

BASH SCRIPTING PROJECT 1

BASH FOR LOOP EXAMPLE -1

The image shows a Windows desktop environment with two terminal windows open. The top window is titled 'bashforloop' and contains a Bash script. The bottom window shows the execution of the script and its output.

Top Terminal Window (bashforloop):

```
karthik@f542ba3b5a78573:~$ nano bashforloop
GNU nano 7.2
#!/bin/bash
#This is the basic example of 'for loop'.
learn="Start learning from Javatpoint."
for learn in $learn
do
echo $learn
done
echo "Thank You."
```

Bottom Terminal Window:

```
karthik@f542ba3b5a78573:~$ rm penauts.txt reading
karthik@f542ba3b5a78573:~$ nano bashforloop
karthik@f542ba3b5a78573:~$ chmod a+x bashforloop
karthik@f542ba3b5a78573:~$ ./bashforloop
Start
learning
from
Javatpoint.
Thank You.
karthik@f542ba3b5a78573:~$ nano bashforloop
karthik@f542ba3b5a78573:~$ nano bashforloop1
karthik@f542ba3b5a78573:~$ chmod +x bashforloop1
karthik@f542ba3b5a78573:~$ ./bashforloop1
1
2
3
4
5
6
7
8
9
10
Series of numbers from 1 to 10.
```

BASH FOR LOOP WITH RANGE EXAMPLE -2

A screenshot of a Windows desktop environment. At the top, there's a taskbar with icons for File Explorer, Edge browser, Mail, and others. The system tray shows the date as 27-01-2025 and the time as 13:59. In the center, a terminal window titled "bashforloop1" is open in the "GNU nano 7.2" editor. It contains the following code:

```
#!/bin/bash
# This is the basic example to print a series of numbers from 1 to 10.
for num in {1..10}
do
    echo $num
done
echo "Series of numbers from 1 to 10."
```

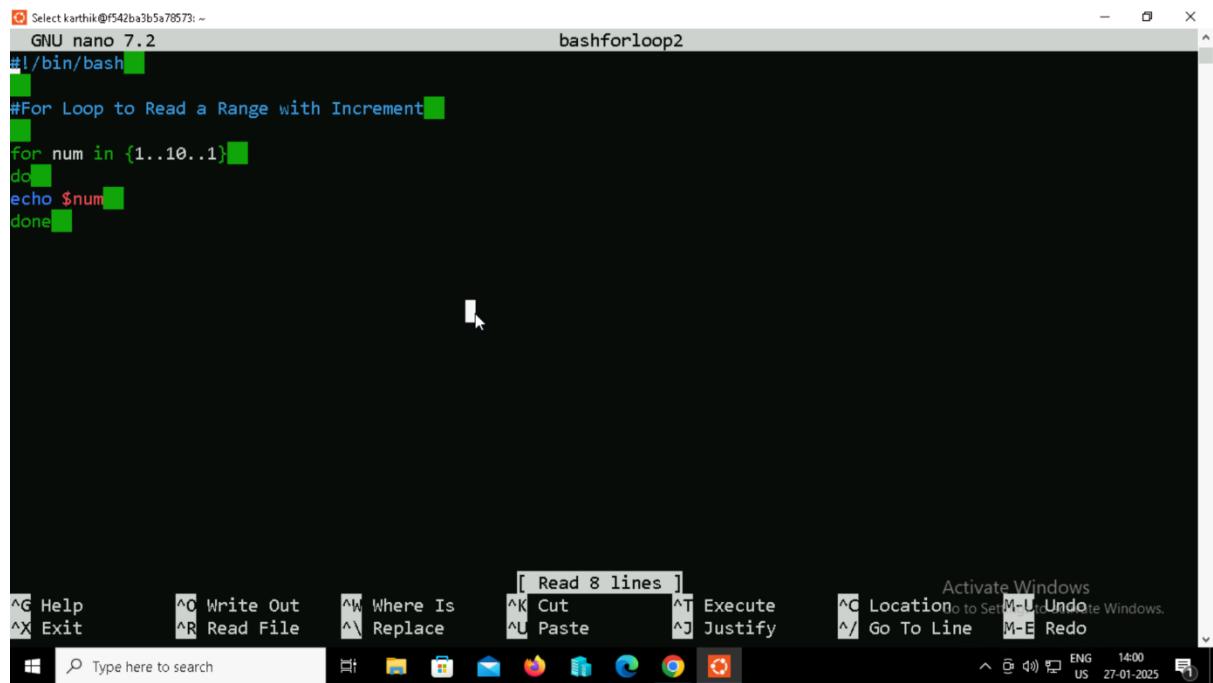
The terminal window has a menu bar at the bottom with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Go To Line, and Redo. A status bar at the bottom right shows "Activate Windows" and "Go to Settings to activate Windows".

A screenshot of a Windows desktop environment, similar to the one above. The taskbar and system tray are identical. The terminal window now displays the output of the script:

```
Series of numbers from 1 to 10.
karthik@f542ba3b5a78573:~$ chmod +x bashforloop1
karthik@f542ba3b5a78573:~$ ./bashforloop1
1
2
3
4
5
6
7
8
9
10
Series of numbers from 1 to 10.
karthik@f542ba3b5a78573:~$
```

The terminal window interface is the same as in the first screenshot, with its own menu bar and status bar.

BASH FOR LOOP INCREMENT WITH RANGE EXAMPLE -3

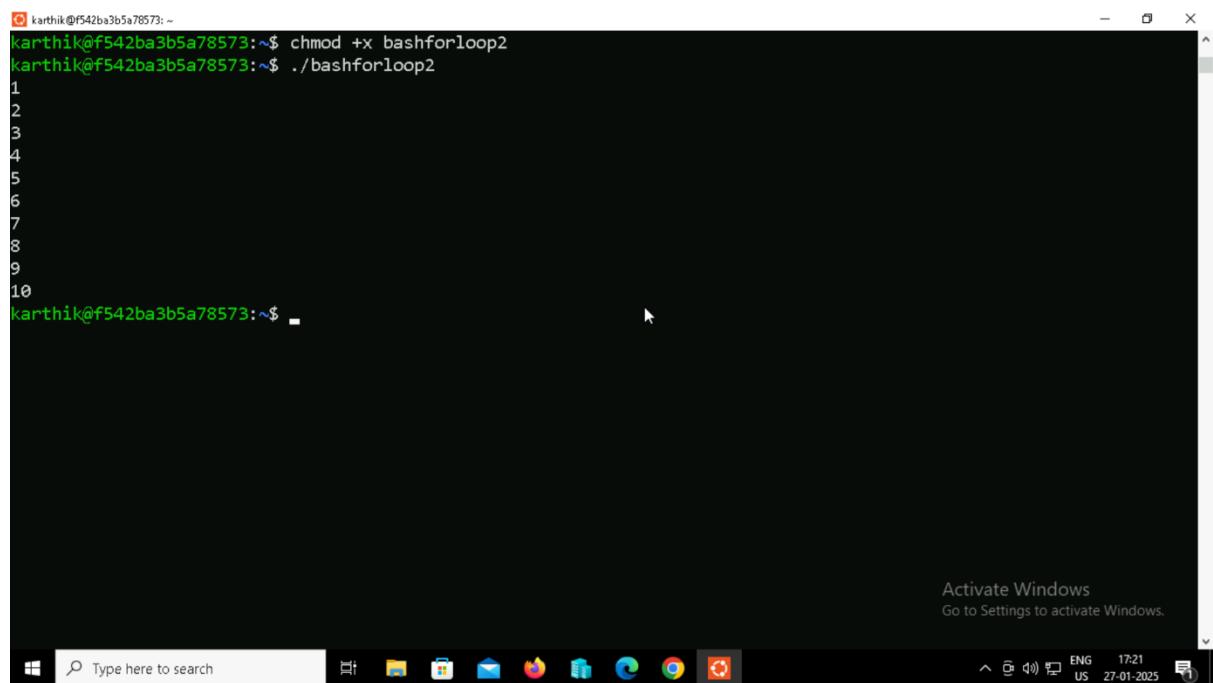


```
>Select karthik@f542ba3b5a78573: ~
GNU nano 7.2
#!/bin/bash
#For Loop to Read a Range with Increment
for num in {1..10..1}
do
echo $num
done
```

The screenshot shows a Windows desktop environment with a terminal window titled "bashforloop2". The terminal is running the nano text editor. Inside the editor, there is a single line of code:

```
for num in {1..10..1}
```

The terminal window has a menu bar at the top with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, and Activate Windows. Below the menu is a search bar and a taskbar with various icons.



```
karthik@f542ba3b5a78573:~$ chmod +x bashforloop2
karthik@f542ba3b5a78573:~$ ./bashforloop2
1
2
3
4
5
6
7
8
9
10
```

The screenshot shows a Windows desktop environment with a terminal window. The terminal window displays the output of a bash script named "bashforloop2". The output consists of the numbers 1 through 10, each on a new line. The terminal window has a menu bar at the top with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, and Activate Windows. Below the menu is a search bar and a taskbar with various icons.

BASH FOR LOOP DECREMENT WITH RANGE EXAMPLE -3

A screenshot of a Windows desktop environment. In the center is a terminal window titled 'Terminal' with the command prompt 'karthik@f542ba3b5a78573:~\$'. The terminal displays the output of a bash script named 'bashforloop3'. The script uses a for loop to print numbers from 10 down to 0. The terminal window has a dark background and white text. At the bottom of the screen is a taskbar with various icons for apps like File Explorer, Edge, and Google Chrome. On the right side of the screen, there is a system tray with icons for battery, signal, and volume, along with the date and time '27-01-2025 14:01'. A watermark 'Activate Windows Go to Settings to activate Windows.' is visible in the middle-right area of the desktop.

```
karthik@f542ba3b5a78573:~$ nano bashforloop3
karthik@f542ba3b5a78573:~$ chmod +x bashforloop3
karthik@f542ba3b5a78573:~$ ./bashforloop3
10
9
8
7
6
5
4
3
2
1
0
karthik@f542ba3b5a78573:~$
```

A screenshot of a Windows desktop environment, identical to the one above. It features a terminal window titled 'Terminal' with the command prompt 'karthik@f542ba3b5a78573:~\$'. This terminal shows the execution of a bash script named 'bashforloop4'. The script prints 'Welcome to Javatpoint' followed by a blank line. The terminal window has a dark background and white text. The taskbar at the bottom includes icons for File Explorer, Edge, and Google Chrome. The system tray on the right shows the date and time '27-01-2025 16:49'. A watermark 'Activate Windows Go to Settings to activate Windows.' is present in the middle-right of the desktop.

```
karthik@f542ba3b5a78573:~$ nano bashforloop4
karthik@f542ba3b5a78573:~$ ./bashforloop4
karthik@f542ba3b5a78573:~$ nano bashforloop4
karthik@f542ba3b5a78573:~$ chmod +x bashforloop4
karthik@f542ba3b5a78573:~$ ./bashforloop4
Welcome
to
Javatpoint
karthik@f542ba3b5a78573:~$
```

BASH FOR LOOP READ ARRAY VARIABLE

A screenshot of a Windows terminal window titled "bashforloop4". The window shows a terminal session with the command "karthik@f542ba3b5a78573:~\$". Inside the terminal, a file named "bashforloop4" is being edited in the "GNU nano 7.2" editor. The script contains the following code:

```
#!/bin/bash
# Array Declaration
arr=( "Welcome" "to" "Javatpoint" )
for i in "${arr[@]}"
do
    echo $i
done
```

The terminal window includes standard Windows-style keyboard shortcuts at the bottom, such as **^G Help**, **^X Exit**, **^O Write Out**, **^R Read File**, **^W Where Is**, **^\\ Replace**, **^K Cut**, **^U Paste**, **^T Execute**, **^J Justify**, **^C Location**, **^/ Go To Line**, **M-U Undo**, **M-R Redo**, and a search bar at the bottom left.

A screenshot of a Windows terminal window showing the execution of the "bashforloop4" script. The terminal session starts with "karthik@f542ba3b5a78573:~\$". The user runs the command `./bashforloop4`. The output of the script is displayed in the terminal:

```
Welcome
to
Javatpoint
```

The terminal window includes standard Windows-style keyboard shortcuts at the bottom, such as **^G Help**, **^X Exit**, **^O Write Out**, **^R Read File**, **^W Where Is**, **^\\ Replace**, **^K Cut**, **^U Paste**, **^T Execute**, **^J Justify**, **^C Location**, **^/ Go To Line**, **M-U Undo**, **M-R Redo**, and a search bar at the bottom left. A watermark for "Activate Windows" is visible in the center of the terminal window.

BASH FOR LOOP READ WHITE SPACES IN STRING AS WORD SEPARATOR

The image shows a Windows desktop environment with two terminal windows open. Both terminals are running on a Linux system (Ubuntu) via a virtual machine or container.

Top Terminal:

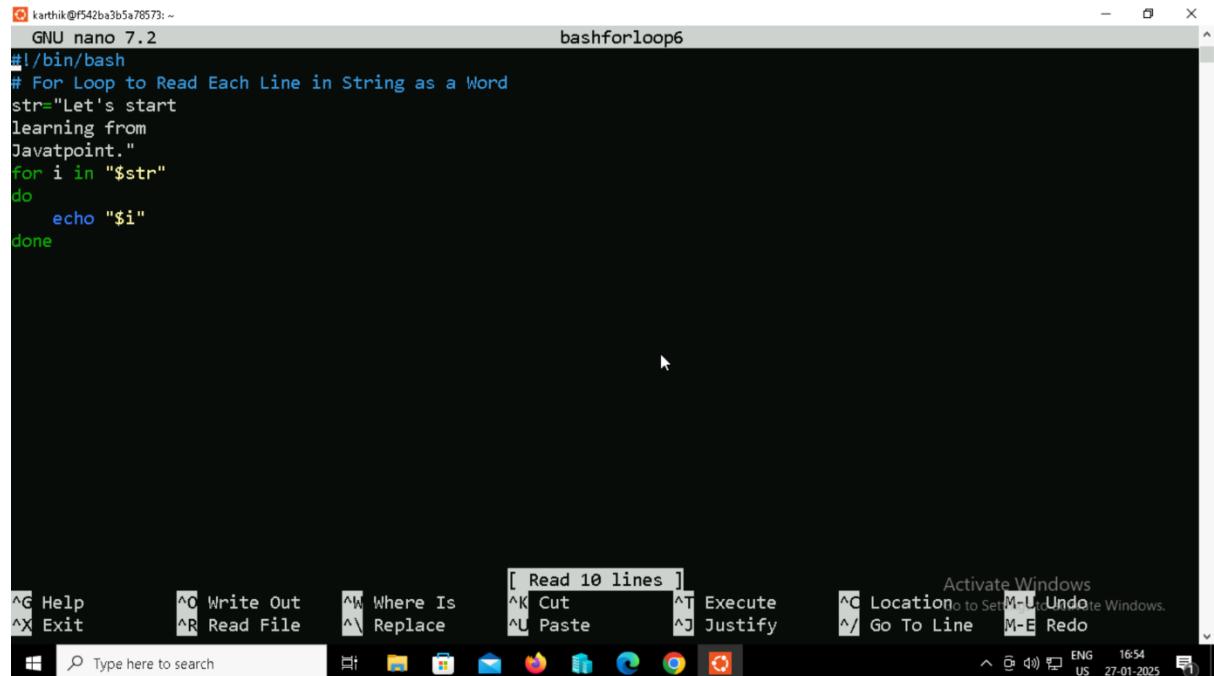
```
karthik@f542ba3b5a78573:~$ nano bashforloop3
9
10
karthik@f542ba3b5a78573:~$ nano bashforloop5
karthik@f542ba3b5a78573:~$ chmod +x bashforloop5
karthik@f542ba3b5a78573:~$ ./bashforloop5
Let's
start
learning
from
Javatpoint.
karthik@f542ba3b5a78573:~$
```

Bottom Terminal:

```
karthik@f542ba3b5a78573:~$ GNU nano 7.2          bashforloop5
#!/bin/bash
# For Loop to Read White Spaces in String as Word Separators
str="Let's start learning from Javatpoint."
for i in $str
do
    echo "$i"
done
```

The bottom terminal window shows the script being edited. The status bar indicates "Read 8 lines". The desktop taskbar at the bottom includes icons for File Explorer, Task View, Mail, Edge, and Google Chrome. The system tray shows battery level, signal strength, and the date/time (27-01-2025). A watermark for "Activate Windows" is visible in the center of the screen.

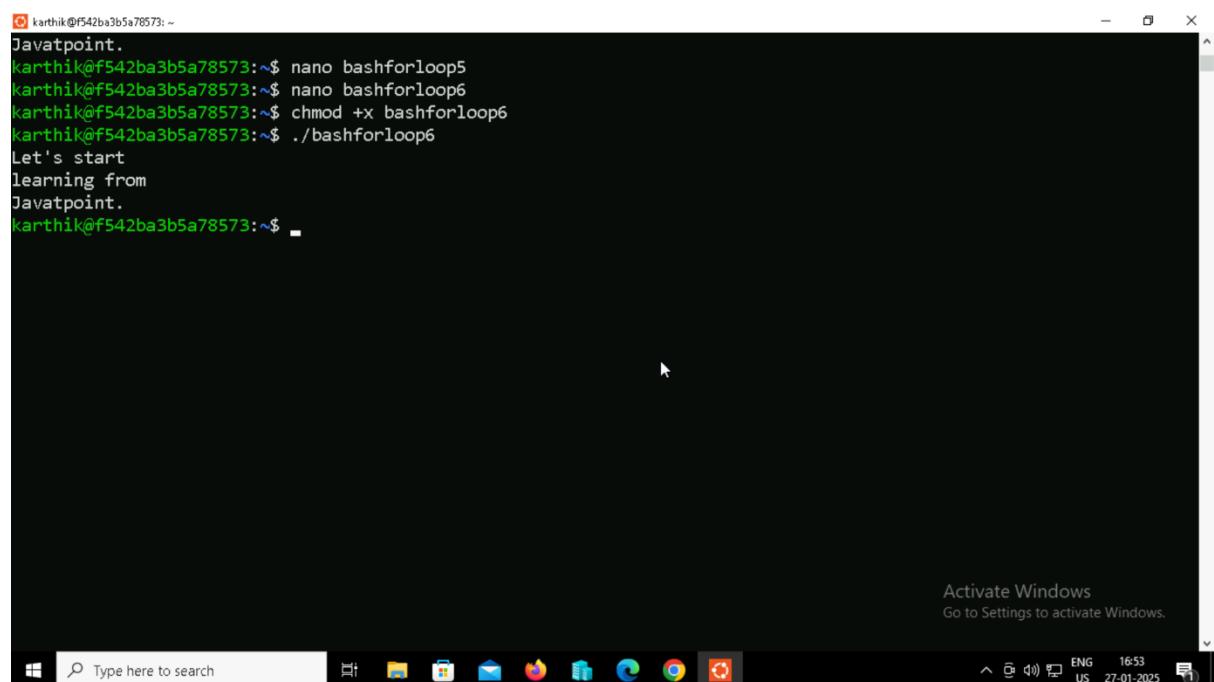
BASH FOR LOOP TO READ EACH LINE IN STRING AS WORD



A screenshot of a Windows desktop environment showing a terminal window. The terminal window title is "bashforloop6". Inside the window, a bash script is displayed:

```
GNU nano 7.2
#!/bin/bash
# For Loop to Read Each Line in String as a Word
str="Let's start
learning from
Javatpoint."
for i in "$str"
do
    echo "$i"
done
```

The terminal window has a menu bar at the top with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, and Redo. Below the menu bar is a toolbar with icons for file operations. At the bottom of the terminal window, there is a status bar showing "Activate Windows Go to Settings to activate Windows.", the date "27-01-2025", and the time "16:54".

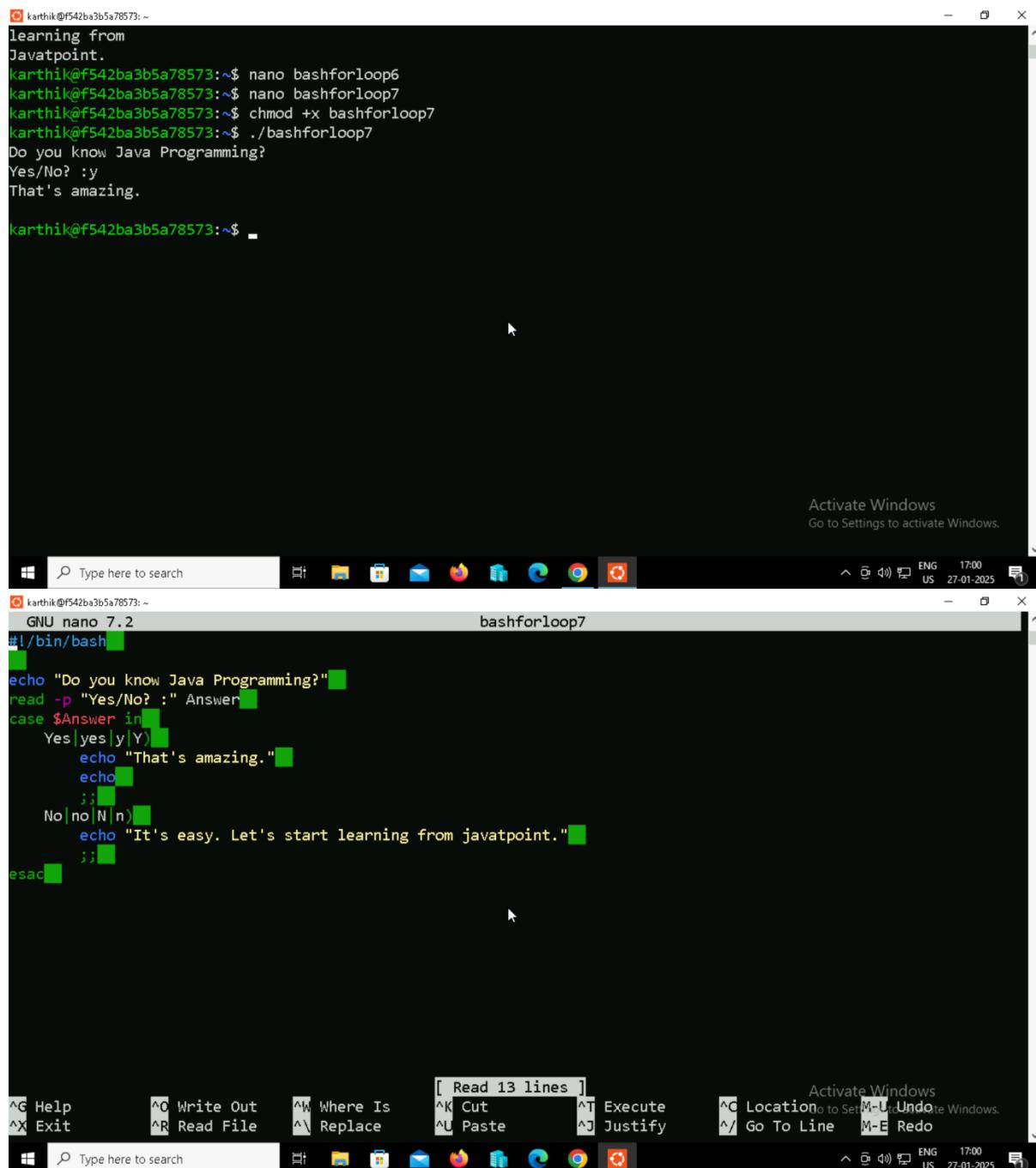


A screenshot of a Windows desktop environment showing a terminal window. The terminal window title is "bashforloop6". Inside the window, the output of the bash script is shown:

```
[Read 10 lines]
karthik@f542ba3b5a78573:~$ nano bashforloop5
karthik@f542ba3b5a78573:~$ nano bashforloop6
karthik@f542ba3b5a78573:~$ chmod +x bashforloop6
karthik@f542ba3b5a78573:~$ ./bashforloop6
Let's start
learning from
Javatpoint.
karthik@f542ba3b5a78573:~$
```

The terminal window has a menu bar at the top with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, and Redo. Below the menu bar is a toolbar with icons for file operations. At the bottom of the terminal window, there is a status bar showing "Activate Windows Go to Settings to activate Windows.", the date "27-01-2025", and the time "16:53".

BASH CASE EXAMPLE -1



The screenshot shows a Windows desktop environment with three windows open. The top window is a terminal window titled 'bashforloop7' with the command 'nano bashforloop7' running. It displays a Bash script that asks if the user knows Java Programming and provides an amazing response if 'y' or 'Y' is chosen. The middle window is another terminal window titled 'bashforloop7' with the command 'nano bashforloop7' running, showing the same script. The bottom window is a file explorer window titled 'File Explorer' showing a folder structure.

```
karthik@f542ba3b5a78573:~$ nano bashforloop6
learning from
Javatpoint.
karthik@f542ba3b5a78573:~$ nano bashforloop7
karthik@f542ba3b5a78573:~$ chmod +x bashforloop7
karthik@f542ba3b5a78573:~$ ./bashforloop7
Do you know Java Programming?
Yes/No? :y
That's amazing.

karthik@f542ba3b5a78573:~$
```

```
GNU nano 7.2
#!/bin/bash

echo "Do you know Java Programming?" 
read -p "Yes/No? :" Answer
case $Answer in
  Yes|yes|y|Y)
    echo "That's amazing."
    echo ;;
  No|no|N|n)
    echo "It's easy. Let's start learning from javatpoint."
    ;;
esac
```

BASH CASE EXAMPLE -2

```
karthik@f542ba3b5a78573: ~$ nano bashcase1
karthik@f542ba3b5a78573: ~$ chmod +x bashcase1
karthik@f542ba3b5a78573: ~$ ./bashcase1
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name:Linux
You might be serious about security!!

karthik@f542ba3b5a78573: ~$ ./bashcase1
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name:Chrome
Cool!!! It's for pro users. Amazing Choice.

karthik@f542ba3b5a78573: ~$ ./bashcase1
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name:Android
This is my favorite. It has lots of applications.

karthik@f542ba3b5a78573: ~$ ./bashcase1
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name:Windows
That's common. You should try something new.

karthik@f542ba3b5a78573: ~$ bashcase1
Select karthik@f542ba3b5a78573: ~
GNU nano 7.2                                bashcase1
#!/bin/bash

echo "Which Operating System are you using?"
echo "Windows, Android, Chrome, Linux, Others?"
read -p "Type your OS Name:" OS

case $OS in
    Windows|windows)
        echo "That's common. You should try something new."
        echo ;;
    ;;
    Android|android)
        echo "This is my favorite. It has lots of applications."
        echo ;;
    ;;
    Chrome|chrome)
        echo "Cool!!! It's for pro users. Amazing Choice."
        echo ;;
    ;;
    Linux|linux)
        echo "You might be serious about security!!"
        echo ;;
    ;;
esac

Activate Windows
Go to Settings to activate Windows.

^G Help      ^O Write Out     ^W Where Is      ^K Cut          ^T Execute
^X Exit      ^R Read File     ^\ Replace       ^U Paste         ^J Justify
^C Location   ^L Go To Line    ^A Undo          ^V Redo          Activate Windows
^/ Go To Line M-E Redo      Go to Settings to activate Windows.
```

BASH WHILE LOOP EXAMPLES

BASH WHILE LOOP WITH SINGLE CONDITIONS

The screenshot shows a Windows desktop environment with two terminal windows open. Both terminals are running on a Linux system (Ubuntu 22.04 LTS) via WSL (Windows Subsystem for Linux).

Top Terminal:

```
karthik@f542ba3b5a78573:~$ nano bashforloop3
karthik@f542ba3b5a78573:~$ nano bashforloop5
karthik@f542ba3b5a78573:~$ chmod +x bashforloop5
karthik@f542ba3b5a78573:~$ ./bashforloop5
Let's
start
learning
from
Javatpoint.
karthik@f542ba3b5a78573:~$ nano whileloop1
karthik@f542ba3b5a78573:~$ chmod a+x whileloop1
karthik@f542ba3b5a78573:~$ ./whileloop1
Enter starting number: 5
Enter ending number: 15
5
6
7
8
9
10
11
12
13
14
15
This is the sequence that you wanted.
karthik@f542ba3b5a78573:~$
```

Bottom Terminal:

```
GNU nano 7.2                               whileloop1
#!/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."
```

Both terminals show the same script content, which is a Bash script named `whileloop1` that prints a sequence of numbers from a user-specified start to an end value. The top terminal shows the execution of the script, while the bottom terminal shows the script being edited.

BASH WHILE LOOP WITH MULTIPLE CONDITIONS

```
karthik@f542ba3b5a78573:~  
This is the sequence that you wanted.  
karthik@f542ba3b5a78573:~$ nano whileloop1  
karthik@f542ba3b5a78573:~$ sed -i 's/[:blank:]\+$//' filename  
sed: can't read filename: No such file or directory  
karthik@f542ba3b5a78573:~$ sed -i 's/[:blank:]\+$//' filename  
sed: can't read filename: No such file or directory  
karthik@f542ba3b5a78573:~$ sed -i 's/[:blank:]\+$//' whileloop1  
karthik@f542ba3b5a78573:~$ ./whileloop1  
Enter starting number: ^C  
karthik@f542ba3b5a78573:~$ nano whileloop1  
karthik@f542ba3b5a78573:~$ nano whileloop2  
karthik@f542ba3b5a78573:~$ chmod a+x whileloop2  
karthik@f542ba3b5a78573:~$ ./whileloop2  
Enter starting number: 1  
Enter ending number: 10  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
This is the sequence that you wanted.  
karthik@f542ba3b5a78573:~$   
  
Windows Type here to search ENG 09:29  
Activate Windows Go to Settings to activate Windows.  
US 28-01-2025  
  
karthik@f542ba3b5a78573:~  
GNU nano 7.2 whileloop2  
#!/bin/bash  
#Script to get specified numbers  
  
read -p "Enter starting number: " snum  
read -p "Enter ending number: " enum  
  
while [[ $snum -lt $enum || $snum == $enum ]];  
do  
echo $snum  
((snum++))  
done  
  
echo "This is the sequence that you wanted."  
  
[ Read 13 lines ]  
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location ^U Undo  
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line ^M-E Redo  
Windows Type here to search ENG 09:30  
Activate Windows Go to Settings to activate Windows.  
US 28-01-2025
```

BASH INFINITE WHILE LOOP CONDITION

The image shows a Windows desktop environment with two terminal windows open. The top window is a terminal session titled "whileloop3" running in a "GNU nano 7.2" editor. It contains the following code:

```
#!/bin/bash
#An infinite while loop

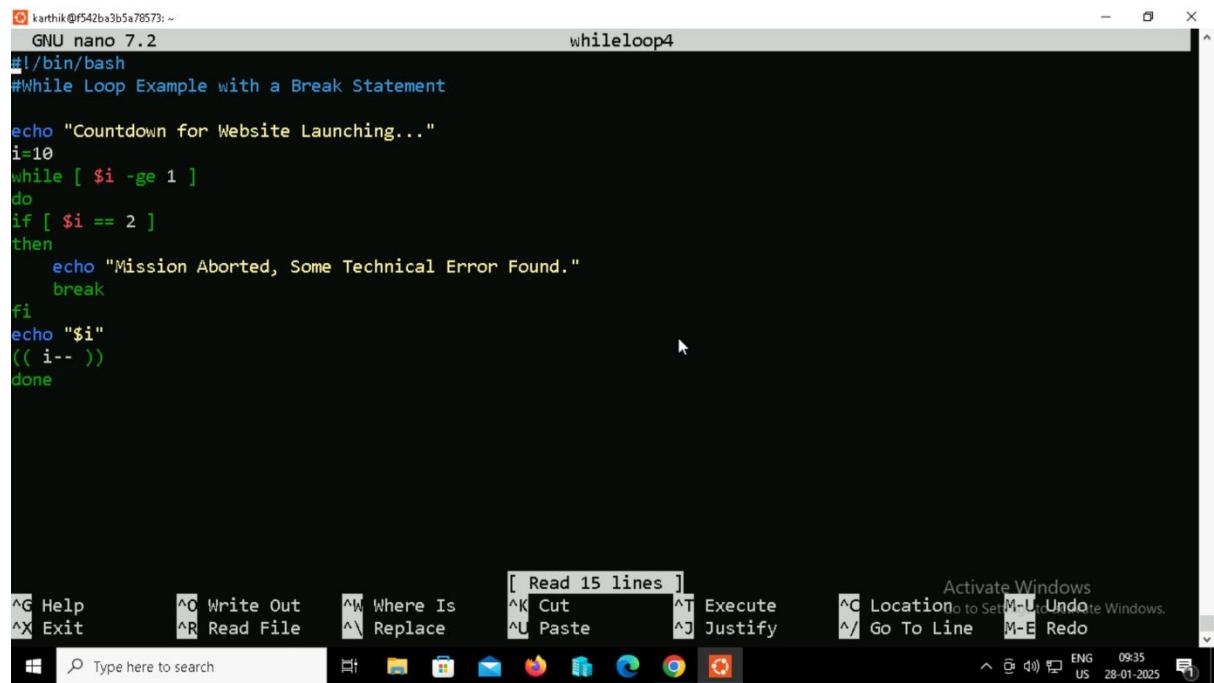
while :
do
echo "Welcome to Javatpoint."
done
```

The bottom window is another terminal session. It shows the output of the script being run, which is the continuous repetition of the message "Welcome to Javatpoint.". The terminal prompt at the bottom indicates the user has pressed ^C to stop the loop.

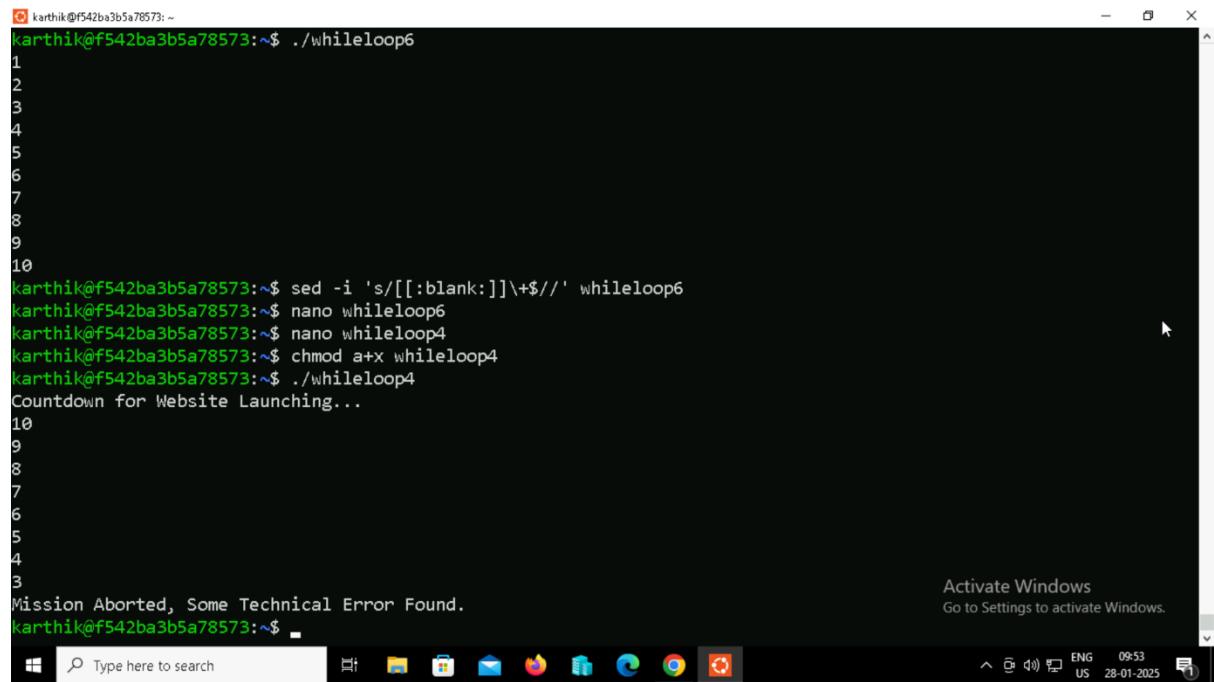
```
karthik@f542ba3b5a78573:~$ sed -i 's/[:blank:]$/+/`' whileloop3
karthik@f542ba3b5a78573:~$ nano whileloop3
karthik@f542ba3b5a78573:~$ chmod a+x whileloop3
karthik@f542ba3b5a78573:~$ ./whileloop3
```

Both terminals show a taskbar at the bottom with various application icons (File Explorer, Edge, Mail, etc.) and system status indicators (language: ENG, location: US, date: 28-01-2025, time: 09:32).

BASH WHILE LOOP WITH BREAK STATEMENT



```
karthik@f542ba3b5a78573:~  
GNU nano 7.2  
#!/bin/bash  
#While Loop Example with a Break Statement  
  
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
```



```
karthik@f542ba3b5a78573:~$ ./whileloop6  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
karthik@f542ba3b5a78573:~$ sed -i 's/[:blank:]\+\$//' whileloop6  
karthik@f542ba3b5a78573:~$ nano whileloop4  
karthik@f542ba3b5a78573:~$ chmod a+x whileloop4  
karthik@f542ba3b5a78573:~$ ./whileloop4  
Countdown for Website Launching...  
10  
9  
8  
7  
6  
5  
4  
3  
Mission Aborted, Some Technical Error Found.  
karthik@f542ba3b5a78573:~$
```

BASH WHILE LOOP WITH CONTINUE STATEMENT

The screenshot shows a Windows desktop environment with a terminal window open. The terminal window title is "whileloop5". Inside, a Bash script is displayed:

```
karthik@f542ba3b5a78573:~  
GNU nano 7.2  
#!/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."
```

The terminal window includes standard Windows-style menu options like Help, Exit, Write Out, Read File, etc., and a status bar at the bottom showing the date and time.

The screenshot shows the same Windows desktop environment with the terminal window now displaying the output of the script execution:

```
karthik@f542ba3b5a78573:~  
Countdown for Website Launching...  
10  
9  
8  
7  
6  
5  
4  
3  
Mission Aborted, Some Technical Error Found.  
karthik@f542ba3b5a78573:~$ sed -i 's/[:blank:]']\+$//'' whileloop4  
karthik@f542ba3b5a78573:~$ nano whileloop4  
karthik@f542ba3b5a78573:~$ nano whileloop5  
karthik@f542ba3b5a78573:~$ chmod a+x whileloop5  
karthik@f542ba3b5a78573:~$ ./whileloop5  
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.  
karthik@f542ba3b5a78573:~$
```

The terminal window includes standard Windows-style menu options and a status bar at the bottom showing the date and time.

BASH WHILE LOOP WITH C STYLE

```
karthik@f542ba3b5a78573:~$ Current Number : 1
Current Number : 2
Current Number : 3
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.
karthik@f542ba3b5a78573:~$ sed -i 's/[:blank:]]+\$/\' whileloop5
karthik@f542ba3b5a78573:~$ nano whileloop5
karthik@f542ba3b5a78573:~$ nano whileloop6
karthik@f542ba3b5a78573:~$ chmod a+x whileloop6
karthik@f542ba3b5a78573:~$ ./whileloop6
1
2
3
4
5
6
7
8
9
10
karthik@f542ba3b5a78573:~$
```

Activate Windows
Go to Settings to activate Windows.

```
karthik@f542ba3b5a78573:~$ Type here to search 09:38
ENG US 28-01-2025
```

```
GNU nano 7.2          whileloop6
#!/bin/bash
#While loop example in C style

i=1
while((i <= 10))
do
echo $i
let i++
done
```

[Read 9 lines]

```
^G Help      ^O Write Out    ^W Where Is    [ Read 9 lines ]    ^C Location  Activate Windows
^X Exit      ^R Read File     ^\ Replace     ^K Cut        ^T Execute    M-U Undo
                                         ^U Paste      ^J Justify    ^/ Go To Line M-E Redo
                                         ^L Location  O to Sett...  Undo Windows.
                                         ^/ Go To Line M-E Redo
Type here to search 09:38
ENG US 28-01-2025
```

UNTILL LOOP WITH SINGLE CONDITION

```
karthik@f542ba3b5a78573:~$ nano whileloop4
karthik@f542ba3b5a78573:~$ chmod a+x whileloop4
karthik@f542ba3b5a78573:~$ ./whileloop4
Countdown for Website Launching...
10
9
8
7
6
5
4
3
Mission Aborted, Some Technical Error Found.

karthik@f542ba3b5a78573:~$ nano untilloop1
karthik@f542ba3b5a78573:~$ chmod a+x untilloop1
karthik@f542ba3b5a78573:~$ ./untilloop1
1
2
3
4
5
6
7
8
9
10

Activate Windows
Go to Settings to activate Windows.

karthik@f542ba3b5a78573:~$ ./.untilloop1
GNU nano 7.2
untilloop1
#!/bin/bash
#Bash Until Loop example with a single condition

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

[ Read 9 lines ]
```

Activate Windows
Go to Settings to activate Windows.

Help Write Out Where Is Cut Execute Location Go To Line Undo Redo

Type here to search

Activate Windows
Go to Settings to activate Windows.

Help Write Out Where Is Cut Execute Location Go To Line Undo Redo

Type here to search

UNTIL LOOP WITH MULTIPLE CONDITION

BASH STRING EQUAL OPERATOR

```
karthik@f542ba3b5a78573:~$ nano bashstring1
GNU nano 7.2                                bashstring1
#!/bin/bash
#Script to check whether two strings are equal.

str1="WelcometoJavatpoint."
str2="javatpoint"

if [ $str1 = $str2 ];
then
echo "Both the strings are equal."
else
echo "Strings are not equal."
fi

[ Read 12 lines ]
^G Help      ^C Write Out   ^W Where Is   ^K Cut        ^T Execute   ^C Location  Activate Windows
^X Exit      ^R Read File   ^\ Replace    ^U Paste      ^J Justify   ^/ Go To Line M-U Undo
                                         No to Settings to activate Windows.
                                         M-E Redo
Type here to search
Windows Start button  File  Home  Mail  Photos  Videos  Downloads  Google Chrome  Microsoft Edge  Task View  Refresh  Stop  Search bar
ENG 13:01 28-01-2025

karthik@f542ba3b5a78573:~$ chmod a+x untilloop1
karthik@f542ba3b5a78573:~$ ./untilloop1
1
2
3
4
5
6
7
8
9
10
karthik@f542ba3b5a78573:~$ nano untilloop1
karthik@f542ba3b5a78573:~$ nano untilloop2
karthik@f542ba3b5a78573:~$ nano untilloop2
karthik@f542ba3b5a78573:~$ chmod a+x untilloop2
karthik@f542ba3b5a78573:~$ ./untilloop2
a = 1 & b = 0.
a = 2 & b = 1.
a = 3 & b = 2.
a = 4 & b = 3.
a = 5 & b = 4.
karthik@f542ba3b5a78573:~$ nano bashstring1
karthik@f542ba3b5a78573:~$ chmod a+x bashstring1
karthik@f542ba3b5a78573:~$ ./bashstring1
Strings are not equal.
karthik@f542ba3b5a78573:~$ nano bashstring1
Windows Start button  File  Home  Mail  Photos  Videos  Downloads  Google Chrome  Microsoft Edge  Task View  Refresh  Stop  Search bar
ENG 13:01 28-01-2025
```

BASH STRING NOT EQUAL OPERATOR

The image shows a Windows desktop environment with two terminal windows open. Both terminals are titled 'bashstring2' and have a dark theme.

Terminal Window 1 (Top):

```
karthik@f542ba3b5a78573:~$ nano 7.2                                bashstring2 *
#!/bin/bash
#Script to check whether two strings are equal.

str1="WelcometoJavatpoint."
str2="javatpoint"

if [[ $str1 != $str2 ]];
then
echo "Strings are not equal."
else
echo "Strings are equal."
fi
```

Terminal Window 2 (Bottom):

```
karthik@f542ba3b5a78573:~$ chmod a+x bashstring1
karthik@f542ba3b5a78573:~$ ./bashstring1
Strings are not equal.
karthik@f542ba3b5a78573:~$ nano bashstring1
karthik@f542ba3b5a78573:~$ sed -i 's/[:blank:]]\+$//'' bashstring1
karthik@f542ba3b5a78573:~$ nano bashstring1
karthik@f542ba3b5a78573:~$ nano bashstring2
karthik@f542ba3b5a78573:~$ chmod a+x bashstring2
karthik@f542ba3b5a78573:~$ ./bashstring2
Strings are not equal.
karthik@f542ba3b5a78573:~$
```

The taskbar at the bottom of the screen shows several icons for common applications like File Explorer, Mail, and a browser. The system tray indicates the date as 28-01-2025 and the time as 13:03. A tooltip 'Activate Windows Go to Settings to activate Windows.' is visible in the top right corner of the bottom terminal window.

BASH STRING LESS THAN OPERATOR

```
karthik@f542ba3b5a78573: ~
GNU nano 7.2                                bashstring3 *

#!/bin/sh

str1="WelcometoJavatpoint"
str2="Javatpoint"
if [ $str1 < $str2 ];
then
    echo "$str1 is less then $str2"
else
    echo "$str1 is not less then $str2"
fi
```

BASH STRING GREATER THAN OPERATOR

The screenshot shows a Windows desktop environment with two terminal windows open. Both terminals are running on a Linux system (Ubuntu) via a virtual machine or container.

Terminal 1 (Top):

```
karthik@f542ba3b5a78573:~$ ./bashstring2
Strings are not equal.
karthik@f542ba3b5a78573:~$
```

Terminal 2 (Bottom):

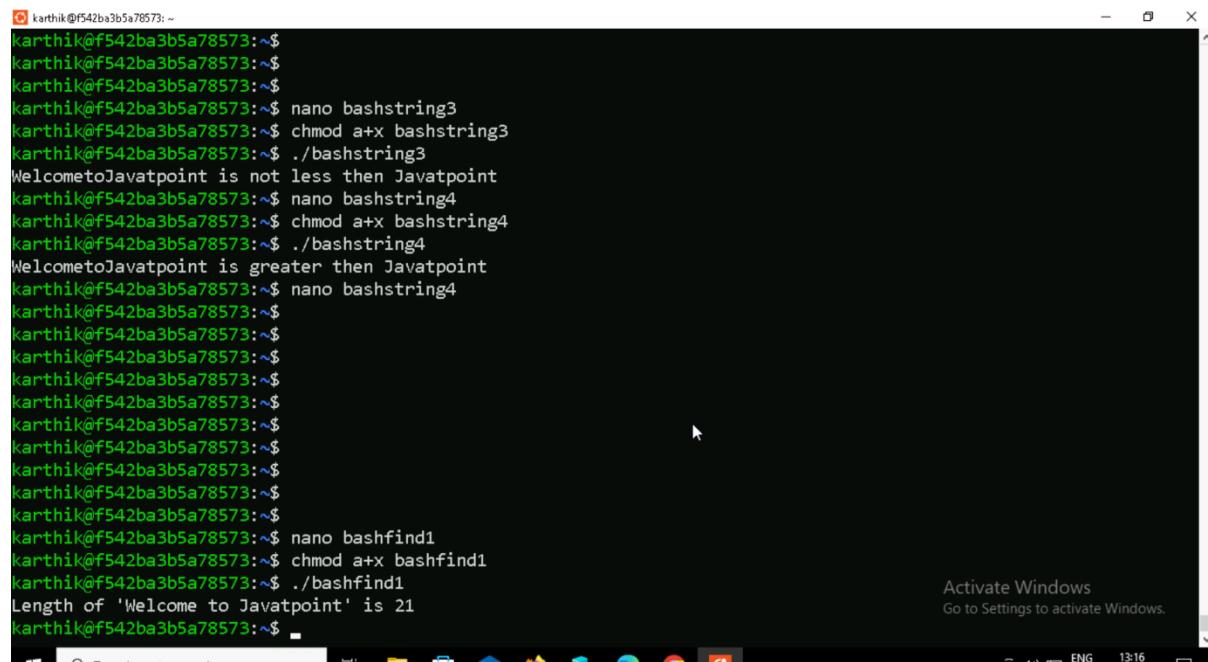
```
Select karthik@f542ba3b5a78573:~$ nano bashstring4
GNU nano 7.2                                bashstring4
#!/bin/sh
[Read 10 lines]
^G Help      ^O Write Out    ^W Where Is      [ Read 10 lines ]
^X Exit      ^R Read File    ^\ Replace      ^K Cut          ^T Execute
                                         ^U Paste      ^J Justify      ^C Location
                                         ^/ Go To Line  M-U Undo  Activate Windows
                                         M-L Redo  Go to Settings to activate Windows.
                                         M-U Undo
                                         M-L Redo
```

The bottom terminal window displays the script code for `bashstring4`:

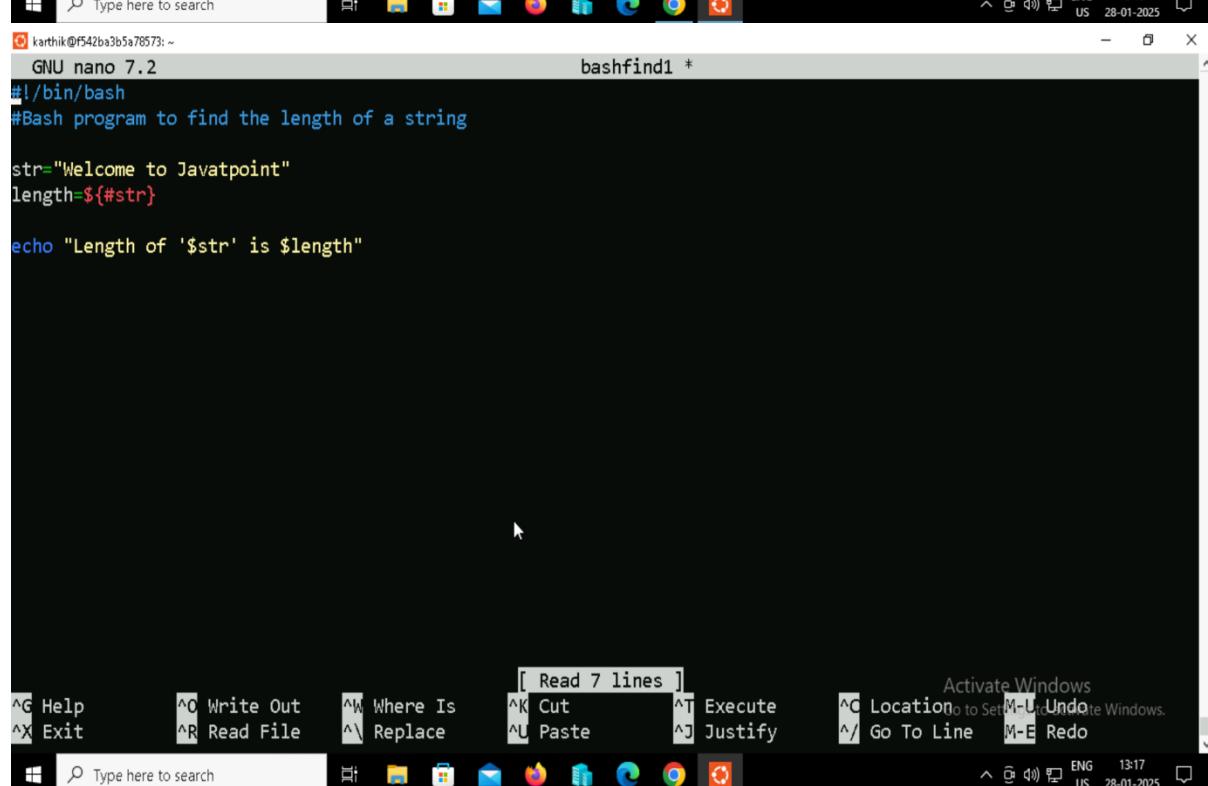
```
#!/bin/sh
str1="WelcometoJavatpoint"
str2="Javatpoint"
if [ $str1 > $str2 ];
then
  echo "$str1 is greater than $str2"
else
  echo "$str1 is less than $str2"
fi
```

BASH FIND

BASH FIND STRING LENGTH IN BASH



```
karthik@f542ba3b5a78573:~$ nano bashstring3
karthik@f542ba3b5a78573:~$ chmod a+x bashstring3
karthik@f542ba3b5a78573:~$ ./bashstring3
WelcometoJavatpoint is not less then Javatpoint
karthik@f542ba3b5a78573:~$ nano bashstring4
karthik@f542ba3b5a78573:~$ chmod a+x bashstring4
karthik@f542ba3b5a78573:~$ ./bashstring4
WelcometoJavatpoint is greater then Javatpoint
karthik@f542ba3b5a78573:~$ nano bashstring4
karthik@f542ba3b5a78573:~$ nano bashfind1
karthik@f542ba3b5a78573:~$ chmod a+x bashfind1
karthik@f542ba3b5a78573:~$ ./bashfind1
Length of 'Welcome to Javatpoint' is 21
karthik@f542ba3b5a78573:~$
```



```
GNU nano 7.2                                bashfind1 *
#!/bin/bash
#Bash program to find the length of a string

str="Welcome to Javatpoint"
length=${#str}

echo "Length of '$str' is $length"
```

[Read 7 lines] [Cut ^K Execute ^T] [Location ^C to Set Undo ^D] [Go To Line ^G ^/] [Paste ^U Justify ^J] [Redo ^M-E] [Undo ^M-U]

BASH FIND STRING LENGTH IN BASH EXAMPLE 2

```
karthik@f542ba3b5a78573:~$ nano bashfind1
karthik@f542ba3b5a78573:~$ nano bashfind2
karthik@f542ba3b5a78573:~$ chmod a+x bashfind2
karthik@f542ba3b5a78573:~$ ./bashfind2
Length of 'Welcome to Javatpoint' is 21
karthik@f542ba3b5a78573:~$
```

The screenshot shows a Windows terminal window titled "bashfind2 *". Inside the window, a Bash script named "bashfind2" is being run. The script defines a variable "str" with the value "Welcome to Javatpoint", calculates its length using the "expr length" command, and then prints the result. The terminal window has a dark background and white text. At the bottom, there is a standard Windows taskbar with icons for various applications like File Explorer, Mail, and a search bar.

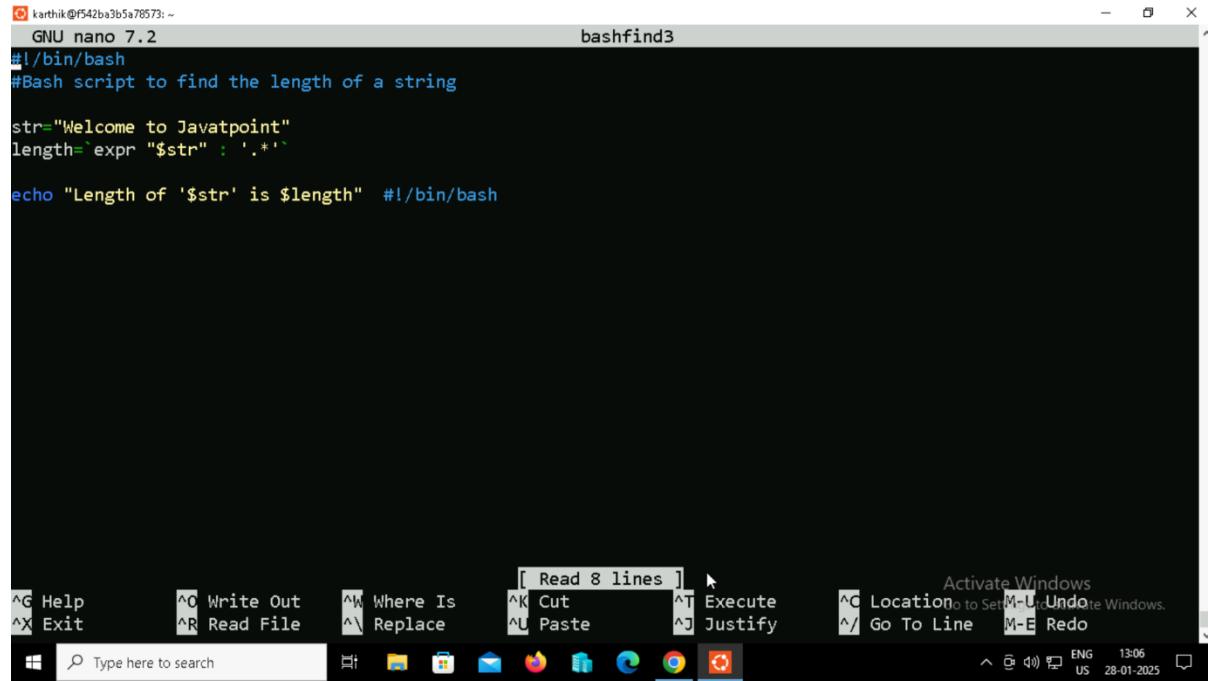
```
karthik@f542ba3b5a78573:~$ GNU nano 7.2                                bashfind2 *
#!/bin/bash
#Bash script to find the length of a string

str="Welcome to Javatpoint"
length=`expr length "$str"`

echo "Length of '$str' is $length" -
```

Activate Windows
^G Help ^C Write Out ^W Where Is ^K Cut ^T Execute ^C Location to Set M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo
Type here to search ENG 13:20 28-01-2025

BASH FIND STRING EXAMPLE 3



The screenshot shows a Windows desktop environment. At the top, there is a taskbar with various icons for common applications like File Explorer, Edge, and File History. To the right of the taskbar, there is a system tray showing the date (28-01-2025), time (13:06), and language (US). Above the taskbar, a terminal window titled "bashfind3" is open. The terminal shows the following content:

```
karthik@f542ba3b5a78573:~$ nano bashfind3
GNU nano 7.2
#!/bin/bash
#Bash script to find the length of a string

str="Welcome to Javatpoint"
length=`expr "$str" : '.*'` 

echo "Length of '$str' is $length" #!/bin/bash
```

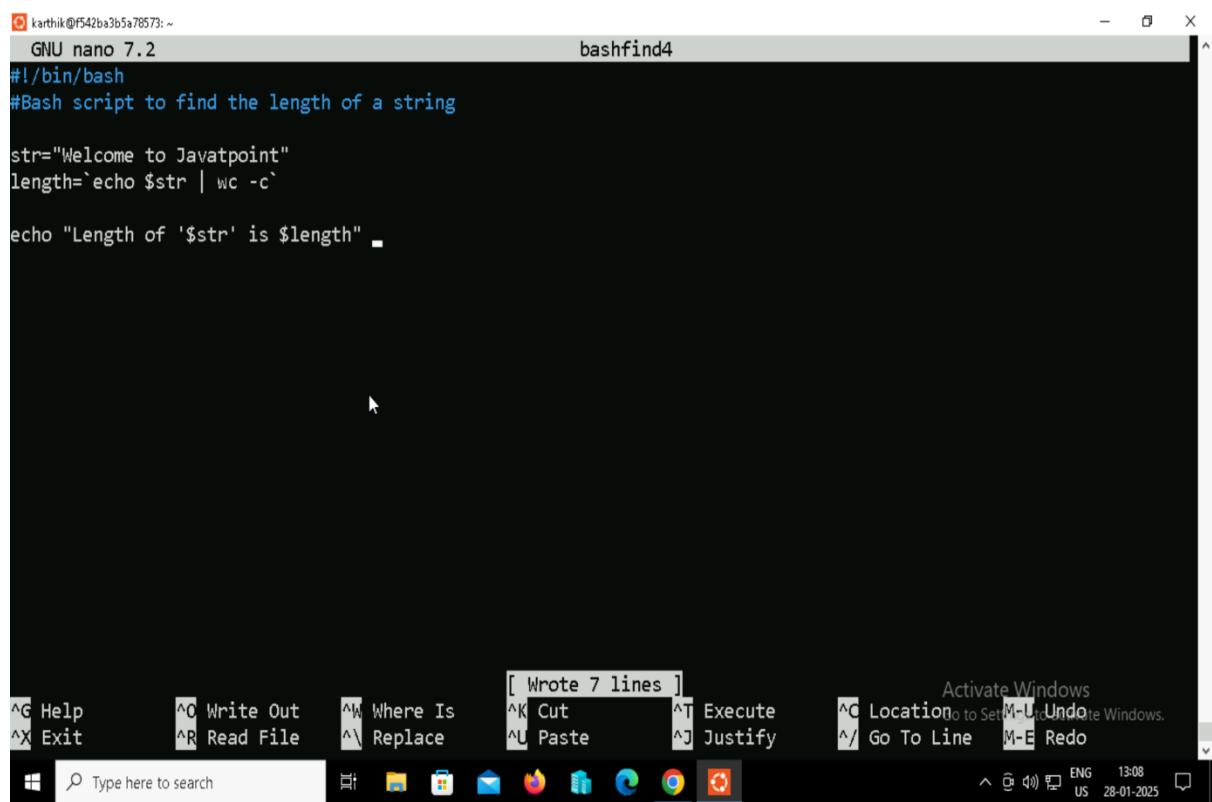
Below the terminal window, the command line prompt shows the script being saved and then executed:

```
karthik@f542ba3b5a78573:~$ nano bashfind3
karthik@f542ba3b5a78573:~$ chmod a+x bashfind3
karthik@f542ba3b5a78573:~$ ./bashfind3
Length of 'Welcome to Javatpoint' is 21
karthik@f542ba3b5a78573:~$
```

At the bottom right of the screen, there is a message from Microsoft: "Activate Windows Go to Settings to activate Windows."

BASH FIND EXAMPLE 4

```
arthik@f542ba3b5a78573:~$ nano bashfind4
arthik@f542ba3b5a78573:~$ chmod a+x bashfind4
arthik@f542ba3b5a78573:~$ ./bashfind4
length of 'Welcome to Javatpoint' is 22
arthik@f542ba3b5a78573:~$
```



BASH FIND EXAMPLE 5

The screenshot shows a Windows desktop environment with two terminal windows open. The top terminal window is a standard black terminal window with a white border. It displays a message from MicroK8s about edge and IoT security, followed by a URL: <https://ubuntu.com/engage/secure-kubernetes-at-the-edge>. Below this, it shows a series of commands being run in a shell:

```
* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s just raised the bar for easy, resilient and secure K8s cluster deployment.  
https://ubuntu.com/engage/secure-kubernetes-at-the-edge  
This message is shown once a day. To disable it please create the /home/karthik/.hushlogin file.  
karthik@f542ba3b5a78573:~$ nano bashfind4  
karthik@f542ba3b5a78573:~$ nano bashfind5  
karthik@f542ba3b5a78573:~$ chmod +x bashfind5  
karthik@f542ba3b5a78573:~$ ./bashfind5  
Length of 'Welcome to Javatpoint' is 21  
karthik@f542ba3b5a78573:~$
```

The bottom terminal window is a nano editor window titled "bashfind5 *". It contains a Bash script to calculate the length of a string:

```
GNU nano 7.2                                bashfind5 *  
#!/bin/bash  
#Bash script to find the length of a string  
  
str="Welcome to Javatpoint"  
length=`echo $str |awk '{print length}'`  
  
echo "Length of '$str' is $length"
```

The desktop taskbar at the bottom shows several pinned icons, including File Explorer, Edge, Mail, and Google Chrome. The system tray indicates the date as 28-01-2025 and the time as 15:47.

BASH SPLIT STRINGS

BASH SPLIT STRING BY SPACE

The image shows a Windows desktop environment with two terminal windows open. Both terminals are running on a black background with white text.

The top terminal window shows the execution of a script named `bashsplit1`. The user enters the string "we welcome you on javatpoint" and the script prints each word on a new line:

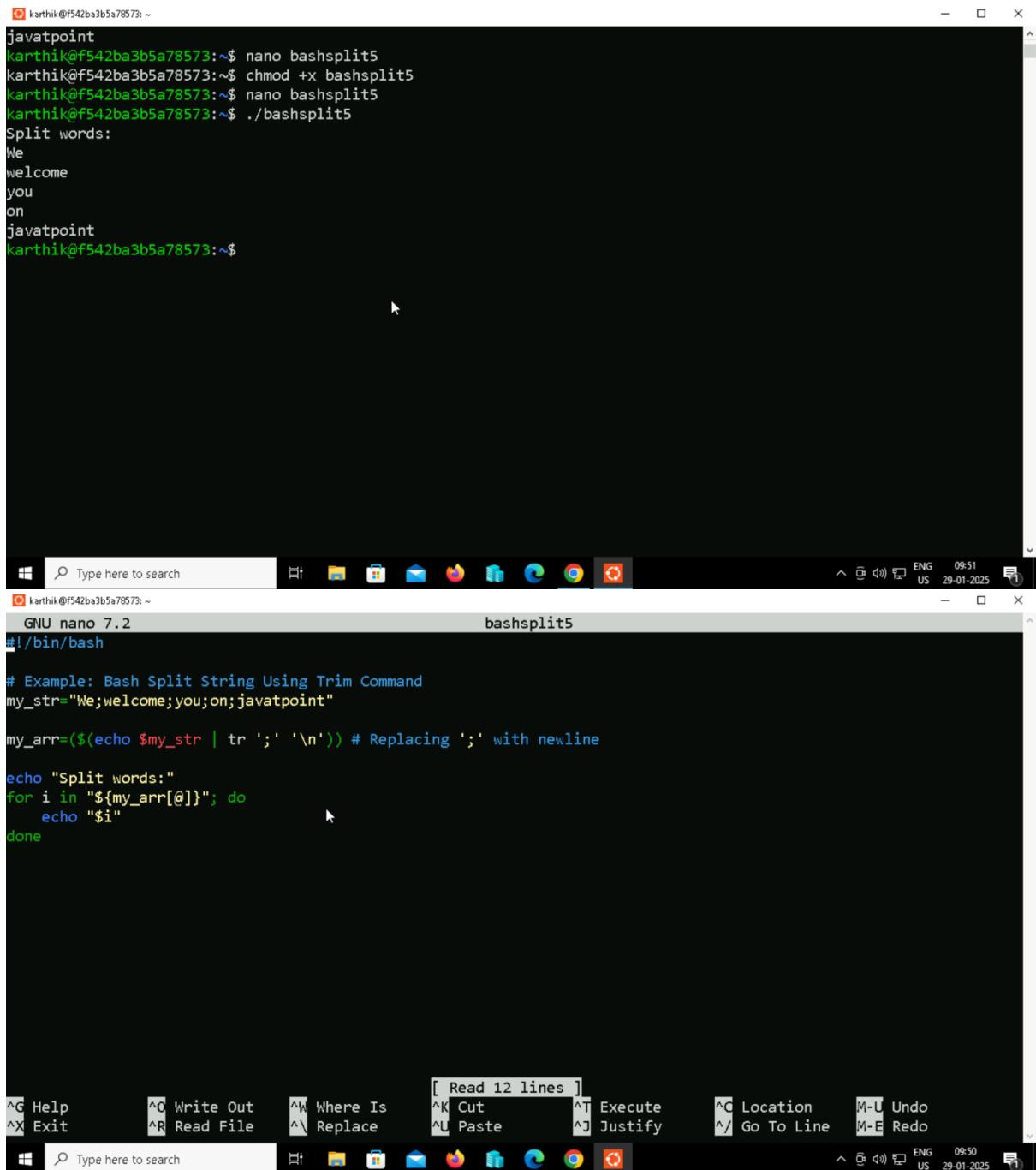
```
karthik@f542ba3b5a78573:~$ nano bashsplit1
karthik@f542ba3b5a78573:~$ ./bashsplit1
Enter any string separated by space: we welcome you on javatpoint
Split words:
we
welcome
you
on
javatpoint
karthik@f542ba3b5a78573:~$
```

The bottom terminal window shows the source code of the `bashsplit1` script. It uses the `read` command with the `-p` option to prompt for input, and the `IFS` variable is set to an empty string to split the input by whitespace. A loop then iterates over the tokens stored in an array `ADDR`.

```
#!/bin/bash
#Example for bash split string by space
read -p "Enter any string separated by space: " str #reading string value
IFS=' ' #setting space as delimiter
read -ra ADDR <<< "$str" #reading str as an array as tokens separated by IFS
for i in "${ADDR[@]}"; #accessing each element of array
do
echo "$i"
done
```

The bottom terminal also displays a menu bar with various keyboard shortcuts for navigating the nano editor.

BASH SPLIT STRING USING TRIM COMMAND



```
javatpoint
karthik@f542ba3b5a78573:~$ nano bashsplit5
karthik@f542ba3b5a78573:~$ chmod +x bashsplit5
karthik@f542ba3b5a78573:~$ nano bashsplit5
karthik@f542ba3b5a78573:~$ ./bashsplit5
Split words:
We
welcome
you
on
javatpoint
karthik@f542ba3b5a78573:~$
```



```
GNU nano 7.2                                bashsplit5
#!/bin/bash

# Example: Bash Split String Using Trim Command
my_str="We;welcome;you;on;javatpoint"

my_arr=($(echo $my_str | tr ';' '\n')) # Replacing ';' with newline

echo "Split words:"
for i in "${my_arr[@]}"; do
    echo "$i"
done
```



```
[ Read 12 lines ]
^G Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute   ^C Location   M-U Undo
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line M-E Redo
^I           ^P             ^L             ^S           ^D           ^F             ^V           ^H
```

BASH SPLIT STRING BY ANOTHER STRING

```
#!/bin/bash

# Example: Bash Split String by Another String
str="WeLearnWelcomeLearnYouLearnOnLearnJavatpoint"
delimiter="Learn"
s=$str$delimiter
array=()

while [[ $s ]]; do
    array+=(" ${s%%$delimiter}*") # Extract substring before delimiter
    s=${s#$delimiter} # Remove processed part
done

echo "Split words:"
for word in "${array[@]}"; do
    echo "$word"
done

declare -p array
```

[Read 20 lines]

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo

Type here to search ENG 09:47 29-01-2025

```
karthik@f542ba3b5a78573:~$ ./bashsplit4
Split words:
We
Welcome
You
On
Javatpoint
karthik@f542ba3b5a78573:~$ nano bashsplit4
karthik@f542ba3b5a78573:~$ chmod +x bashsplit4
karthik@f542ba3b5a78573:~$ ./bashsplit4
Split words:
We
Welcome
You
On
Javatpoint
declare -a array=([0]="We" [1]="Welcome" [2]="You" [3]="On" [4]="Javatpoint")
karthik@f542ba3b5a78573:~$
```

Type here to search ENG 09:47 29-01-2025

BASH SPLIT STRING BY SYMBOL -2

The screenshot shows a Windows desktop environment with three terminal windows open in separate windows.

Top Terminal: A standard black terminal window. It starts with the command to enter name, state, and age separated by commas. Then it runs the nano editor to edit a file named bashsplit3, changes its mode to executable, and runs it. The output shows the input and the resulting split words.

```
karthik@f542ba3b5a78573:~$ Enter Name, State and Age separated by a comma: ALankar,uttar pradesh,22
Name : ALankar
State : uttar pradesh
Age : 22
karthik@f542ba3b5a78573:~$ nano bashsplit3
karthik@f542ba3b5a78573:~$ chmod +x bashsplit3
karthik@f542ba3b5a78573:~$ ./bashsplit3
Enter any string separated by colon(:) welcome:you:on:javapoint
Split words:
welcome
you
on
javapoint
```

Middle Terminal: A terminal window titled "bashsplit3" running in the nano editor. It contains a script to read a string from the user, split it into an array using colons as delimiters, and then print each element of the array.

```
GNU nano 7.2                                bashsplit3
#!/bin/bash

# Example: Bash Split String without $IFS
read -p "Enter any string separated by colon(:) " str

readarray -d : -t strarr <<<"$str" # Splitting string using readarray

echo "Split words:"
for (( i=0; i<#${strarr[@]}; i++ )); do
    echo "${strarr[i]}"
done
```

Bottom Terminal: Another standard black terminal window. It shows the same process as the top terminal, but with a different timestamp in the status bar (09:42). The status bar also shows ENG US 29-01-2025.

BASH SPLIT STRING BY COMMA

The image shows a Windows desktop environment with three terminal windows open, illustrating how to split a string by commas in Bash.

Terminal 1: A standard terminal window titled "Split words:" showing the output of a script that prints each word on a new line. The script reads from standard input and splits it by whitespace.

```
karthik@f542ba3b5a78573:~$ nano bashsplit2
Split words:
we
welcome
you
on
javatpoint
karthik@f542ba3b5a78573:~$ chmod +x bashsplit2
karthik@f542ba3b5a78573:~$ ./bashsplit2
Enter Name, State and Age separated by a comma: ALankar,uttar pradesh,22
Name : ALankar
State : uttar pradesh
Age : 22
karthik@f542ba3b5a78573:~$
```

Terminal 2: A terminal window titled "bashsplit2" showing the source code of the script. It uses the `read` command with a prompt to get input from the user, then splits the input by commas into an array using `IFS=','` and prints each element.

```
GNU nano 7.2                                bashsplit2
#!/bin/bash

# Example: Bash Split String by Comma
read -p "Enter Name, State and Age separated by a comma: " entry

IFS=',' # Setting comma as delimiter
read -ra strarr <<< "$entry" # Splitting string into array

echo "Name : ${strarr[0]}"
echo "State : ${strarr[1]}"
echo "Age : ${strarr[2]}"
```

Terminal 3: A terminal window showing the result of running the script. The user entered "ALankar,uttar pradesh,22" and the script printed "Name : ALankar", "State : uttar pradesh", and "Age : 22".

```
[ Wrote 12 lines ]
```

Keyboard shortcuts displayed at the bottom of the terminal windows:

- Help (^G)
- Exit (^X)
- Write Out (^C)
- Read File (^R)
- Where Is (^W)
- Replace (^R)
- Cut (^K)
- Paste (^U)
- Execute (^T)
- Justify (^J)
- Location (^C)
- Go To Line (^/)
- Undo (M-U)
- Redo (M-E)

BASH SUB STRING

Bash substring to extract last 11 characters

The image shows a Windows desktop environment with two terminal windows open. The top window is a terminal session titled 'bashsubstring4' running in a 'GNU nano 7.2' editor. It contains the following code:

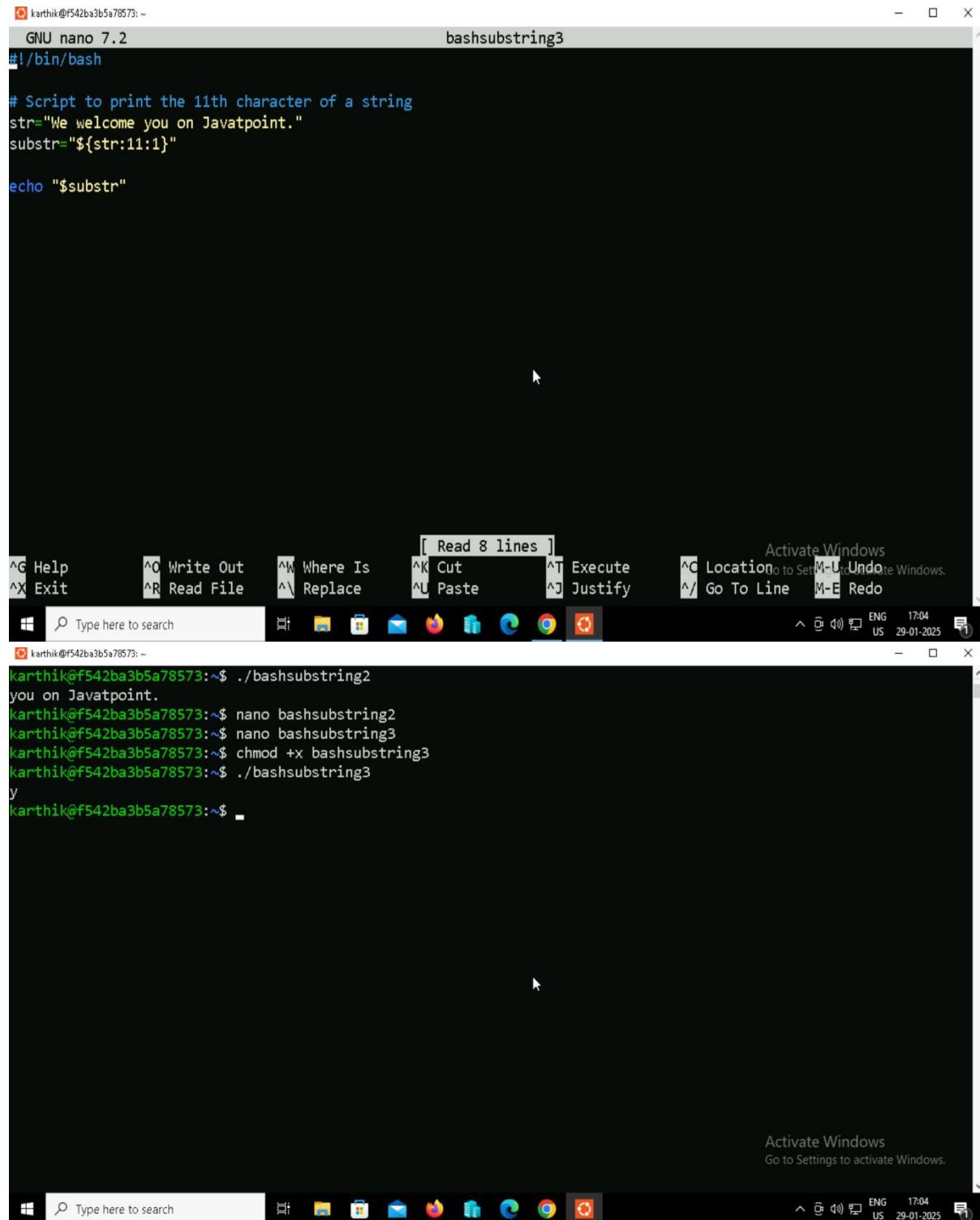
```
# Script to extract the last 11 characters
str="We welcome you on Javatpoint."
substr="${str: -11}"
echo "$substr"
```

The bottom window is a standard command-line terminal window. The user runs the script and then executes it:

```
karthik@f542ba3b5a78573:~$ nano bashsubstring4
karthik@f542ba3b5a78573:~$ chmod +x bashsubstring4
karthik@f542ba3b5a78573:~$ ./bashsubstring4
Javatpoint.
karthik@f542ba3b5a78573:~$
```

The output 'Javatpoint.' is displayed in the terminal.

Bash script to print the 11th characters of string



```
karthik@f542ba3b5a78573: ~
GNU nano 7.2                                bashsubstring3
#!/bin/bash

# Script to print the 11th character of a string
str="We welcome you on Javatpoint."
substr="${str:11:1}"

echo "$substr"

[ Read 8 lines ]
^G Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute   ^C Location  Activate Windows
^X Exit      ^R Read File    ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line M-U Undo Go to Settings to activate Windows.
                                         M-E Redo
Type here to search
Windows Start button  File Explorer  Mail  Edge  File  Task View  Start  Taskbar  System  Date/Time  Language  ENG US  29-01-2025
```

```
karthik@f542ba3b5a78573:~$ ./bashsubstring2
you on Javatpoint.
karthik@f542ba3b5a78573:~$ nano bashsubstring2
karthik@f542ba3b5a78573:~$ nano bashsubstring3
karthik@f542ba3b5a78573:~$ chmod +x bashsubstring3
karthik@f542ba3b5a78573:~$ ./bashsubstring3
y
karthik@f542ba3b5a78573:~$
```

```
Activate Windows
Go to Settings to activate Windows.

[ Read 8 lines ]
^G Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute   ^C Location  Activate Windows
^X Exit      ^R Read File    ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line M-U Undo Go to Settings to activate Windows.
                                         M-E Redo
Type here to search
Windows Start button  File Explorer  Mail  Edge  File  Task View  Start  Taskbar  System  Date/Time  Language  ENG US  29-01-2025
```

Bash script to print from 11th characters onwards

```
karthik@f542ba3b5a78573: ~
GNU nano 7.2                                bashsubstring2
#!/bin/bash

# Script to print from 11th character onwards
str="We welcome you on Javatpoint."
substr="${str:11}"

echo "$substr"

[ Read 8 lines ] [ Read 8 lines ] Activate Windows
^G Help          ^O Write Out      ^W Where Is      ^K Cut           ^T Execute      ^C Location Go to Settings to activate Windows.
^X Exit          ^R Read File      ^\ Replace       ^U Paste         ^J Justify      M-U Undo
                                                ^/ Go To Line    M-E Redo

Type here to search  17:03
ENG 29-01-2025  US

karthik@f542ba3b5a78573: ~$ nano bashsubstring1
karthik@f542ba3b5a78573: ~$ nano bashsubstring2
karthik@f542ba3b5a78573: ~$ chmod +x bashsubstring2
karthik@f542ba3b5a78573: ~$ ./bashsubstring2
you on Javatpoint.
karthik@f542ba3b5a78573: ~$ -
```

Activate Windows
Go to Settings to activate Windows.

```
karthik@f542ba3b5a78573: ~$ Type here to search  17:03
ENG 29-01-2025  US
```

Bash substring to extract first 11 characters

The image shows a Windows desktop environment with two terminal windows open. The top window is a terminal session titled "bashsubstring1" running in a nano editor. It contains a script to extract the first 10 characters from a string. The bottom window is a standard command-line terminal window showing the execution of the script and its output.

```
karthik@f542ba3b5a78573: ~
GNU nano 7.2
#!/bin/bash
#Script to extract first 10 characters of a string
echo "String: We welcome you on Javatpoint."
str="We welcome you on Javatpoint."
echo "Total characters in a String: ${#str} "
substr="${str:0:10}"
echo "Substring: $substr"
echo "Total characters in Substring: ${#substr}"
```

```
[ Read 12 lines ]
^G Help      ^C Write Out   ^W Where Is      [ Activate Windows
^X Exit      ^R Read File    ^\ Replace      Go to Settings M-U Undo Windows.
                                                ^C Location M-G Set M-E Redo
                                                ^/ Go To Line M-E Redo
Windows Type here to search  File Explorer Mail Edge Google Chrome Microsoft Edge
Activate Windows
Go to Settings to activate Windows.
ENG 16:42
US 29-01-2025
```

```
on
javatpoint
karthik@f542ba3b5a78573:~$ nano bashsubstring1
karthik@f542ba3b5a78573:~$ chmod +x bashsubstring1
karthik@f542ba3b5a78573:~$ ./bashsubstring1
String: We welcome you on Javatpoint.
Total characters in a String: 29
Substring: We welcome
Total characters in Substring: 10
karthik@f542ba3b5a78573:~$
```

```
Activate Windows
Go to Settings to activate Windows.
Windows Type here to search  File Explorer Mail Edge Google Chrome Microsoft Edge
Activate Windows
Go to Settings to activate Windows.
ENG 16:42
US 29-01-2025
```

BASH CONCATENATE STRING

Using the printf Function for String Concatenation

```
karthik@f542ba3b5a78573: ~
GNU nano 7.2                                bashconcat4
#!/bin/bash

# Defining a string
str="Welcome"

# Using printf to concatenate
printf -v new_str "%s to Javatpoint." "$str"

# Printing the concatenated string
echo "$new_str"

[ Read 11 lines ]
^G Help      ^C Write Out   ^W Where Is   ^K Cut        ^T Execute   ^C Location Go to Settings M-U Undo
^X Exit      ^R Read File    ^\ Replace    ^U Paste     ^J Justify   ^V Go To Line M-E Redo
Activate Windows
Type here to search
Windows Start button  File  Home  Mail  Photos  Firefox  Edge  Chrome  Task View  Refresh  17:17
ENG US 29-01-2025
```

```
Java Python C C++
karthik@f542ba3b5a78573:~$ nano bashconcat3
karthik@f542ba3b5a78573:~$ nano bashconcat4
karthik@f542ba3b5a78573:~$ chmod +x bashconcat4
karthik@f542ba3b5a78573:~$ ./bashconcat4
Welcome to Javatpoint.
karthik@f542ba3b5a78573:~$
```

```
Activate Windows
Go to Settings to activate Windows.

Type here to search
Windows Start button  File  Home  Mail  Photos  Firefox  Edge  Chrome  Task View  Refresh  17:17
ENG US 29-01-2025
```

Bash Using the Append Operator (+=) with a Loop

The image shows a Windows desktop environment with two terminal windows open. Both terminals are titled "bashconcat3".

Terminal Window 1 (Top):

```
karthik@f542ba3b5a78573: ~
GNU nano 7.2
#!/bin/bash

echo "Printing the names of programming languages"

# Initializing an empty variable
lang=""

# Using a for loop to concatenate strings
for value in "Java" "Python" "C" "C++"; do
    lang+="$value " # Appending values with space
done

# Printing the concatenated values
echo "$lang"
```

Terminal Window 2 (Bottom):

```
We welcome you on Javatpoint.
karthik@f542ba3b5a78573:~$ nano bashconcat2
karthik@f542ba3b5a78573:~$ nano bashconcat3
karthik@f542ba3b5a78573:~$ chmod +x bashconcat3
karthik@f542ba3b5a78573:~$ ./bashconcat3
Printing the names of programming languages
Java Python C C++
karthik@f542ba3b5a78573:~$
```

The desktop interface includes a taskbar at the bottom with icons for File Explorer, Edge, Mail, and others. The system tray shows the date (29-01-2025), time (17:16), and language (ENG US).

Bash Concatenating Strings Using a Variable

The screenshot shows a Windows desktop environment with two terminal windows open. The top window is a terminal session titled "bashconcat2" running "GNU nano 7.2". It contains a script named "bashconcat2" with the following content:

```
#!/bin/bash

# Script to concatenate strings
str="We welcome you"

# Adding the variable within the string
echo "$str on Javatpoint."
```

The bottom window is a standard terminal session with the command line prompt "karthik@f542ba3b5a78573:~\$". It shows the execution of three scripts: "bashsubstring4", "bashconcat1", and "bashconcat2", all resulting in the output "We welcome you on Javatpoint.". The desktop taskbar at the bottom includes icons for File Explorer, Mail, Edge, and Google Chrome.

BASH SCRIPT TO CONCATENATE STRING

The image shows a Windows desktop environment with two terminal windows open. The top window is a standard terminal window titled 'bashsubstring4'. It contains the following command history:

```
karthik@f542ba3b5a78573:~$ nano bashsubstring4
karthik@f542ba3b5a78573:~$ chmod +x bashsubstring4
karthik@f542ba3b5a78573:~$ ./bashsubstring4
Javatpoint.
karthik@f542ba3b5a78573:~$ nano bashsubstring4
karthik@f542ba3b5a78573:~$ nano bashconcat1
karthik@f542ba3b5a78573:~$ chmod +x bashconcat1
karthik@f542ba3b5a78573:~$ ./bashconcat1
We welcome you on Javatpoint.
karthik@f542ba3b5a78573:~$
```

The bottom window is a terminal window titled 'bashconcat1'. It contains the following script code:

```
GNU nano 7.2
#!/bin/bash
#Script to Concatenate Strings

#Declaring the first String
str1="We welcome you"

#Declaring the Second String
str2=" on Javatpoint."

#Combining first and second string
str3="$str1$str2"

#printing a new string by combining both
echo $str3
```

The desktop interface includes a taskbar at the bottom with icons for File Explorer, Edge, Mail, and others. The system tray shows the date and time as 29-01-2025, 17:14. A watermark for 'Activate Windows' is visible in the center of the screen.

BASH FUNCTIONS

Bash functions with return status

The image shows a Windows desktop environment with two terminal windows open. The top window is titled 'bashfunction3' and contains the following code:

```
karthik@f542ba3b5a78573: ~
GNU nano 7.2
#!/bin/bash

# Function with return status
print_it() {
    echo "Hello $1"
    return 5 # Setting return status
}

# Calling the function
print_it User
print_it Reader

# Printing return value of the last executed function
echo "The previous function returned a value of $?"
```

The bottom window shows the output of running the script:

```
v1 is C.
v2 is D.
After Executing the Function
v1 is A.
v2 is D.
karthik@f542ba3b5a78573:~$ nano bashfunction2
karthik@f542ba3b5a78573:~$ nano bashfunction3
karthik@f542ba3b5a78573:~$ chmod +x bashfunction3
karthik@f542ba3b5a78573:~$ ./bashfunction3
Hello User
Hello Reader
The previous function returned a value of 5
karthik@f542ba3b5a78573:~$
```

The taskbar at the bottom of the screen shows various application icons, and the system tray indicates the date and time as 29-01-2025.

Understanding Variable Scope in Bash

The image shows a Windows desktop environment with two terminal windows open. Both terminals have a dark theme and are running on a virtual machine with the IP address f542ba3b5a78573.

Terminal 1 (Top): bashfunction2

```
karthik@f542ba3b5a78573: ~
GNU nano 7.2                                bashfunction2
#!/bin/bash

# Global variables
v1='A'
v2='B'

my_var() {
    local v1='C'  # Local variable
    v2='D'        # Modifying global variable
    echo "Inside Function"
    echo "v1 is $v1."
    echo "v2 is $v2."
}

echo "Before Executing the Function"
echo "v1 is $v1."
echo "v2 is $v2."

my_var

echo "After Executing the Function"
echo "v1 is $v1." # Global variable remains unchanged
echo "v2 is $v2." # Global variable modified inside function
```

Terminal 2 (Bottom): Javatpoint

```
karthik@f542ba3b5a78573:~$ nano bashfunction1
karthik@f542ba3b5a78573:~$ nano bashfunction2
karthik@f542ba3b5a78573:~$ chmod +x bashfunction2
karthik@f542ba3b5a78573:~$ ./bashfunction2
Before Executing the Function
v1 is A.
v2 is B.
Inside Function
v1 is C.
v2 is D.
After Executing the Function
v1 is A.
v2 is D.
```

The taskbar at the bottom of the screen shows several icons, including File Explorer, Mail, and a browser. The system tray indicates the date as 29-01-2025 and the time as 17:26. A tooltip in the top right corner of the bottom terminal window says "Activate Windows Go to Settings to activate Windows."

Passing and Accessing Function Arguments In Bash

```
karthik@f542ba3b5a78573:~$ nano bashfunction1
GNU nano 7.2
#!/bin/bash

# Function to print passed arguments
function_arguments() {
    echo "$1"
    echo "$2"
    echo "$3"
    echo "$4"
    echo "$5"
}

# Calling function_arguments with parameters
function_arguments "We" "welcome" "you" "on" "Javatpoint."
[ Read 14 lines ]
^G Help      ^C Write Out   ^W Where Is   ^K Cut        ^T Execute   ^C Location to Sett... M-U Undo
^X Exit       ^R Read File    ^\ Replace     ^U Paste      ^J Justify   ^/ Go To Line  M-E Redo
^O Type here to search
Activate Windows
Go to Settings to activate Windows.
ENG 17:25
US 29-01-2025

karthik@f542ba3b5a78573:~$ ./bashconcat4
Welcome to Javatpoint.
karthik@f542ba3b5a78573:~$ nano bashconcat4
karthik@f542ba3b5a78573:~$ nano bashfunction1
karthik@f542ba3b5a78573:~$ chmod +x bashfunction1
karthik@f542ba3b5a78573:~$ ./bashfunction1
We
welcome
you
on
Javatpoint.
karthik@f542ba3b5a78573:~$
```

BASH ARRAY

BASH SCRIPT TO PRINT ALL THE ARRAY ELEMENT

The image shows a Windows desktop environment with a terminal window open in the background. The terminal window title is "basharray1" and contains the following Bash script:

```
GNU nano 7.2
#!/bin/bash
# Script to print all the elements of the array

# Declaring the array
declare -a example_array=("Welcome" "To" "Javatpoint")

# Printing all the elements
echo "${example_array[@]}"
```

The terminal window has a menu bar with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, and Redo. A status bar at the bottom shows the date and time as 30-01-2025.

In the foreground, there is a Java Update Checker window titled "Java Update Available". It displays the message: "A new version of Java is ready to be installed. Click here to continue." The Java Update Checker window has a close button (X) in the top right corner.

The desktop taskbar at the bottom includes icons for File Explorer, Mail, and several browser windows (Firefox, Chrome, Edge).

BASH SCRIPT TO DELETE ARRAY ELEMENT

```
karthik@f542ba3b5a78573:~$ nano 7.2
GNU nano 7.2                                basharray5
#!/bin/bash
# Script to delete the element from the array

# Declaring the array
declare -a example_array=("Java" "Python" "HTML" "CSS" "JavaScript")

# Removing the element
unset example_array[1]

# Printing all the elements after deletion
echo "${example_array[@]}"

[ Read 12 lines ]
^G Help      ^O Write Out    ^W Where Is    ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File     ^\ Replace     ^U Paste      ^J Justify    ^I Go To Line M-E Redo
                                         Type here to search          ENG 09:45
                                         US 30-01-2025
```

```
karthik@f542ba3b5a78573:~$ ./basharray4
We welcome you on Javatpoint
karthik@f542ba3b5a78573:~$ nano basharray4
karthik@f542ba3b5a78573:~$ nano basharray5
karthik@f542ba3b5a78573:~$ chmod +x basharray5
karthik@f542ba3b5a78573:~$ ./basharray5
Java HTML CSS JavaScript
karthik@f542ba3b5a78573:~$
```

```
karthik@f542ba3b5a78573:~$ nano 7.2
GNU nano 7.2                                basharray5
#!/bin/bash
# Script to delete the element from the array

# Declaring the array
declare -a example_array=("Java" "Python" "HTML" "CSS" "JavaScript")

# Removing the element
unset example_array[1]

# Printing all the elements after deletion
echo "${example_array[@]}"

[ Read 12 lines ]
^G Help      ^O Write Out    ^W Where Is    ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File     ^\ Replace     ^U Paste      ^J Justify    ^I Go To Line M-E Redo
                                         Type here to search          ENG 09:45
                                         US 30-01-2025
```

```
karthik@f542ba3b5a78573:~$ ./basharray4
We welcome you on Javatpoint
karthik@f542ba3b5a78573:~$ nano basharray4
karthik@f542ba3b5a78573:~$ nano basharray5
karthik@f542ba3b5a78573:~$ chmod +x basharray5
karthik@f542ba3b5a78573:~$ ./basharray5
Java HTML CSS JavaScript
karthik@f542ba3b5a78573:~$
```

```
karthik@f542ba3b5a78573:~$ nano 7.2
GNU nano 7.2                                basharray5
#!/bin/bash
# Script to delete the element from the array

# Declaring the array
declare -a example_array=("Java" "Python" "HTML" "CSS" "JavaScript")

# Removing the element
unset example_array[1]

# Printing all the elements after deletion
echo "${example_array[@]}"

[ Read 12 lines ]
^G Help      ^O Write Out    ^W Where Is    ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File     ^\ Replace     ^U Paste      ^J Justify    ^I Go To Line M-E Redo
                                         Type here to search          ENG 09:45
                                         US 30-01-2025
```

```
karthik@f542ba3b5a78573:~$ ./basharray4
We welcome you on Javatpoint
karthik@f542ba3b5a78573:~$ nano basharray4
karthik@f542ba3b5a78573:~$ nano basharray5
karthik@f542ba3b5a78573:~$ chmod +x basharray5
karthik@f542ba3b5a78573:~$ ./basharray5
Java HTML CSS JavaScript
karthik@f542ba3b5a78573:~$
```

BASH SCRIPT TO UPDATE ARRAY ELEMENT

The screenshot displays a Windows desktop environment with two terminal windows.

Top Terminal Window:

- Terminal title: basharray4
- Content:

```
GNU nano 7.2
#!/bin/bash
# Script to update array element

# Declaring the array
declare -a example_array=("We" "welcome" "you" "on" "SSSIT")

# Updating the Array Element
example_array[4]="Javatpoint"

# Printing all the elements of the Array
echo ${example_array[@]}
```
- Bottom status bar:
 - [Read 12 lines]
 - Keyboard shortcuts: ^G Help, ^O Write Out, ^W Where Is, ^X Exit, ^R Read File, ^\ Replace, ^K Cut, ^T Execute, ^U Paste, ^J Justify, ^C Location, ^/ Go To Line, M-U Undo, M-E Redo
 - System status: ENG 09:44 US 30-01-2025

Bottom Terminal Window:

- Terminal title: basharray4
- Content:

```
karthik@f542ba3b5a78573:~$ nano basharray4
karthik@f542ba3b5a78573:~$ chmod +x basharray4
karthik@f542ba3b5a78573:~$ ./basharray4
We welcome you on Javatpoint
karthik@f542ba3b5a78573:~$
```
- Bottom status bar:
 - System status: ENG 09:44 US 30-01-2025

BASH SCRIPT TO DECLARE THE ARRAY

The key value of element To is 1
The key value of element Javatpoint is 2

```
karthik@f542ba3b5a78573:~$ nano basharray2
karthik@f542ba3b5a78573:~$ nano basharray3
karthik@f542ba3b5a78573:~$ chmod +x basharray3
karthik@f542ba3b5a78573:~$ ./basharray3
Java Python PHP HTML JavaScript
karthik@f542ba3b5a78573:~$
```

BASH SCRIPT TO PRINT ALL THE KEYS AND VALUES USING LOOPS

```
karthik@f542ba3b5a78573: ~
GNU nano 7.2                                basharray2
#!/bin/bash
# Script to print all keys and values using loop through the array

declare -a example_array=("Welcome" "To" "Javatpoint")

# Array Loop
for i in "${!example_array[@]}"
do
    echo The key value of element "${example_array[$i]}" is "$i"
done
```

[Read 11 lines]

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^I Go To Line M-E Redo

Type here to search

karthik@f542ba3b5a78573: ~\$ nano basharray1
karthik@f542ba3b5a78573: ~\$ nano basharray2
karthik@f542ba3b5a78573: ~\$ chmod +x basharray2
karthik@f542ba3b5a78573: ~\$./basharray2
The key value of element Welcome is 0
The key value of element To is 1
The key value of element Javatpoint is 2
karthik@f542ba3b5a78573: ~\$

Type here to search

ENG 09:42 US 30-01-2025