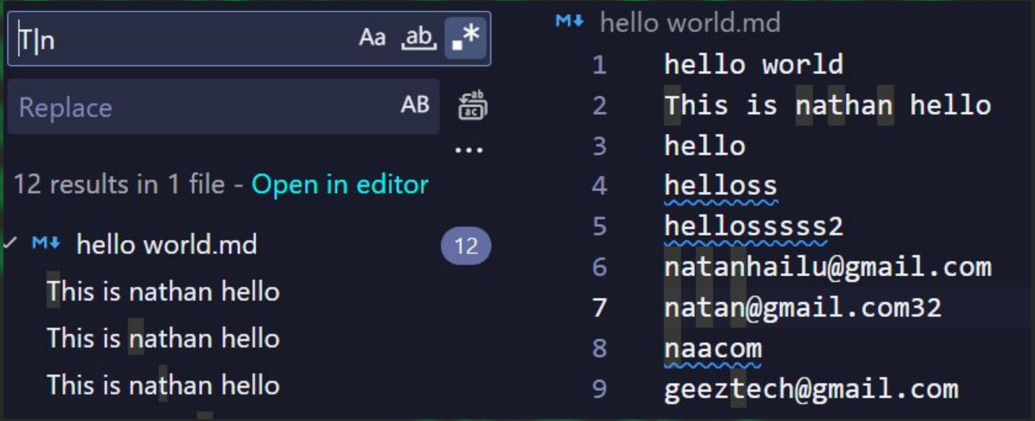


```
hello world
hello world
hello world
hello world
hello world
```

```
hello world
This is nathan hello
hello
helloss
hellosssss2
natanhailu@gmail.com
natan@gmail.com32
naacom
geeztech@gmail.com

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

Pipe (|) - OR

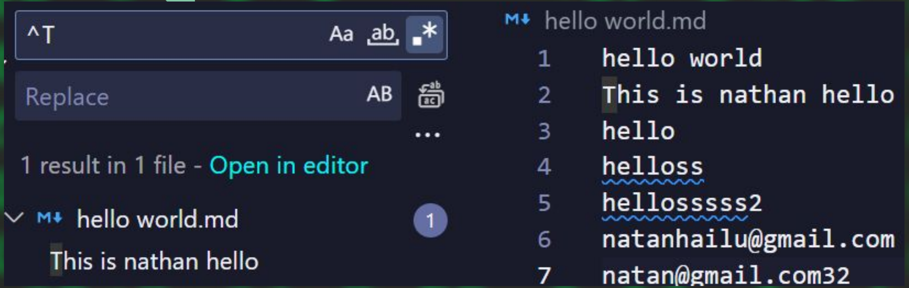
- Used to search 2 different things.
- Syntax: a|b



```
T|n
Replace
12 results in 1 file - Open in editor
hello world.md 12
This is nathan hello
This is nathan hello
This is nathan hello
```

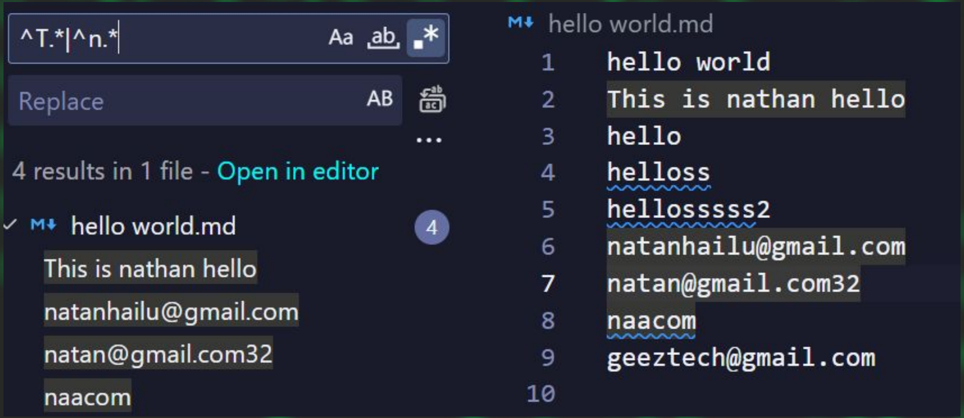
```
hello world.md
hello world
This is nathan hello
hello
helloss
hellosssss2
natanhailu@gmail.com
natan@gmail.com32
naacom
geeztech@gmail.com
```

Cont...



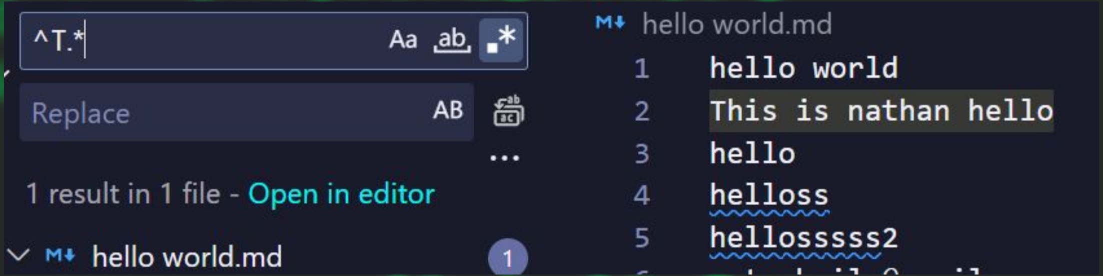
```
^T
Replace
1 result in 1 file - Open in editor
hello world.md  1
This is nathan hello
```

```
hello world.md
hello world
This is nathan hello
hello
helloss
hellosssss2
natanhailu@gmail.com
natan@gmail.com32
```



```
^T.*|^n.*
Replace
4 results in 1 file - Open in editor
hello world.md  4
This is nathan hello
natanhailu@gmail.com
natan@gmail.com32
naacom
```

```
hello world.md
hello world
This is nathan hello
hello
helloss
hellosssss2
natanhailu@gmail.com
natan@gmail.com32
naacom
geeztech@gmail.com
```

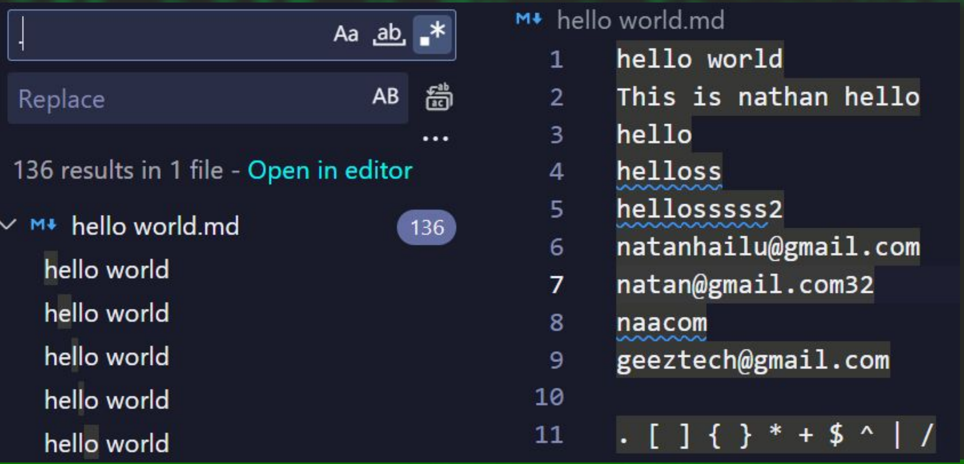


```
^T.*
Replace
1 result in 1 file - Open in editor
hello world.md  1
```

```
hello world.md
hello world
This is nathan hello
hello
helloss
hellosssss2
```

Escape (\ )

- Used to search symbols that are metacharacters.
- Syntax: \sign



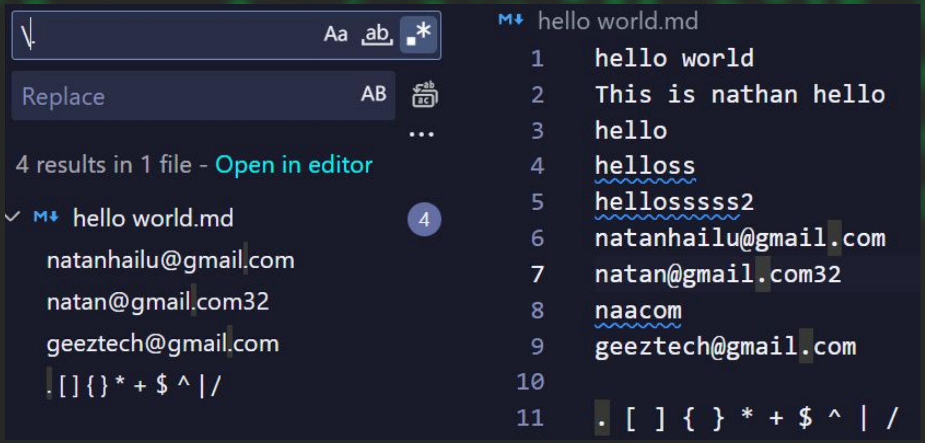
136 results in 1 file - Open in editor

hello world.md 136

```
hello world
hello world
hello world
hello world
hello world
```

hello world.md

```
hello world
This is nathan hello
hello
helloss
hellosssss2
natanhailu@gmail.com
natan@gmail.com32
naacom
geeztech@gmail.com

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



4 results in 1 file - Open in editor

hello world.md 4

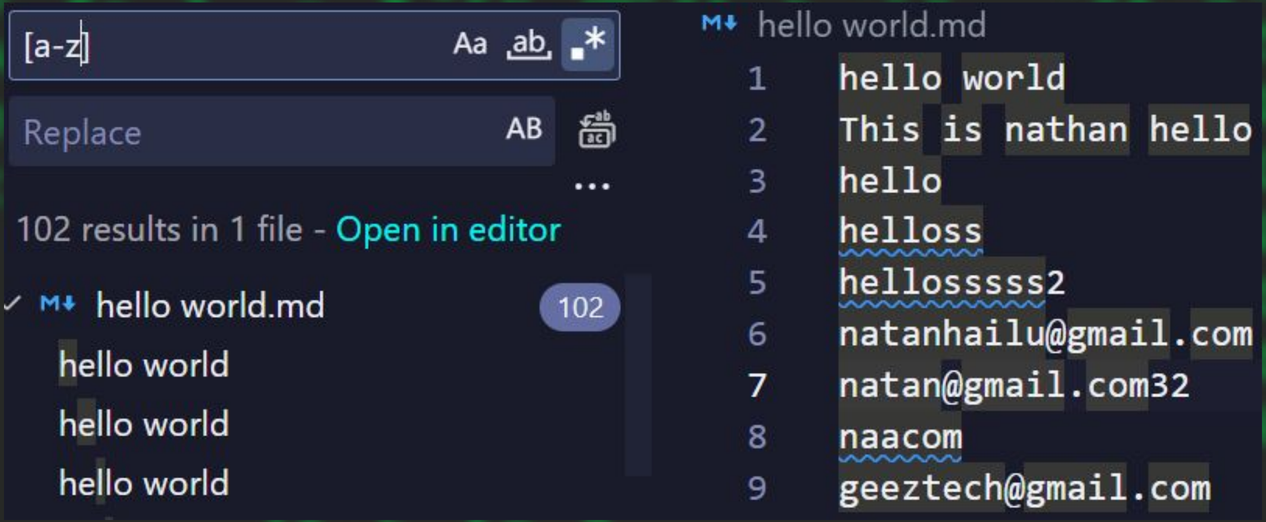
```
natanhailu@gmail.com
natan@gmail.com32
geeztech@gmail.com
. [ ] { } * + $ ^ | /
```

hello world.md

```
hello world
This is nathan hello
hello
helloss
hellosssss2
natanhailu@gmail.com
natan@gmail.com32
naacom
geeztech@gmail.com

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

Square Brackets ([ ]) - Custom pattern

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



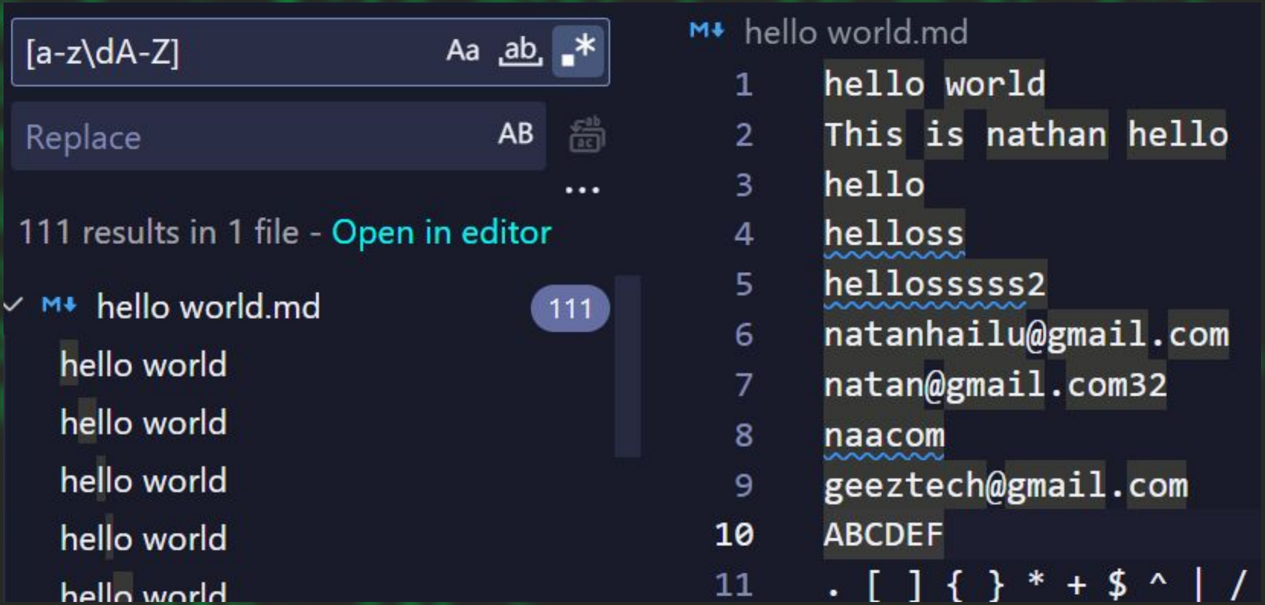
```
[a-z]                    Aa  ab  .*
Replace                  AB

102 results in 1 file - Open in editor

hello world.md   102
  hello world
  hello world
  hello world
```

```
hello world.md
hello world
This is nathan hello
hello
helloss
hellosssss2
natanhailu@gmail.com
natan@gmail.com32
naacom
geeztech@gmail.com
```

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.



```
[a-z\dA-Z]
Replace
111 results in 1 file - Open in editor
hello world.md 111
hello world
hello world
hello world
hello world
hello world
```

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

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

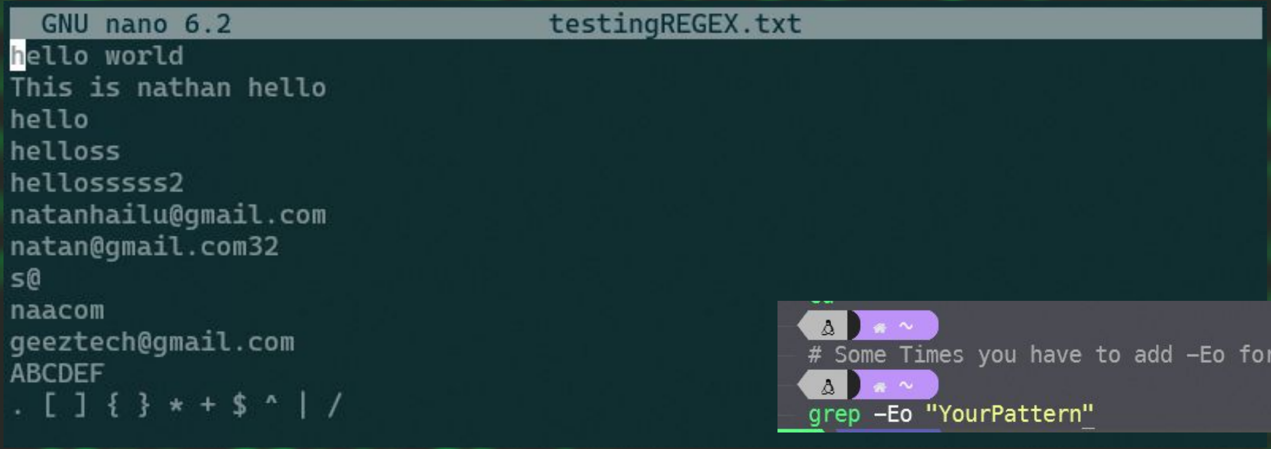
rexder@gmail.com

Hello there this is me

My phone number 0987654321

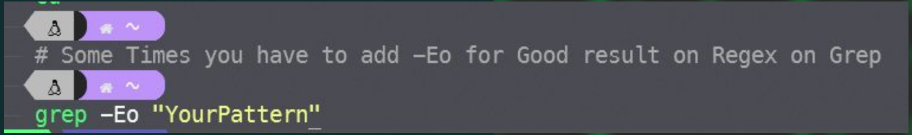
Bash regex

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

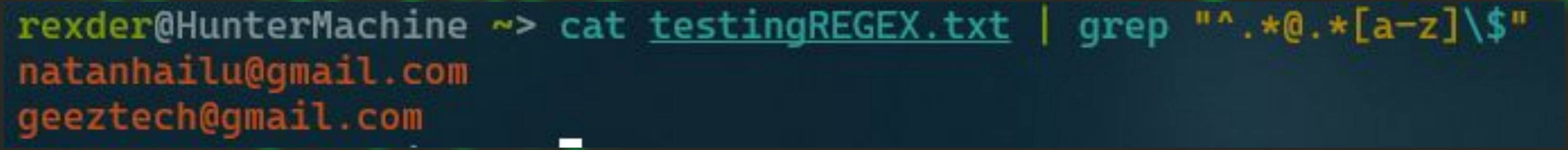


```
 GNU nano 6.2                testingREGEX.txt
hello world
This is nathan hello
hello
helloss
hellossss2
natanhailu@gmail.com
natan@gmail.com32
s@
naacom
geeztech@gmail.com
ABCDEF
. [ ] { } * + $ ^ | /
```

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



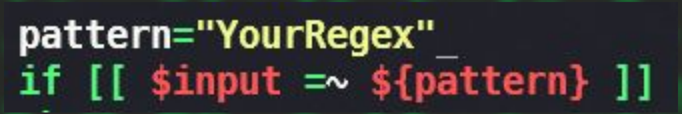
```
# Some Times you have to add -Eo for Good result on Regex on Grep
grep -Eo "YourPattern"
```



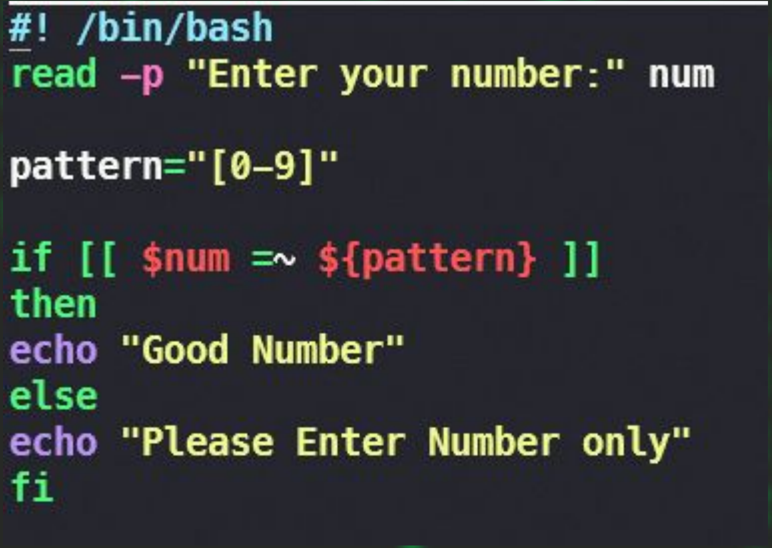
```
rexder@HunterMachine ~> cat testingREGEX.txt | grep "^.*@.*[a-z]\$"
natanhailu@gmail.com
geeztech@gmail.com
```

BASH FOR REGEX

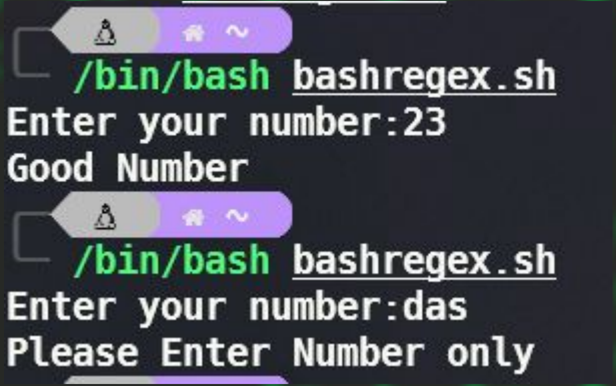
- We use =~ operator for regex check with if condition statements
- Here we use double Brackets for our conditional Statemens.
- SYNTAX:
-



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



```
#! /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
```

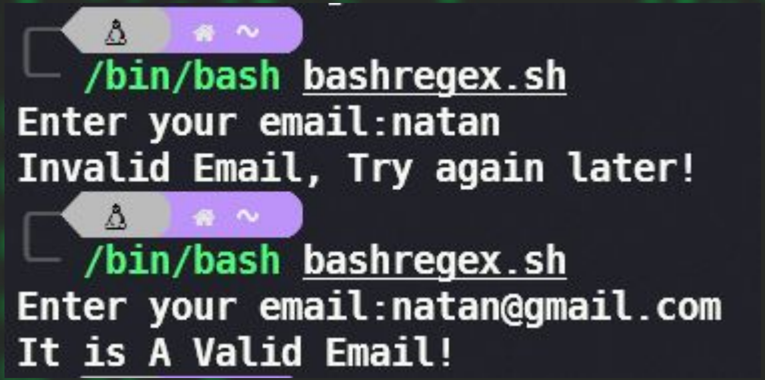


```
/bin/bash bashregex.sh
Enter your number:23
Good Number
/bin/bash bashregex.sh
Enter your number:das
Please Enter Number only
```

Task

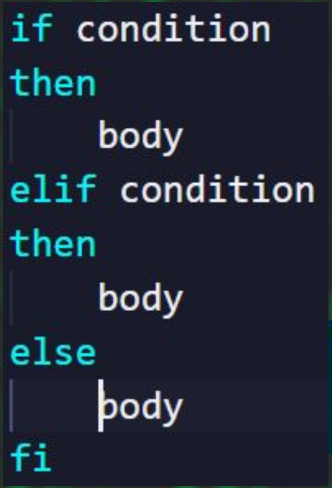
10min

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

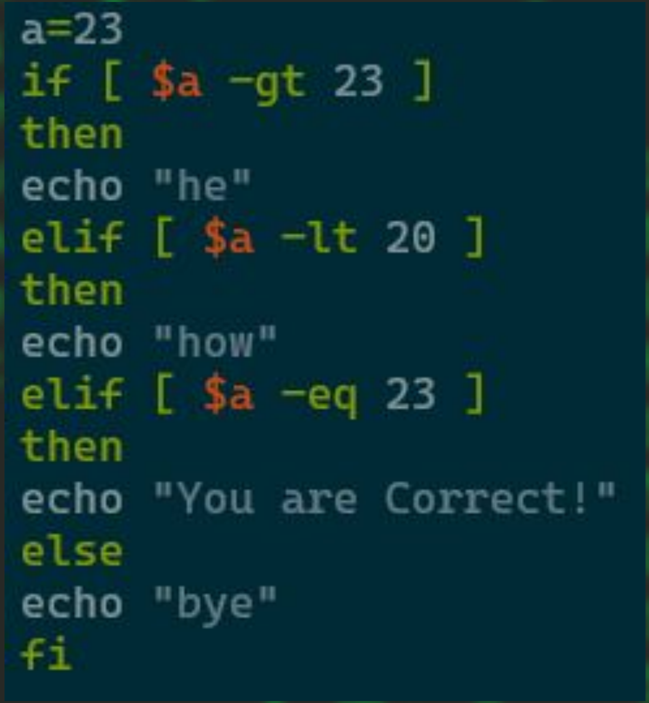


```
/bin/bash bashregex.sh
Enter your email:natan
Invalid Email, Try again later!
/bin/bash bashregex.sh
Enter your email:natan@gmail.com
It is A Valid Email!
```

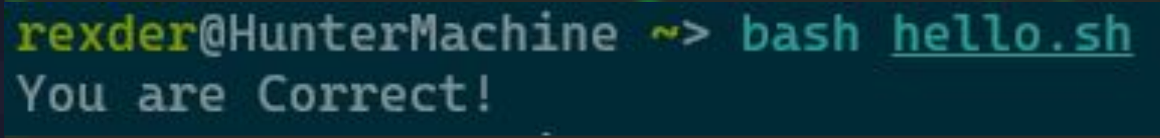
Bash else if

- To do more than 1 comparing.
- It is same with python is 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
```



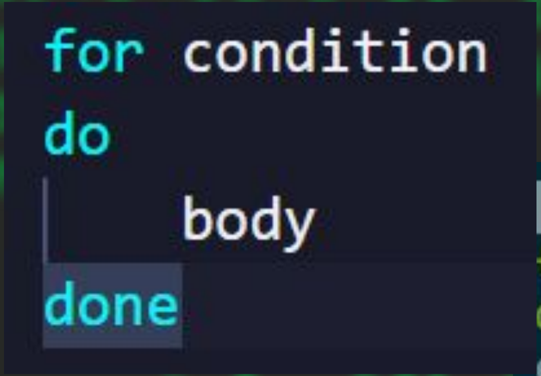
```
rexder@HunterMachine ~> bash hello.sh
You are Correct!
```

Loops

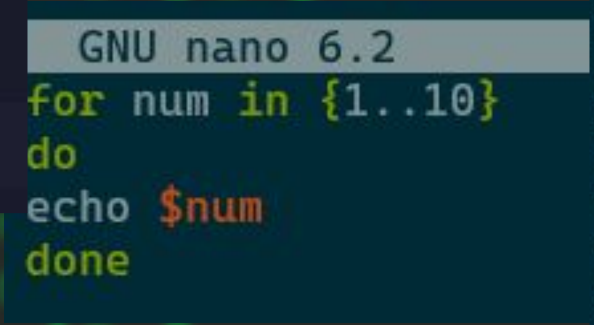
- On Bash, there are 3 types of loops
  a. For loop
  b. While loop
  c. Until loop

For loops

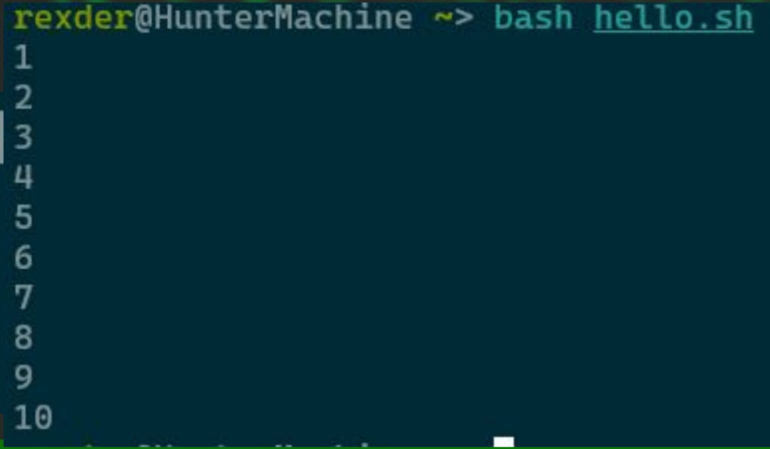
- The iteration may be different here. We can use arrays like list on python but we dont have range in bash but we can do {a..b} this iterates from a to b
- Syntax:



```
for condition
do
    body
done
```

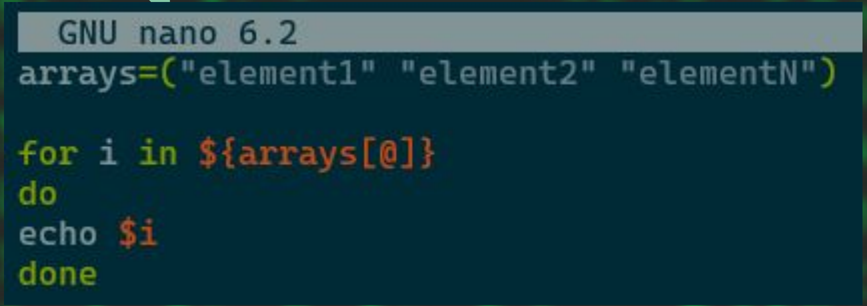


```
  GNU nano 6.2
for num in {1..10}
do
echo $num
done
```

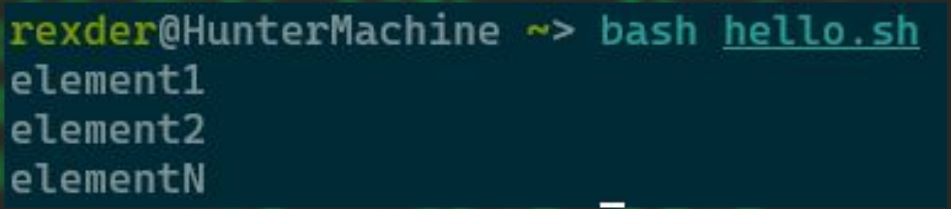


```
rexder@HunterMachine ~> bash hello.sh
1
2
3
4
5
6
7
8
9
10
```

Cont...

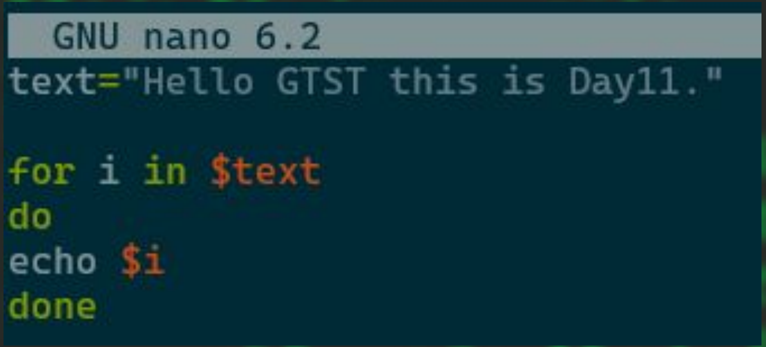


```
  GNU nano 6.2
arrays=("element1" "element2" "elementN")

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

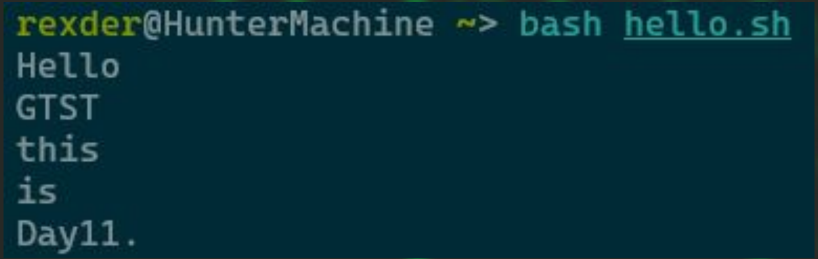


```
rexder@HunterMachine ~> bash hello.sh
element1
element2
elementN
```

Works with words in strings too.

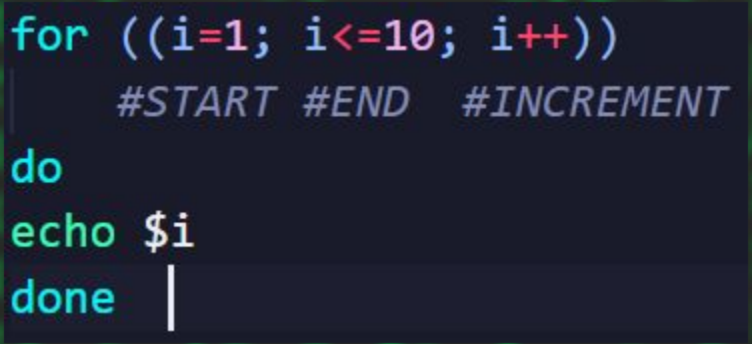


```
  GNU nano 6.2
text="Hello GTST this is Day11."

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

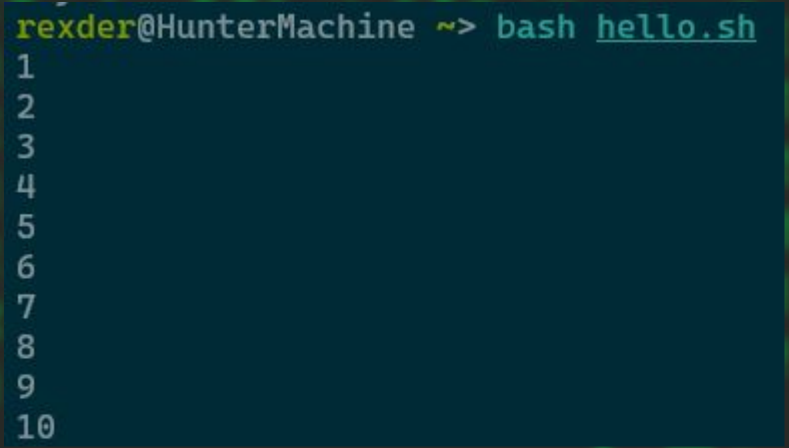


```
rexder@HunterMachine ~> bash hello.sh
Hello
GTST
this
is
Day11.
```

For loop without sequence data



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

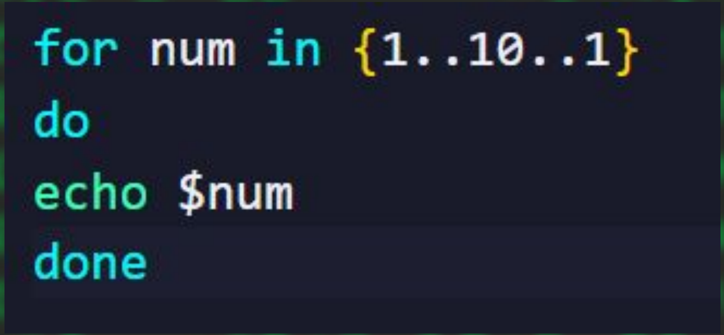


```
rexder@HunterMachine ~> bash hello.sh
1
2
3
4
5
6
7
8
9
10
```

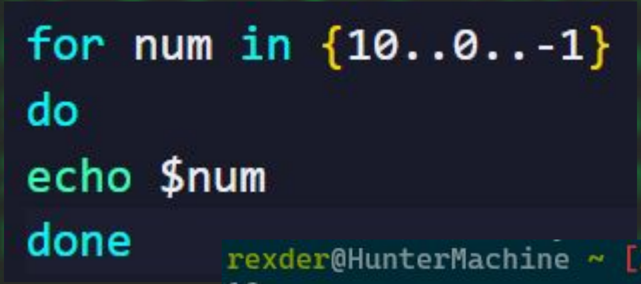
On bash
i++ = i +=1
i- - = i -= 1

For Loop with increment/decrement

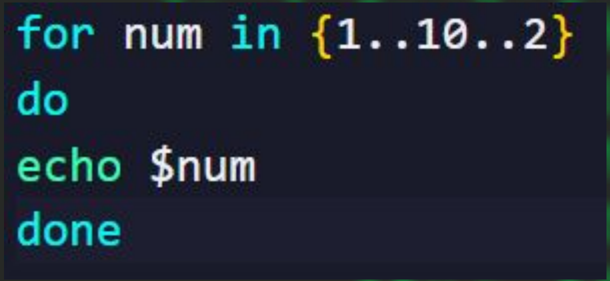
Same syntax with index slicing but here it is for looping. - {start..stop..step}



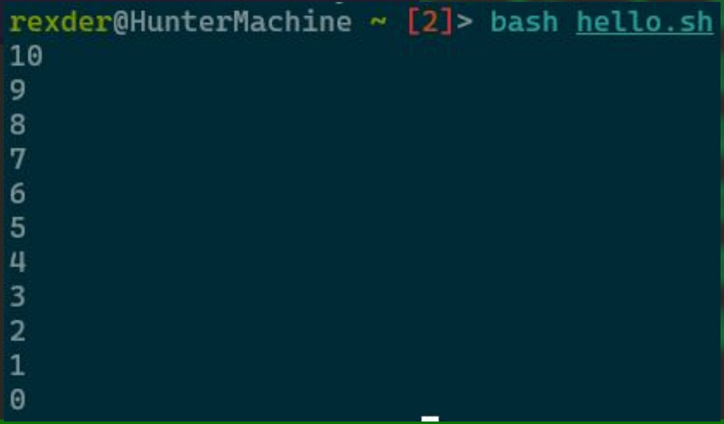
```
for num in {1..10..1}
do
echo $num
done
```



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

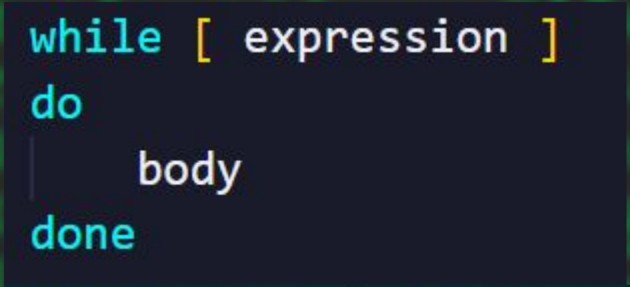


```
for num in {1..10..2}
do
echo $num
done
```

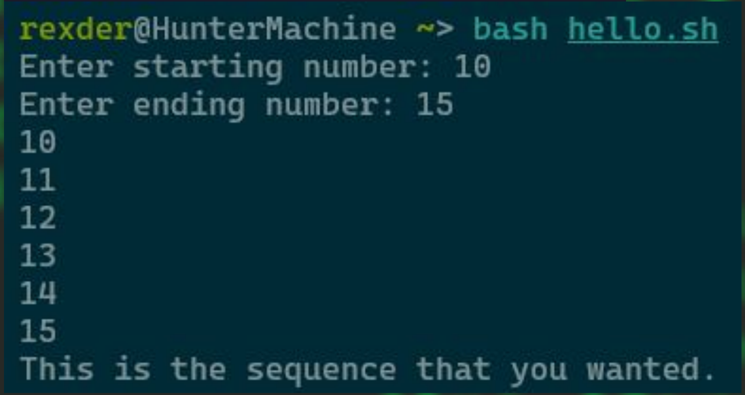


```
rexder@HunterMachine ~ [2]> bash hello.sh
10
9
8
7
6
5
4
3
2
1
0
```

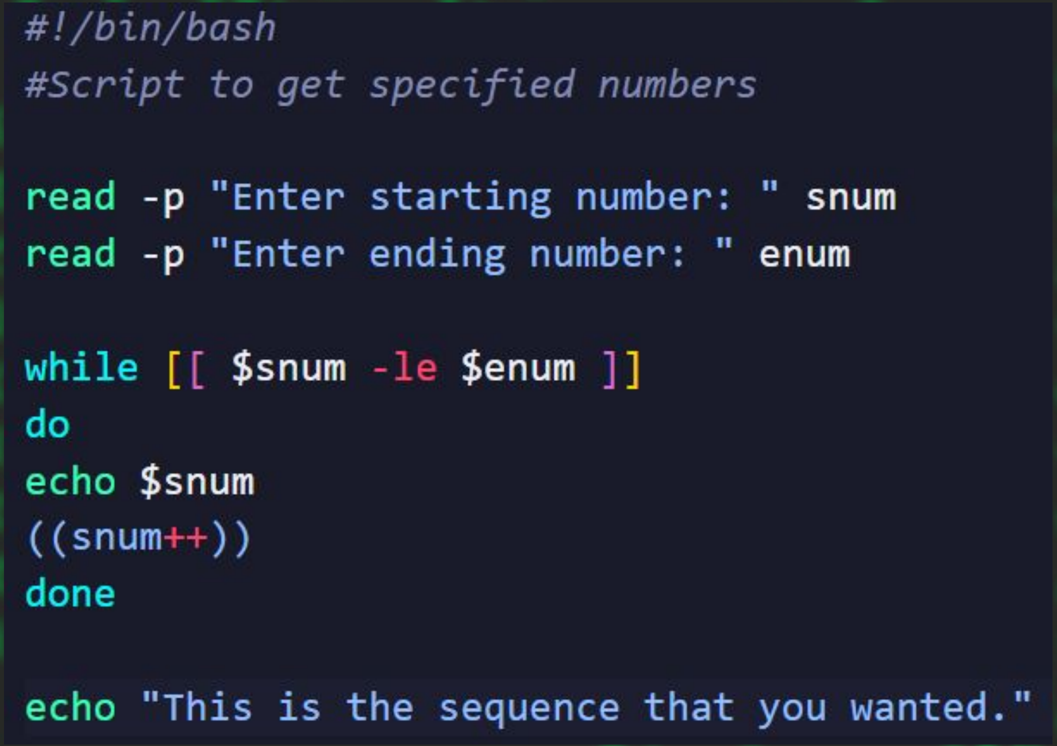
While loop



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



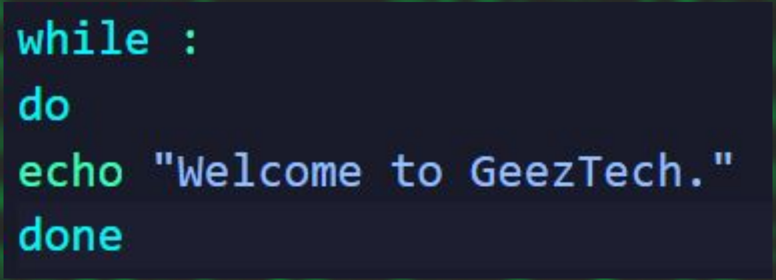
```
rexder@HunterMachine ~> bash hello.sh
Enter starting number: 10
Enter ending number: 15
10
11
12
13
14
15
This is the sequence that you wanted.
```



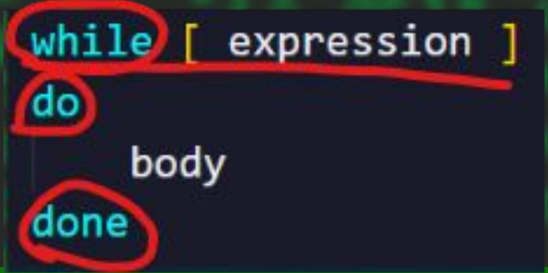
```bash
#!/bin/bash
#Script to get specified numbers

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++))

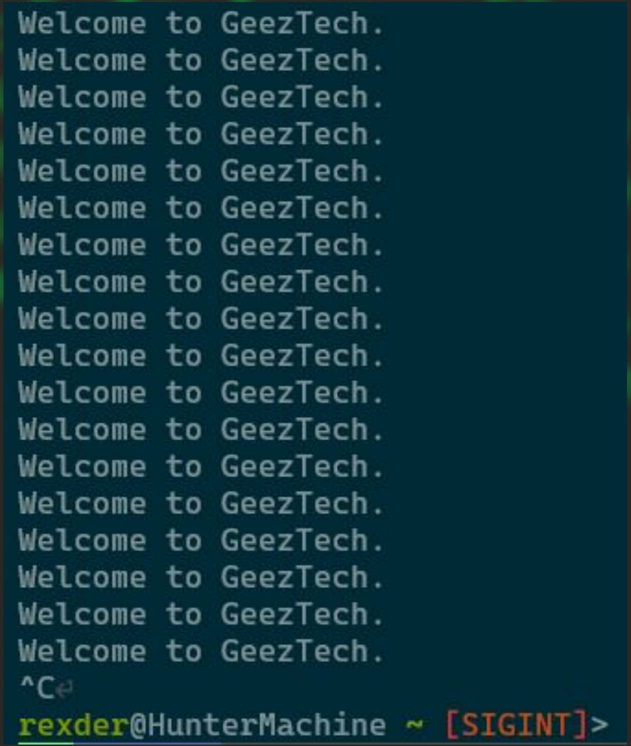
Infinite loop



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

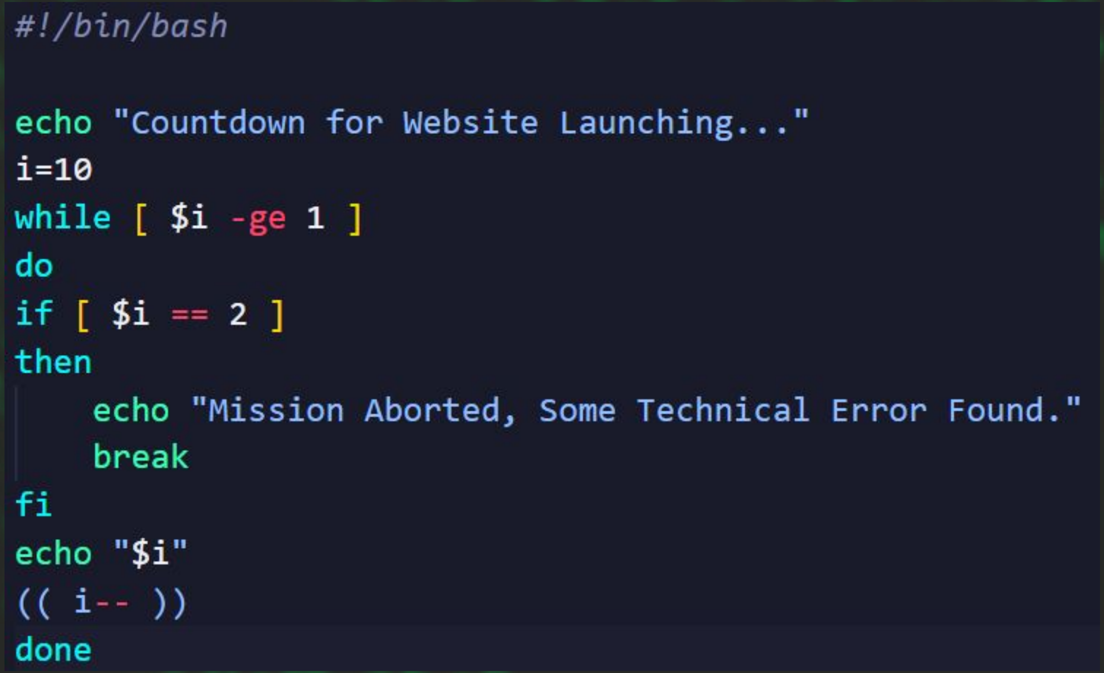


```
while [ expression ]
do
    body
done
```



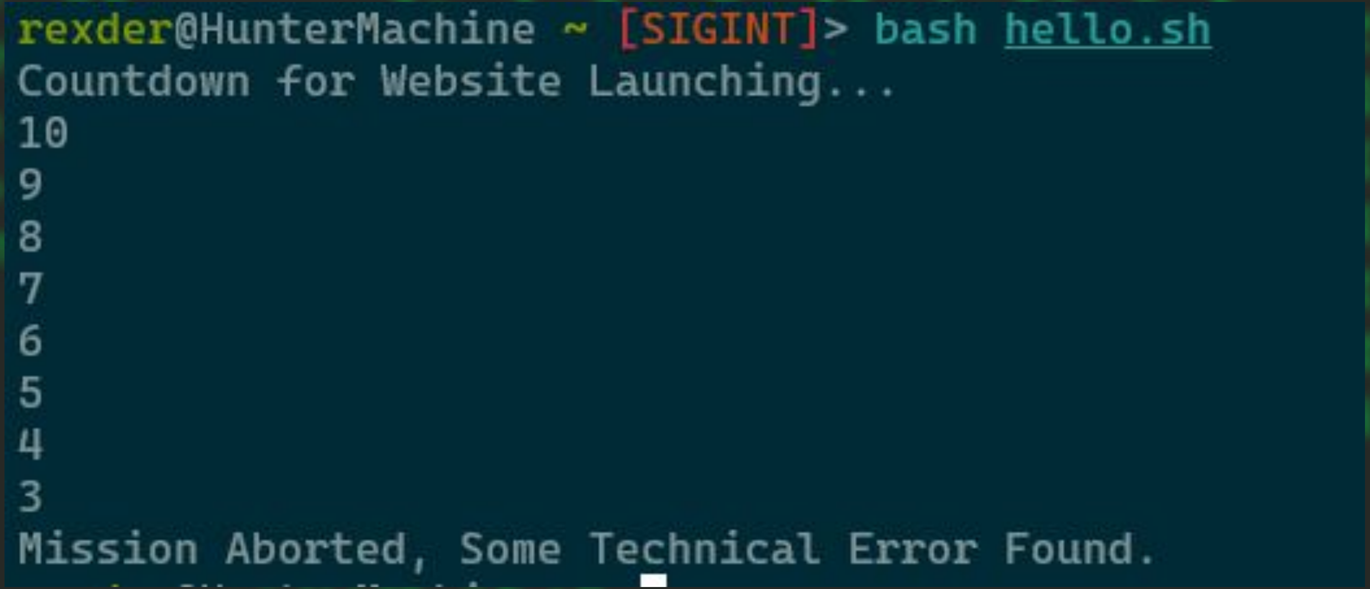
```
Welcome to GeezTech.
Welcome to GeezTech.
Welcome to GeezTech.
Welcome to GeezTech.
Welcome to GeezTech.
Welcome to GeezTech.
Welcome to GeezTech.
Welcome to GeezTech.
Welcome to GeezTech.
Welcome to GeezTech.
Welcome to GeezTech.
Welcome to GeezTech.
Welcome to GeezTech.
Welcome to GeezTech.
Welcome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

Break Statement



```bash
#!/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
```

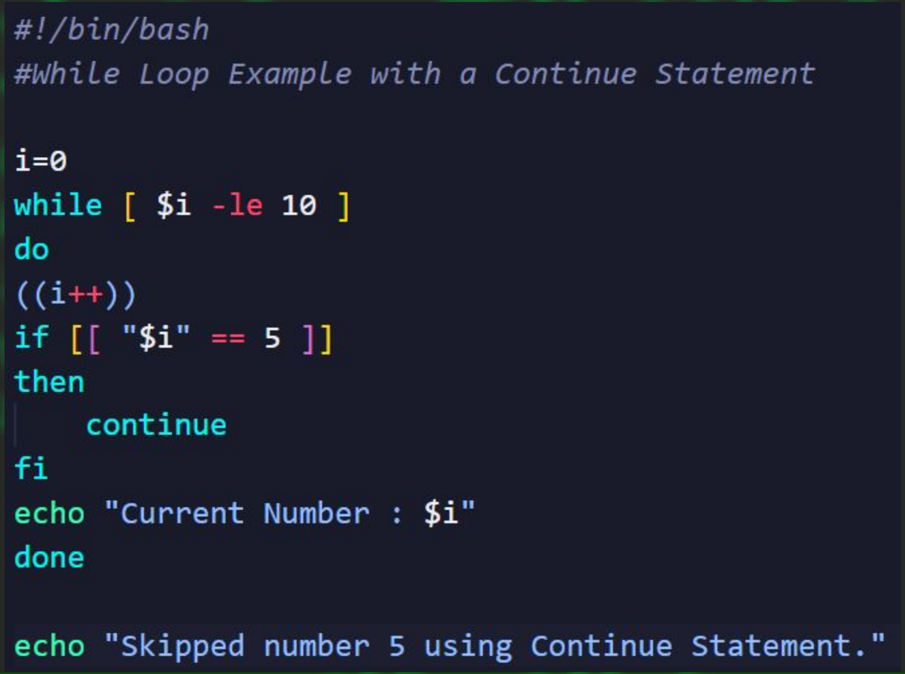
...Output



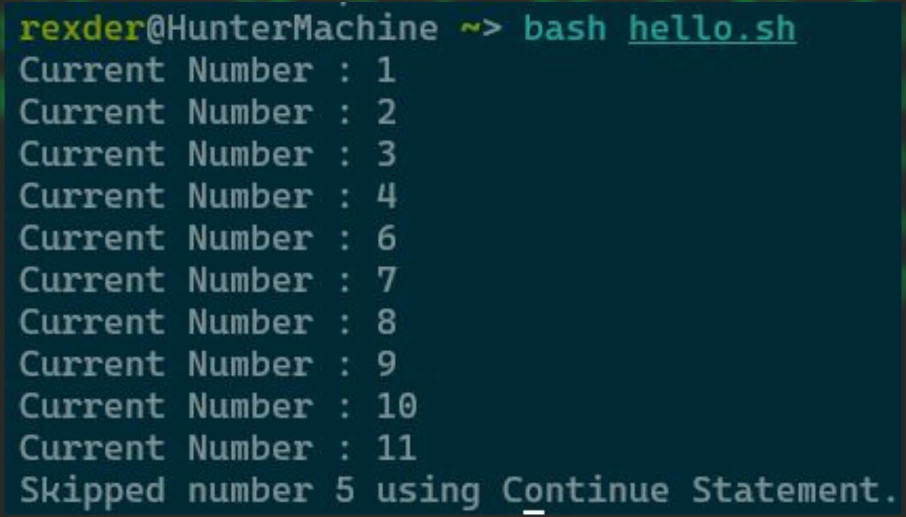
```
rexder@HunterMachine ~ [SIGINT]> bash hello.sh
Countdown for Website Launching...
10
9
8
7
6
5
4
3
Mission Aborted, Some Technical Error Found.
```

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."
```



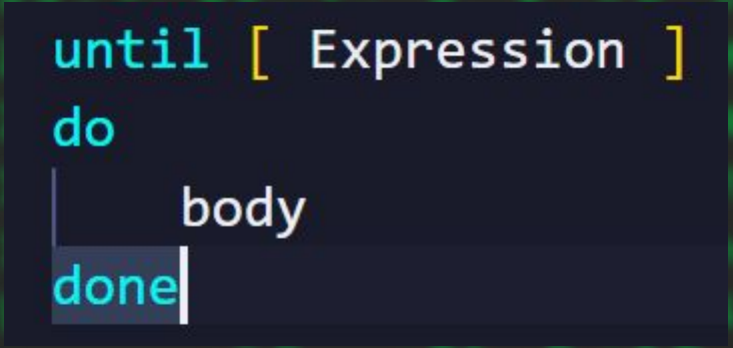
```
rexder@HunterMachine ~> bash hello.sh
Current Number : 1
Current Number : 2
Current Number : 3
Current Number : 4
Current Number : 6
Current Number : 7
Current Number : 8
Current Number : 9
Current Number : 10
Current Number : 11
Skipped number 5 using Continue Statement.
```

Exercise 2

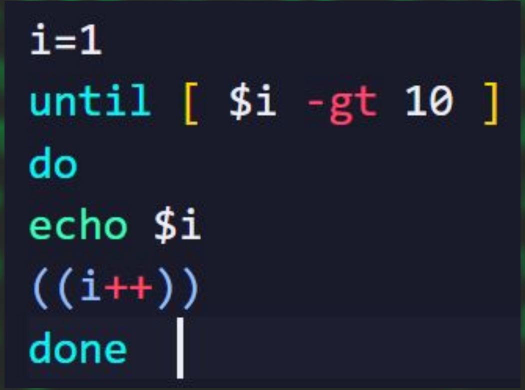
1) Create a loop that prints your name 10 times
2) Accept 2 input from user 1 his name and 2 amount of repeat he needs and display it for the size he wanted to repeat.
3) Create a infinite loop that displays, “You are BASH MASTER!!!”

Until loop

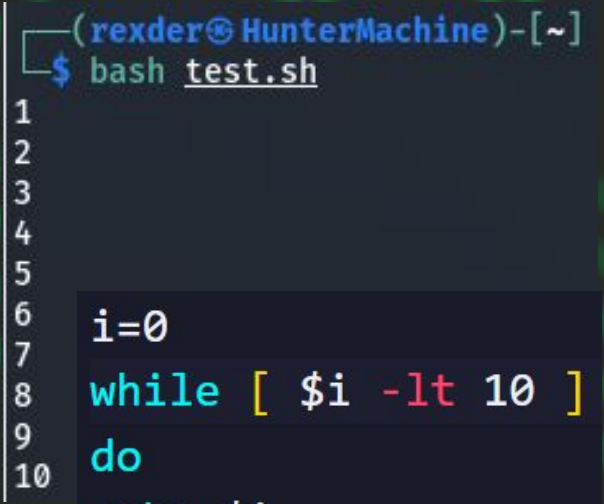
- It is same but this one exits the loop when the expression is true.
- The name until means እስከ



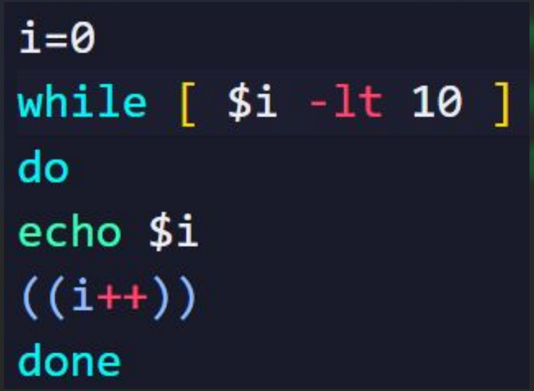
```
until [ Expression ]
do
    body
done
```



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

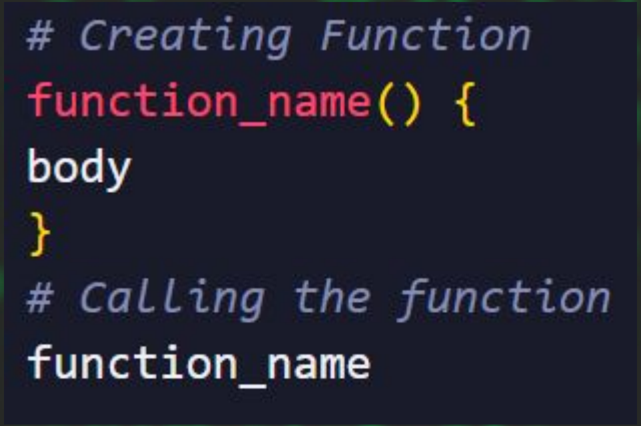


```
┌──(rexder㉿HunterMachine)-[~]
└─$ bash test.sh
1
2
3
4
5
6
7
8
9
10
```

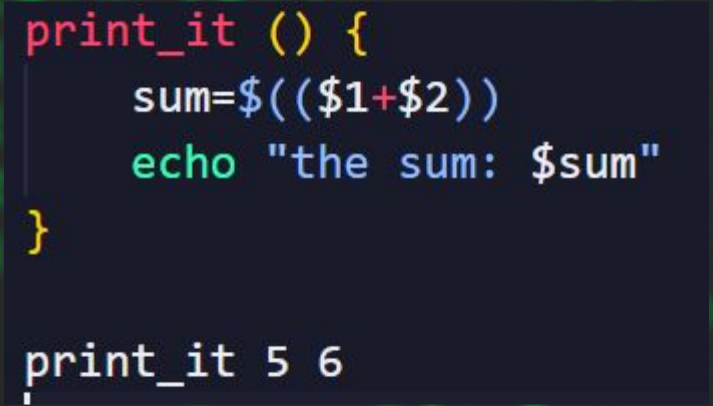


```
i=0
while [ $i -lt 10 ]
do
echo $i
((i++))
done
```

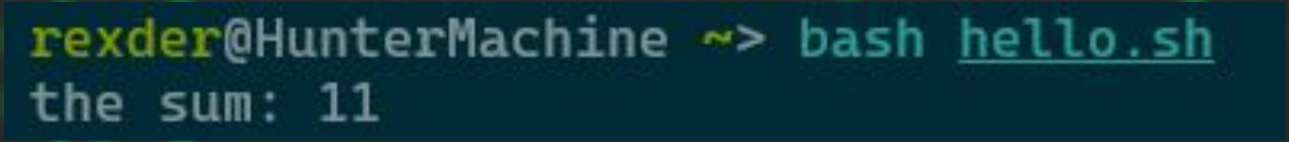
Function



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



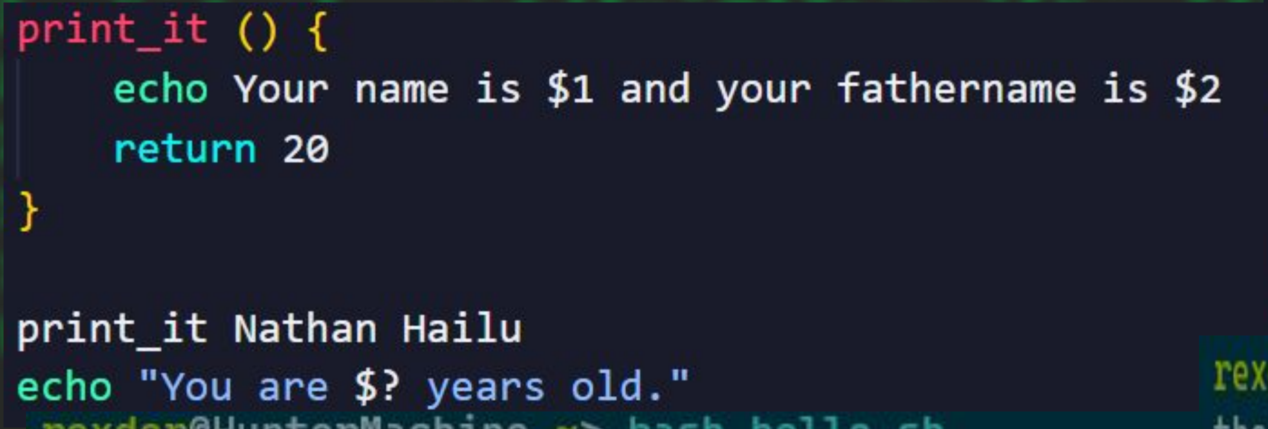
```
print_it () {
    sum=$(($1+$2))
    echo "the sum: $sum"
}

print_it 5 6
```



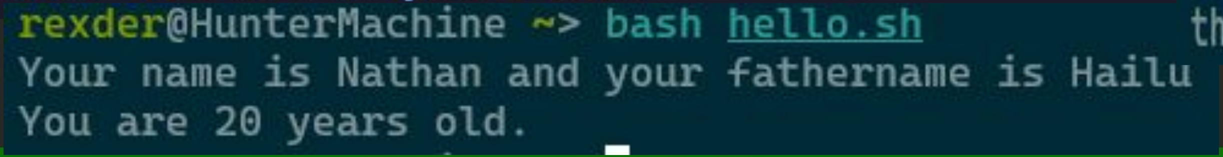
```
rexder@HunterMachine ~> bash hello.sh
the sum: 11
```

Cont...

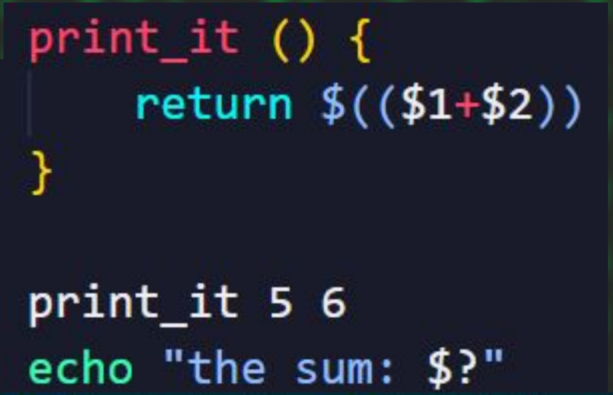
- When you have function and the arguments will be $1, $2 ....
- If we use $? it will call the return value.



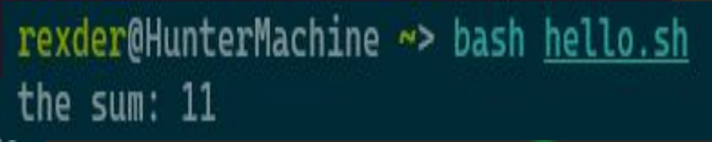
```
print_it () {
    echo Your name is $1 and your fathername is $2
    return 20
}

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.
```



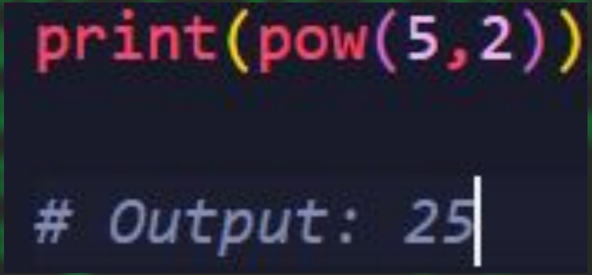
```
print_it () {
    return $(($1+$2))
}

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. It u do pow(4,2) => it gives 16 , means the power.

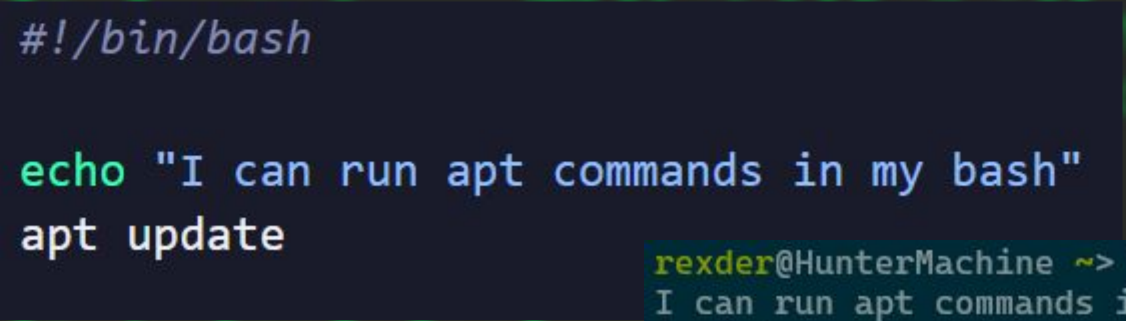


```
print(pow(5,2))

# Output: 25
```

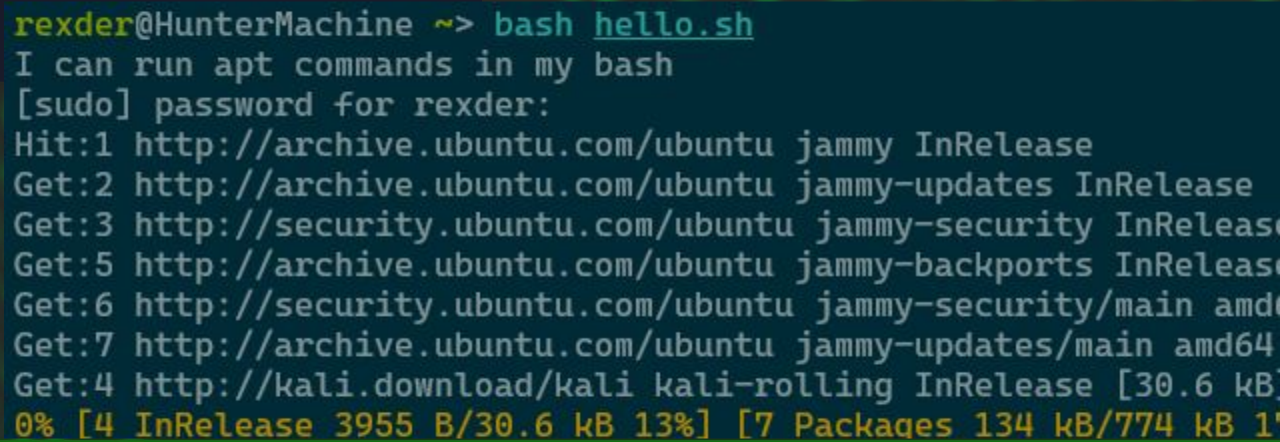
1) Create A function
2) Accept to arguments
3) Do a calculation to power
4) Return the value
5) Call the function and print it.

BASH and Linux

- We can run linux commands inside our bash



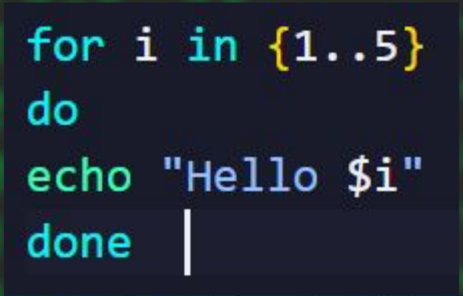
```
#!/bin/bash

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



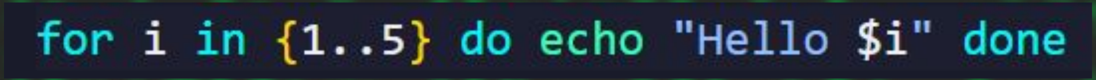
```
rexder@HunterMachine ~> bash hello.sh
I can run apt commands in my bash
[sudo] password for rexder:
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Get:3 http://security.ubuntu.com/ubuntu jammy-security InReleas
Get:5 http://archive.ubuntu.com/ubuntu jammy-backports InReleas
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd
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 1
```

Cont...

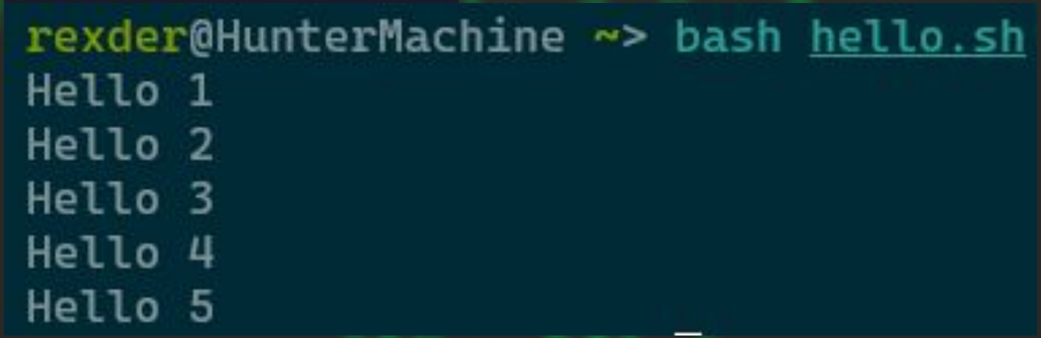
- You can run bash codes in 1 line.



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



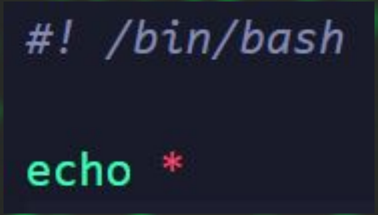
```
for i in {1..5} do echo "Hello $i" done
```



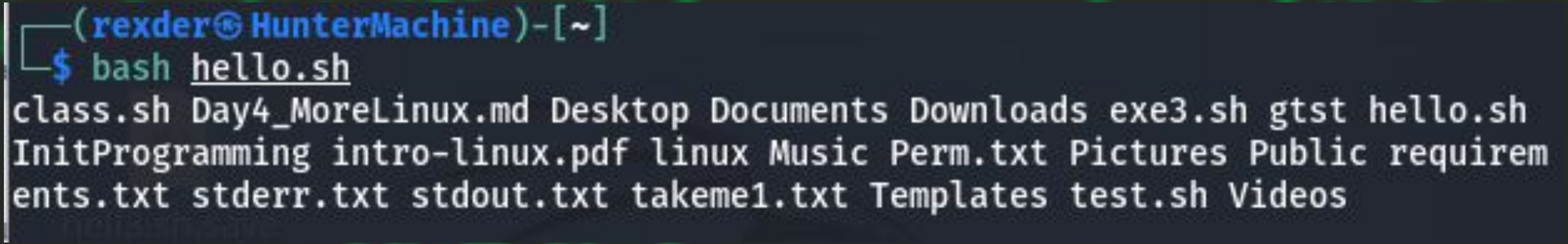
```
rexder@HunterMachine ~> bash hello.sh
Hello 1
Hello 2
Hello 3
Hello 4
Hello 5
```

Cont...

- You can access files in current directory using * sign



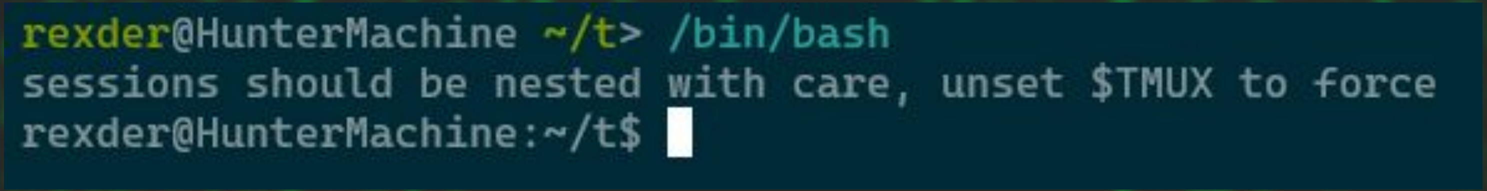
```
#! /bin/bash

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

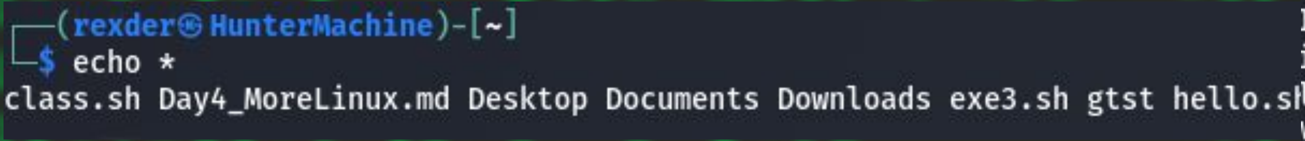
- You can interact with linux terminal using bash.
- For this our codes are bash so, your terminal have to be on bash
- Command: /bin/bash



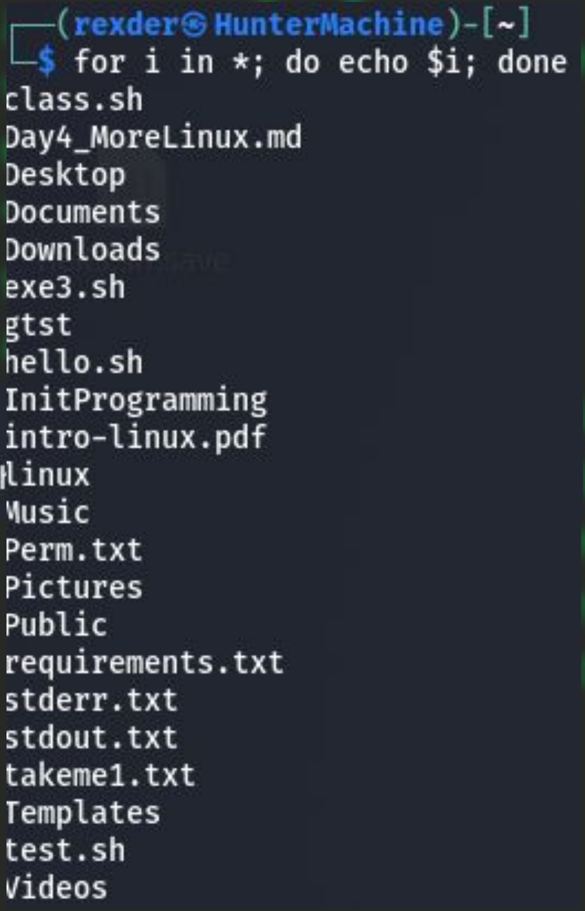
```
rexder@HunterMachine ~/t> /bin/bash
sessions should be nested with care, unset $TMUX to force
rexder@HunterMachine:~/t$ █
```

For loop on terminal

Syntax: `for command; do body; done`



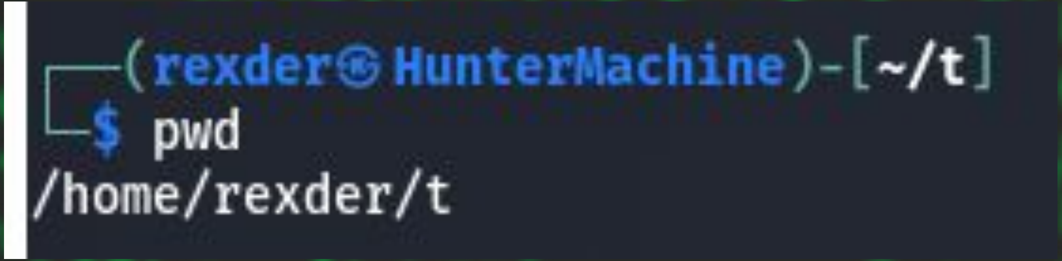
```
┌──(rexder㉿HunterMachine)-[~]
└─$ echo *
class.sh Day4_MoreLinux.md Desktop Documents Downloads exe3.sh gtst hello.sh
```



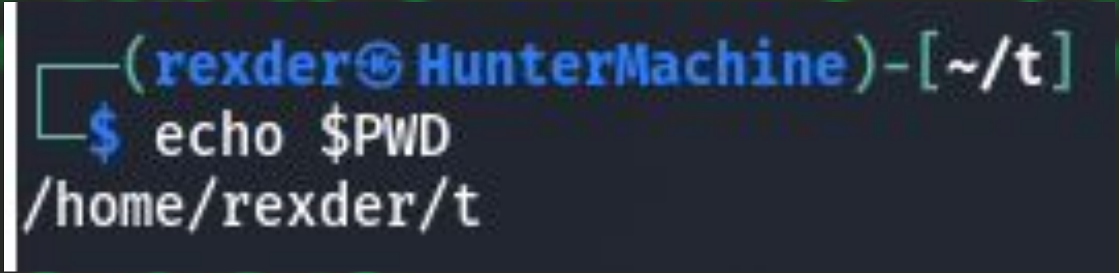
```
┌──(rexder㉿HunterMachine)-[~]
└─$ for i in *; do echo $i; done
class.sh
Day4_MoreLinux.md
Desktop
Documents
Downloads
exe3.sh
gtst
hello.sh
InitProgramming
intro-linux.pdf
linux
Music
Perm.txt
Pictures
Public
requirements.txt
stderr.txt
stdout.txt
takeme1.txt
Templates
test.sh
Videos
```

Tasks #1

Do the pwd command by your own.



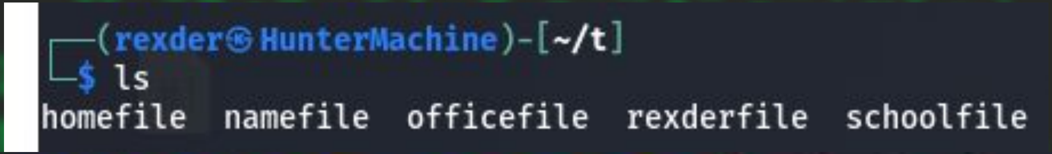
```
┌──(rexder㉿HunterMachine)-[~/t]
└─$ pwd
/home/rexder/t
```



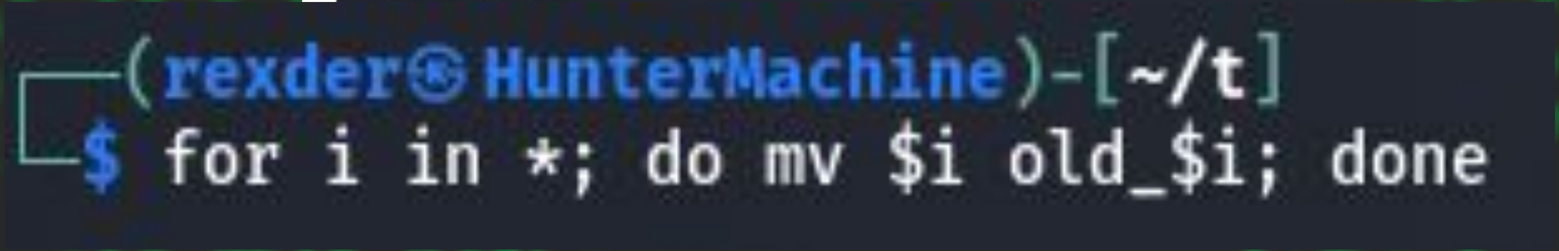
```
┌──(rexder㉿HunterMachine)-[~/t]
└─$ echo $PWD
/home/rexder/t
```

Tasks #2

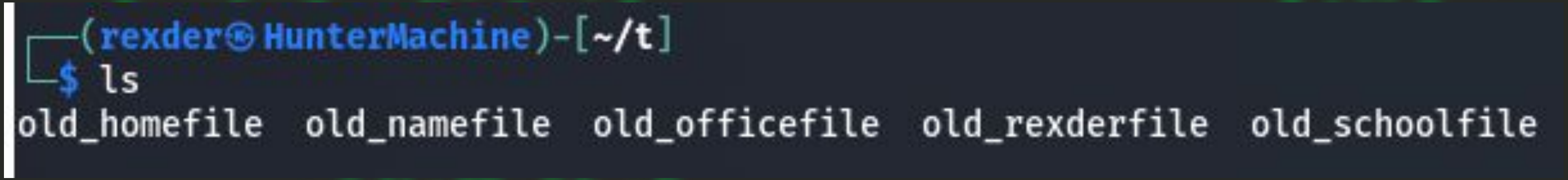
Adding “old” text before filenames.



```
┌──(rexder㉿HunterMachine)-[~/t]
└─$ ls
homefile  namefile  officefile  rexderfile  schoolfile
```



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



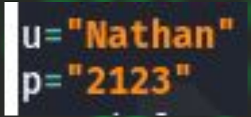
```
┌──(rexder㉿HunterMachine)-[~/t]
└─$ ls
old_homefile  old_namefile  old_officefile  old_rexderfile  old_schoolfile
```

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