

Bash case example1

```
bhargava@23372bc849b55ff:~$ nano bash_case1
bhargava@23372bc849b55ff:~$ chmod +x bash_case1
bhargava@23372bc849b55ff:~$ cat bash_case1
#!/bin/bash
echo "Do you know python programming?"
read -p "Yes/No?:" Answer
case $Answer in
Yes|yes|y|Y)
echo "That's Amazing."
echo ;;
No|no|N|n)
echo "It's easy.Let's start Learning from Python Modules"
;;
esac
bhargava@23372bc849b55ff:~$ ./bash_case1
Do you know python programming?
Yes/No?:n
It's easy.Let's start Learning from Python Modules
bhargava@23372bc849b55ff:~$
```



Bash case example 2

```
bhargava@23372bc849b55ff:~$ nano bash_case2
bhargava@23372bc849b55ff:~$ chmod +x bash_case2
bhargava@23372bc849b55ff:~$ cat bash_case2
#!/bin/bash

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

case $OS in
    Windows|windows)
        echo "That's common. You should try something new."
        echo ;;
    Android|android)
        echo "This is my favorite. It has lots of applications."
        echo ;;
    Chrome|chrome)
        echo "Cool!!! It's for pro users. Amazing Choice."
        echo ;;
    Linux|linux)
        echo "You might be serious about security!!"
        echo ;;
    *)
        echo "Sounds interesting. I will try that."
        echo ;;
esac
bhargava@23372bc849b55ff:~$ ./bash_case2
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name:chrome
Cool!!! It's for pro users. Amazing Choice.
bhargava@23372bc849b55ff:~$
```



Bash for loop example 1

```
bhargava@23372bc849b55ff:~$ nano bash_for_loop1
bhargava@23372bc849b55ff:~$ chmod +x bash_for_loop1
bhargava@23372bc849b55ff:~$ cat bash_for_loop1
#!/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."
bhargava@23372bc849b55ff:~$ ./bash_for_loop1
Start
learning
from
Javatpoint.
Thank You.
bhargava@23372bc849b55ff:~$
```

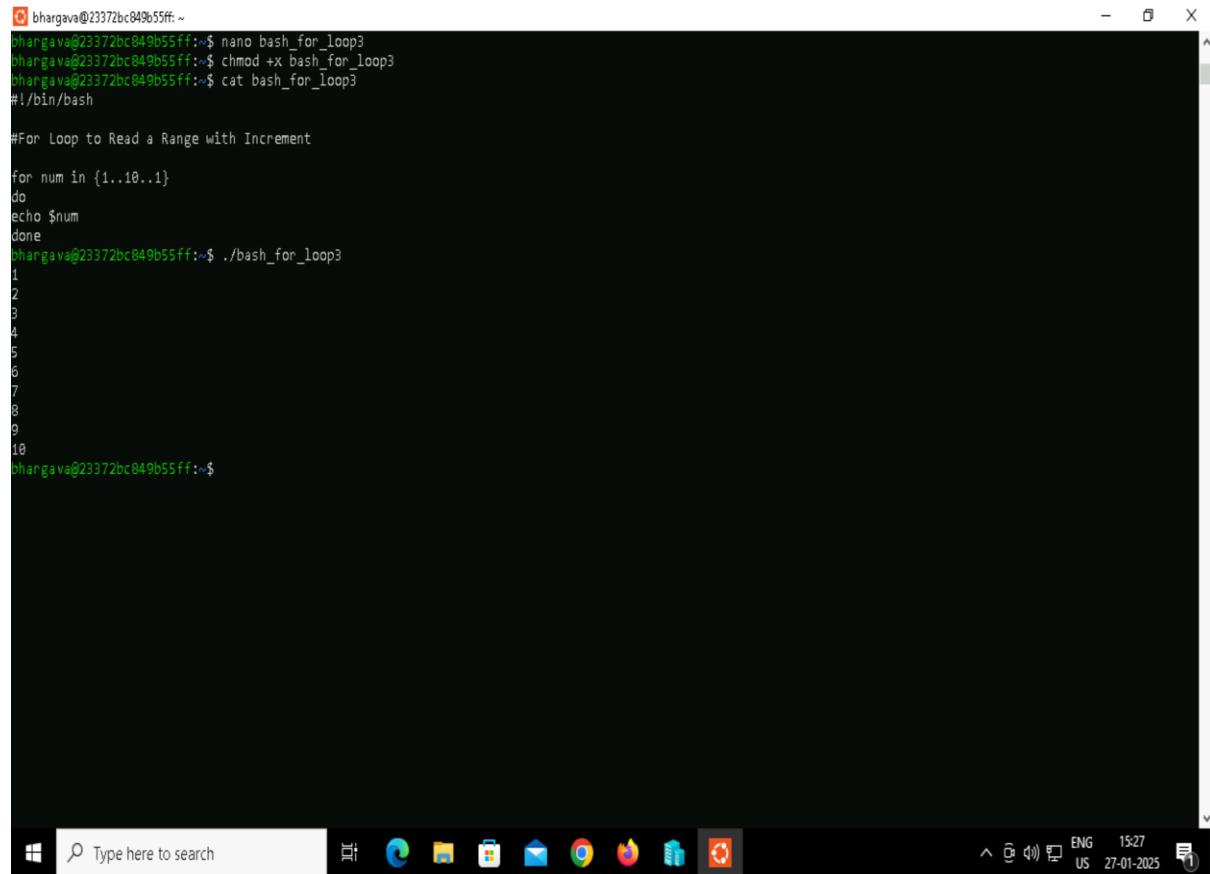
Bash for loop example 2

```
bhargava@23372bc849b55ff:~$ nano bash_for_loop2
bhargava@23372bc849b55ff:~$ chmod +x bash_for_loop2
bhargava@23372bc849b55ff:~$ cat bash_for_loop2
#!/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."
bhargava@23372bc849b55ff:~$ ./bash_for_loop2
1
2
3
4
5
6
7
8
9
10
Series of numbers from 1 to 10.
bhargava@23372bc849b55ff:~$
```

Bash for loop example 3

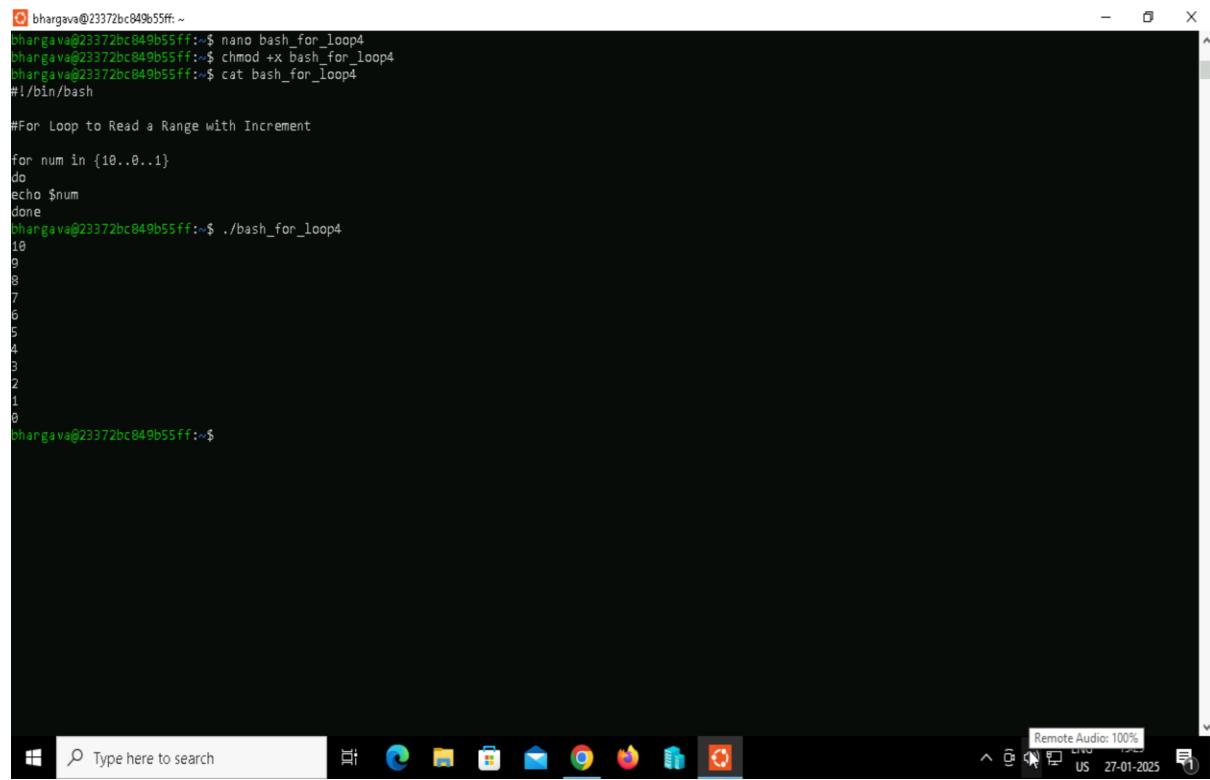


```
bhargava@23372bc849b55ff:~$ nano bash_for_loop3
bhargava@23372bc849b55ff:~$ chmod +x bash_for_loop3
bhargava@23372bc849b55ff:~$ cat bash_for_loop3
#!/bin/bash

#For Loop to Read a Range with Increment

for num in {1..10..1}
do
echo $num
done
bhargava@23372bc849b55ff:~$ ./bash_for_loop3
1
2
3
4
5
6
7
8
9
10
bhargava@23372bc849b55ff:~$
```

Bash for loop example 4



```
bhargava@23372bc849b55ff:~$ nano bash_for_loop4
bhargava@23372bc849b55ff:~$ chmod +x bash_for_loop4
bhargava@23372bc849b55ff:~$ cat bash_for_loop4
#!/bin/bash

#For Loop to Read a Range with Increment

for num in {10..0..1}
do
echo $num
done
bhargava@23372bc849b55ff:~$ ./bash_for_loop4
10
9
8
7
6
5
4
3
2
1
0
bhargava@23372bc849b55ff:~$
```

Bash for loop example 5

```
bhargava@23372bc849b55ff:~$ nano bash_for_loop5
bhargava@23372bc849b55ff:~$ chmod +x bash_for_loop5
bhargava@23372bc849b55ff:~$ cat bash_for_loop5
#!/bin/bash

##Array Declaration
arr=("Welcome" "to" "Javatpoint")

for i in "${arr[@]}"
do
echo $i
done
bhargava@23372bc849b55ff:~$ ./bash_for_loop5
Welcome
to
Javatpoint
bhargava@23372bc849b55ff:~$
```

Bash for loop example 6

```
bhargava@23372bc849b55ff:~$ nano bash_for_loop6
bhargava@23372bc849b55ff:~$ chmod +x bash_for_loop6
bhargava@23372bc849b55ff:~$ cat bash_for_loop6
#!/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
bhargava@23372bc849b55ff:~$ ./bash_for_loop6
Let's
start
learning
from
Javatpoint.
bhargava@23372bc849b55ff:~$
```

Bash for loop example 7

```
bhargava@23372bc849b55ff:~$ nano bash_for_loop7
bhargava@23372bc849b55ff:~$ chmod +x bash_for_loop7
bhargava@23372bc849b55ff:~$ cat bash_for_loop7
#!/bin/bash
#For Loop to Read Three-expression

for ((i=1; i<=10; i++))
do
echo "$i"
done
bhargava@23372bc849b55ff:~$ ./bash_for_loop7
1
2
3
4
5
6
7
8
9
10
bhargava@23372bc849b55ff:~$
```

The screenshot shows a Windows desktop environment with a terminal window open. The terminal window has a black background and white text. It displays a Bash script named 'bash_for_loop7' which prints numbers from 1 to 10. The terminal window is titled with its file name. Below the terminal is a taskbar with various icons and system status indicators.

Bash for loop example 8

```
bhargava@23372bc849b55ff:~$ nano bash_for_loop8
bhargava@23372bc849b55ff:~$ chmod +x bash_for_loop8
bhargava@23372bc849b55ff:~$ cat bash_for_loop8
#!/bin/bash
#Table of 2

for table in {2..100..2}
do
echo $table
if [ $table == 20 ]; then
break
fi
done
bhargava@23372bc849b55ff:~$ ./bash_for_loop8
2
4
6
8
10
12
14
16
18
20
bhargava@23372bc849b55ff:~$
```

The screenshot shows a Windows desktop environment with a terminal window open. The terminal window has a black background and white text. It displays a Bash script named 'bash_for_loop8' which prints a multiplication table of 2. The terminal window is titled with its file name. Below the terminal is a taskbar with various icons and system status indicators. A 'Hyper-V Manager' icon is visible in the taskbar.

Bash for loop example 9

```
bhargava@23372bc849b55ff:~$ nano bash_for_loop9
bhargava@23372bc849b55ff:~$ chmod +x bash_for_loop9
bhargava@23372bc849b55ff:~$ cat bash_for_loop9
#!/bin/bash
#Numbers from 1 to 20, ignoring from 6 to 15 using continue statement"

for ((i=1; i<=20; i++));
do
if [[ $i -gt 5 && $i -lt 16 ]];
then
continue
fi
echo $i
done
bhargava@23372bc849b55ff:~$ ./bash_for_loop9
1
2
3
4
5
16
17
18
19
20
bhargava@23372bc849b55ff:~$
```

The screenshot shows a Windows desktop environment with a terminal window open. The terminal window has a black background and white text. It displays a Bash script named 'bash_for_loop9' which prints numbers from 1 to 20, skipping the range from 6 to 15. The terminal window is titled with the user's session ID. Below the terminal is the Windows taskbar, which includes icons for File Explorer, Edge browser, Task View, Mail, Photos, and File Explorer again. The system tray shows the date and time as 27-01-2025.

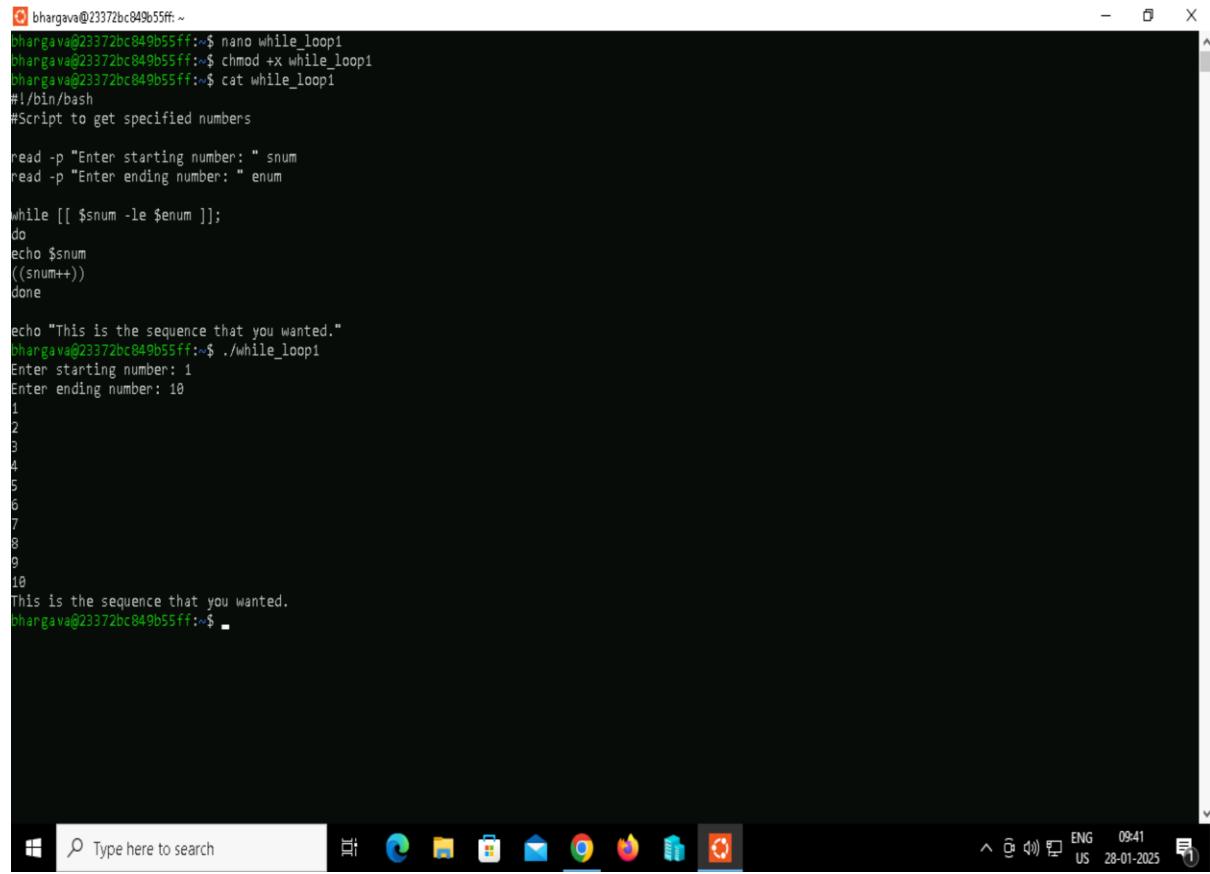
Bash for loop example 10

```
bhargava@23372bc849b55ff:~$ nano bash_for_loop10
bhargava@23372bc849b55ff:~$ chmod +x bash_for_loop10
bhargava@23372bc849b55ff:~$ cat bash_for_loop10
#!/bin/bash

i=1;
for (( ; ; ))
do
sleep 1s
echo "Current Number: $((i++))"
done
bhargava@23372bc849b55ff:~$ ./bash_for_loop10
Current Number: 1
Current Number: 2
Current Number: 3
Current Number: 4
Current Number: 5
Current Number: 6
Current Number: 7
Current Number: 8
Current Number: 9
Current Number: 10
Current Number: 11
Current Number: 12
Current Number: 13
Current Number: 14
Current Number: 15
Current Number: 16
Current Number: 17
Current Number: 18
Current Number: 19
Current Number: 20
Current Number: 21
Current Number: 22
Current Number: 23
Current Number: 24
Current Number: 25
Current Number: 26
Current Number: 27
Current Number: 28
^C
```

This screenshot is similar to the previous one, showing a Windows desktop with a terminal window displaying a Bash script that prints current numbers from 1 to 28. The script uses a for loop with an empty condition and a sleep command. The terminal window is titled with the user's session ID. The Windows taskbar and system tray are visible at the bottom.

Bash While loop example 1



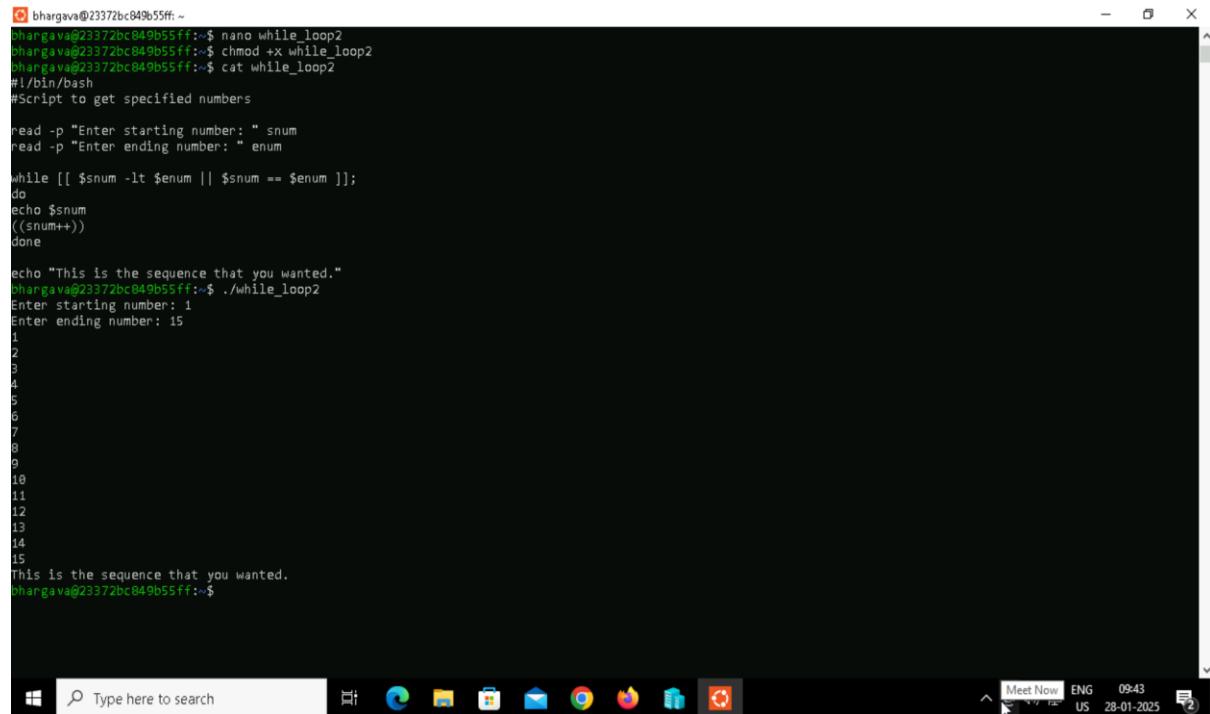
```
bhargava@23372bc849b55ff:~$ nano while_loop1
bhargava@23372bc849b55ff:~$ chmod +x while_loop1
bhargava@23372bc849b55ff:~$ cat while_loop1
#!/bin/bash
#Script to get specified numbers

read -p "Enter starting number: " $num
read -p "Enter ending number: " enum

while [[ $num -le $enum ]];
do
echo $num
((num++))
done

echo "This is the sequence that you wanted."
bhargava@23372bc849b55ff:~$ ./while_loop1
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.
bhargava@23372bc849b55ff:~$
```

Bash While loop example 2



```
bhargava@23372bc849b55ff:~$ nano while_loop2
bhargava@23372bc849b55ff:~$ chmod +x while_loop2
bhargava@23372bc849b55ff:~$ cat while_loop2
#!/bin/bash
#Script to get specified numbers

read -p "Enter starting number: " $num
read -p "Enter ending number: " enum

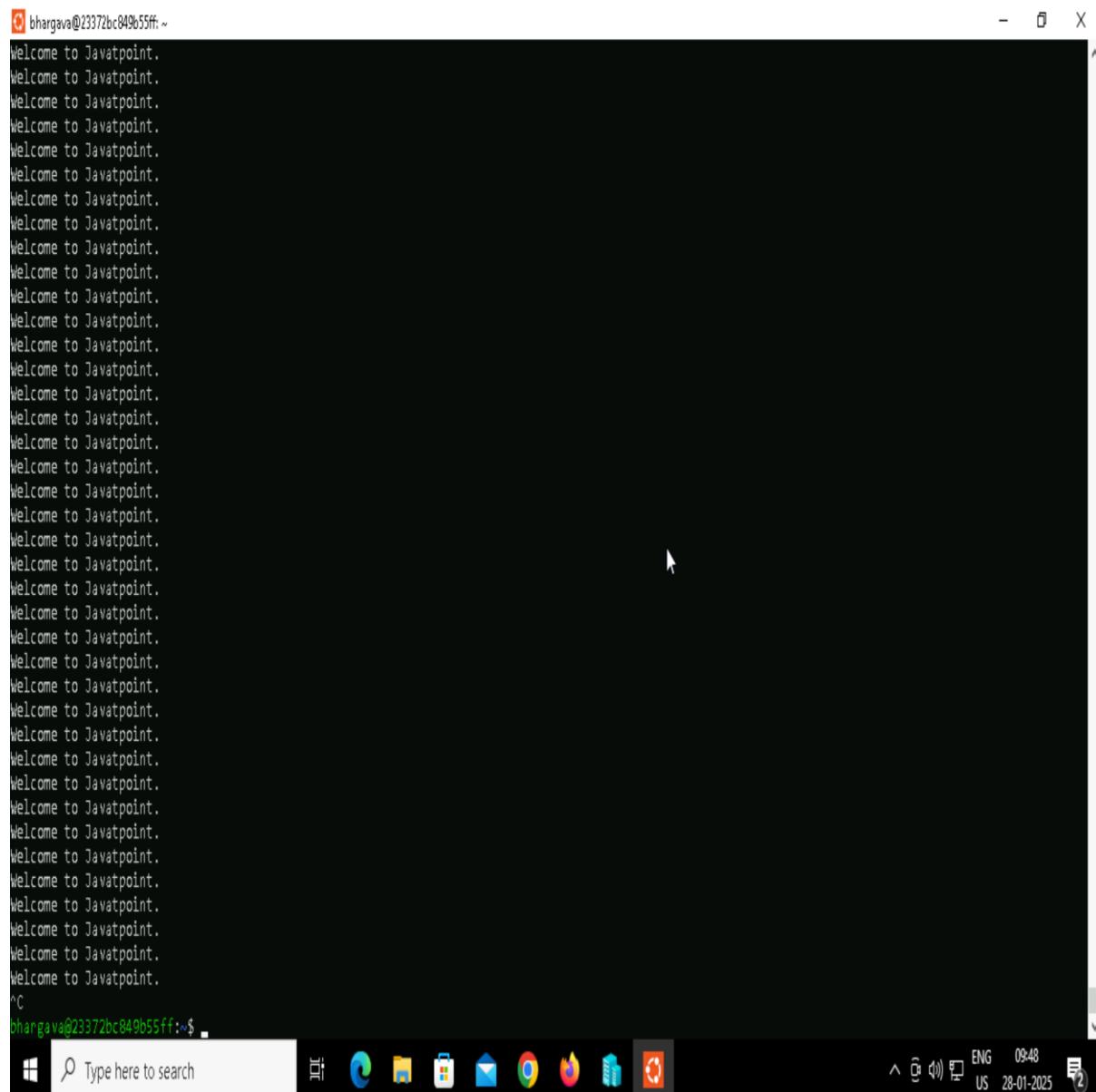
while [[ $num -lt $enum || $num == $enum ]];
do
echo $num
((num++))
done

echo "This is the sequence that you wanted."
bhargava@23372bc849b55ff:~$ ./while_loop2
Enter starting number: 1
Enter ending number: 15
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
This is the sequence that you wanted.
bhargava@23372bc849b55ff:~$
```

Bash While loop example 3

```
bhargava@23372bc849b55ff:~$ nano while_loop3
bhargava@23372bc849b55ff:~$ chmod +x while_loop3
bhargava@23372bc849b55ff:~$ cat while_loop3
#!/bin/bash
#An infinite while loop

while :
do
echo "Welcome to Javatpoint."
done
bhargava@23372bc849b55ff:~$ ./while_loop3
```



Bash While loop example 4

```
phargava@23372bc849b55ff:~$ nano while_loop4
phargava@23372bc849b55ff:~$ chmod +x while_loop4
phargava@23372bc849b55ff:~$ cat while_loop4
#!/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
phargava@23372bc849b55ff:~$ ./while_loop4
Countdown for Website Launching...
10
9
8
7
6
5
4
3
Mission Aborted, Some Technical Error Found.
phargava@23372bc849b55ff:~$
```

Bash While loop example 5

```
bhargava@23372bc849b55ff:~$ nano while_loop5
bhargava@23372bc849b55ff:~$ chmod +x while_loop5
bhargava@23372bc849b55ff:~$ cat while_loop5
#!/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

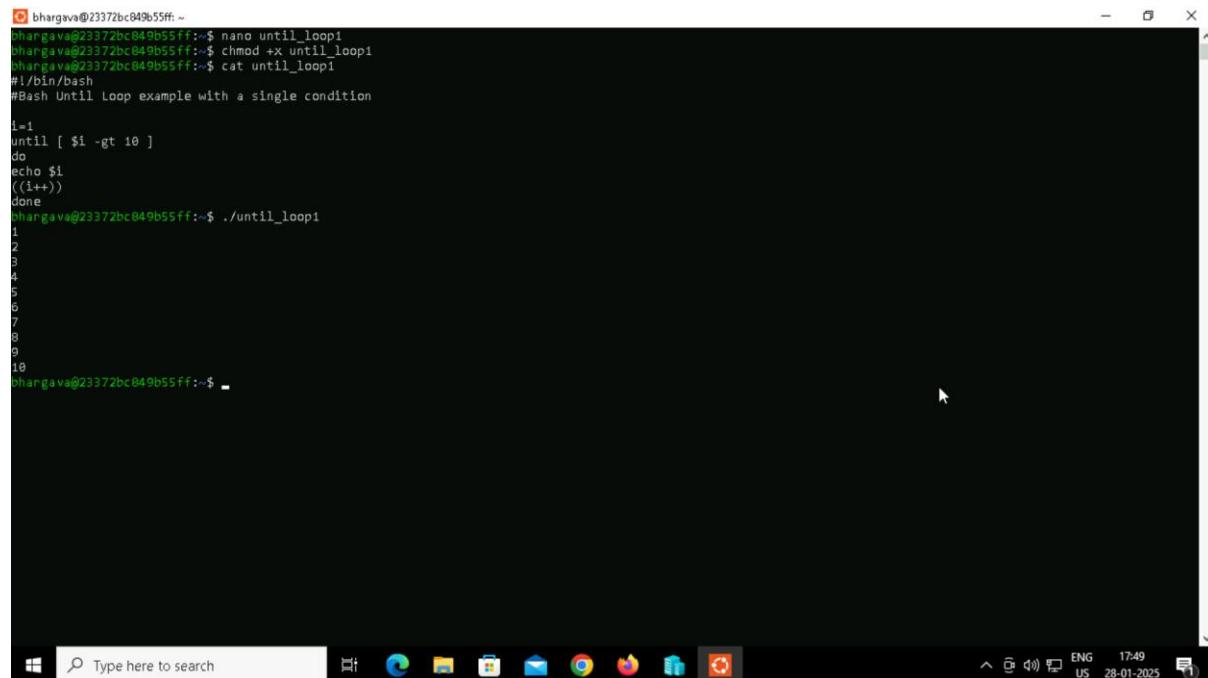
echo "Skipped number 5 using Continue Statement."
bhargava@23372bc849b55ff:~$ ./while_loop5
Current Number : 1
Current Number : 2
Current Number : 3
Current Number : 4
Current Number : 6
Current Number : 7
Current Number : 8
Current Number : 9
Current Number : 10
Current Number : 11
Skipped number 5 using Continue Statement.
```

Bash While loop example 6

```
bhargava@23372bc849b55ff:~$ nano while_loop6
bhargava@23372bc849b55ff:~$ chmod +x while_loop6
bhargava@23372bc849b55ff:~$ cat while_loop6
#!/bin/bash
#while loop example in C style

i=1
while((i <= 10))
do
echo $i
let i++
done
bhargava@23372bc849b55ff:~$ ./while_loop6
1
2
3
4
5
6
7
8
9
10
bhargava@23372bc849b55ff:~$
```

Bash Until loop example 1



A screenshot of a Windows terminal window titled 'Windows Terminal'. The window shows a command-line session in a black background with white text. The session starts with the user's name 'bhargava' followed by the IP address '233.72.0.1' and the port '849b55ff'. The user runs several commands: 'nano until_loop1', 'chmod +x until_loop1', 'cat until_loop1', and finally './until_loop1'. The output of the script is displayed, showing the numbers 1 through 10 on separate lines. The terminal window has a standard Windows title bar and taskbar at the bottom.

```
bhargava@23372bc849b55ff:~$ nano until_loop1
bhargava@23372bc849b55ff:~$ chmod +x until_loop1
bhargava@23372bc849b55ff:~$ cat until_loop1
#!/bin/bash
#Bash Until Loop example with a single condition

i=1
until [ $i -gt 10 ]
do
echo $i
((i++))
done
bhargava@23372bc849b55ff:~$ ./until_loop1
1
2
3
4
5
6
7
8
9
10
bhargava@23372bc849b55ff:~$
```

Bash Until loop example 2

```
bhargava@23372bc849b55ff:~$ nano until_loop2
bhargava@23372bc849b55ff:~$ chmod +x until_loop2
bhargava@23372bc849b55ff:~$ cat until_loop2
#!/bin/bash
#Bash Until Loop example with multiple conditions

max=5
a=1
b=0

until [[ $a -gt $max || $b -gt $max ]];
do
echo "a = $a & b = $b."
((a++))
((b++))
done
bhargava@23372bc849b55ff:~$ ./until_loop2
a = 1 & b = 0.
a = 2 & b = 1.
a = 3 & b = 2.
a = 4 & b = 3.
a = 5 & b = 4.
bhargava@23372bc849b55ff:~$
```

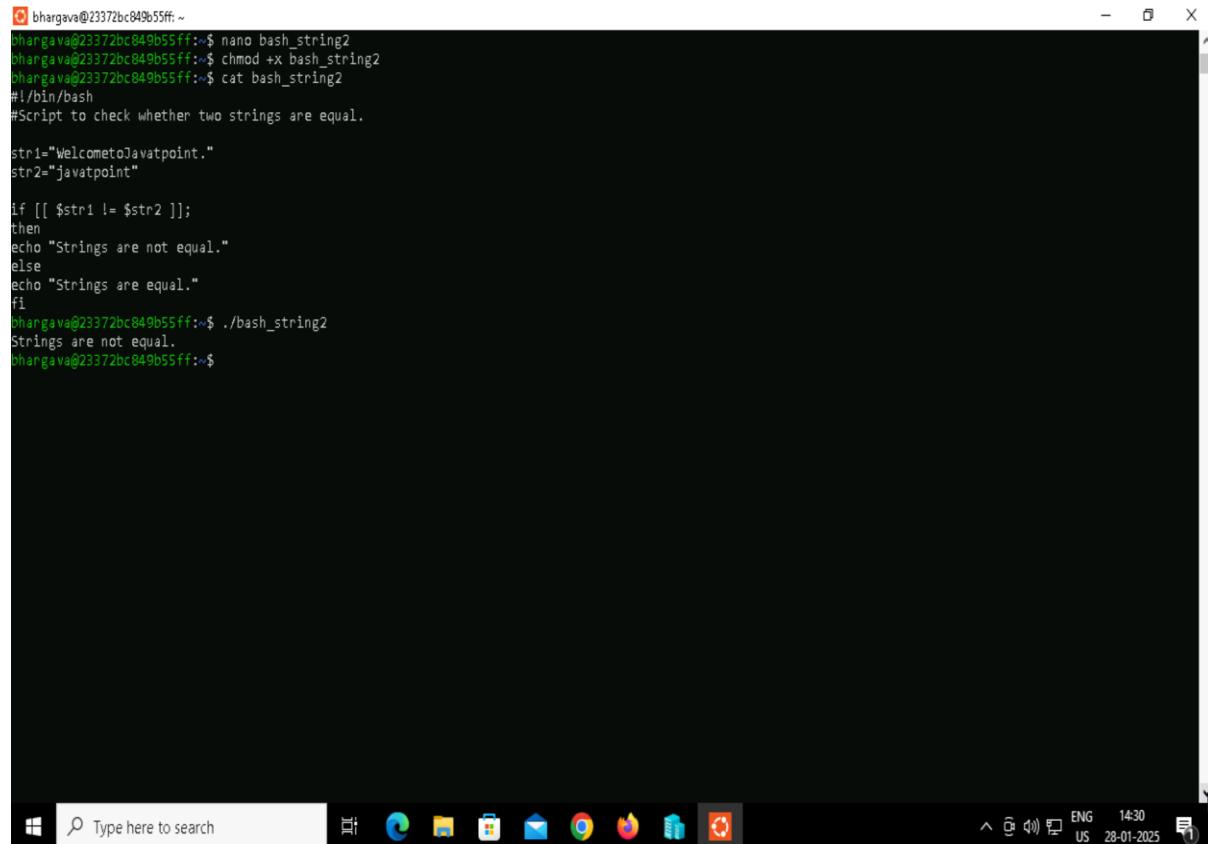
Bash string example 1

```
bhargava@23372bc849b55ff:~$ nano bash_string1
bhargava@23372bc849b55ff:~$ chmod +x bash_string1
bhargava@23372bc849b55ff:~$ cat bash_string1
#!/bin/bash
#Script to check whether two strings are equal.

str1="WelcometoJavatpoint."
str2="javatpoint"

if [ $str1 = $str2 ];
then
echo "Both the strings are equal."
else
echo "Strings are not equal."
fi
bhargava@23372bc849b55ff:~$ ./bash_string1
Strings are not equal.
bhargava@23372bc849b55ff:~$
```

Bash string example 2

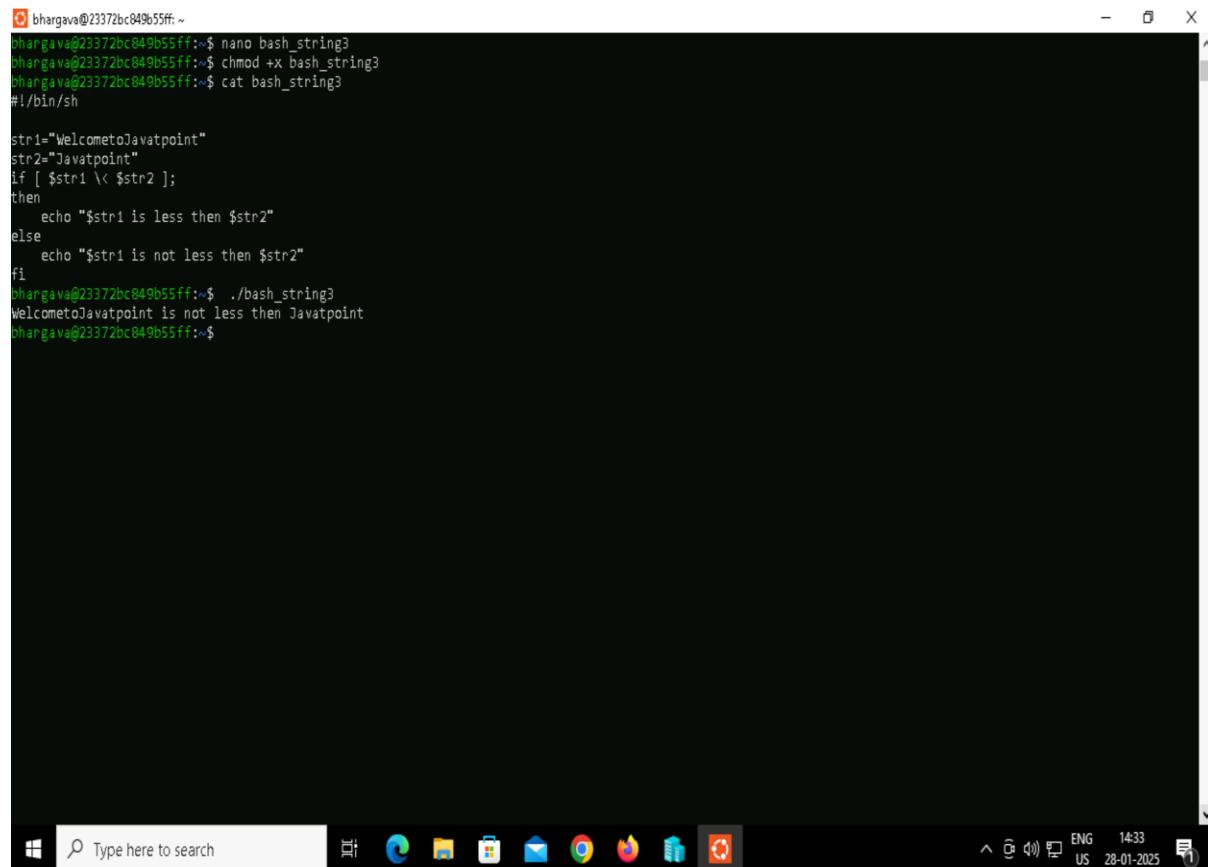


```
bhargava@23372bc849b55ff:~$ nano bash_string2
bhargava@23372bc849b55ff:~$ chmod +x bash_string2
bhargava@23372bc849b55ff:~$ cat bash_string2
#!/bin/bash
#Script to check whether two strings are equal.

str1="WelcometoJavatpoint."
str2="javatpoint"

if [[ $str1 != $str2 ]];
then
echo "Strings are not equal."
else
echo "Strings are equal."
fi
bhargava@23372bc849b55ff:~$ ./bash_string2
Strings are not equal.
bhargava@23372bc849b55ff:~$
```

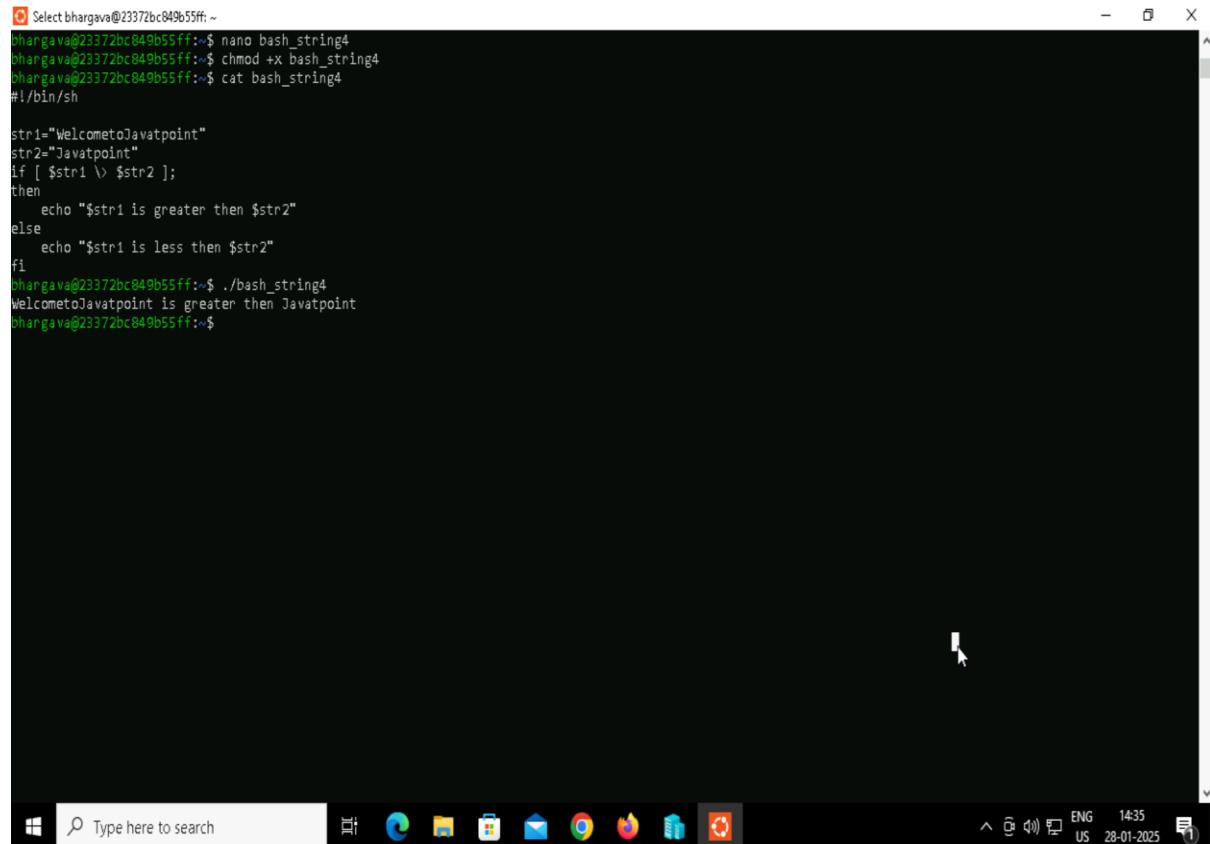
Bash string example 3



```
bhargava@23372bc849b55ff:~$ nano bash_string3
bhargava@23372bc849b55ff:~$ chmod +x bash_string3
bhargava@23372bc849b55ff:~$ cat bash_string3
#!/bin/sh

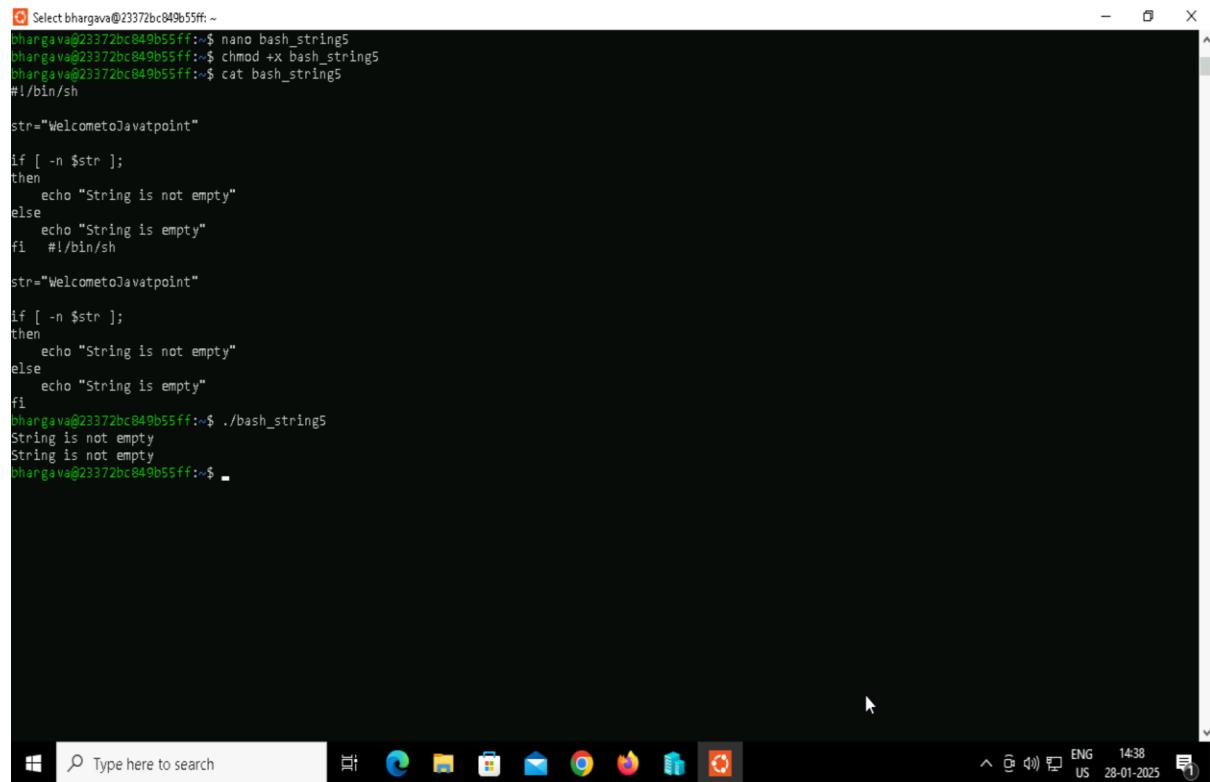
str1="WelcometoJavatpoint"
str2="Javatpoint"
if [ $str1 < $str2 ];
then
    echo "$str1 is less than $str2"
else
    echo "$str1 is not less than $str2"
fi
bhargava@23372bc849b55ff:~$ ./bash_string3
WelcometoJavatpoint is not less than Javatpoint
bhargava@23372bc849b55ff:~$
```

Bash string example 4



```
>Select bhargava@23372bc849b55ff:~  
bhargava@23372bc849b55ff:~$ nano bash_string4  
bhargava@23372bc849b55ff:~$ chmod +x bash_string4  
bhargava@23372bc849b55ff:~$ cat bash_string4  
#!/bin/sh  
  
str1="WelcometoJavatpoint"  
str2="Javatpoint"  
if [ $str1 > $str2 ];  
then  
    echo "$str1 is greater than $str2"  
else  
    echo "$str1 is less than $str2"  
fi  
bhargava@23372bc849b55ff:~$ ./bash_string4  
WelcometoJavatpoint is greater than Javatpoint  
bhargava@23372bc849b55ff:~$
```

Bash string example 5



```
>Select bhargava@23372bc849b55ff:~  
bhargava@23372bc849b55ff:~$ nano bash_string5  
bhargava@23372bc849b55ff:~$ chmod +x bash_string5  
bhargava@23372bc849b55ff:~$ cat bash_string5  
#!/bin/sh  
  
str="WelcometoJavatpoint"  
  
if [ -n $str ];  
then  
    echo "String is not empty"  
else  
    echo "String is empty"  
fi #!/bin/sh  
  
str="WelcometoJavatpoint"  
  
if [ -n $str ];  
then  
    echo "String is not empty"  
else  
    echo "String is empty"  
fi  
bhargava@23372bc849b55ff:~$ ./bash_string5  
String is not empty  
String is not empty  
bhargava@23372bc849b55ff:~$
```

Bash string example 6

```
bhargava@23372bc849b55ff:~$ nano bash_string6
bhargava@23372bc849b55ff:~$ chmod +x bash_string6
bhargava@23372bc849b55ff:~$ cat bash_string6
#!/bin/sh

str=""

if [ -z $str ];
then
    echo "String is empty."
else
    echo "String is non-empty."
fi
bhargava@23372bc849b55ff:~$ ./bash_string6
String is empty.
bhargava@23372bc849b55ff:~$
```

The screenshot shows a Windows terminal window with a black background. It displays a Bash script named 'bash_string6'. The script checks if a variable 'str' is empty using the condition [-z \$str]. If it is, it prints 'String is empty.'. Otherwise, it prints 'String is non-empty.'. The terminal window includes a title bar, a scroll bar on the right, and a taskbar at the bottom with icons for File Explorer, Edge, File Manager, Mail, Google Chrome, and Task View.

Bash String length example 1

```
bhargava@23372bc849b55ff:~$ nano string_length1
bhargava@23372bc849b55ff:~$ chmod +x string_length1
bhargava@23372bc849b55ff:~$ cat string_length1
#!/bin/bash
#Bash program to find the length of a string

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

echo "Length of '$str' is $length"
bhargava@23372bc849b55ff:~$ ./string_length1
Length of 'Welcome to Javatpoint' is 21
bhargava@23372bc849b55ff:~$
```

The screenshot shows a Windows terminal window with a black background. It displays a Bash script named 'string_length1'. The script defines a string 'str' and calculates its length using \${#str}. It then prints the length followed by a message. The terminal window includes a title bar, a scroll bar on the right, and a taskbar at the bottom with icons for File Explorer, Edge, File Manager, Mail, Google Chrome, and Task View.

Bash String length example 2

```
bhargava@23372bc849b55ff:~$ nano string_length2
bhargava@23372bc849b55ff:~$ chmod +x string_length2
bhargava@23372bc849b55ff:~$ cat string_length2
#!/bin/bash
#Bash script to find the length of a string

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

echo "Length of '$str' is $length"
bhargava@23372bc849b55ff:~$ ./string_length2
Length of 'Welcome to Javatpoint' is 21
bhargava@23372bc849b55ff:~$
```

Bash String length example 3

```
bhargava@23372bc849b55ff:~$ nano string_length3
bhargava@23372bc849b55ff:~$ chmod +x string_length3
bhargava@23372bc849b55ff:~$ cat string_length3
#!/bin/bash
#Bash script to find the length of a string

str="I am Kulla Bhargava Ram"
length=`expr "$str" : '\.*'`

echo "Length of '$str' is $length"
bhargava@23372bc849b55ff:~$ ./string_length3
Length of 'I am Kulla Bhargava Ram' is 23
bhargava@23372bc849b55ff:~$
```

Bash String length example 4

```
bhargava@23372bc849b55ff:~$ nano string_length4
bhargava@23372bc849b55ff:~$ chmod +x string_length4
bhargava@23372bc849b55ff:~$ cat string_length4
#!/bin/bash
#Bash script to find the length of a string

str="Welcome to Javatpoint"
length=`echo $str | wc -c`

echo "Length of '$str' is $length"
bhargava@23372bc849b55ff:~$ ./string_length4
Length of 'Welcome to Javatpoint' is 22
bhargava@23372bc849b55ff:~$
```

The screenshot shows a Windows desktop environment with a terminal window open. The terminal window has a dark background and white text. It displays a Bash script named 'string_length4'. The script uses the 'wc' command to count the number of characters in the string 'str'. The output of the script is 'Length of 'Welcome to Javatpoint' is 22'. The system tray at the bottom shows various icons and the date/time as 28-01-2025.

Bash String length example 5

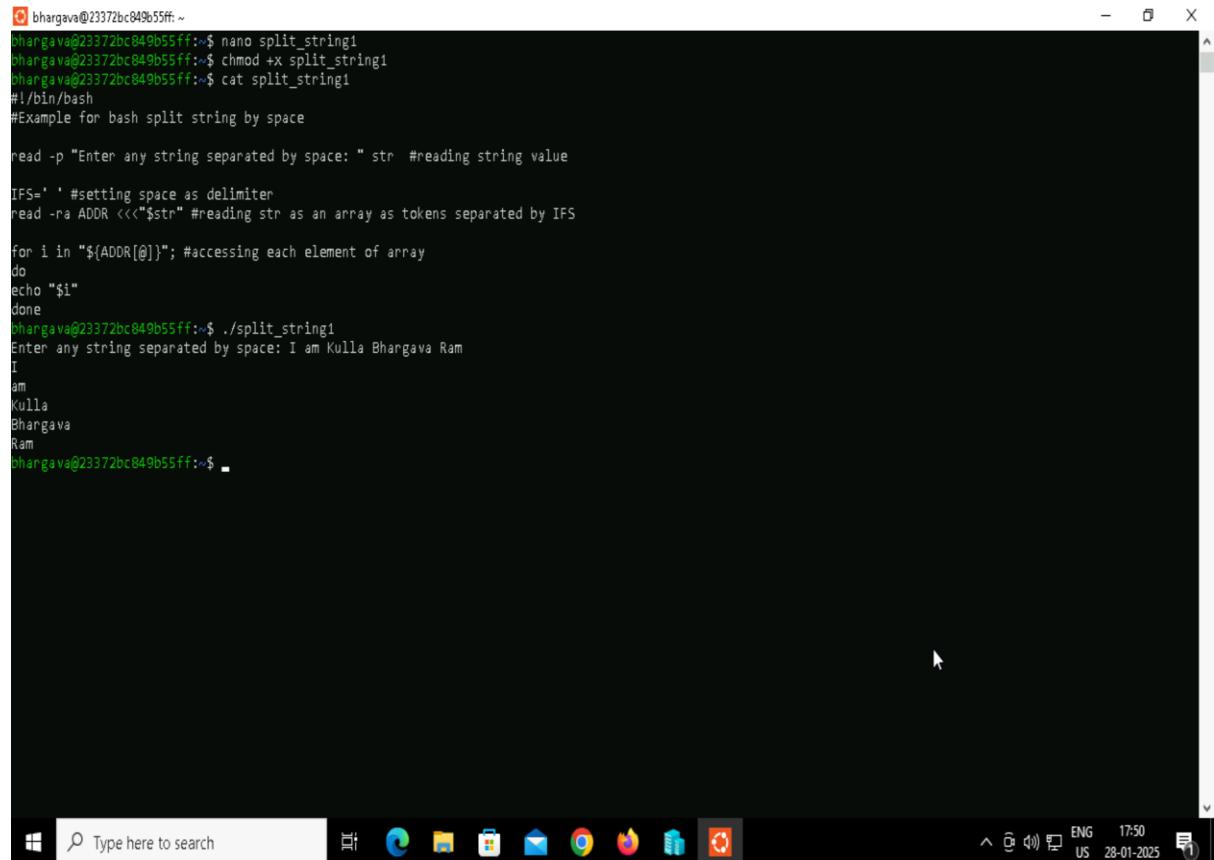
```
bhargava@23372bc849b55ff:~$ nano string_length5
bhargava@23372bc849b55ff:~$ chmod +x string_length5
bhargava@23372bc849b55ff:~$ cat string_length5
#!/bin/bash
#Bash script to find the length of a string

str="Welcome to Javatpoint"
length=`echo $str | awk '{print length}'``

echo "Length of '$str' is $length"
bhargava@23372bc849b55ff:~$ ./string_length5
Length of 'Welcome to Javatpoint' is 21
bhargava@23372bc849b55ff:~$
```

This screenshot is identical to the one above, showing the same terminal window with the Bash script 'string_length5'. The only difference is the output, which shows a length of 21 for the string 'Welcome to Javatpoint'. The system tray at the bottom shows the date/time as 28-01-2025.

Bash Split String Example 1



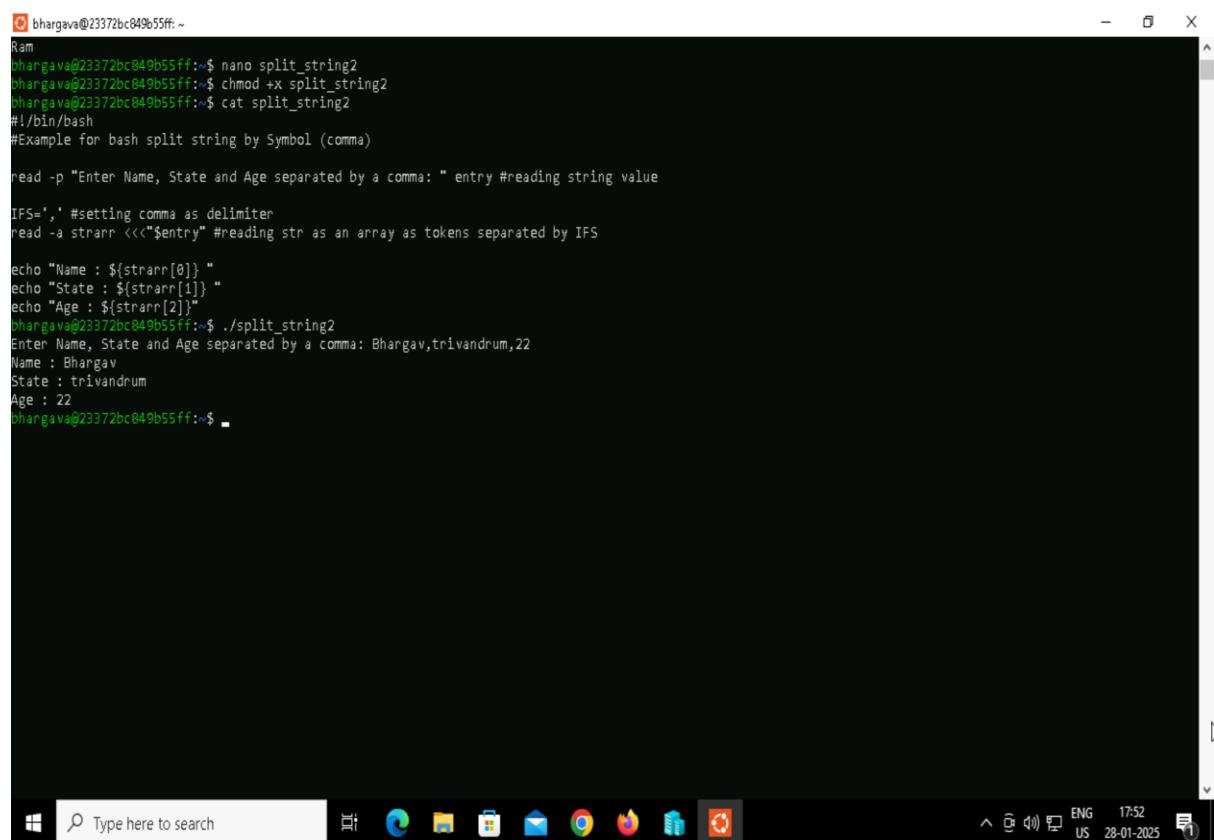
```
bhargava@23372bc849b55ff:~$ nano split_string1
bhargava@23372bc849b55ff:~$ chmod +x split_string1
bhargava@23372bc849b55ff:~$ cat split_string1
#!/bin/bash
#Example for bash split string by space

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
bhargava@23372bc849b55ff:~$ ./split_string1
Enter any string separated by space: I am Kulla Bhargava Ram
I
am
Kulla
Bhargava
Ram
bhargava@23372bc849b55ff:~$
```

Bash Split String Example 2



```
Ram
bhargava@23372bc849b55ff:~$ nano split_string2
bhargava@23372bc849b55ff:~$ chmod +x split_string2
bhargava@23372bc849b55ff:~$ cat split_string2
#!/bin/bash
#Example for bash split string by Symbol (comma)

read -p "Enter Name, State and Age separated by a comma: " entry #reading string value

IFS=',' #setting comma as delimiter
read -a strarr <<<"$entry" #reading str as an array as tokens separated by IFS

echo "Name : ${strarr[0]}"
echo "State : ${strarr[1]}"
echo "Age : ${strarr[2]}"
bhargava@23372bc849b55ff:~$ ./split_string2
Enter Name, State and Age separated by a comma: Bhargav,trivandrum,22
Name : Bhargav
State : trivandrum
Age : 22
bhargava@23372bc849b55ff:~$
```

Bash Split String Example 3

```
bhargava@23372bc849b55ff:~$ nano split_string3
bhargava@23372bc849b55ff:~$ chmod +x split_string3
bhargava@23372bc849b55ff:~$ cat split_string3
#!/bin/bash
#Example for bash split string without $IFS

read -p "Enter any string separated by colon(:) " str #reading string value
readarray -d : -t strarr <<< "$str" #split a string based on the delimiter ':'
printf "\n"

#Print each value of Array with the help of loop
for (( n=0; n < ${#strarr[*]}; n++ ))
do
echo "${strarr[n]}"
done
bhargava@23372bc849b55ff:~$ ./split_string3
Enter any string separated by colon(:) I:Love:Trivandrum

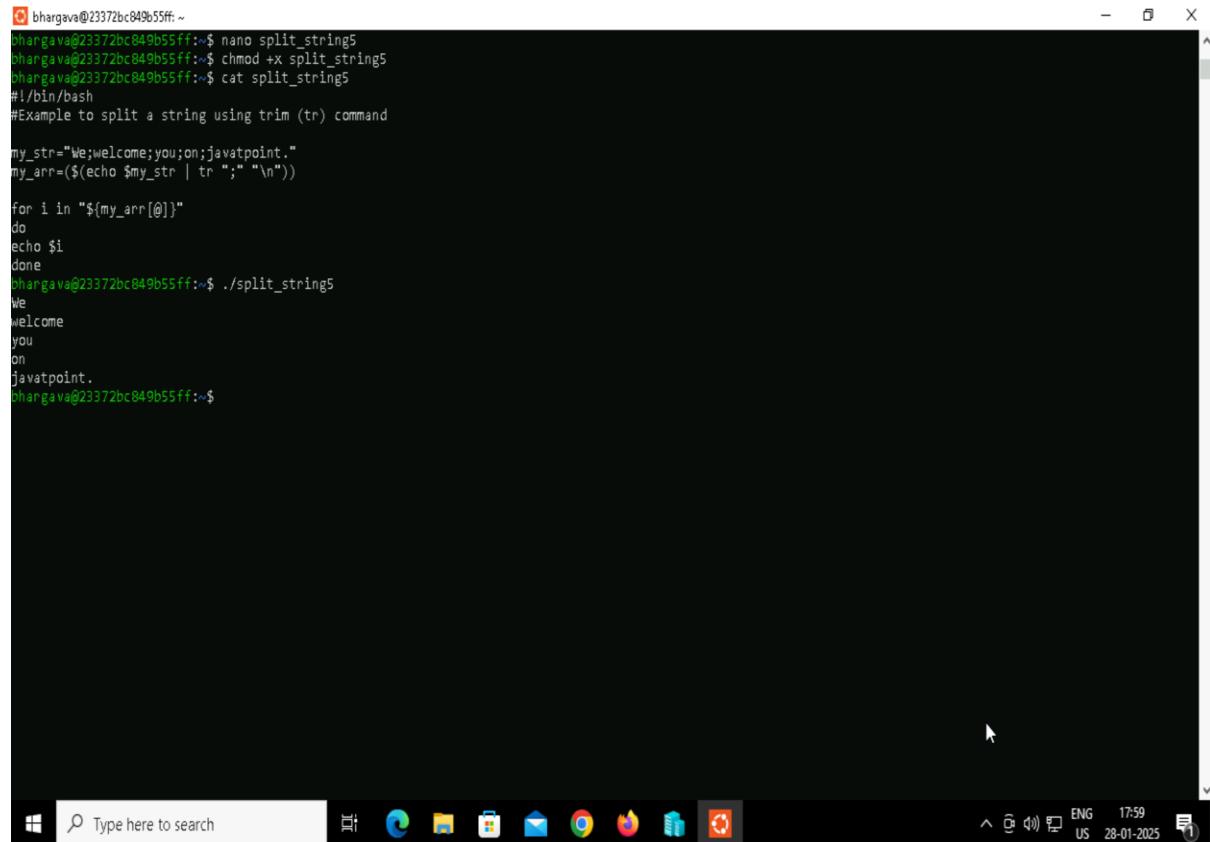
I
Love
Trivandrum
bhargava@23372bc849b55ff:~$
```

Bash Split String Example 4

```
bhargava@23372bc849b55ff:~$ nano split_string4
bhargava@23372bc849b55ff:~$ chmod +x split_string4
bhargava@23372bc849b55ff:~$ cat split_string4
#!/bin/bash
#Example for bash split string by another string

str="WeLearnWelcomeLearnYouLearnOnLearnJavatpoint"
delimiter=Learn
s=$str$delimiter
array=()
while [[ $s ]];
do
array+=("$(s%%$delimiter*)");
s=${s%$delimiter};
done;
declare -p array
bhargava@23372bc849b55ff:~$ ./split_string4
declare -a array=( [0]="We" [1]="Welcome" [2]="You" [3]="On" [4]="Javatpoint")
bhargava@23372bc849b55ff:~$
```

Bash Split String Example 5

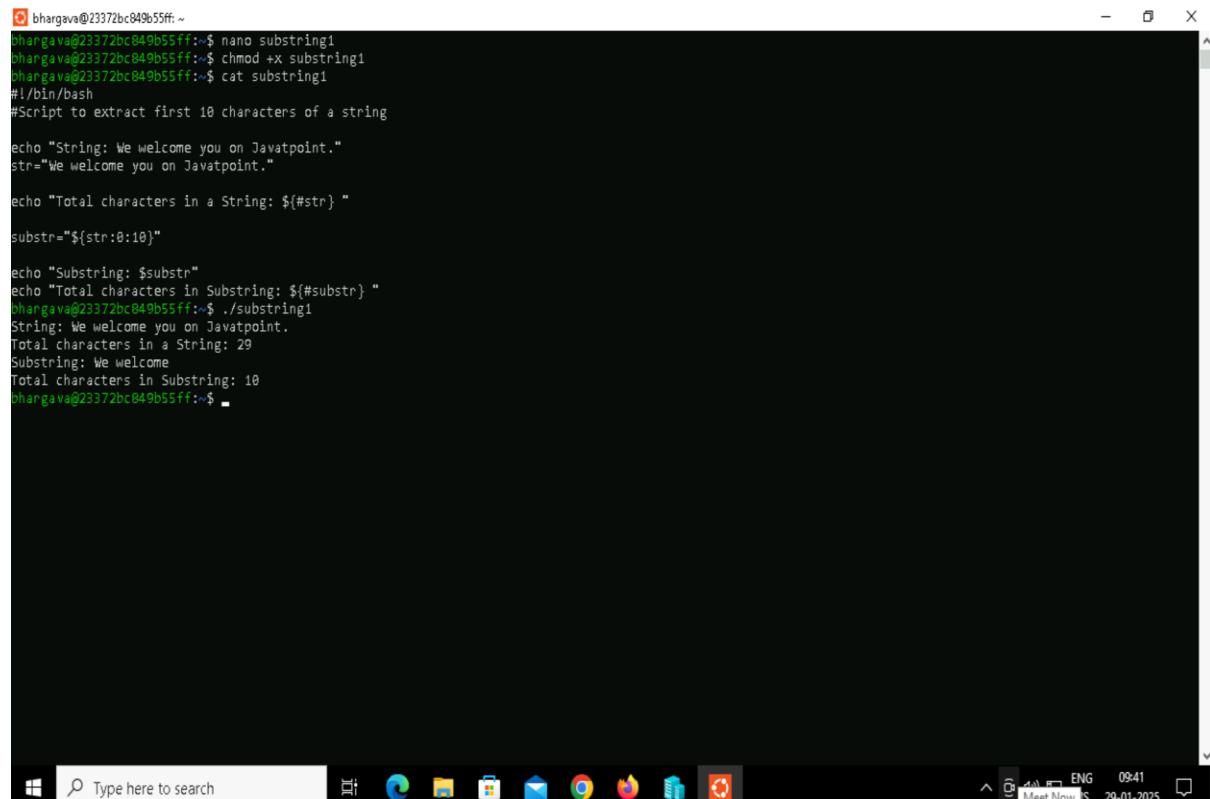


```
bhargava@23372bc849b55ff:~$ nano split_string5
bhargava@23372bc849b55ff:~$ chmod +x split_string5
bhargava@23372bc849b55ff:~$ cat split_string5
#!/bin/bash
#Example to split a string using trim (tr) command

my_str="We;welcome;you;on;javatpoint."
my_arr=($(echo $my_str | tr ";" "\n"))

for i in "${my_arr[@]}"
do
echo $i
done
bhargava@23372bc849b55ff:~$ ./split_string5
We
welcome
you
on
javatpoint.
bhargava@23372bc849b55ff:~$
```

Bash Substring Example 1



```
bhargava@23372bc849b55ff:~$ nano substring1
bhargava@23372bc849b55ff:~$ chmod +x substring1
bhargava@23372bc849b55ff:~$ cat substring1
#!/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: ${#substr} "
bhargava@23372bc849b55ff:~$ ./substring1
String: We welcome you on Javatpoint.
Total characters in a String: 29
Substring: We welcome
Total characters in Substring: 10
bhargava@23372bc849b55ff:~$
```

Bash Substring Example 2

```
bhargava@23372bc849b55ff:~$ nano substring2
bhargava@23372bc849b55ff:~$ chmod +x substring2
bhargava@23372bc849b55ff:~$ cat substring2
#!/bin/bash
#Script to print from 11th character onwards

str="We welcome you on Javatpoint."
substr="${str:11}"
echo "$substr"
bhargava@23372bc849b55ff:~$ ./substring2
you on Javatpoint.
bhargava@23372bc849b55ff:~$
```

The screenshot shows a Windows terminal window with a black background and white text. It displays a Bash script named 'substring2' which prints the substring from the 11th character of the string 'We welcome you on Javatpoint.' The output is 'you on Javatpoint.'. The terminal window has a title bar, a scroll bar on the right, and a taskbar at the bottom with various icons.

Bash Substring Example 3

```
bhargava@23372bc849b55ff:~$ nano substring3
bhargava@23372bc849b55ff:~$ chmod +x substring3
bhargava@23372bc849b55ff:~$ cat substring3
#!/bin/bash
#Script to print 11th character of a String

str="We welcome you on Javatpoint."
substr="${str:11:1}"
echo "$substr"
bhargava@23372bc849b55ff:~$ ./substring3
y
bhargava@23372bc849b55ff:~$
```

The screenshot shows a Windows terminal window with a black background and white text. It displays a Bash script named 'substring3' which prints the 11th character of the string 'We welcome you on Javatpoint.', resulting in 'y'. The terminal window has a title bar, a scroll bar on the right, and a taskbar at the bottom with various icons.

Bash Substring Example 4

```
bhargava@23372bc849b55ff:~$ nano substring4
bhargava@23372bc849b55ff:~$ chmod +x substring4
bhargava@23372bc849b55ff:~$ cat substring4
#!/bin/bash
#Script to extract 11 characters from last

str="We welcome you on Javatpoint."
substr="${str: -11}"
echo "$substr"
bhargava@23372bc849b55ff:~$ ./substring4
Javatpoint.
bhargava@23372bc849b55ff:~$
```

Bash Concatenate String Example 1

```
bhargava@23372bc849b55ff:~$ nano concatenate1
bhargava@23372bc849b55ff:~$ chmod +x concatenate1
bhargava@23372bc849b55ff:~$ cat concatenate1
#!/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 $str3
bhargava@23372bc849b55ff:~$ ./concatenate1
We welcome you on Javatpoint.
bhargava@23372bc849b55ff:~$
```

Bash Concatenate String Example 2

```
bhargava@23372bc849b55ff:~$ nano concatenate2
bhargava@23372bc849b55ff:~$ chmod +x concatenate2
bhargava@23372bc849b55ff:~$ cat concatenate2
#!/bin/bash
#Script to Concatenate Strings

#Declaring String Variable
str="We welcome you"

#Add the variable within the string
echo "$str on Javatpoint."
bhargava@23372bc849b55ff:~$ ./concatenate2
We welcome you on Javatpoint.
bhargava@23372bc849b55ff:~$
```

Activate Windows
Go to Settings to activate Windows.

Type here to search Show hidden icons 17:37
29-01-2025

Bash Concatenate String Example 3

```
bhargava@23372bc849b55ff:~$ nano concatenate3
bhargava@23372bc849b55ff:~$ chmod +x concatenate3
bhargava@23372bc849b55ff:~$ cat concatenate3
#!/bin/bash
echo "Printing the name of the programming languages"
#Initializing the variable before combining
lang=" "
#for loop for reading the list
for value in 'java''python''C''C++';
do
lang+="$value " #Combining the list values using append operator
done
#printing the combined values
echo "$lang"
bhargava@23372bc849b55ff:~$ ./concatenate3
Printing the name of the programming languages
javapythonCC++
bhargava@23372bc849b55ff:~$
```

Activate Windows
Go to Settings to activate Windows.

Type here to search Show hidden icons 17:51
US 29-01-2025

Bash Concatenate String Example 4

```
bhargava@23372bc849b55ff:~$ nano concatenate4
bhargava@23372bc849b55ff:~$ nano concatenate4
bhargava@23372bc849b55ff:~$ chmod +x concatenate4
bhargava@23372bc849b55ff:~$ cat concatenate4
#!/bin/bash

str="Welcome"
printf -v new_str "$str to Javatpoint."
echo $new_str
bhargava@23372bc849b55ff:~$ ./concatenate4
Welcome to Javatpoint.
bhargava@23372bc849b55ff:~$
```

The screenshot shows a terminal window on a Windows operating system. The terminal window has a black background and white text. It displays a Bash script named 'concatenate4'. The script defines a variable 'str' with the value 'Welcome', then uses the 'printf' command with a format string '-v' and a variable 'new_str' to append 'to Javatpoint.' to the end of 'str'. Finally, it prints the concatenated string using 'echo'. The terminal window also shows the command 'cat concatenate4' being run to view the script's contents. The desktop taskbar at the bottom includes icons for File Explorer, Edge, File Manager, Mail, Google Chrome, and Firefox. The system tray shows the date and time as 29-01-2025, 17:57, and language settings as ENG US. An 'Activate Windows' watermark is visible in the center of the screen.

Bash Concatenate String Example 5

```
bhargava@23372bc849b55ff:~$ nano concatenate5
bhargava@23372bc849b55ff:~$ chmod +x concatenate5
bhargava@23372bc849b55ff:~$ cat concatenate5
#!/bin/bash

str="Welcome to"
newstr="${str} Javatpoint."
echo "$newstr"
bhargava@23372bc849b55ff:~$ ./concatenate5
Welcome to Javatpoint.
bhargava@23372bc849b55ff:~$
```

This screenshot shows a similar terminal setup to the previous one, but the script content has changed. It uses brace expansion to create a new string 'newstr' by concatenating 'str' and ' Javatpoint.'. The terminal window shows the command 'cat concatenate5' being run to view the script. The desktop taskbar and system tray are identical to the first screenshot, including the 'Activate Windows' watermark.

Bash Concatenate String Example 6

```
bhargava@23372bc849b55ff:~$ nano concatenate6
bhargava@23372bc849b55ff:~$ chmod +x concatenate6
bhargava@23372bc849b55ff:~$ cat concatenate6
#!/bin/bash

str1="Hello"
str2="World!"

echo "${str1} ${str2}"
bhargava@23372bc849b55ff:~$ ./concatenate6
Hello_World!
bhargava@23372bc849b55ff:~$
```

The screenshot shows a Windows terminal window with a black background and white text. It displays a Bash script named 'concatenate6' which concatenates two strings, 'Hello' and 'World!', separated by a space. The terminal window has a title bar with the window name and standard minimize, maximize, and close buttons. At the bottom, there's a taskbar with various icons and a system tray showing the date and time.

Bash Concatenate String Example 7

```
bhargava@23372bc849b55ff:~$ nano concatenate7
bhargava@23372bc849b55ff:~$ chmod +x concatenate7
bhargava@23372bc849b55ff:~$ cat concatenate7
#!/bin/bash
#String Concatenation by Character (,) with User Input

read -p "Enter First Name: " name
read -p "Enter State: " state
read -p "Enter Age: " age

combine="$name,$state,$age"

echo "Name, State, Age: $combine"
bhargava@23372bc849b55ff:~$ ./concatenate7
Enter First Name: Bhargav
Enter State: Trivandrum
Enter Age: 22
Name, State, Age: Bhargav,Trivandrum,22
bhargava@23372bc849b55ff:~$
```

This screenshot shows a similar Windows terminal setup. It demonstrates a Bash script 'concatenate7' that reads three pieces of user input: First Name, State, and Age, and then concatenates them using a comma as a separator. The terminal window includes a status message about activating Windows.

Bash Functions Example 1

```
bhargava@23372bc849b55ff:~$ nano functions1
bhargava@23372bc849b55ff:~$ chmod +x functions1
bhargava@23372bc849b55ff:~$ cat functions1
#!/bin/bash
#Script to pass and access arguments

function_arguments()
{
    echo $1
    echo $2
    echo $3
    echo $4
    echo $5
}

#Calling function_arguments
function_arguments "We" "Welcome" "you" "on" "Javatpoint."
bhargava@23372bc849b55ff:~$ ./functions1
We
Welcome
you
on
Javatpoint.

bhargava@23372bc849b55ff:~$
```

The screenshot shows a Windows terminal window with a black background and white text. It displays the execution of a Bash script named 'functions1'. The script defines a function 'function_arguments' that prints its five arguments. It then calls this function with the words 'We', 'Welcome', 'you', 'on', and 'Javatpoint.' respectively. The terminal window has a title bar at the top and a taskbar at the bottom with various icons for Microsoft applications like File Explorer, Edge, and Mail. A watermark for 'Activate Windows' is visible in the center of the screen.

Bash Functions Example 2

```
bhargava@23372bc849b55ff:~$ nano functions2
bhargava@23372bc849b55ff:~$ chmod +x functions2
bhargava@23372bc849b55ff:~$ cat functions2
#!/bin/bash

v1='A'
v2='B'

my_var () {
    local v1='C'
    v2='D'
    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."
echo "v2 is $v2."
bhargava@23372bc849b55ff:~$ ./functions2
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.

bhargava@23372bc849b55ff:~$
```

This screenshot shows a similar Windows terminal window to the previous one, displaying the execution of 'functions2'. This script uses a function 'my_var' which changes the values of 'v1' and 'v2' to 'C' and 'D' respectively. It then prints the original values of 'v1' and 'v2', executes the function, and finally prints the modified values. The terminal window interface and taskbar are identical to the first screenshot, with the 'Activate Windows' watermark present.

Bash Functions Example 3

```
bhargava@23372bc849b55ff:~$ nano functions3
bhargava@23372bc849b55ff:~$ chmod +x functions3
bhargava@23372bc849b55ff:~$ cat functions3
#!/bin/bash
#Setting up a return status for a function

print_it () {
    echo Hello $1
    return 5
}

print_it User
print_it Reader
echo The previous function returned a value of $?
bhargava@23372bc849b55ff:~$ ./functions3
Hello User
Hello Reader
The previous function returned a value of 5
bhargava@23372bc849b55ff:~$
```

The terminal window shows the execution of a Bash script named 'functions3'. It defines a function 'print_it' that prints 'Hello \$1' and returns a status of 5. It then calls 'print_it' with arguments 'User' and 'Reader', and prints the return value of 5. The taskbar at the bottom shows various application icons.

Bash Functions Example 4

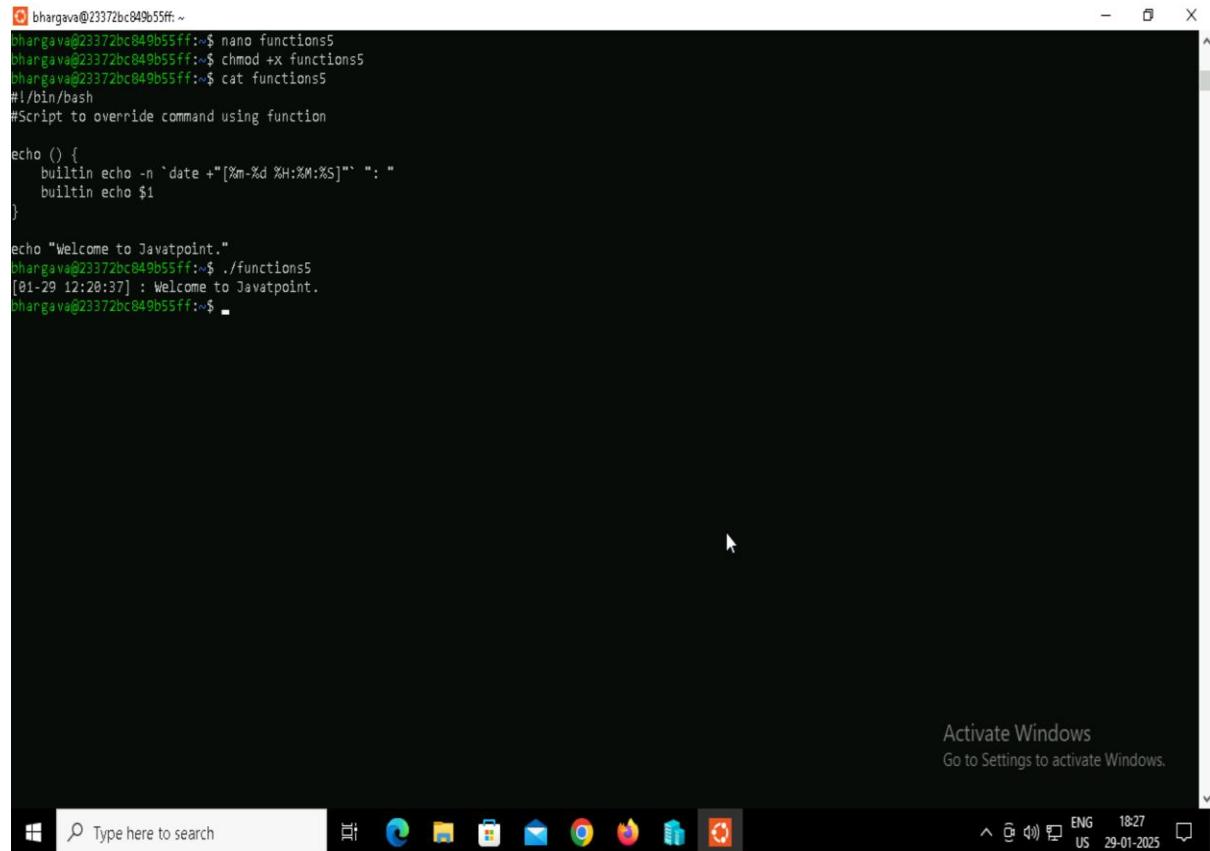
```
bhargava@23372bc849b55ff:~$ nano functions4
bhargava@23372bc849b55ff:~$ chmod +x functions4
bhargava@23372bc849b55ff:~$ cat functions4
#!/bin/bash

print_it () {
    local my_greet="Welcome to Javatpoint."
    echo "$my_greet"
}

my_greet=$(print_it)
echo $my_greet
bhargava@23372bc849b55ff:~$ ./functions4
Welcome to Javatpoint.
bhargava@23372bc849b55ff:~$
```

The terminal window shows the execution of a Bash script named 'functions4'. It defines a function 'print_it' that sets a local variable 'my_greet' to 'Welcome to Javatpoint.' and prints it. It then calls 'print_it' and prints the result. The taskbar at the bottom shows various application icons.

Bash Functions Example 5



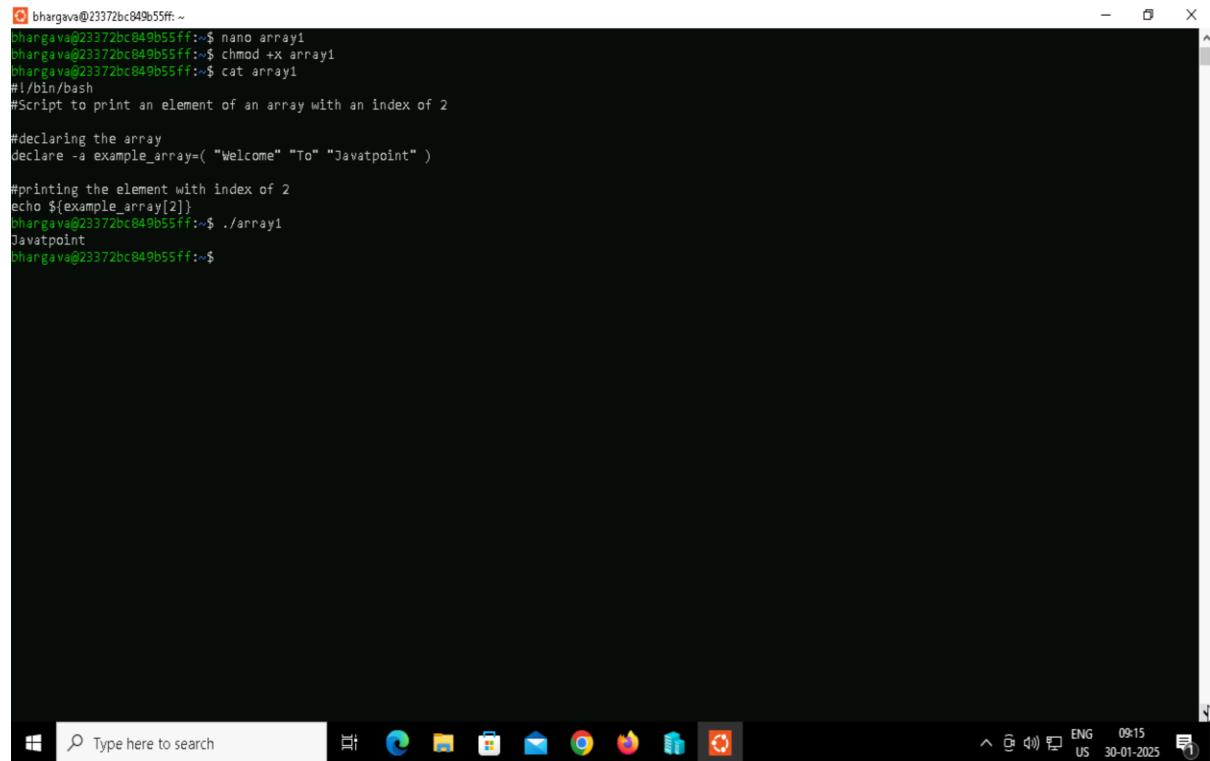
```
bhargava@23372bc849b55ff:~$ nano functions5
bhargava@23372bc849b55ff:~$ chmod +x functions5
bhargava@23372bc849b55ff:~$ cat functions5
#!/bin/bash
#Script to override command using function

echo () {
    builtin echo -n `date +"%m-%d %H:%M:%S"`" ":" "
    builtin echo $1
}

echo "Welcome to Javatpoint."
bhargava@23372bc849b55ff:~$ ./functions5
[01-29 12:20:37] : Welcome to Javatpoint.
bhargava@23372bc849b55ff:~$
```

The screenshot shows a Windows desktop environment with a terminal window open. The terminal window has a black background and white text. It displays a Bash script named 'functions5' which overrides the 'echo' command. When run, it prints the current date and time followed by the original 'echo' content. The terminal window is titled with its file name. Below the terminal is a taskbar with various icons and system status indicators.

Bash Array Example 1



```
bhargava@23372bc849b55ff:~$ nano array1
bhargava@23372bc849b55ff:~$ chmod +x array1
bhargava@23372bc849b55ff:~$ cat array1
#!/bin/bash
#Script to print an element of an array with an index of 2

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

#printing the element with index of 2
echo ${example_array[2]}
bhargava@23372bc849b55ff:~$ ./array1
Javatpoint
bhargava@23372bc849b55ff:~$
```

The screenshot shows a Windows desktop environment with a terminal window open. The terminal window has a black background and white text. It displays a Bash script named 'array1' which creates an array and prints its third element. The terminal window is titled with its file name. Below the terminal is a taskbar with various icons and system status indicators.

Bash Array Example 2

```
bhargava@23372bc849b55ff:~$ nano array2
bhargava@23372bc849b55ff:~$ chmod +x array2
bhargava@23372bc849b55ff:~$ cat array2
#!/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[@]}"

bhargava@23372bc849b55ff:~$ ./array2
WelcomeToJavatpoint
bhargava@23372bc849b55ff:~$
```

Bash Array Example 3

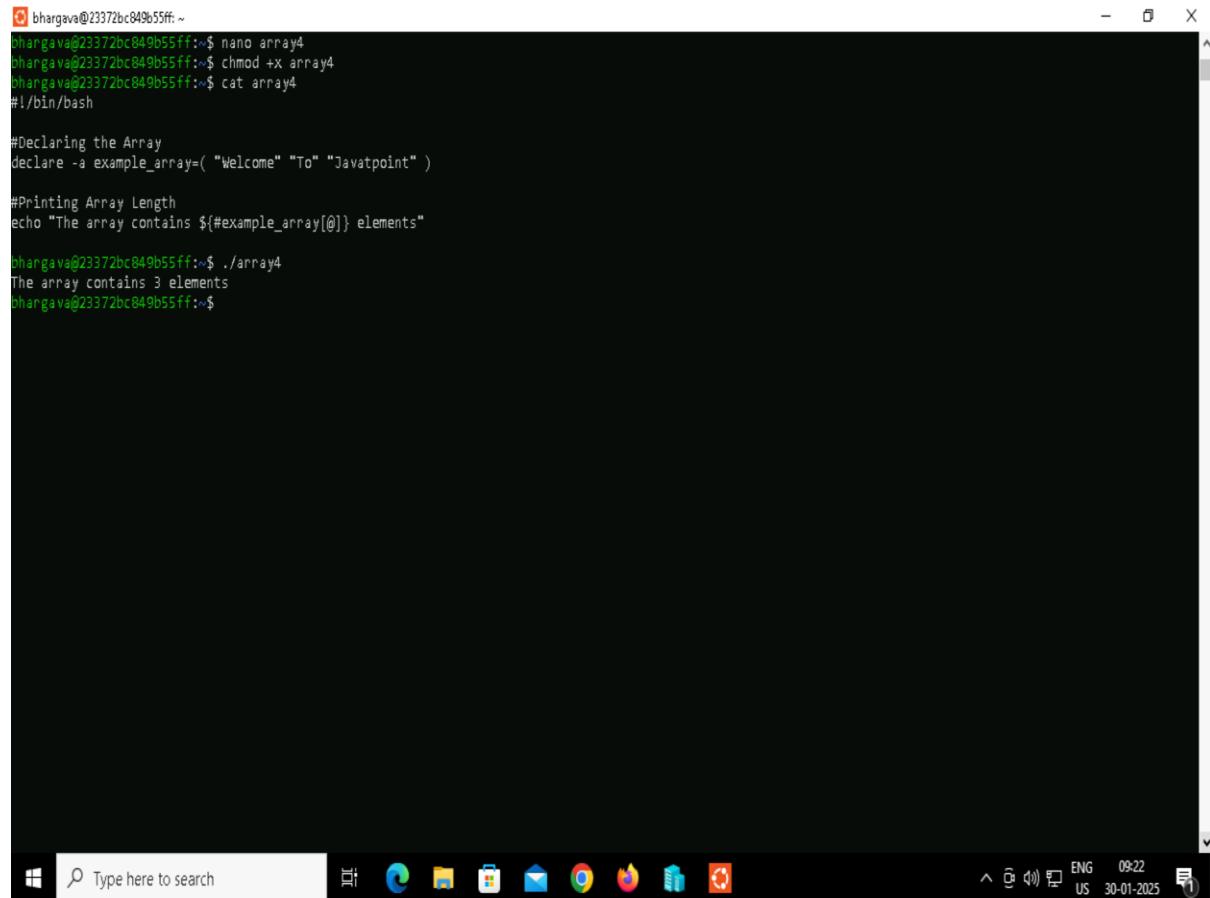
```
bhargava@23372bc849b55ff:~$ nano array3
bhargava@23372bc849b55ff:~$ chmod +x array3
bhargava@23372bc849b55ff:~$ cat array3
#!/bin/bash
#Script to print the keys of the array

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

#Printing the Keys
echo "${!example_array[@]}"

bhargava@23372bc849b55ff:~$ ./array3
0 1 2
bhargava@23372bc849b55ff:~$
```

Bash Array Example 4



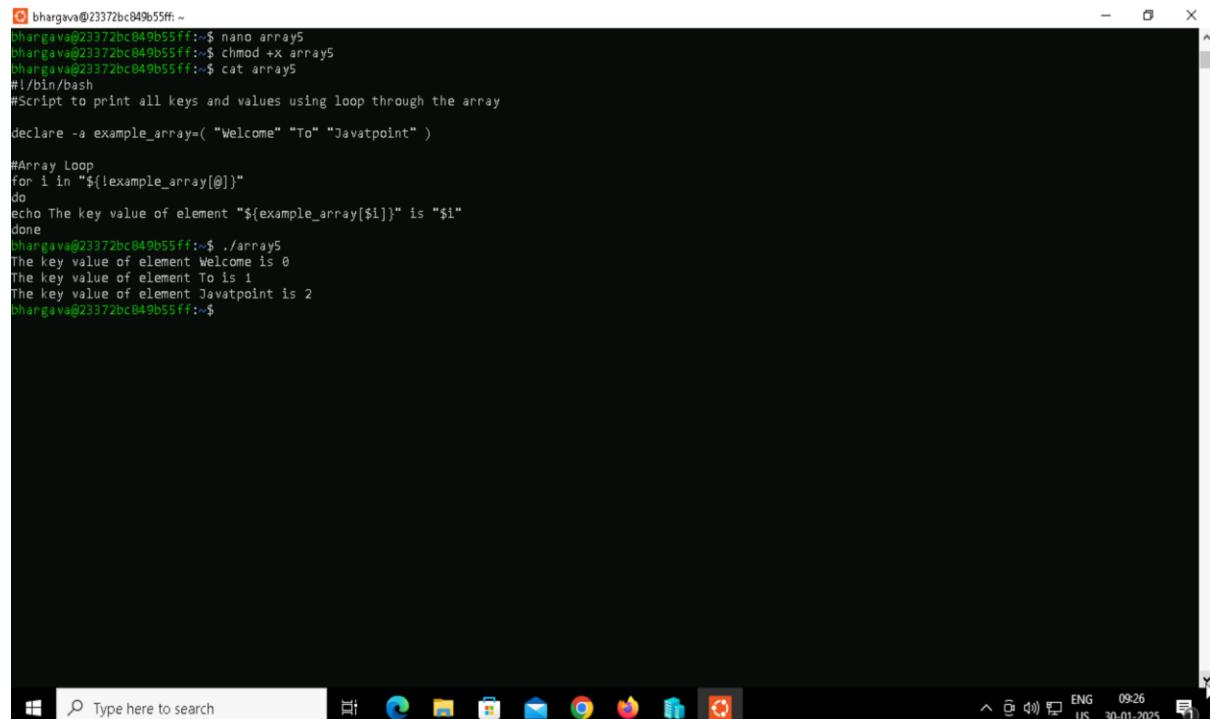
```
bhargava@23372bc849b55ff:~$ nano array4
bhargava@23372bc849b55ff:~$ chmod +x array4
bhargava@23372bc849b55ff:~$ cat array4
#!/bin/bash

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

#Printing Array Length
echo "The array contains ${#example_array[@]} elements"

bhargava@23372bc849b55ff:~$ ./array4
The array contains 3 elements
bhargava@23372bc849b55ff:~$
```

Bash Array Example 5

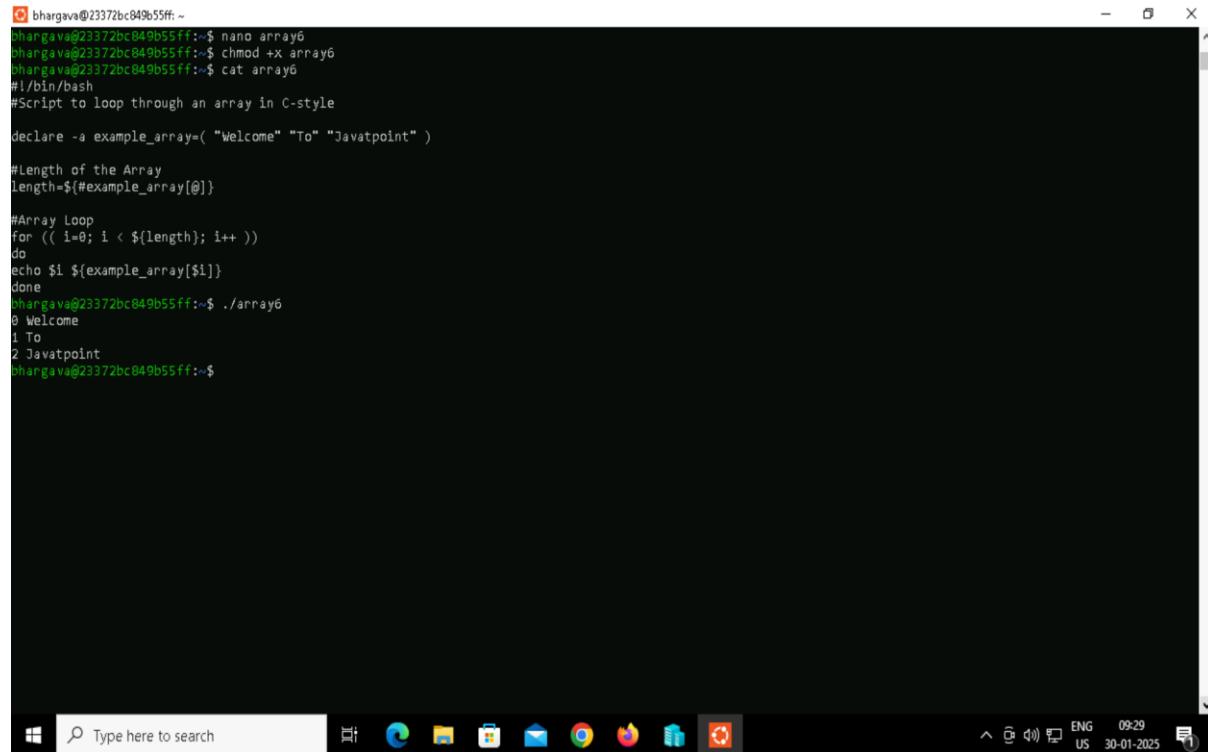


```
bhargava@23372bc849b55ff:~$ nano array5
bhargava@23372bc849b55ff:~$ chmod +x array5
bhargava@23372bc849b55ff:~$ cat array5
#!/bin/bash
#Script to print all keys and values using loop through the array

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
bhargava@23372bc849b55ff:~$ ./array5
The key value of element Welcome is 0
The key value of element To is 1
The key value of element Javatpoint is 2
bhargava@23372bc849b55ff:~$
```

Bash Array Example 6



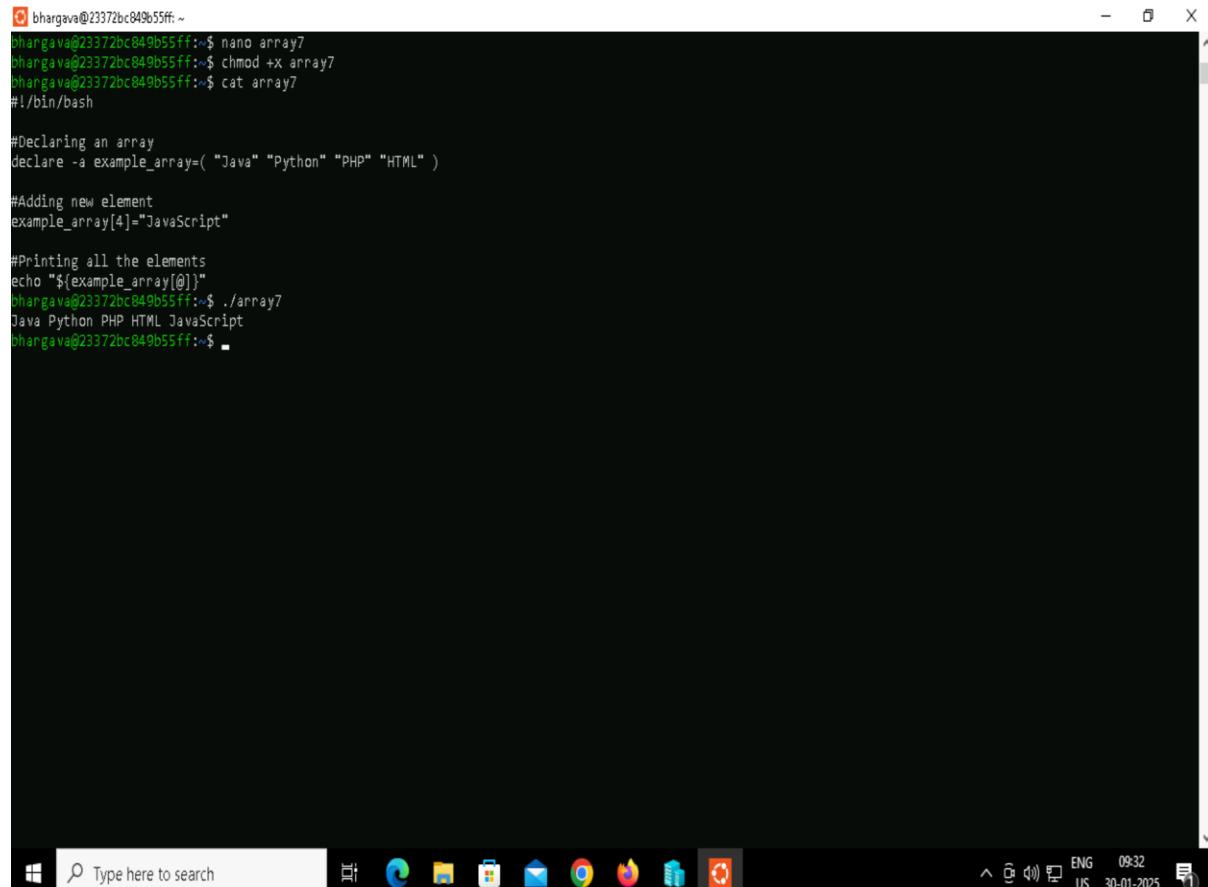
```
bhargava@23372bc849b55ff:~$ nano array6
bhargava@23372bc849b55ff:~$ chmod +x array6
bhargava@23372bc849b55ff:~$ cat array6
#!/bin/bash
#Script to loop through an array in C-style

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

#Length of the Array
length=${#example_array[@]}

#Array Loop
for (( i=0; i < ${length}; i++ ))
do
echo ${example_array[$i]}
done
bhargava@23372bc849b55ff:~$ ./array6
0 Welcome
1 To
2 Javatpoint
bhargava@23372bc849b55ff:~$
```

Bash Array Example 7



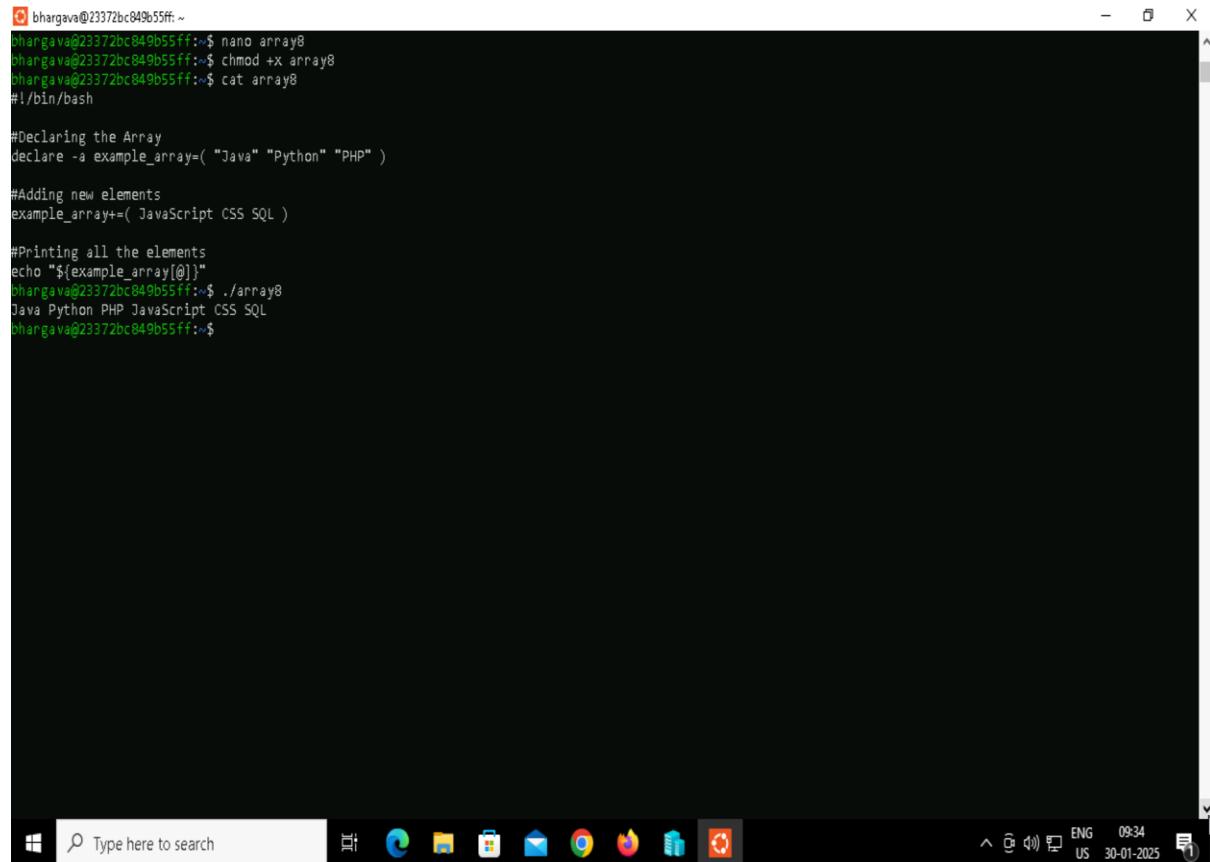
```
bhargava@23372bc849b55ff:~$ nano array7
bhargava@23372bc849b55ff:~$ chmod +x array7
bhargava@23372bc849b55ff:~$ cat array7
#!/bin/bash

#Declaring an array
declare -a example_array=( "Java" "Python" "PHP" "HTML" )

#Adding new element
example_array[4]="JavaScript"

#Printing all the elements
echo "${example_array[@]}"
bhargava@23372bc849b55ff:~$ ./array7
Java Python PHP HTML JavaScript
bhargava@23372bc849b55ff:~$
```

Bash Array Example 8



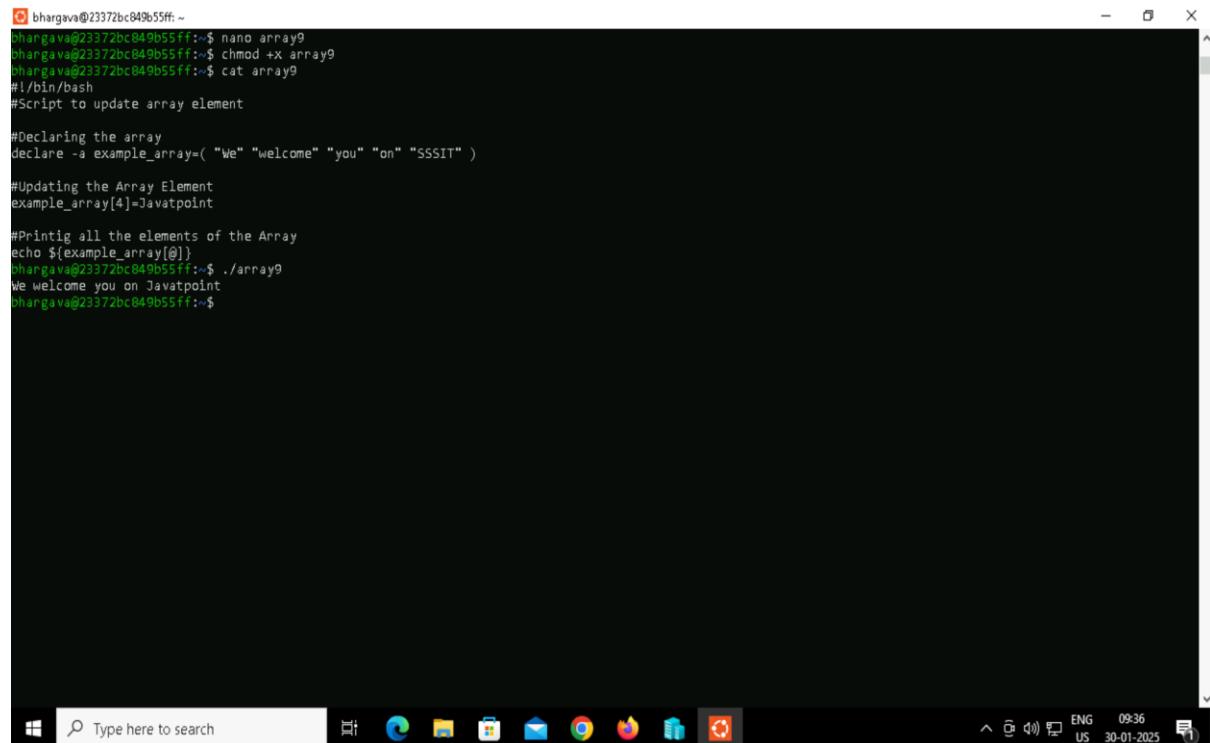
```
bhargava@23372bc849b55ff:~$ nano array8
bhargava@23372bc849b55ff:~$ chmod +x array8
bhargava@23372bc849b55ff:~$ cat array8
#!/bin/bash

#Declaring the Array
declare -a example_array=( "Java" "Python" "PHP" )

#Adding new elements
example_array+=( JavaScript CSS SQL )

#Printing all the elements
echo "${example_array[@]}"
bhargava@23372bc849b55ff:~$ ./array8
Java Python PHP JavaScript CSS SQL
bhargava@23372bc849b55ff:~$
```

Bash Array Example 9



```
bhargava@23372bc849b55ff:~$ nano array9
bhargava@23372bc849b55ff:~$ chmod +x array9
bhargava@23372bc849b55ff:~$ cat array9
#!/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

#Printig all the elements of the Array
echo ${example_array[@]}
bhargava@23372bc849b55ff:~$ ./array9
We welcome you on Javatpoint
bhargava@23372bc849b55ff:~$
```

Bash Array Example 10

```
bhargava@23372bc849b55ff:~$ nano array10
bhargava@23372bc849b55ff:~$ chmod +x array10
bhargava@23372bc849b55ff:~$ cat array10
#!/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 "${example_array[@]}"
bhargava@23372bc849b55ff:~$ ./array10
Java HTML CSS JavaScript
bhargava@23372bc849b55ff:~$
```

Bash Array Example 11

```
Select bhargava@23372bc849b55ff:~$ nano array11
bhargava@23372bc849b55ff:~$ chmod +x array11
bhargava@23372bc849b55ff:~$ cat array11
#!/bin/bash
#Script to delete the entire Array

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

#Deleting Entire Array
unset example_array

#Printing the Array Elements
echo ${!example_array[@]}
bhargava@23372bc849b55ff:~$ ./array11
bhargava@23372bc849b55ff:~$
```

Bash Array Example 12

The screenshot shows a Windows desktop environment with a terminal window open. The terminal window has a black background and white text. It displays a Bash script named 'array12' being run. The script defines an array 'example_array' containing 'Java', 'Python', 'HTML', 'CSS', and 'JavaScript'. It then slices the array from index 1 to index 3, resulting in an array 'sliced_array' containing 'Python', 'HTML', and 'CSS'. Finally, it loops through each element of the sliced array and prints it. The terminal window is titled with the user's session ID. Below the terminal, the Windows taskbar is visible, showing the Start button, a search bar, pinned application icons (File Explorer, Edge, File History, Task View, Task Manager, and a recycle bin icon), and system status icons (Wi-Fi, battery, volume, and date/time).

```
bhargava@23372bc849b55ff:~$ nano array12
bhargava@23372bc849b55ff:~$ chmod +x array12
bhargava@23372bc849b55ff:~$ cat array12
#!/bin/bash
#Script to slice Array Element from index 1 to index 3

#Declaring the Array
example_array( "Java" "Python" "HTML" "CSS" "JavaScript" )

#Slicing the Array
sliced_array=("${example_array[@]:1:3}")

#Applying for loop to iterate over each element in Array
for i in "${sliced_array[@]}"
do
echo $i
done
bhargava@23372bc849b55ff:~$ ./array12
Python
HTML
CSS
bhargava@23372bc849b55ff:~$
```