

# **PROJECT**

## **BASH SCRIPTING PROGRAMS**

## BASH CASE

## **1. A simple scenario to demonstrate the use of the case statement**

CODE

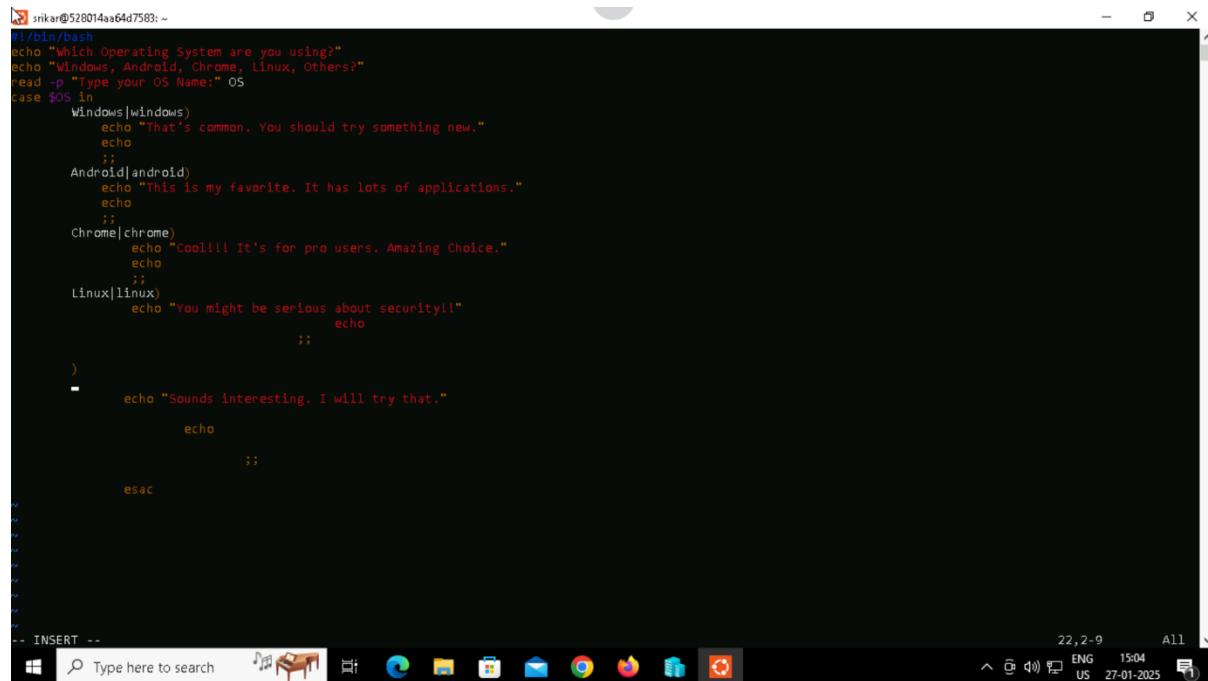
```
srikanth@528014aa64d7583: ~
#!/bin/bash
echo "Do you know Java Programming?"
read -p "Yes/No? :" Answer
case $Answer in
    Yes|yes|y|Y)
        echo "That's amazing."
        echo ;;
    No|no|N|n)
        echo "It's easy. Let's start learning from javatpoint."
        ;;
esac
```

## OUTPUT

```
srikan@528014aa64d7583:~  
srikan@528014aa64d7583:~$ vi f13.sh  
srikan@528014aa64d7583:~$ chmod +x f13.sh  
srikan@528014aa64d7583:~$ ./f13.sh  
Do you know Java Programming?  
Yes/No? :yes  
That's amazing.  
  
srikan@528014aa64d7583:~$ ./f13.sh  
Do you know Java Programming?  
Yes/No? :no  
It's easy. Let's start learning from javatpoint.  
srikan@528014aa64d7583:~$
```

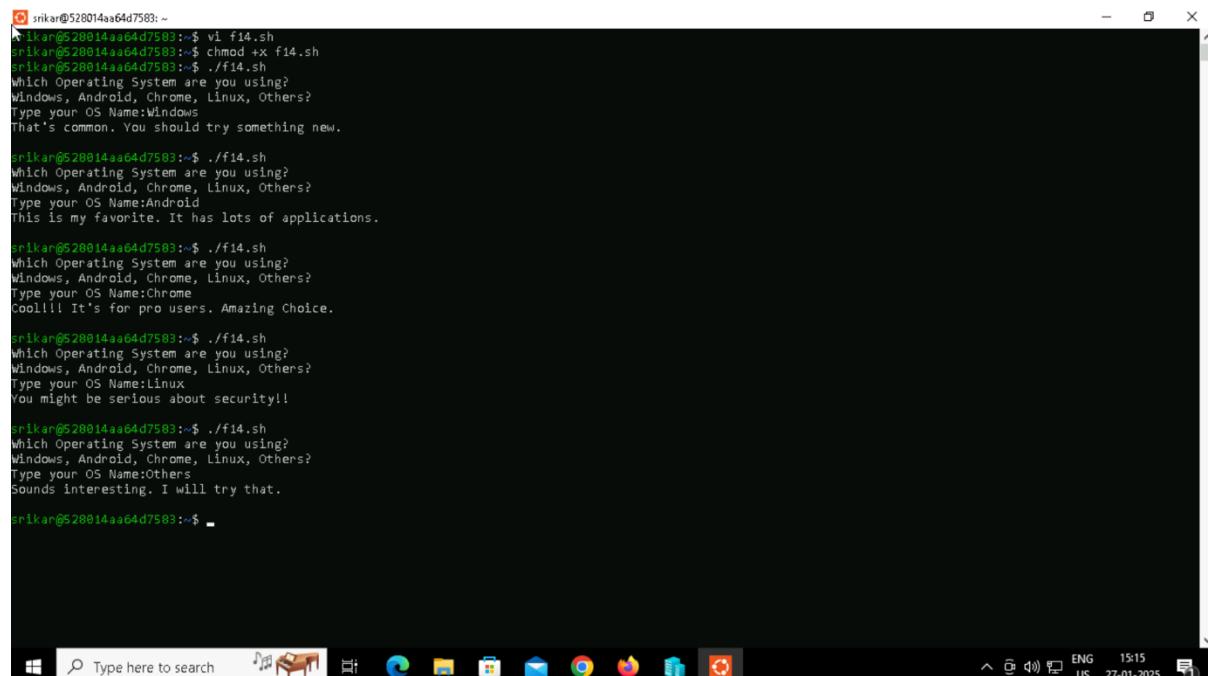
## 2. Define a one-line case statement with a default scenario for unmatched cases.

### CODE



```
#!/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
```

### OUTPUT



```
srikan@528014aa64d7583:~$ vi f14.sh
srikan@528014aa64d7583:~$ chmod +x f14.sh
srikan@528014aa64d7583:~$ ./f14.sh
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name:Windows
That's common. You should try something new.

srikan@528014aa64d7583:~$ ./f14.sh
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.

srikan@528014aa64d7583:~$ ./f14.sh
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name:Chrome
cool!! It's for pro users. Amazing Choice.

srikan@528014aa64d7583:~$ ./f14.sh
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name:Linux
You might be serious about security!!

srikan@528014aa64d7583:~$ ./f14.sh
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name:Others
sounds interesting. I will try that.

srikan@528014aa64d7583:~$
```

## BASH FOR LOOPS

## 1. Basic For Loop example

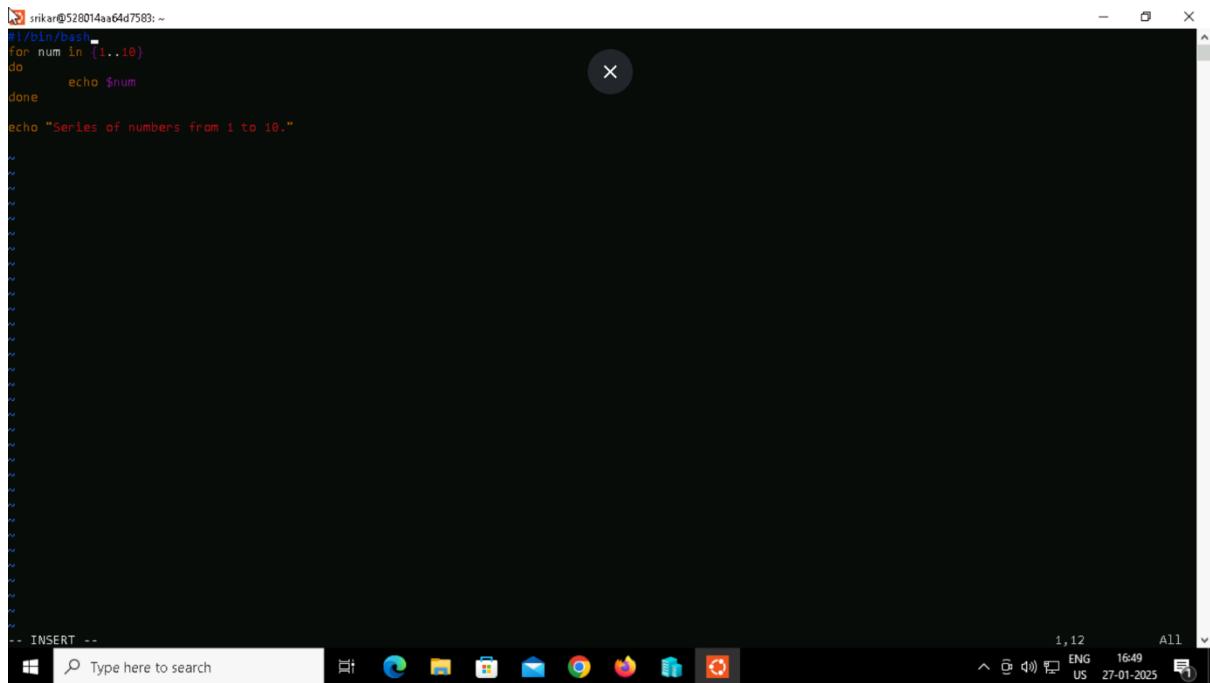
CODE

## OUTPUT

```
srikanth@528014aa64d7583:~  
srikanth@528014aa64d7583:~$ vi f15.sh  
srikanth@528014aa64d7583:~$ chmod +x f15.sh  
srikanth@528014aa64d7583:~$ ./f15.sh  
Start  
learning  
from  
Javaatpoint.  
Thank You.  
srikanth@528014aa64d7583:~$
```

## 2. For loop to read a range

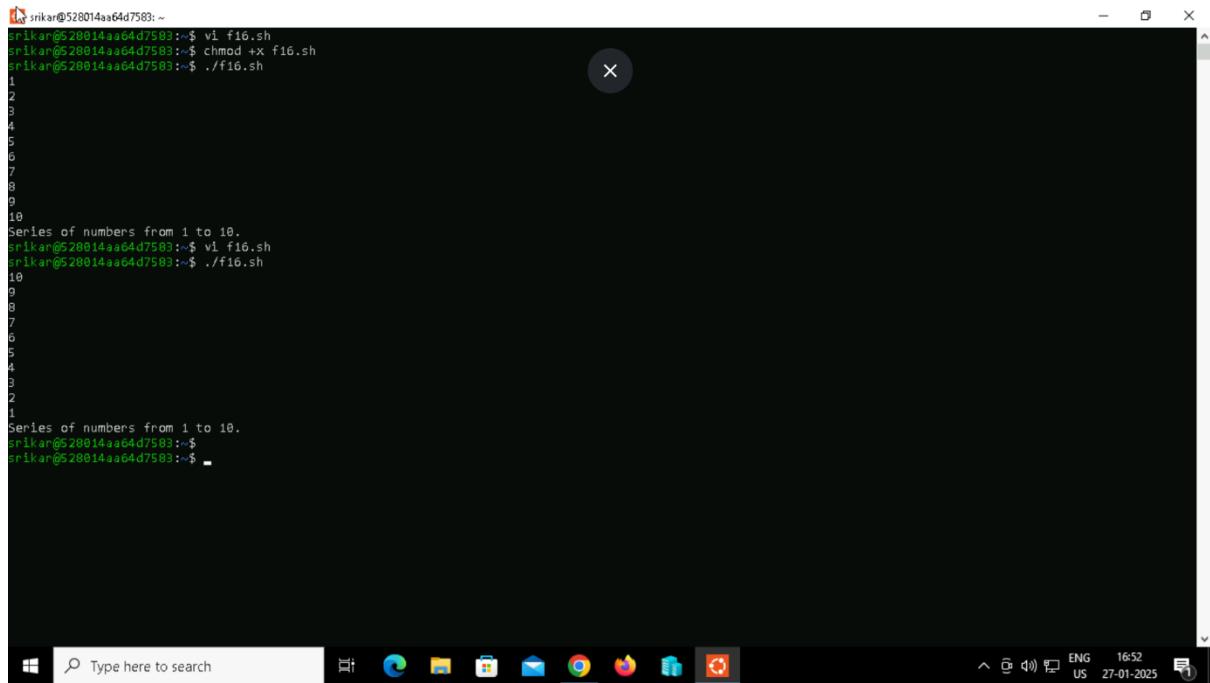
### CODE



```
srikan@528014aa64d7583: ~
#!/bin/bash
for num in {1..10}
do
    echo $num
done
echo "Series of numbers from 1 to 10."
-- INSERT --
```

The terminal window shows a shell script being run. The script uses a for loop to iterate from 1 to 10, printing each number on a new line. After the loop, it prints the message "Series of numbers from 1 to 10." The terminal interface includes a search bar at the bottom, a taskbar with various icons, and system status indicators like battery level and date/time.

### OUTPUT

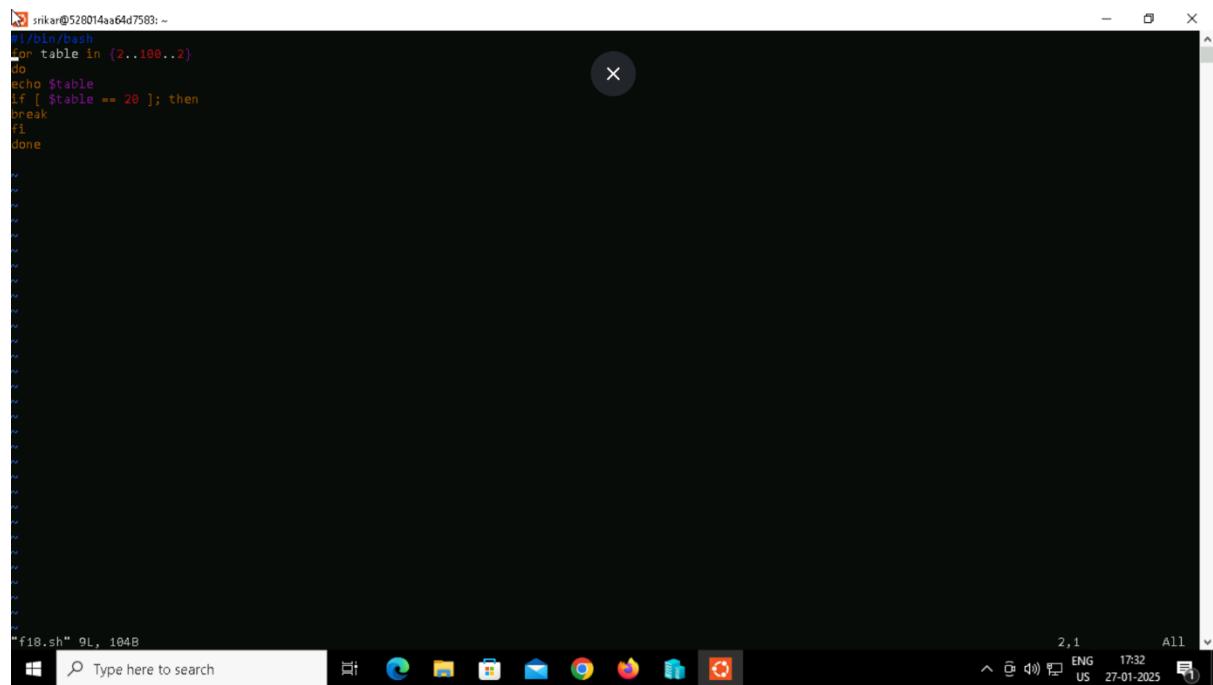


```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583:~$ vi f16.sh
srikan@528014aa64d7583:~$ chmod +x f16.sh
srikan@528014aa64d7583:~$ ./f16.sh
1
2
3
4
5
6
7
8
9
10
series of numbers from 1 to 10.
srikan@528014aa64d7583:~$ vi f16.sh
srikan@528014aa64d7583:~$ ./f16.sh
10
9
8
7
6
5
4
3
2
1
series of numbers from 1 to 10.
srikan@528014aa64d7583:~$ 
srikan@528014aa64d7583:~$
```

The terminal window shows the execution of the script. It first creates and chmods the file, then runs it twice. The output shows the numbers 1 through 10 on two separate lines, followed by the message "series of numbers from 1 to 10." The terminal interface is identical to the one in the code screenshot.

### 3. For Loop with a BREAK Statement

#### CODE

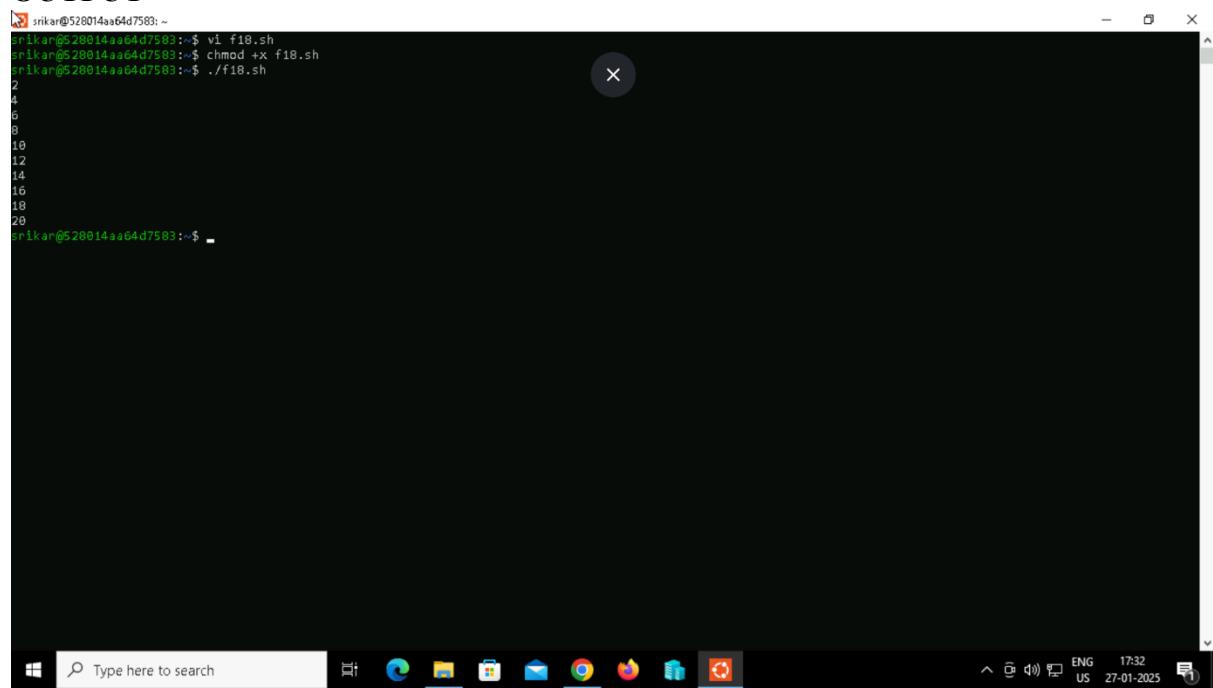


```
srikan@528014aa64d7583: ~
#!/bin/bash
for table in {2..100..2}
do
echo $table
if [ $table == 20 ]; then
break
fi
done

f18.sh" 9L, 104B
```

The terminal window shows a for loop that iterates over even numbers from 2 to 100 in increments of 2. It prints each value to the screen. When it reaches the value 20, it executes the 'break' command, which exits the loop. The terminal window has a dark background and light-colored text. The status bar at the bottom right shows the file name 'f18.sh', line count '9L', byte count '104B', and the current date and time '27-01-2025'. The taskbar at the bottom includes icons for File Explorer, Edge, File, Mail, Photos, and Task View.

#### OUTPUT



```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583: ~$ vi f18.sh
srikan@528014aa64d7583: ~$ chmod +x f18.sh
srikan@528014aa64d7583: ~$ ./f18.sh
2
4
6
8
10
12
14
16
18
20
```

The terminal window shows the execution of the script 'f18.sh'. It first edits the script with 'vi', changes its mode to executable with 'chmod +x', and then runs it with './'. The output is a sequence of even numbers starting from 2 up to 20, each on a new line. The terminal window has a dark background and light-colored text. The status bar at the bottom right shows the current date and time '27-01-2025'. The taskbar at the bottom includes icons for File Explorer, Edge, File, Mail, Photos, and Task View.

## 4. For Loop with a CONTINUE Statement

### CODE

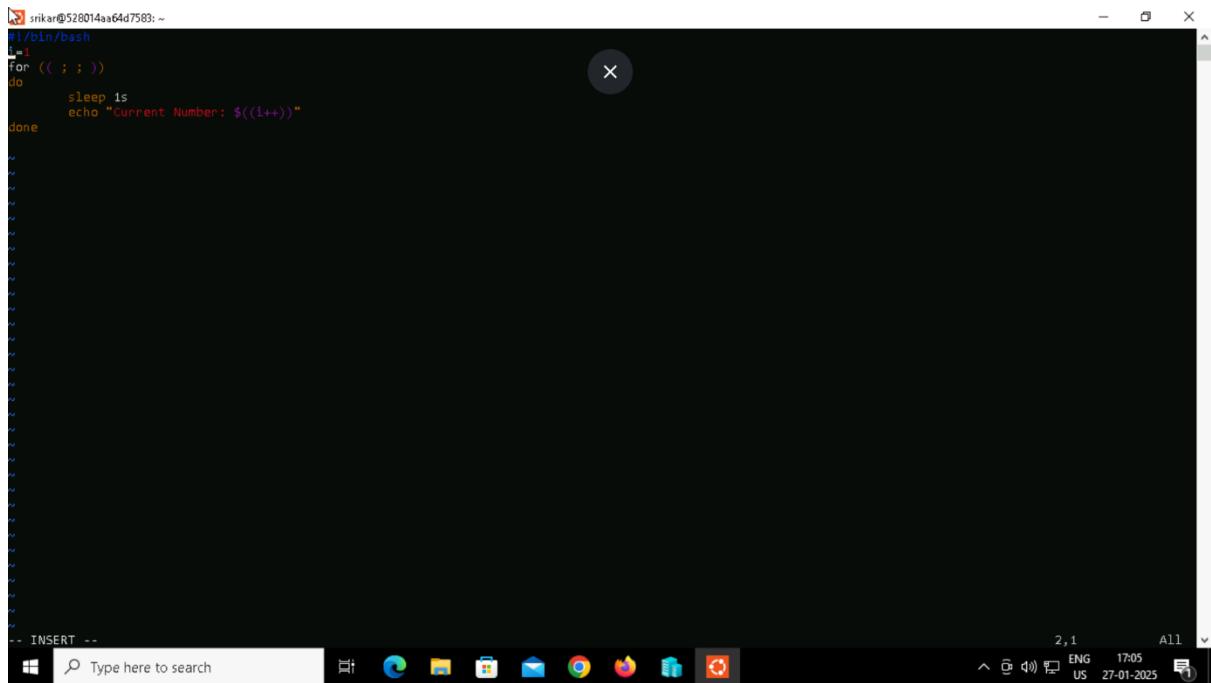
```
srikan@528014aa64d7583: ~
#!/bin/bash
for ((i=1; i<=20; i++));
do
    if [[ $i -gt 5 && $i -lt 16 ]]; then
        continue
    fi
    echo $i
done
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
-- INSERT --
1,12 ENG 17:10 All
^ Q! F11 US 27-01-2025
```

### OUTPUT

```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583:~$ vi f21.sh
srikan@528014aa64d7583:~$ chmod +x f21.sh
srikan@528014aa64d7583:~$ ./f21.sh
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
srikan@528014aa64d7583:~$
```

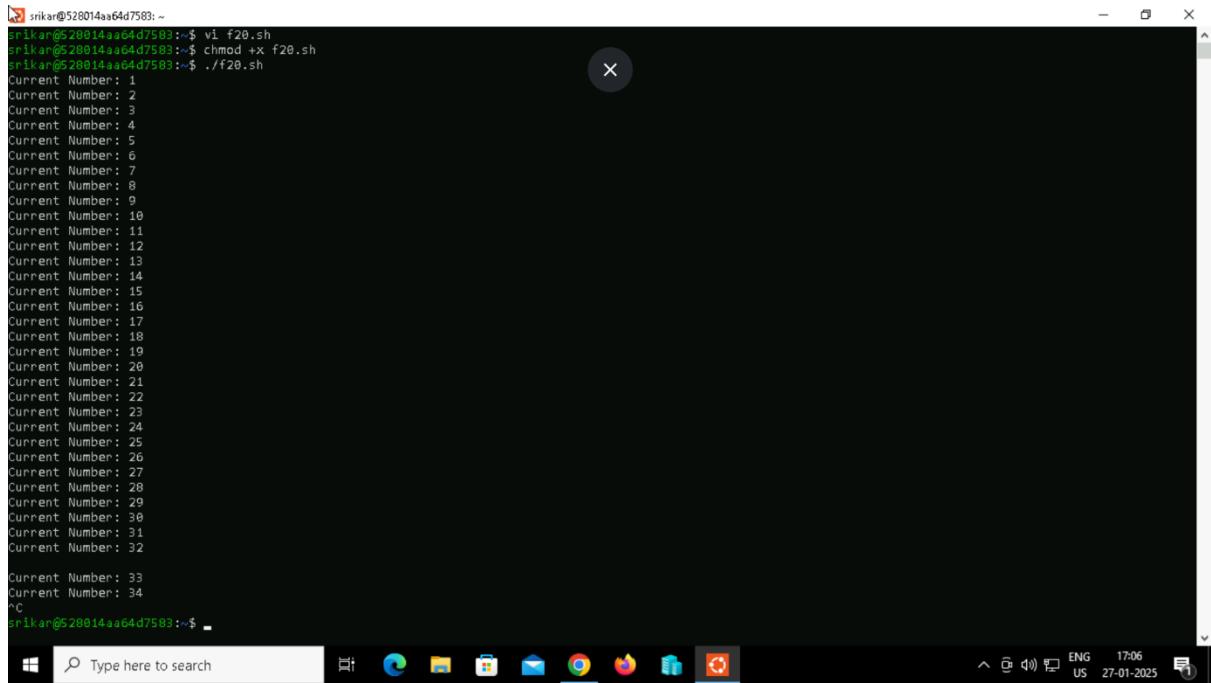
## 5. For Loop with an INFINITE Bash for Loop

### CODE



```
#!/bin/bash
i=1
for (( ; ; ))
do
    sleep 1s
    echo "Current Number: $((i++))"
done
```

### OUTPUT



```
srikan@528014aa64d7583:~$ vi f20.sh
srikan@528014aa64d7583:~$ chmod +x f20.sh
srikan@528014aa64d7583:~$ ./f20.sh
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
Current Number: 29
Current Number: 30
Current Number: 31
Current Number: 32

Current Number: 33
Current Number: 34
^C
srikan@528014aa64d7583:~$
```

## 6. For Loop to read Array variables

### CODE

```
srikanth@520014aa64d7583: ~
#!/bin/bash
arr=( "Welcome" "to" "Javatpoint" )
for i in "${arr[@]}"
do
    echo $i
done
```

The terminal window shows a for loop that iterates over an array named arr. The array contains three elements: "Welcome", "to", and "Javatpoint". The loop prints each element on a new line.

### OUTPUT

```
srikanth@520014aa64d7583: ~
srikanth@520014aa64d7583: ~$ vi f22.sh
srikanth@520014aa64d7583: ~$ chmod +x f22.sh
srikanth@520014aa64d7583: ~$ ./f22.sh
Welcome
to
Javatpoint
srikanth@520014aa64d7583: ~$
```

The terminal window shows the execution of the script f22.sh. The script contains the same for loop as the code above. The output is identical, printing the three elements of the array on separate lines.

## BASH WHILE LOOPS

## 1. While Loop with single condition

CODE

```
srikan@528014aa64d7583: ~
#!/bin/bash

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."
~
```

-- INSERT --

## OUTPUT

```
srikan@528014aa64d7583:~$ vi f23.sh
srikan@528014aa64d7583:~$ chmod +x f23.sh
srikan@528014aa64d7583:~$ ./f23.sh
Enter starting number: 5
Enter ending number: 30
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
This is the sequence that you wanted.
srikan@528014aa64d7583:~$
```

## **2. While Loop with Multiple condition**

CODE

```
srikanth@528014aa64d7583: ~
#!/bin/bash

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."

```

## OUTPUT

```
srikan@528014aa64d7583:~  
srikan@528014aa64d7583:~$ vi f24.sh  
srikan@528014aa64d7583:~$ chmod +x f24.sh  
srikan@528014aa64d7583:~$ ./f24.sh  
Enter starting number: 10  
Enter ending number: 30  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
This is the sequence that you wanted.  
srikan@528014aa64d7583:~$
```

### 3. INFINITE While Loop

CODE

A screenshot of a Windows desktop environment. At the top, there's a dark terminal window titled 'srikanth@528014aa64d7583: ~'. The window contains a shell script with the following code:

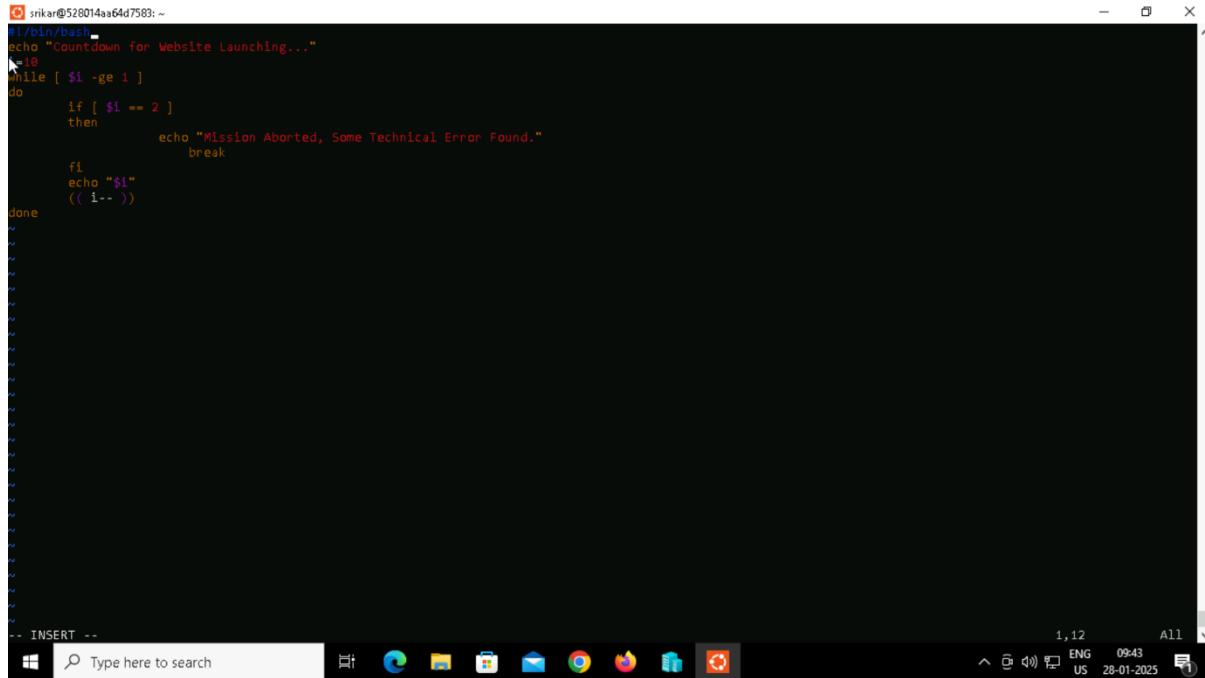
```
#!/bin/bash
while :
do
    echo "Welcome to UST Global."
done
```

The terminal has a small circular close button in the top right corner. Below the terminal is a standard Windows taskbar. On the taskbar, from left to right, are icons for File Explorer, Microsoft Edge, Mail, Photos, Google Chrome, Mozilla Firefox, File Explorer again, and a system tray icon. To the right of the taskbar, the system tray shows the date and time as '28-01-2025 09:42' and includes icons for battery level, signal strength, and language settings. The bottom of the screen features a dark footer bar with the text 'INSERT --' on the left and '5,5 All' on the right.

## OUTPUT

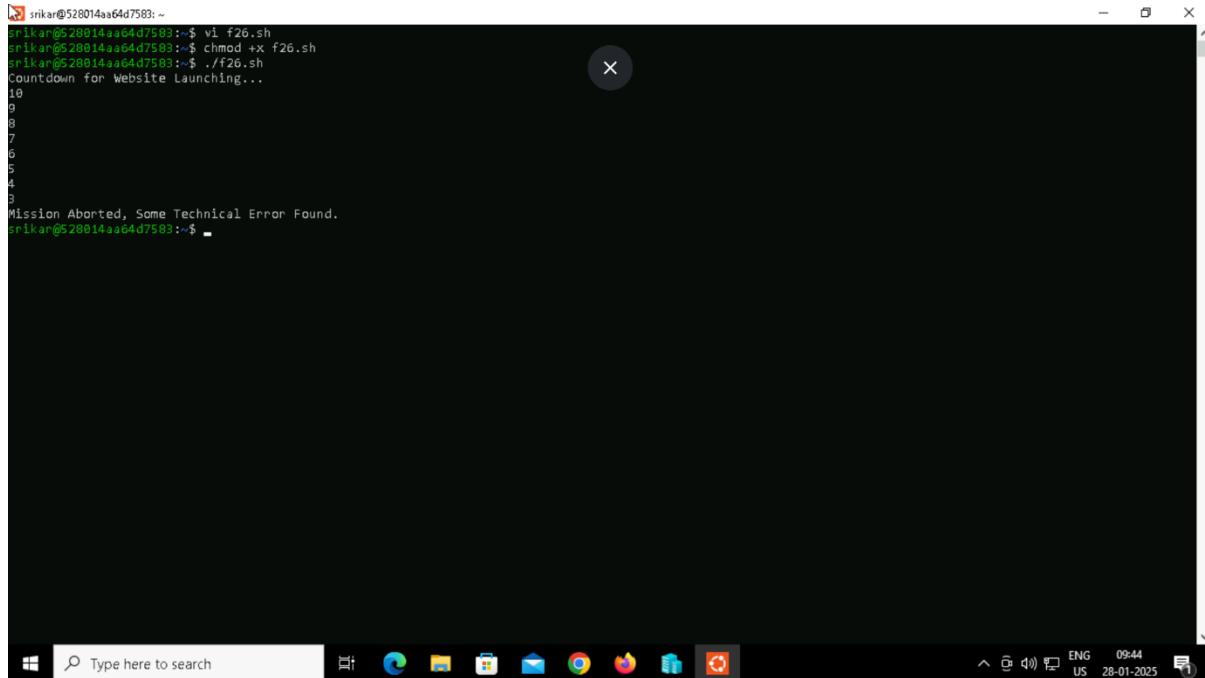
## 4. While Loop using BREAK statement

### CODE



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

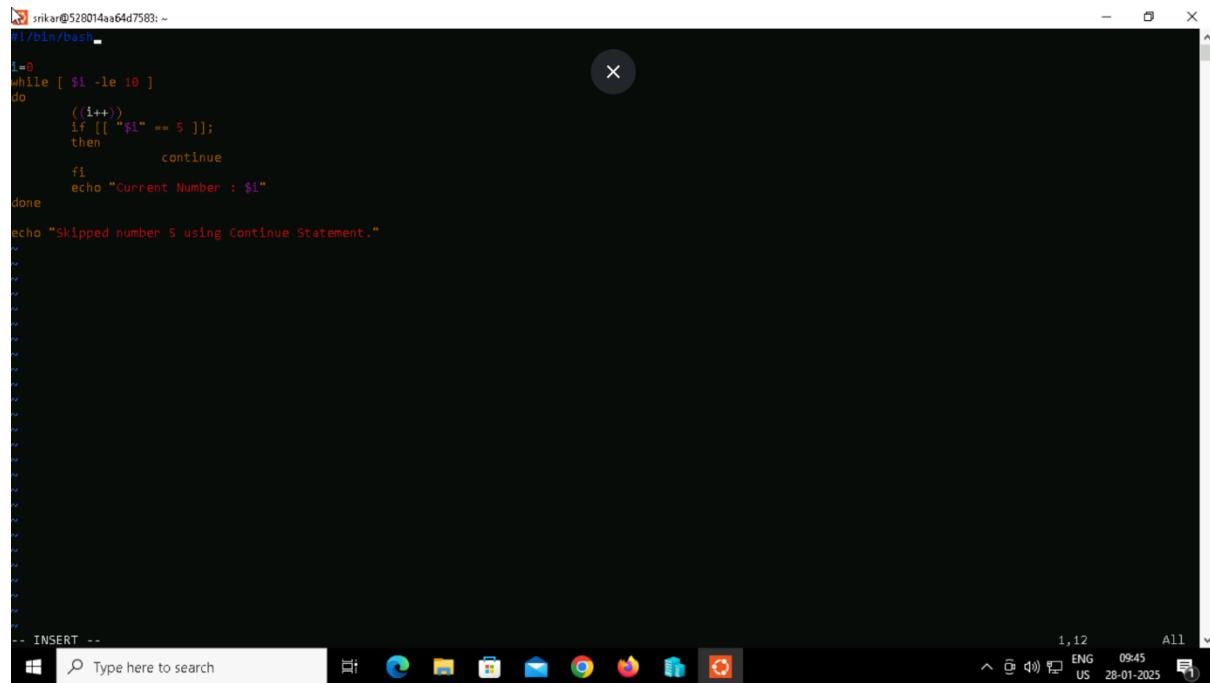
### OUTPUT



```
srikan@528014aa64d7583:~$ vi f26.sh
srikan@528014aa64d7583:~$ chmod +x f26.sh
srikan@528014aa64d7583:~$ ./f26.sh
Countdown for Website Launching...
10
9
8
7
6
5
4
3
Mission Aborted, Some Technical Error Found.
srikan@528014aa64d7583:~$
```

## 5. While Loop using CONTINUE statement

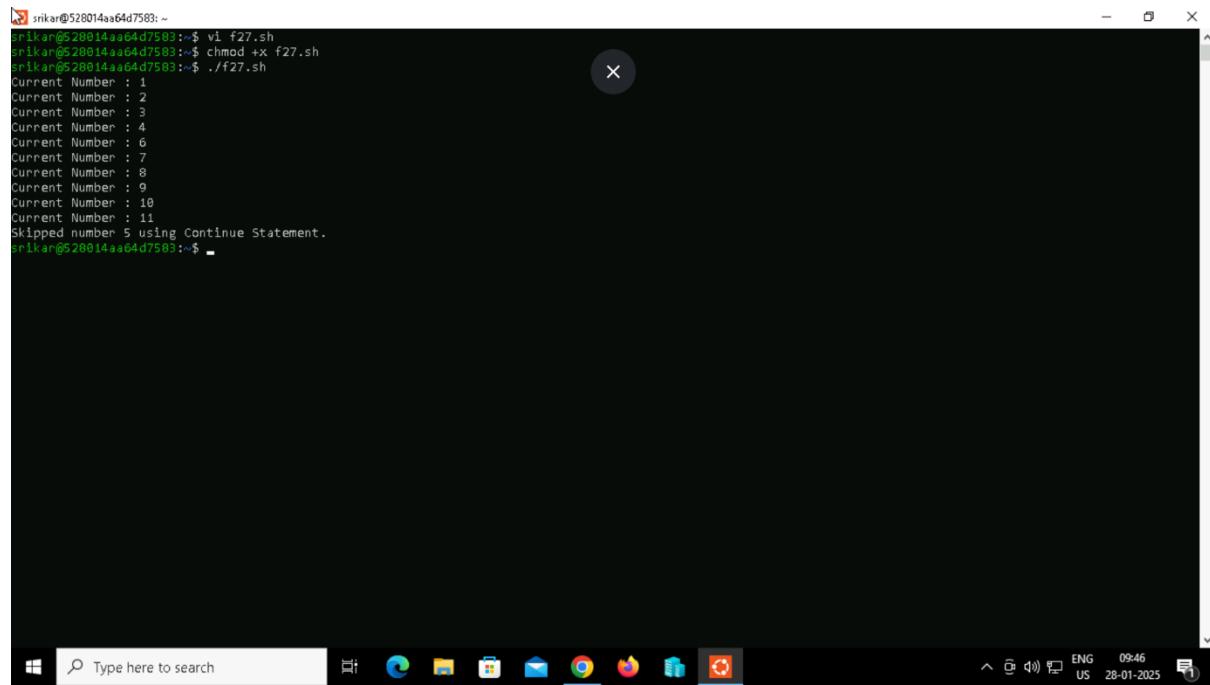
### CODE



```
srikanth@528014aa64d7583: ~
#!/bin/bash

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."
~
```

### OUTPUT



```
srikanth@528014aa64d7583: ~
srikanth@528014aa64d7583: ~$ vi f27.sh
srikanth@528014aa64d7583: ~$ chmod +x f27.sh
srikanth@528014aa64d7583: ~$ ./f27.sh
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
Skipped number 5 using Continue Statement.
srikanth@528014aa64d7583: ~$
```

## BASH UNTIL LOOP

## 1. Bash UNTIL loop using single condition

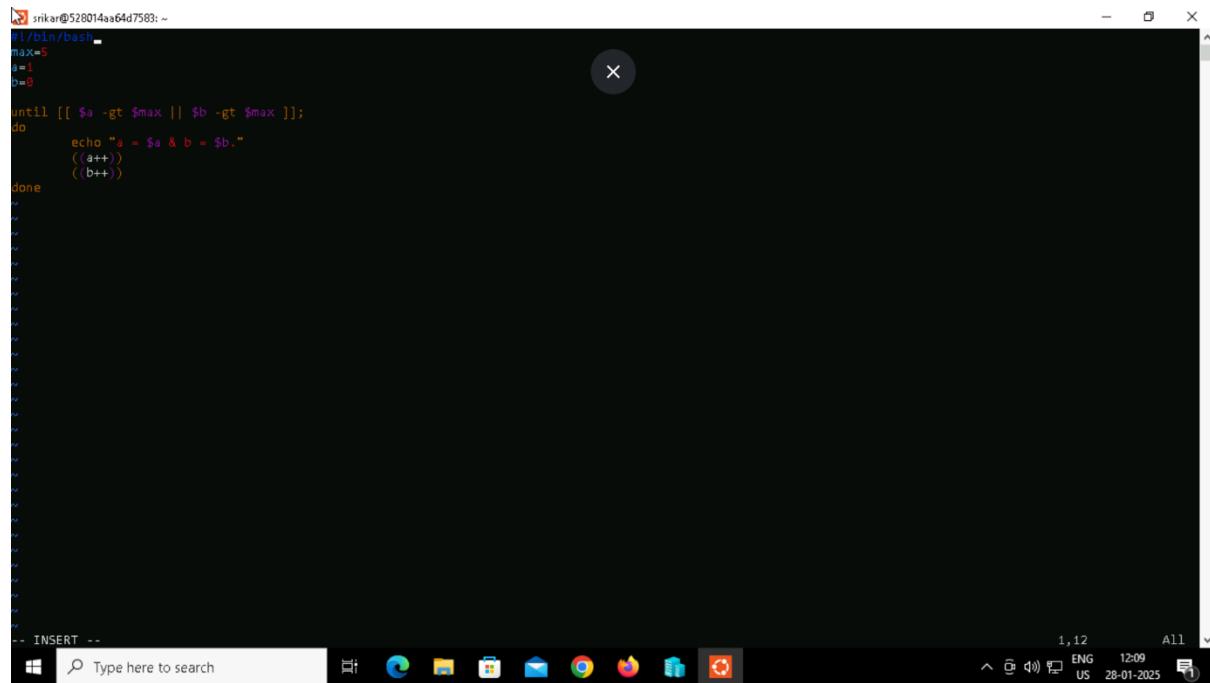
CODE

## OUTPUT

```
srikan@528014aa64d7583:~$ vi f28.sh
srikan@528014aa64d7583:~$ chmod +x f28.sh
srikan@528014aa64d7583:~$ ./f28.sh
./f28.sh: line 1: !/bin/bash: No such file or directory
1
2
3
4
5
6
7
8
9
10
srikan@528014aa64d7583:~$ -
```

## 2. Bash UNTIL loop using multiple condition

### CODE

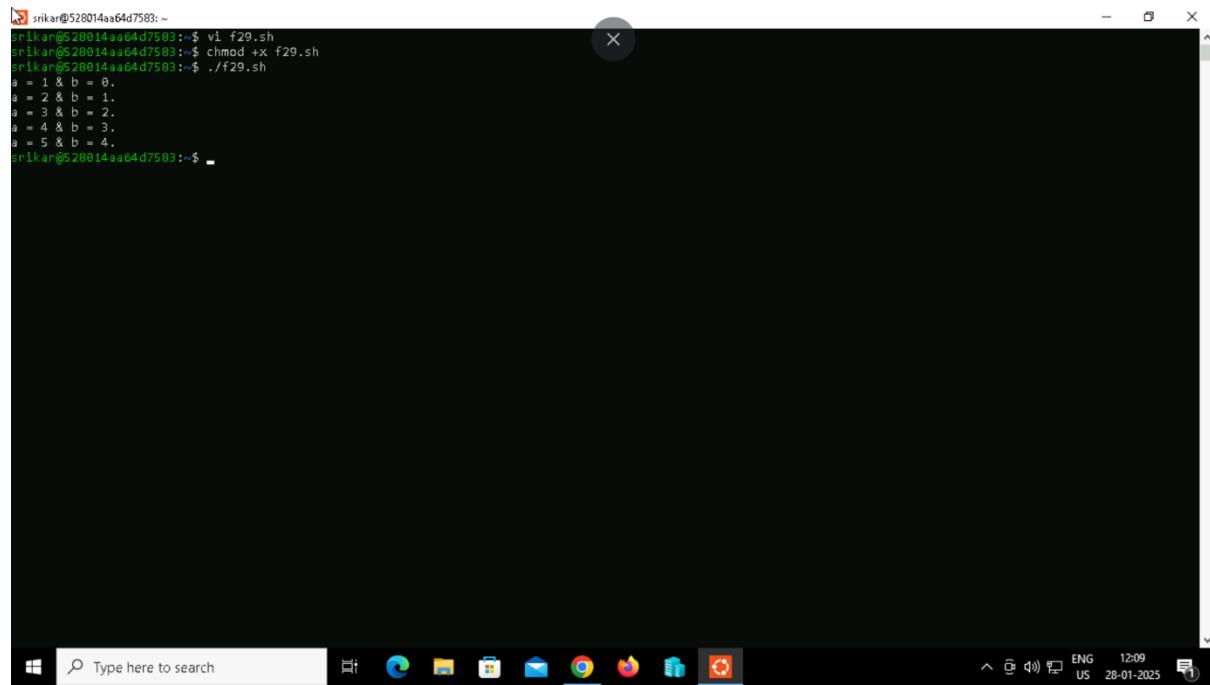


```
srikan@528014aa64d7583: ~
#!/bin/bash
max=5
a=1
b=0

until [[ $a -gt $max || $b -gt $max ]];
do
    echo "a = $a & b = $b."
    ((a++))
    ((b++))
done
```
... (approximately 100 lines of identical output)
```
a = 5 & b = 4.

```

### OUTPUT



```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583: ~$ vi f29.sh
srikan@528014aa64d7583: ~$ chmod +x f29.sh
srikan@528014aa64d7583: ~$ ./f29.sh
a = 1 & b = 0.
a = 2 & b = 1.
a = 3 & b = 2.
a = 4 & b = 3.
a = 5 & b = 4.
srikan@528014aa64d7583: ~$
```

## BASH STRING

### 1. To check whether two strings are equal or not

#### CODE

```
srikan@528014aa64d7583: ~
$ cd /bin/bash
$ str1="WelcometoJavatpoint."
$ str2="javatpoint"
$ if [ $str1 = $str2 ];
then
    echo "Both the strings are equal."
else
    echo "Strings are not equal."
fi
$
```

-- INSERT --

Type here to search

1,12 All

ENG 14:27

US 28-01-2025

#### OUTPUT

```
srikan@528014aa64d7583: ~$ vi f30.sh
srikan@528014aa64d7583: ~$ chmod +x f30.sh
srikan@528014aa64d7583: ~$ ./f30.sh
Strings are not equal.
srikan@528014aa64d7583: ~$ vi f30.sh
srikan@528014aa64d7583: ~$ ./f30.sh
Both the strings are equal.
srikan@528014aa64d7583: ~$
```

Type here to search

14:29

ENG

US 28-01-2025

## 2. To check whether two strings are greater or less

### CODE

```
srikan@528014aa64d7583: ~
#!/bin/bash
str1="WelcometoJavatpoint"
str2="Javatpoint"
if [ $str1 < $str2 ];
then
    echo "$str1 is less than $str2"
else
    echo "$str1 is not less than $str2"
fi
-- INSERT --
```

The terminal window shows a command-line interface with a black background and white text. It displays a bash script that compares two strings, `str1` and `str2`. The script uses an if-then-else construct to output the result. Below the terminal window is a taskbar with various application icons and system status indicators.

### OUTPUT

```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583:~$ vi f31.sh
srikan@528014aa64d7583:~$ chmod +x f31.sh
srikan@528014aa64d7583:~$ ./f31.sh
WelcometoJavatpoint is not less than Javatpoint
srikan@528014aa64d7583:~$ vi f31.sh
srikan@528014aa64d7583:~$ ./f31.sh
WelcometoJavatpoint is greater than Javatpoint
srikan@528014aa64d7583:~$
```

The terminal window shows the execution of the bash script `f31.sh`. It first creates the file with `vi`, changes its mode to executable with `chmod +x`, and then runs it with `./f31.sh`. The output shows two different comparisons: one where `str1` is less than `str2` and one where `str1` is greater than `str2`. Below the terminal window is a taskbar with various application icons and system status indicators.

### 3. To check whether given string is empty or not

#### CODE

```
srikan@528014aa64d7583: ~
#!/bin/bash
str="WelcometoJavatpoint"
if [ -n $str ];
then
    echo "String is not empty"
else
    echo "String is empty"
fi
-- INSERT --
Type here to search 1,12 ENG 14:40 All
Activate Windows Go to Settings to activate Windows.
28-01-2025 US
```

#### OUTPUT

```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583: ~$ vi f32.sh
srikan@528014aa64d7583: ~$ chmod +x f32.sh
srikan@528014aa64d7583: ~$ ./f32.sh
String is not empty
srikan@528014aa64d7583: ~$ vi f32.sh
srikan@528014aa64d7583: ~$ ./f32.sh
String is empty
srikan@528014aa64d7583: ~$ vi f32.sh
srikan@528014aa64d7583: ~$ ./f32.sh
String is empty
srikan@528014aa64d7583: ~$ -
-- INSERT --
Type here to search 1,12 ENG 14:42 All
Activate Windows Go to Settings to activate Windows.
28-01-2025 US
```

## BASH FIND STRING

### 1. To find length of the string using “expr”

#### CODE

```
srikanth@528014aa64d7583: ~
~/bin/bash
str="Welcome to Javatpoint"
length=`expr length "$str"`
echo "Length of '$str' is $length"
```
The screenshot shows a terminal window on a Windows desktop. The terminal has a dark background with white text. It displays a Bash script that 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 title bar, a scroll bar on the right, and a taskbar at the bottom with various icons like File Explorer, Edge, and File Manager.
```

#### OUTPUT

```
srikanth@528014aa64d7583: ~$ vi f33.sh
srikanth@528014aa64d7583: ~$ chmod +x f33.sh
srikanth@528014aa64d7583: ~$ ./f33.sh
Length of 'Welcome to Javatpoint' is 21
srikanth@528014aa64d7583: ~$
```

The screenshot shows the same terminal window after running the script. The output is displayed in white text on the black background. It shows the command to edit the file, change its mode to executable, run the script, and finally the output of the script which is 'Length of 'Welcome to Javatpoint' is 21'. The terminal window and taskbar are identical to the previous screenshot.

## 2. To find length of the string using “wc” command

### CODE

```
srikan@528014aa64d7583: ~
#!/bin/bash
str="Welcome to Javatpoint"
length=`echo $str | wc -c`
echo "Length of '$str' is $length"
```
The terminal window shows a script being run. It defines a variable str with the value "Welcome to Javatpoint". It then uses the wc command with the -c option to count the characters in str and stores the result in the variable length. Finally, it prints the message "Length of '$str' is $length". The terminal window has a dark background and light-colored text. At the bottom, there is a taskbar with various icons and a search bar.
```

### OUTPUT

```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583:~$ vi f44.sh
srikan@528014aa64d7583:~$ vi f34.sh
srikan@528014aa64d7583:~$ chmod +x f34.sh
srikan@528014aa64d7583:~$ ./f34.sh
Length of 'Welcome to Javatpoint' is 22
srikan@528014aa64d7583:~$
```

The terminal window shows the execution of the script f34.sh. It first edits the file f44.sh with vi, then edits f34.sh with vi, changes its mode to executable with chmod, and finally runs it with ./f34.sh. The output of the script is displayed: "Length of 'Welcome to Javatpoint' is 22". The terminal window has a dark background and light-colored text. At the bottom, there is a taskbar with various icons and a search bar.

### 3. To find length of the string using “awk” command

#### CODE

```
srikan@528014aa64d7583: ~
#!/bin/bash
str="Welcome to Javatpoint"
length=`echo $str | awk '{print length}'`
echo "Length of '$str' is $length"
```
The screenshot shows a terminal window on a Windows operating system. The command `#!/bin/bash` is run, followed by defining a variable `str` with the value "Welcome to Javatpoint". Then, the command `length=`echo $str | awk '{print length}'` is run to calculate the length of the string. Finally, the output `Length of 'Welcome to Javatpoint' is 21` is displayed. The terminal window has a dark background and light-colored text. The taskbar at the bottom shows various application icons.
```

#### OUTPUT

```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583: ~$ vi f35.sh
srikan@528014aa64d7583: ~$ chmod +x f35.sh
srikan@528014aa64d7583: ~$ ./f35.sh
Length of 'Welcome to Javatpoint' is 21
srikan@528014aa64d7583: ~$
```

Activate Windows  
Go to Settings to activate Windows.

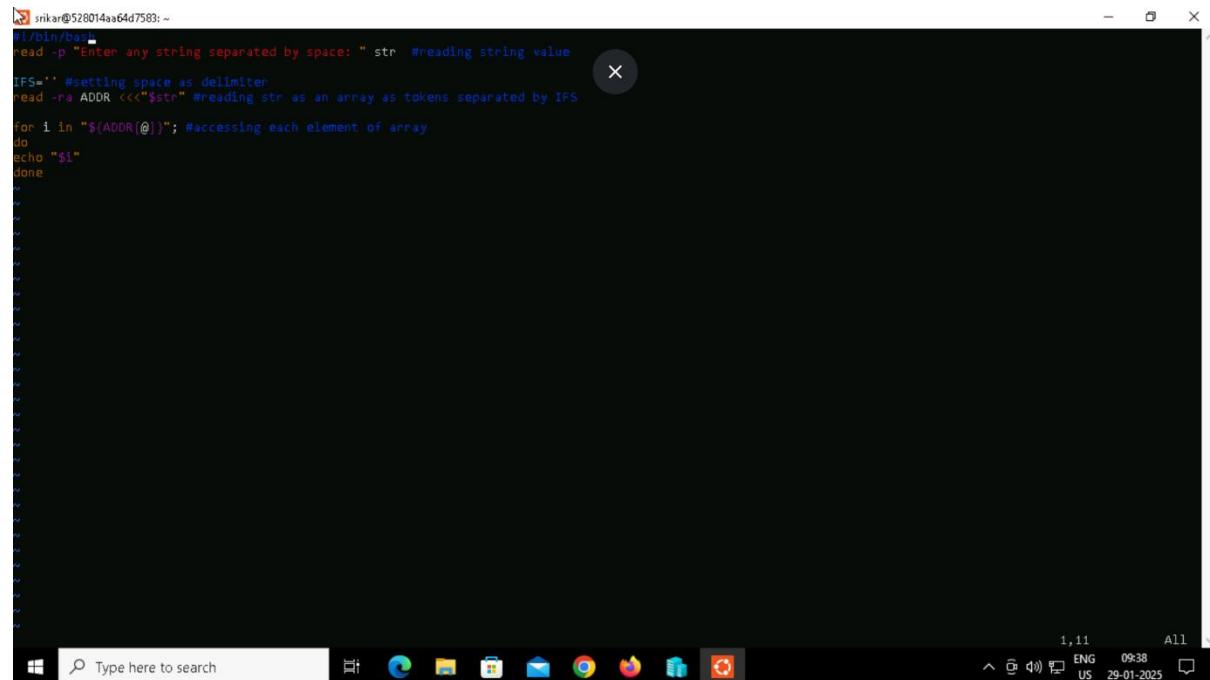
1,12 All  
14:57  
ENG US 28-01-2025

The screenshot shows a terminal window on a Windows operating system. It displays the execution of a shell script `f35.sh` which uses the `awk` command to find the length of the string "Welcome to Javatpoint". The output "Length of 'Welcome to Javatpoint' is 21" is shown. The terminal window has a dark background and light-colored text. The taskbar at the bottom shows various application icons. A watermark for "Activate Windows" is visible in the bottom right corner of the screen.

## BASH SPLIT STRING

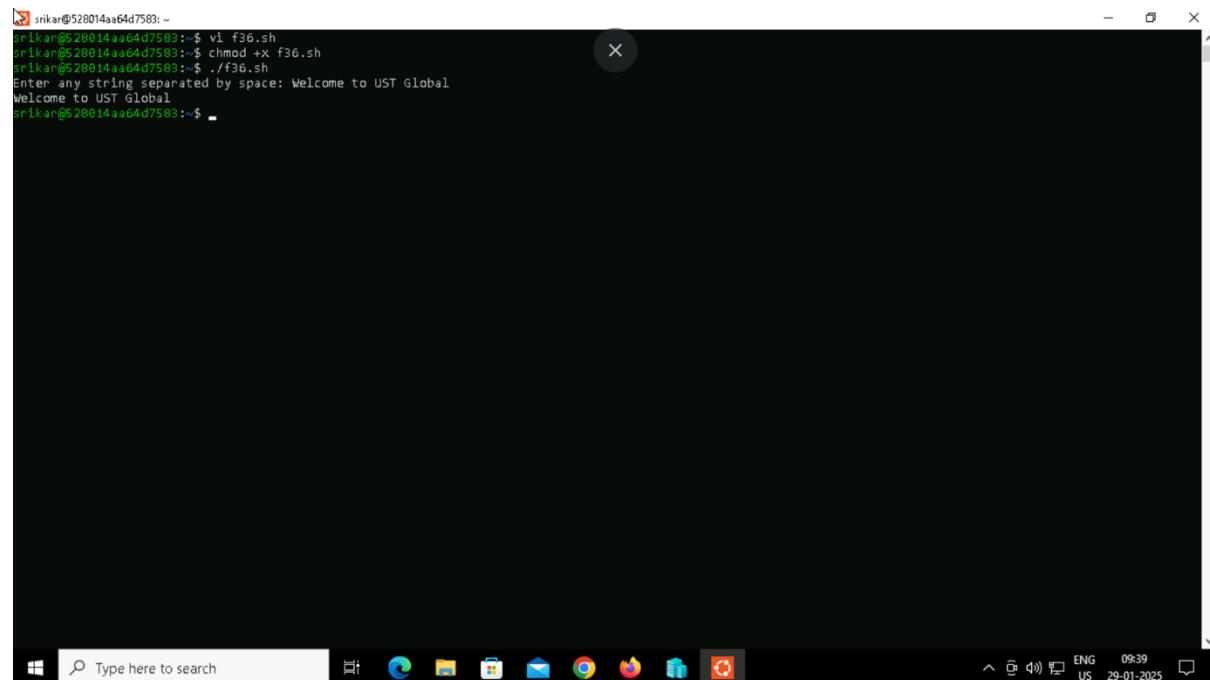
### 1. Bash split string by SPACE

#### CODE



```
srikan@528014aa64d7583: ~
$ cd bin/bash
$ 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
$
```

#### OUTPUT



```
srikan@528014aa64d7583: ~
$ vi f36.sh
$ chmod +x f36.sh
$ ./f36.sh
Enter any string separated by space: Welcome to UST Global
Welcome to UST Global
$
```

## 2. Bash split string by SYMBOL

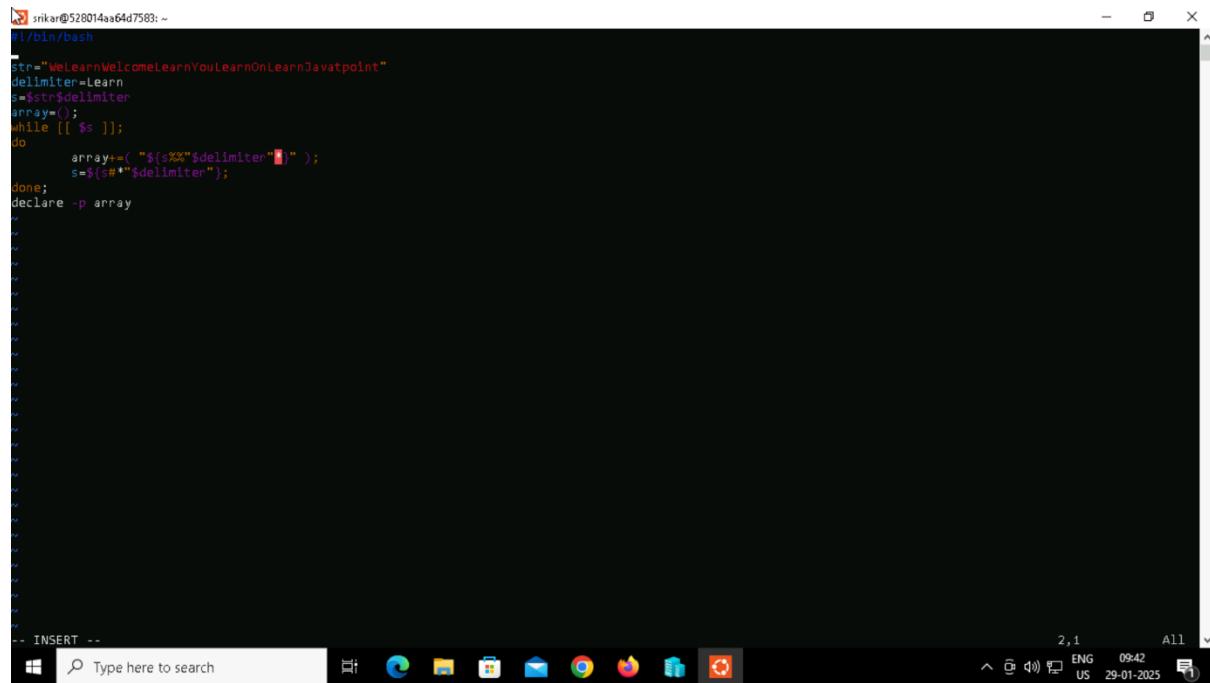
## CODE

## OUTPUT

```
srikan@528014aa64d7583:~  
srikan@528014aa64d7583:~$ vi f37.sh  
srikan@528014aa64d7583:~$ chmod +x f37.sh  
srikan@528014aa64d7583:~$ ./f37.sh  
Enter Name, State and Age separated by a comma: Srikar,Telangana,22  
Name : Srikar  
State : Telangana  
Age : 22  
srikan@528014aa64d7583:~$
```

### 3. Bash split string by ANOTHER string

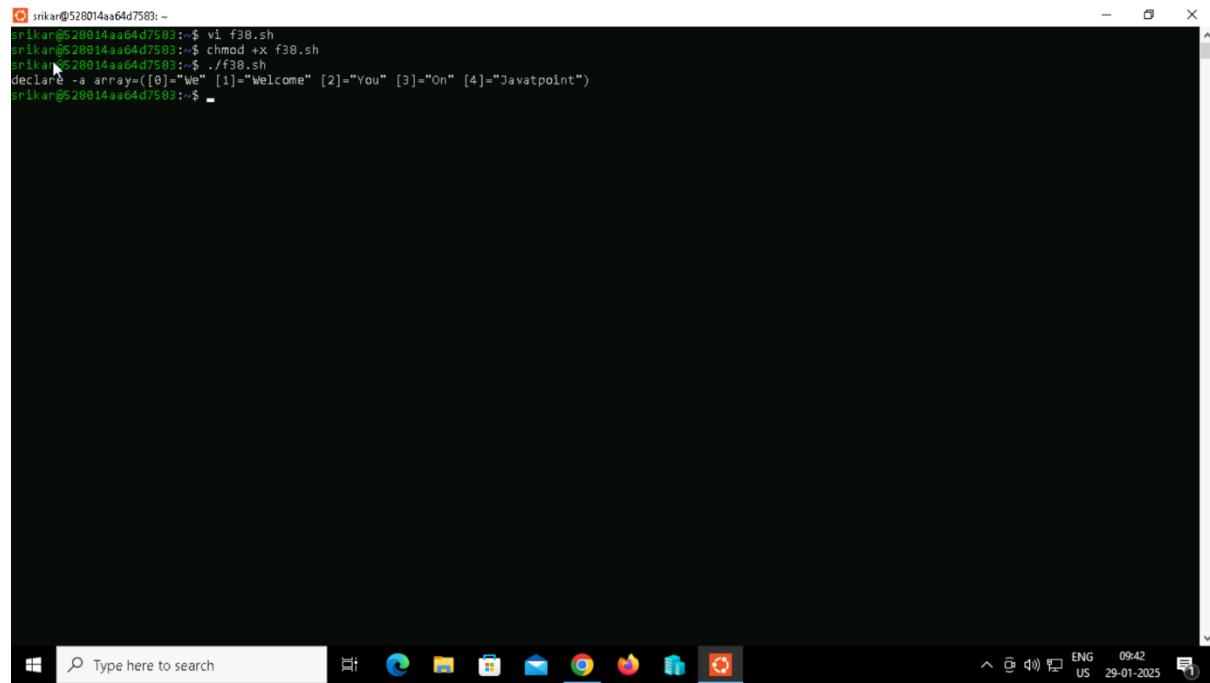
#### CODE



```
srikan@528014aa64d7583: ~
#!/bin/bash

str="WeLearnWelcomeLearnYouLearnOnLearnJavaatpoint"
delimiter=Learn
s=$str$delimiter
array=();
while [[ $s ]];
do
    array+=(" ${s%%$delimiter}" );
    s=${s%"$delimiter"};
done;
declare -p array
-- INSERT --
2,1 ENG 09:42 All
^ D) US 29-01-2025
```

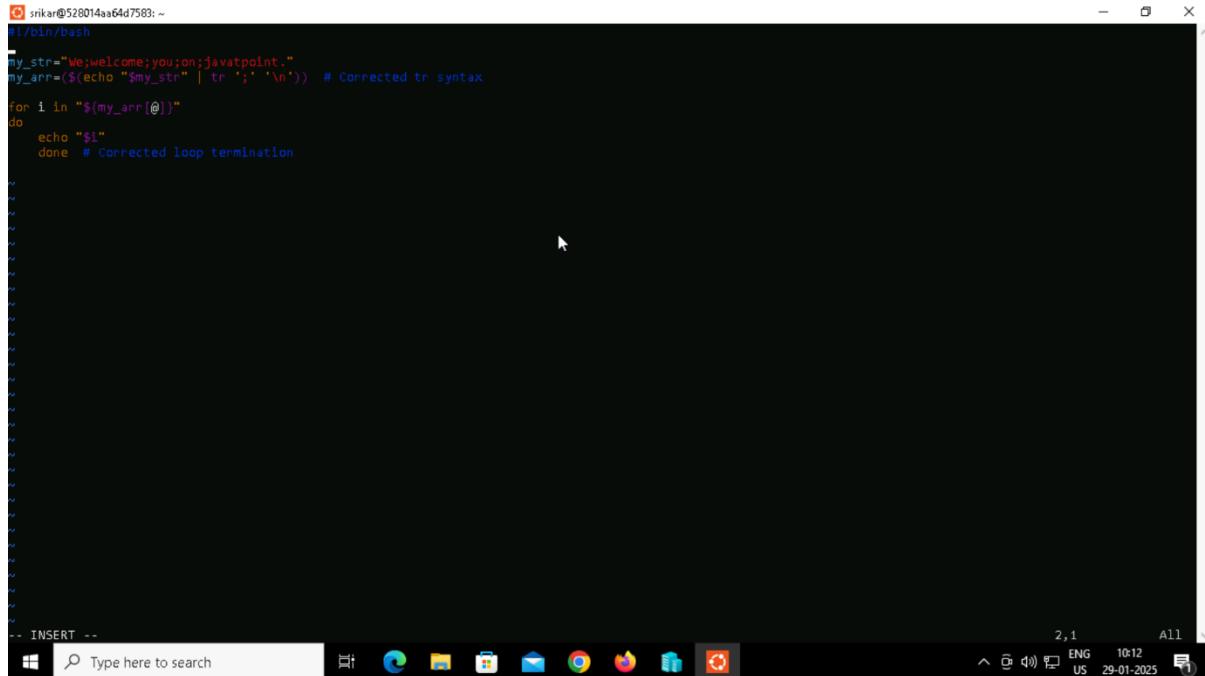
#### OUTPUT



```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583:~$ vi f38.sh
srikan@528014aa64d7583:~$ chmod +x f38.sh
srikan@528014aa64d7583:~$ ./f38.sh
declare -a array=([0]="We" [1]="Welcome" [2]="You" [3]="On" [4]="Javaatpoint")
srikan@528014aa64d7583:~$
```

## 4. Bash split string by TRIM command

### CODE



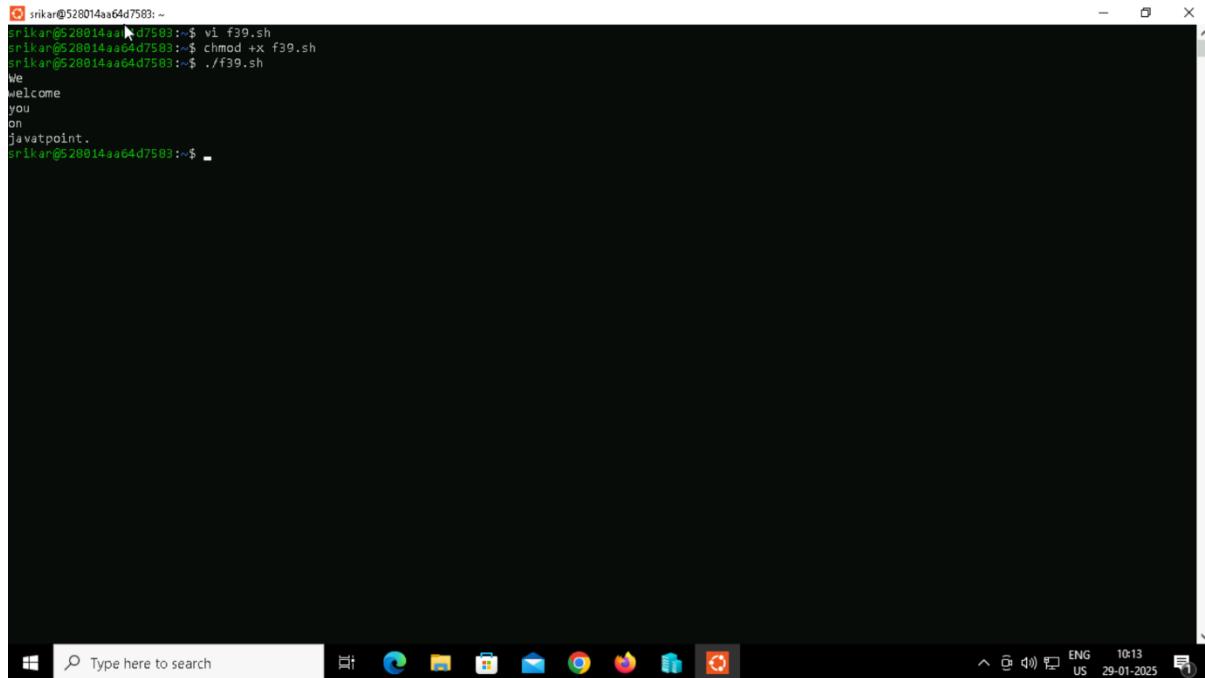
```
srikan@528014aa64d7583: ~
#!/bin/bash

my_str="We;Welcome;you;on;javatpoint."
my_arr=($(echo "$my_str" | tr ';' '\n')) # Corrected tr syntax

for i in "${my_arr[@]}"
do
    echo "$i"
done # Corrected loop termination

-- INSERT --
2,1 All
Type here to search ENG 10:12
US 29-01-2025
```

### OUTPUT

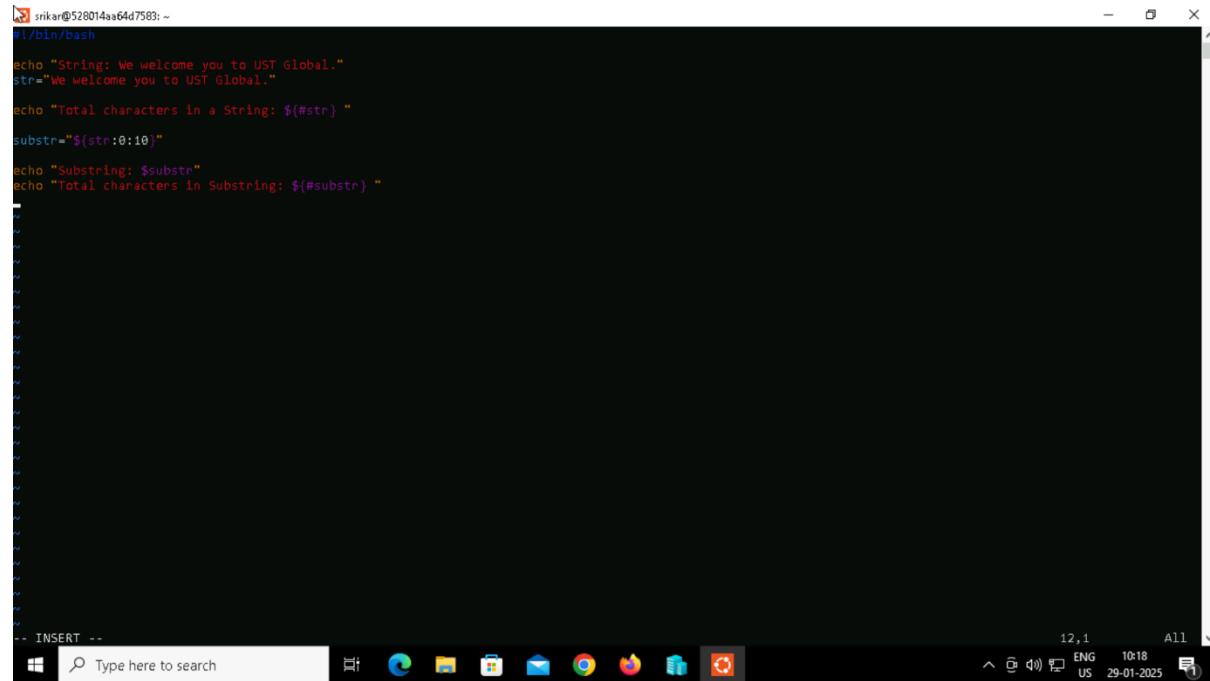


```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583:~$ vi f39.sh
srikan@528014aa64d7583:~$ chmod +x f39.sh
srikan@528014aa64d7583:~$ ./f39.sh
We
welcome
you
on
javatpoint.
srikan@528014aa64d7583:~$
```

## BASH SUBSTRING

### 1. To extract specific characters from starting

#### CODE



```
srinikar@528014aa64d7583: ~
#!/bin/bash

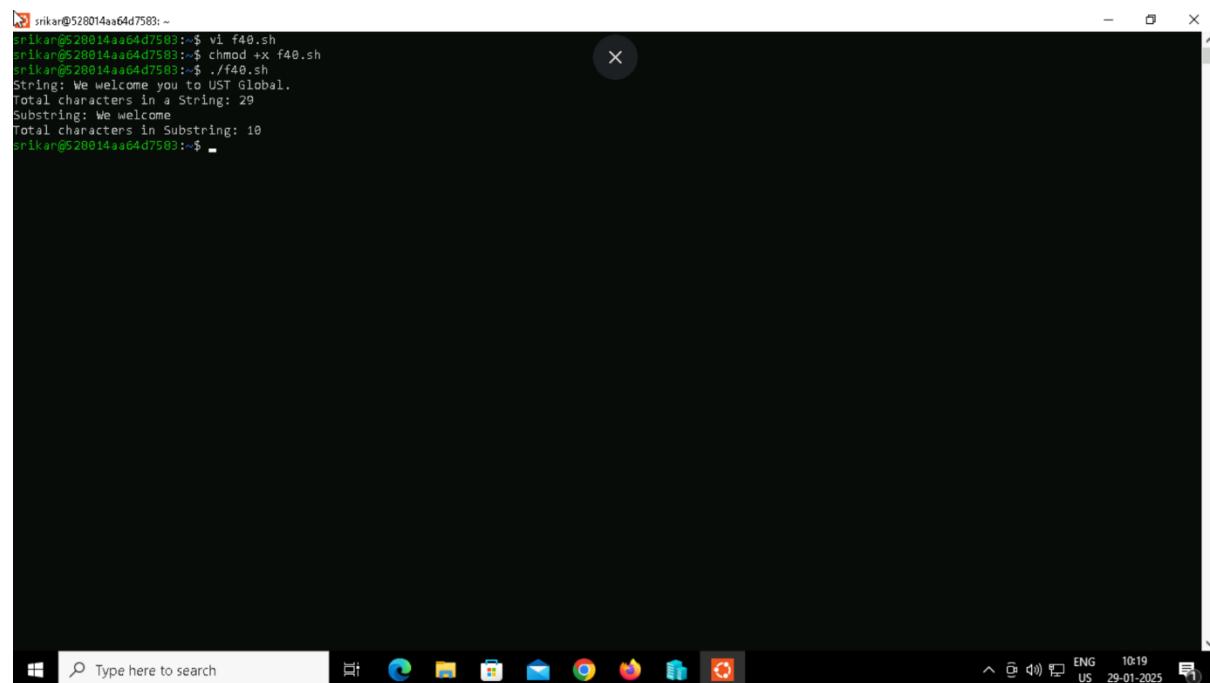
echo "String: We welcome you to UST Global."
str="We welcome you to UST Global."

echo "Total characters in a String: ${#str} "
substr="${str:0:10}"

echo "Substring: $substr"
echo "Total characters in Substring: ${#substr} "
-- INSERT --
```

The screenshot shows a Windows desktop environment with a terminal window open. The terminal window has a dark background and contains a Bash script. The script defines a string 'str' with the value 'We welcome you to UST Global.' It then prints the total length of the string and creates a substring 'substr' containing the first 10 characters ('We welcome'). Finally, it prints the length of the substring. The terminal window is titled with its path and has a standard Windows taskbar at the bottom.

#### OUTPUT

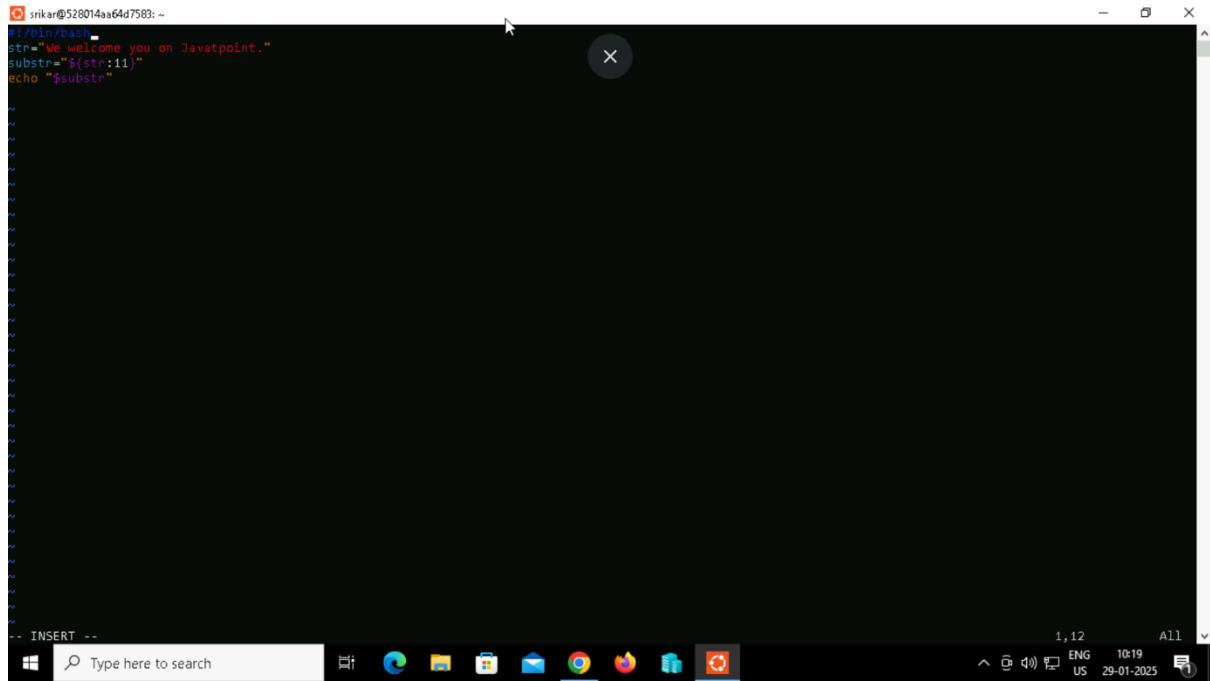


```
srinikar@528014aa64d7583: ~
srinikar@528014aa64d7583: ~$ vi f40.sh
srinikar@528014aa64d7583: ~$ chmod +x f40.sh
srinikar@528014aa64d7583: ~$ ./f40.sh
String: We welcome you to UST Global.
Total characters in a String: 29
Substring: We welcome
Total characters in Substring: 10
srinikar@528014aa64d7583: ~$
```

The screenshot shows the same Windows desktop environment with the terminal window now displaying the output of the script. The output shows the original string, its total length (29), the extracted substring 'We welcome', and the length of that substring (10). The terminal window is titled with its path and has a standard Windows taskbar at the bottom.

## 2. To extract from specific character onwards

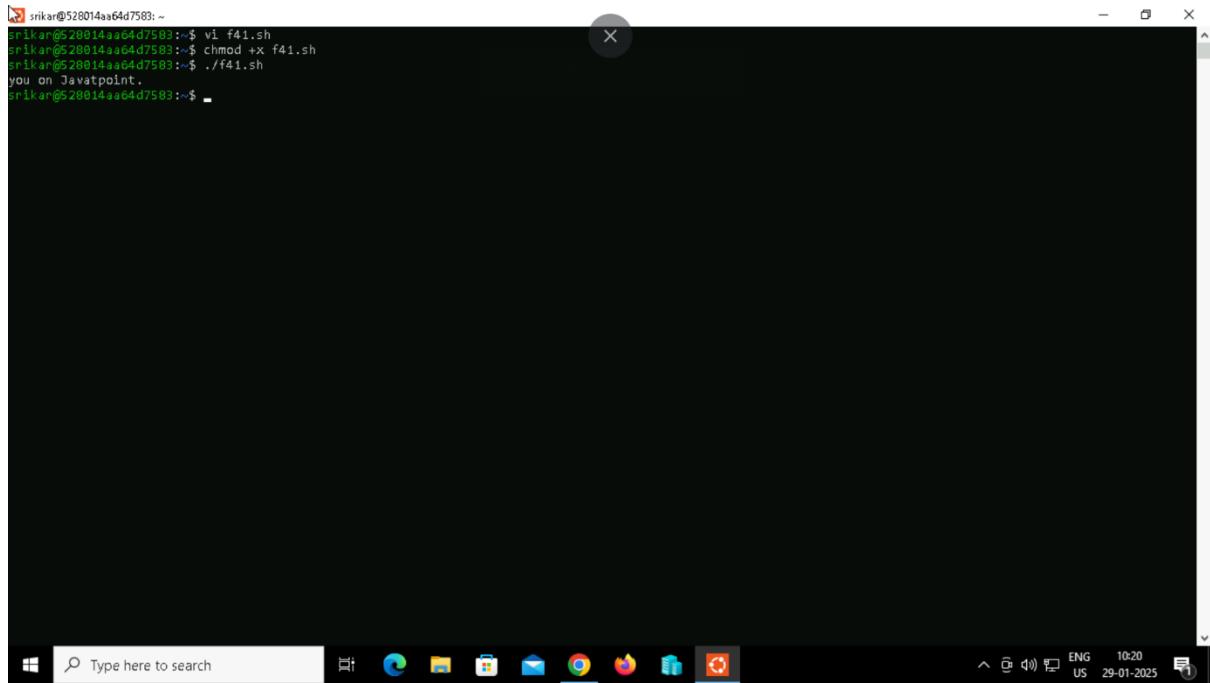
### CODE



```
srikar@528014aa64d7583: ~
#!/bin/bash
str="We welcome you on Javatpoint."
substr=${str:11}
echo "$substr"
-- INSERT --
```

The terminal window shows a Bash script being run. The script defines a variable `str` with the value "We welcome you on Javatpoint.". It then uses parameter expansion to set `substr` to the substring starting at index 11, which is "on Javatpoint.". Finally, it prints `$substr`. The terminal interface includes a status bar at the bottom with system information like battery level, network, and date.

### OUTPUT

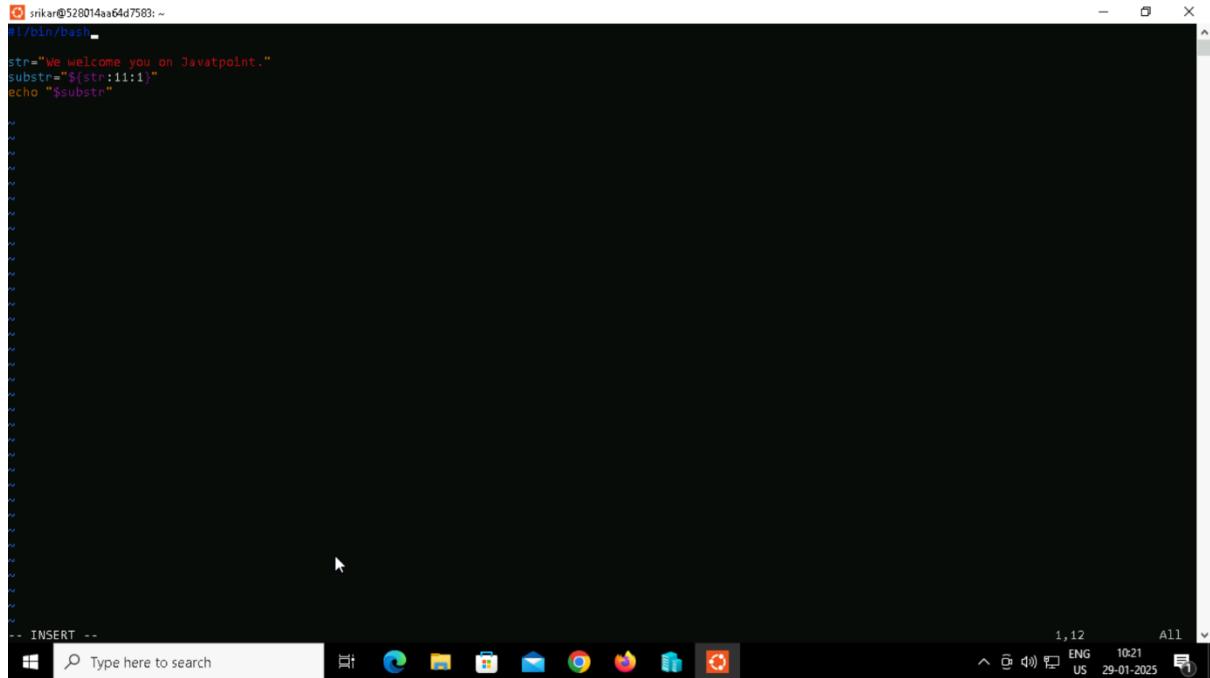


```
srikar@528014aa64d7583: ~
srikar@528014aa64d7583: ~$ vi f41.sh
srikar@528014aa64d7583: ~$ chmod +x f41.sh
srikar@528014aa64d7583: ~$ ./f41.sh
you on Javatpoint.
srikar@528014aa64d7583: ~$
```

The terminal window shows the execution of the script. After creating and saving the file `f41.sh`, changing its mode to executable, and running it, the output "you on Javatpoint." is displayed. The terminal interface includes a status bar at the bottom with system information like battery level, network, and date.

### 3. To extract a single character

#### CODE

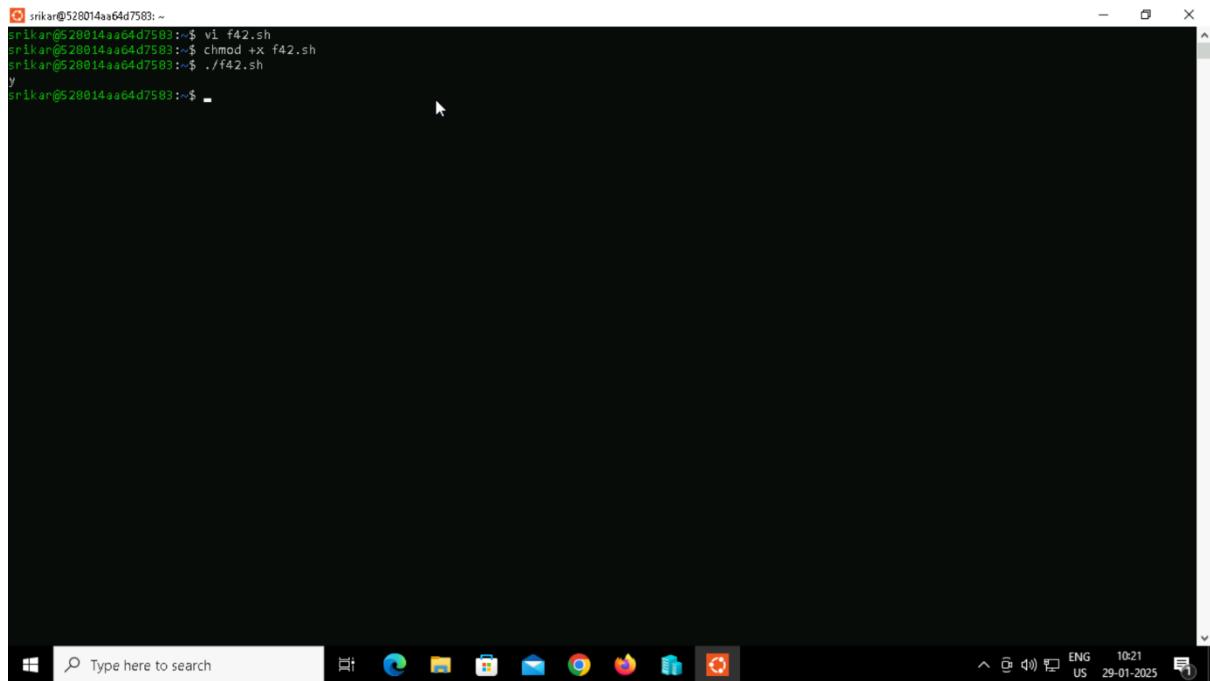


```
srikan@528014aa64d7583: ~
#!/bin/bash

str="We welcome you on Javatpoint."
substr=${str:1:1}
echo "$substr"

"
```

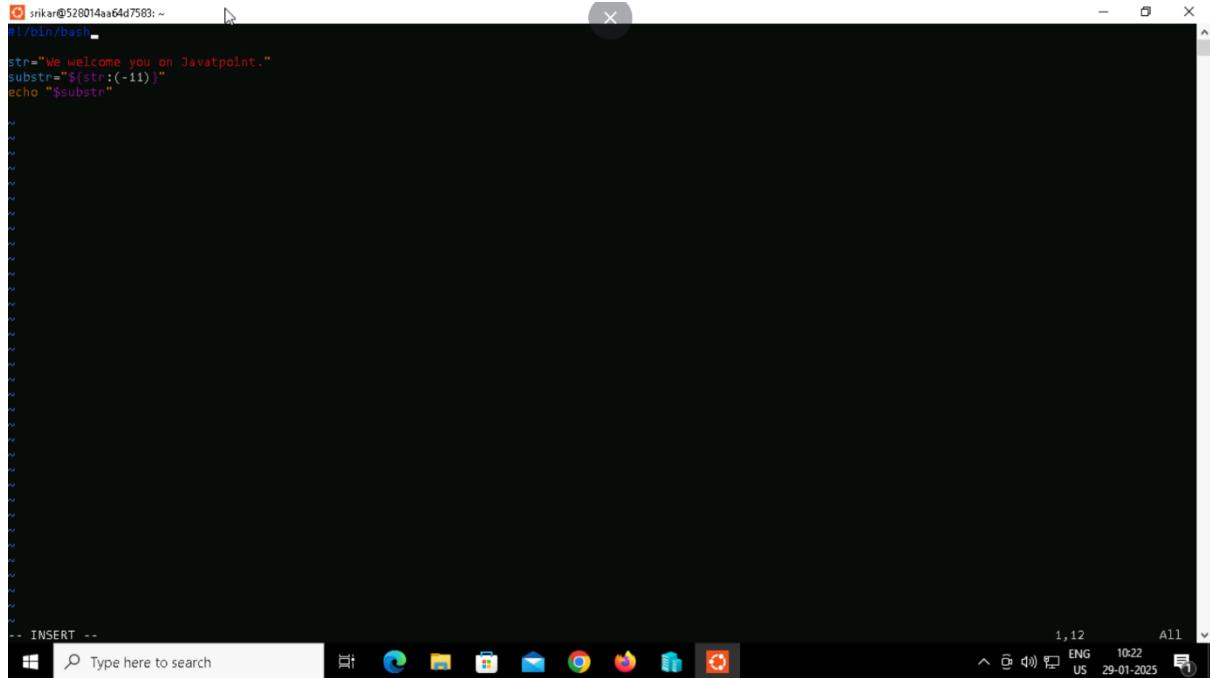
#### OUTPUT



```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583:~$ vi f42.sh
srikan@528014aa64d7583:~$ chmod +x f42.sh
srikan@528014aa64d7583:~$ ./f42.sh
y
srikan@528014aa64d7583:~$ -
```

#### 4. To extract the specific characters from last

##### CODE

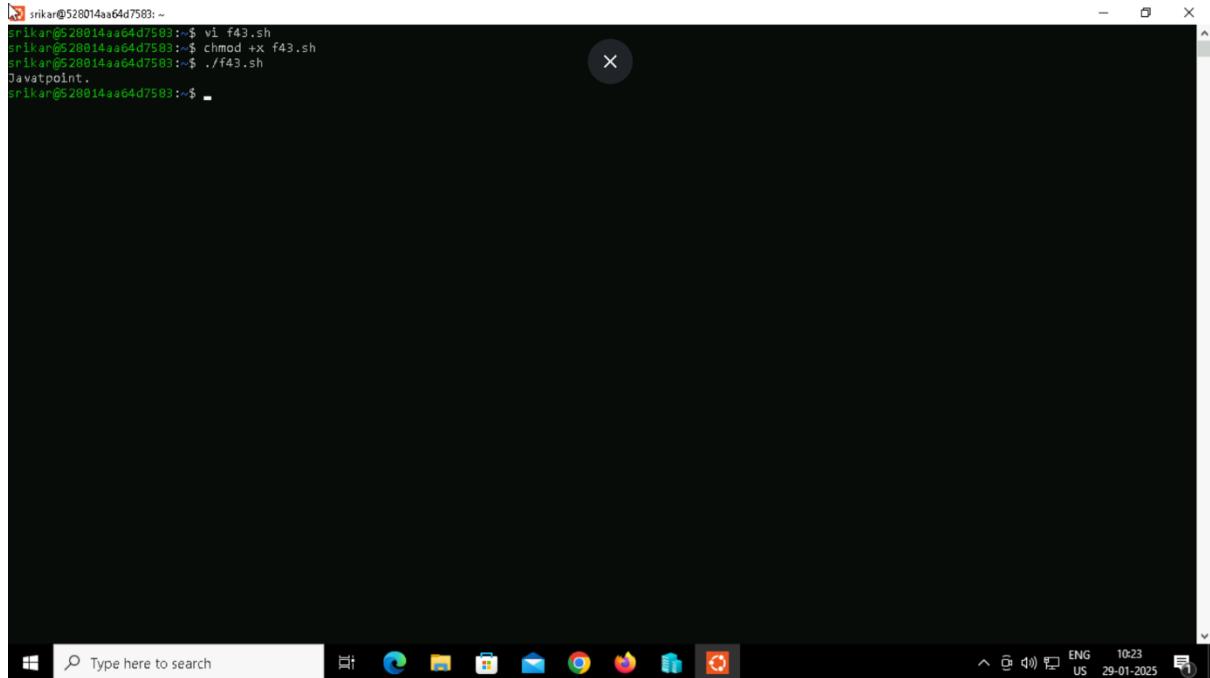


```
srikan@528014aa64d7583: ~
#!/bin/bash

str="We welcome you on Javatpoint."
substr=${str: -11}
echo "$substr"

"
```

##### OUTPUT

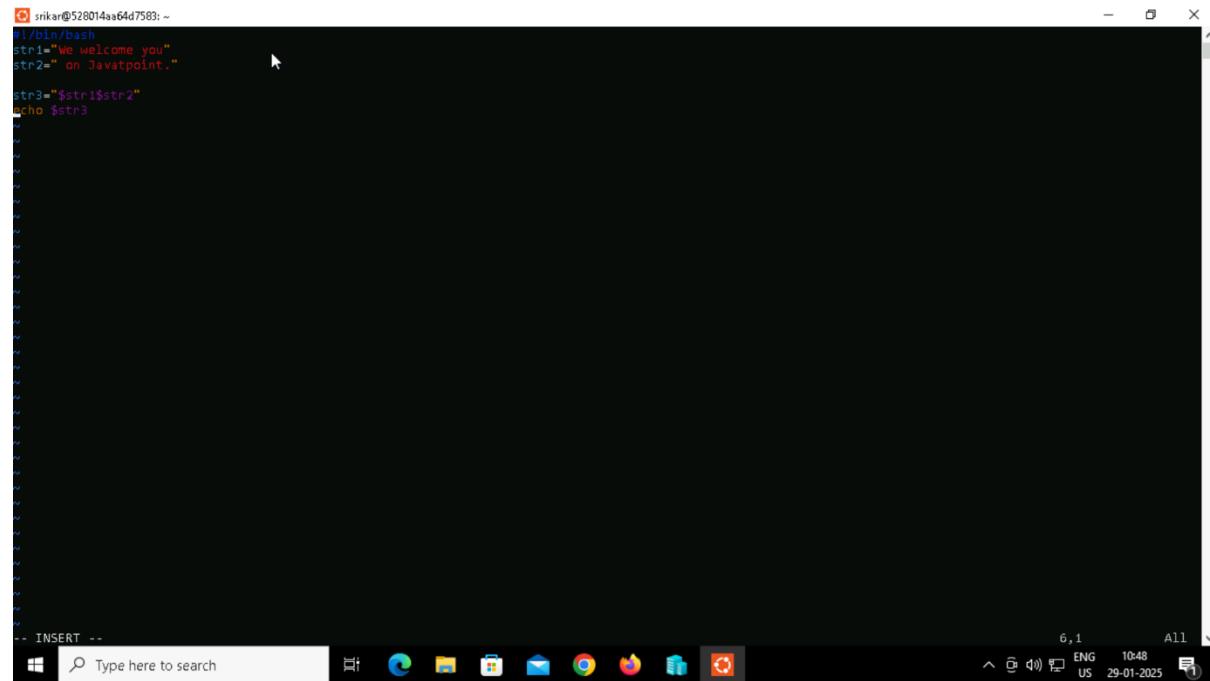


```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583: ~$ vi f43.sh
srikan@528014aa64d7583: ~$ chmod +x f43.sh
srikan@528014aa64d7583: ~$ ./f43.sh
Javatpoint.
srikan@528014aa64d7583: ~$
```

## BASH CONCATENATE STRING

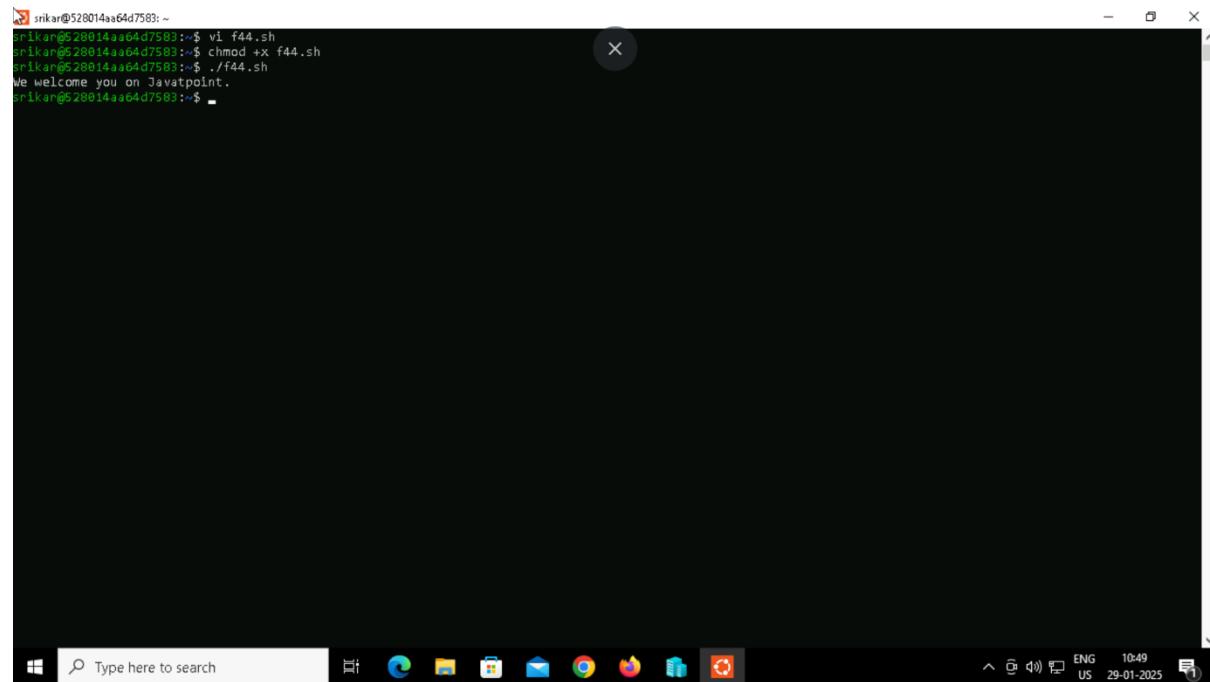
### 1. To write variables side by side

#### CODE



```
srikan@528014aa64d7583: ~
#!/bin/bash
str1="We welcome you"
str2=" on Javatpoint."
str3="$str1$str2"
echo $str3
```

#### OUTPUT



```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583:~$ vi f44.sh
srikan@528014aa64d7583:~$ chmod +x f44.sh
srikan@528014aa64d7583:~$ ./f44.sh
We welcome you on Javatpoint.
srikan@528014aa64d7583:~$
```

## 2. Using Double Quotes

## CODE

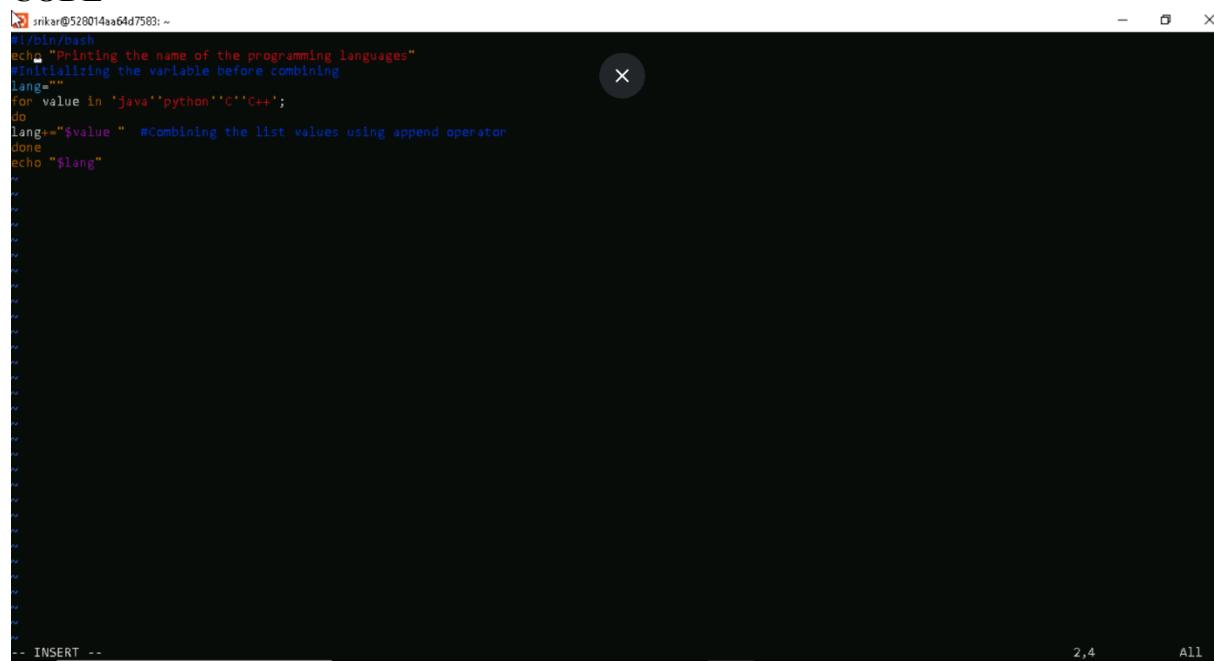
```
srikanth@520014aa64d7583: ~
#!/bin/bash
str="We welcome you"
echo "$str on Javatpoint."
-- INSERT --
```

## OUTPUT

```
srikanth@528014aa64d7583:~$ vi f45.sh
srikanth@528014aa64d7583:~$ chmod +x f45.sh
srikanth@528014aa64d7583:~$ ./f45.sh
We welcome you on Javatpoint.
srikanth@528014aa64d7583:~$ -
```

### 3. Using Append Operator with Loop

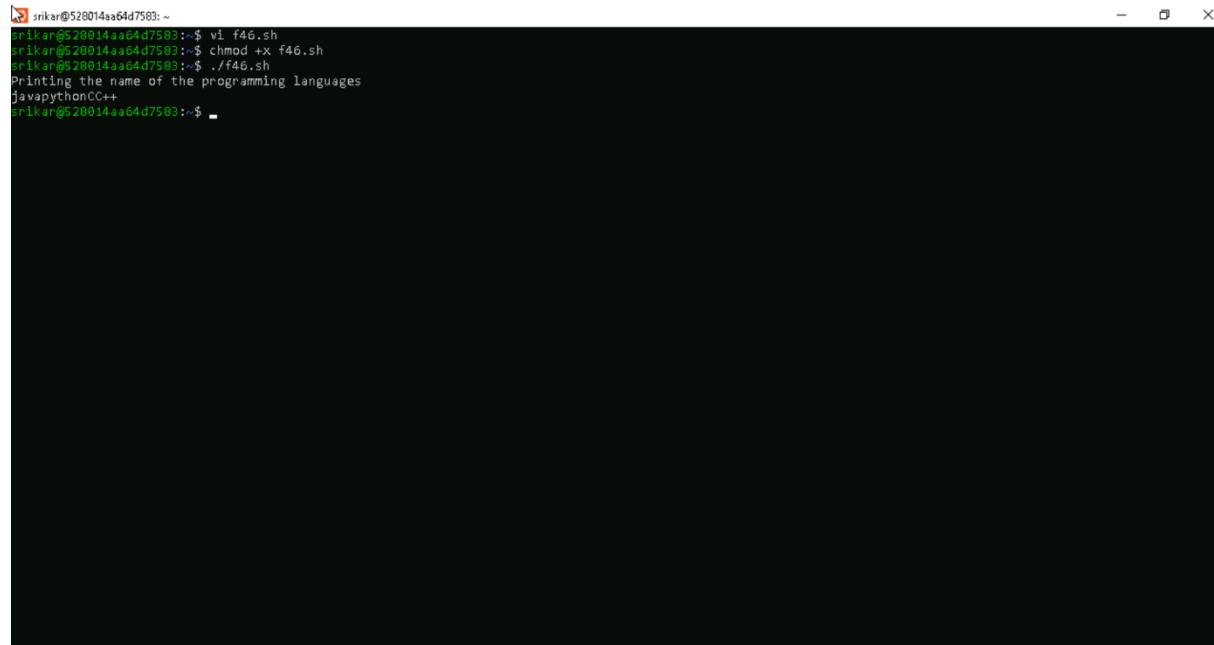
#### CODE



```
srikan@528014aa64d7583:~  
#!/bin/bash  
echo "Printing the name of the programming languages"  
#initializing the variable before combining  
lang=""  
for value in 'java' 'python' 'C' 'C++';  
do  
    lang+="$value " #combining the list values using append operator  
done  
echo "$lang"  
  
-- INSERT --
```

2,4 All

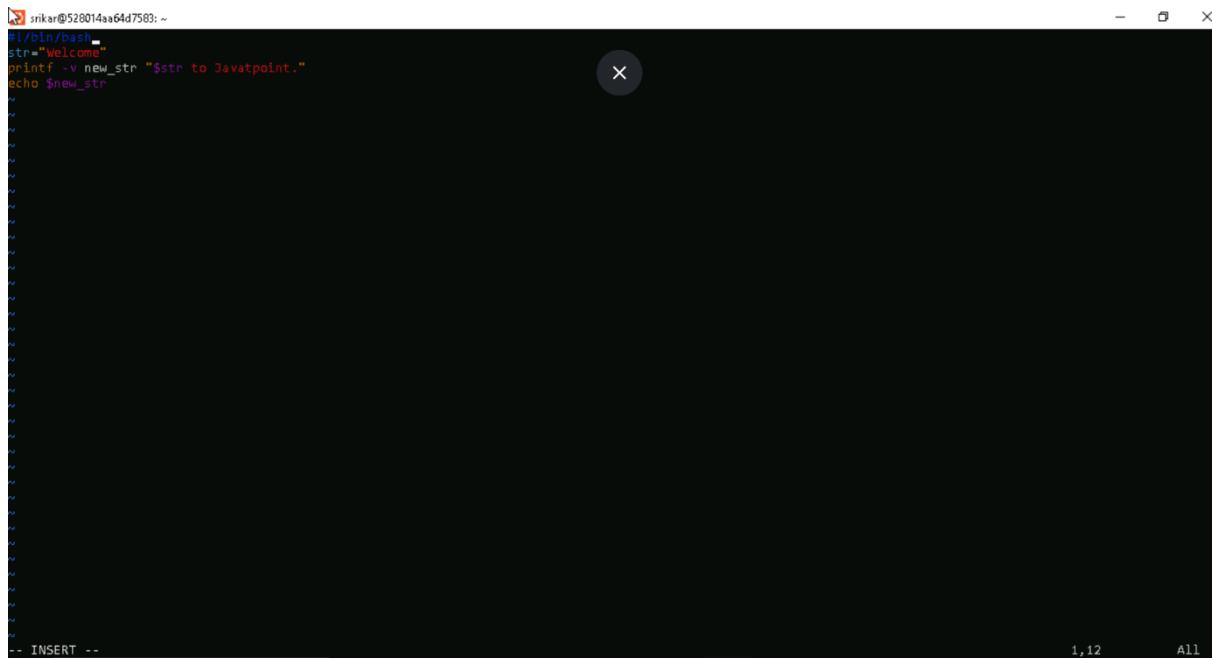
#### OUTPUT



```
srikan@528014aa64d7583:~  
srikan@528014aa64d7583:~$ vi f46.sh  
srikan@528014aa64d7583:~$ chmod +x f46.sh  
srikan@528014aa64d7583:~$ ./f46.sh  
Printing the name of the programming languages  
javapythonCC++  
srikan@528014aa64d7583:~$ -
```

## 4. Using the PrintF function

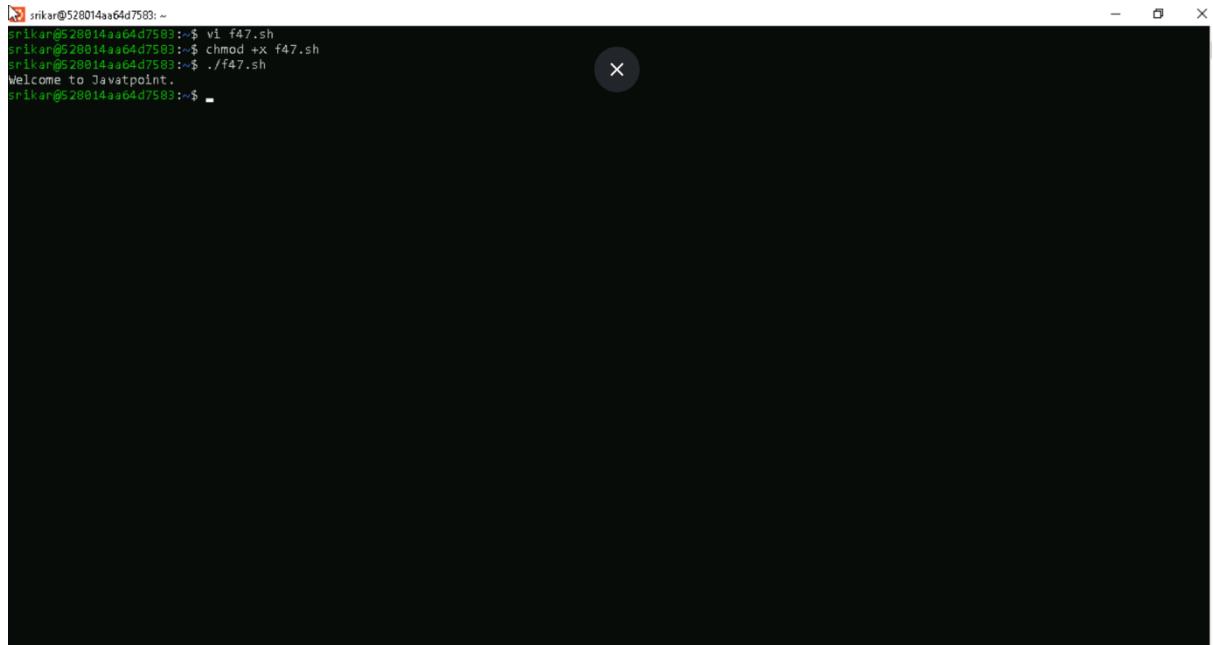
### CODE



```
srikan@528014aa64d7583: ~
#!/bin/bash
str="Welcome"
printf -v new_str "%s to Javatpoint."
echo $new_str
-- INSERT --
```

The terminal window shows a command-line interface with a dark background. The command `printf -v new\_str "%s to Javatpoint."` is entered, followed by `echo \$new\_str`. The output is not visible in the screenshot. The status bar at the bottom right indicates "1,12" and "All".

### OUTPUT



```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583:~$ vi f47.sh
srikan@528014aa64d7583:~$ chmod +x f47.sh
srikan@528014aa64d7583:~$ ./f47.sh
Welcome to Javatpoint.
srikan@528014aa64d7583:~$ -
```

The terminal window shows a command-line interface with a dark background. The script `f47.sh` is created, given execute permissions, and run. The output "Welcome to Javatpoint." is displayed. The status bar at the bottom right indicates "1,12" and "All".

## BASH FUNCTIONS

## 1. Bash function passing the arguments

## CODE

```
srinikar@528014aa64d7583: ~
$!/bin/bash
function_arguments()
{
echo $1
echo $2
echo $3
echo $4
echo $5
}
function_arguments "We""welcome""you""on""Javaatpoint."
```
-- INSERT --

```

## OUTPUT

A screenshot of a Linux terminal window titled 'srikan@528014aa64d7583:~'. The terminal displays the following command-line session:

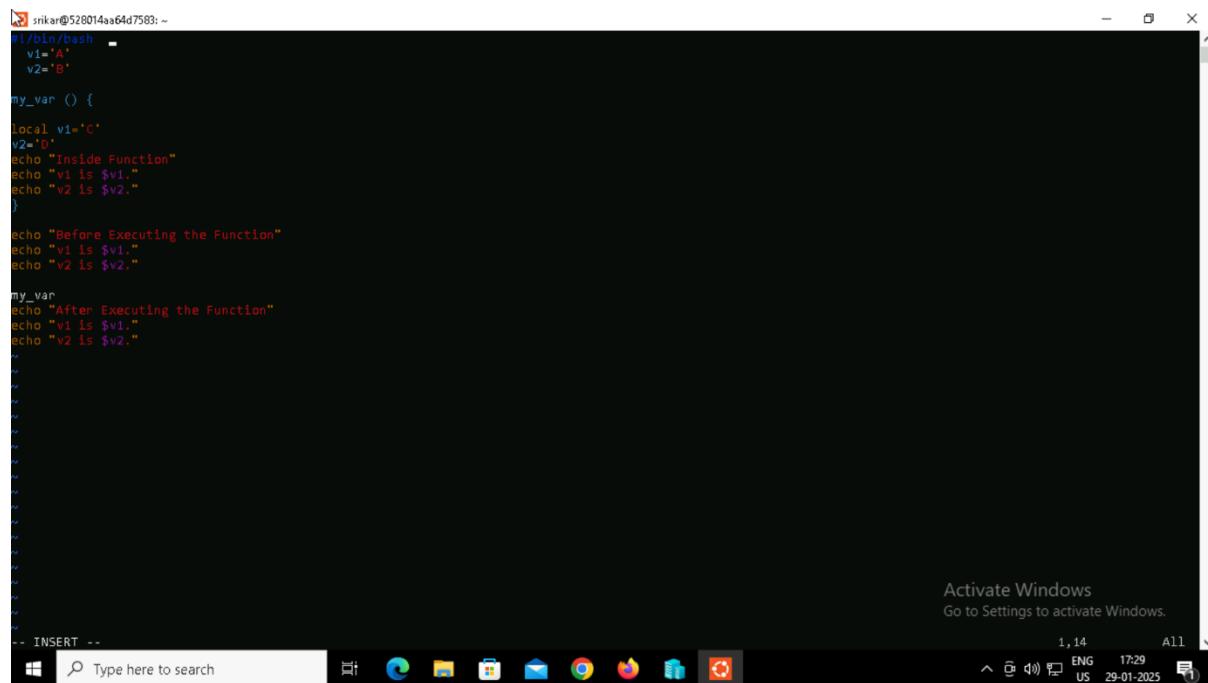
```
srikan@528014aa64d7583:~$ vi f48.sh
srikan@528014aa64d7583:~$ chmod +x f48.sh
srikan@528014aa64d7583:~$ ./f48.sh
Welcome you on Javatpoint.

srikan@528014aa64d7583:~$
```

The terminal has a dark background with light-colored text. A small circular close button is visible in the top right corner. The bottom of the screen shows the Windows taskbar with icons for File Explorer, Edge, Mail, Google Chrome, and others. The system tray shows battery status, network connection, and system settings. A watermark for 'Activate Windows' is present in the bottom right corner.

## 2. Bash Function using Variable Scope

### CODE



```
#!/bin/bash
v1="A"
v2="B"

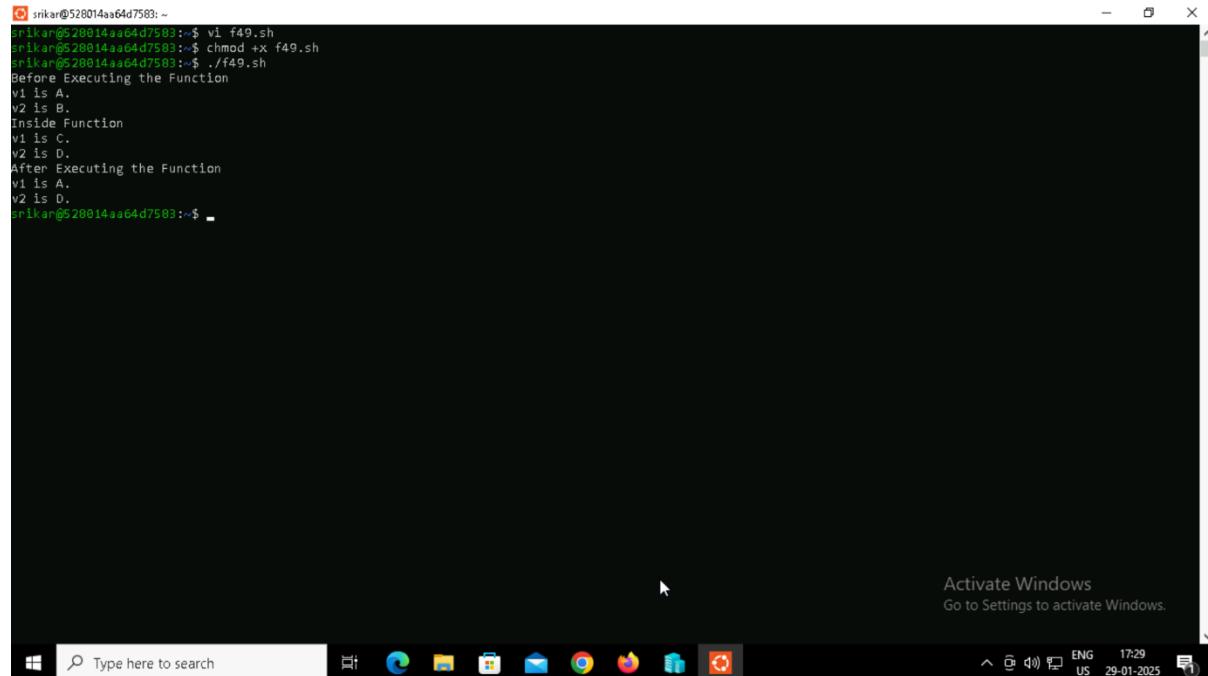
my_var () {
local v1="C"
v2="D"
echo "Inside Function"
echo "v1 is $v1"
echo "v2 is $v2."
}

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."

```

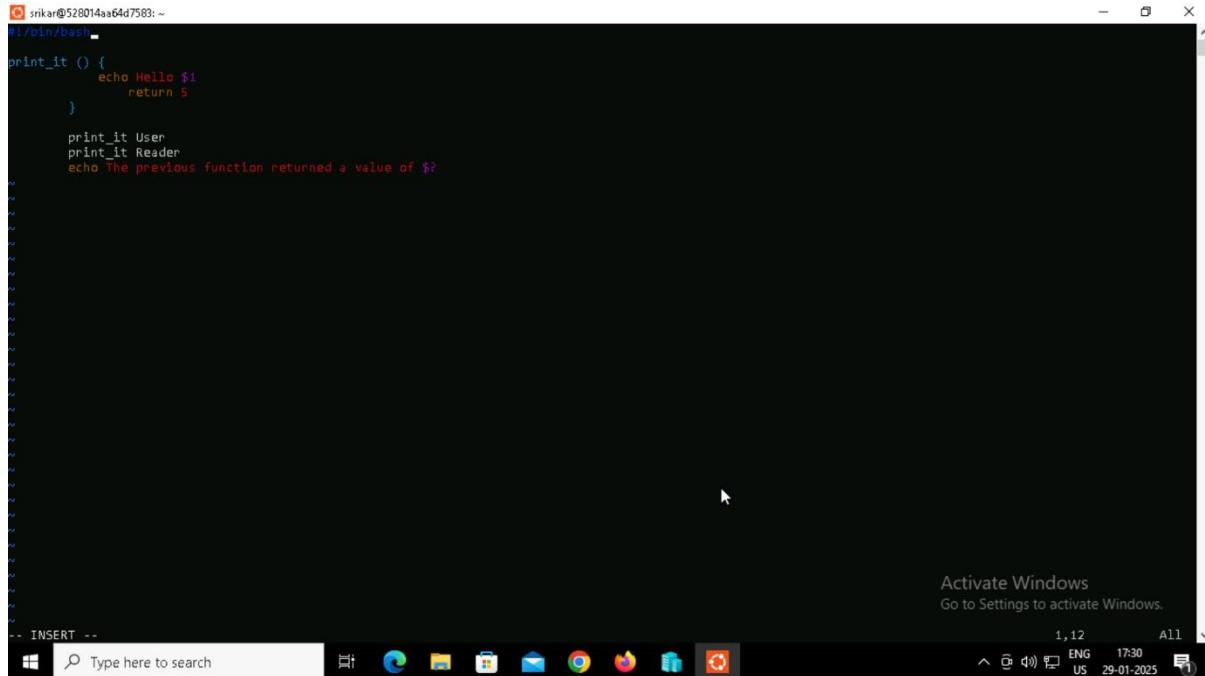
### OUTPUT



```
srikan@528014aa64d7583:~$ vi f49.sh
srikan@528014aa64d7583:~$ chmod +x f49.sh
srikan@528014aa64d7583:~$ ./f49.sh
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.
srikan@528014aa64d7583:~$
```

### 3. Bash function Returning values

#### CODE

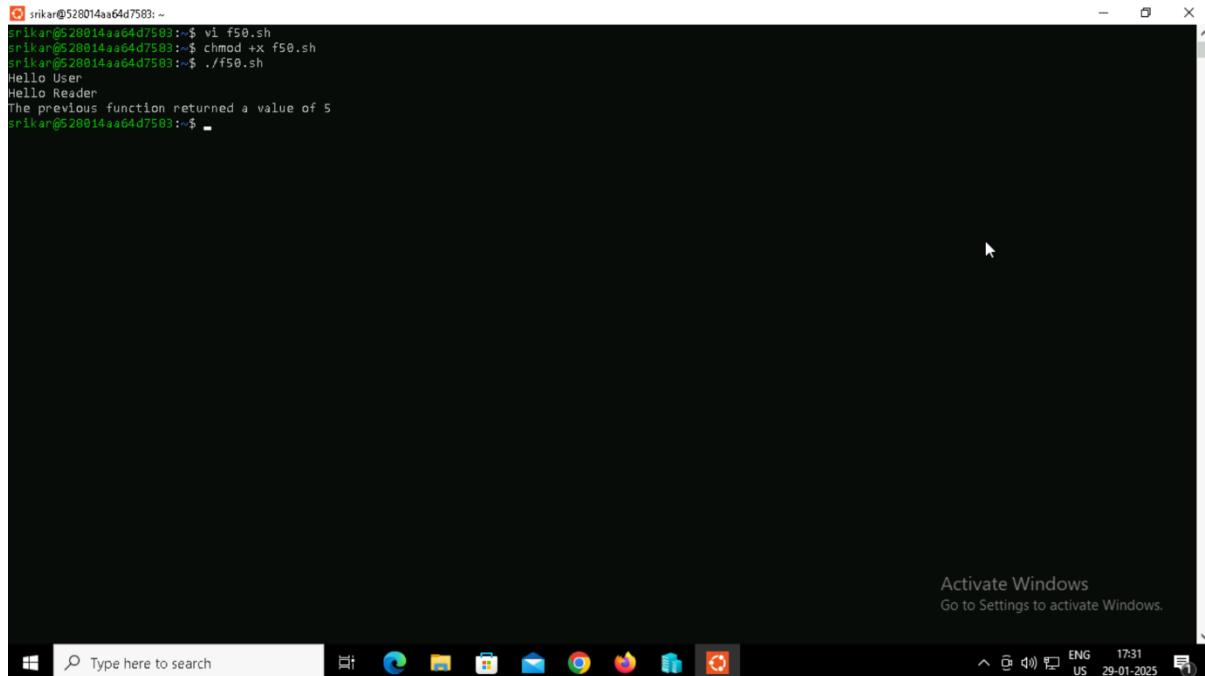


```
srikan@528014aa64d7583: ~
#!/bin/bash

print_it () {
    echo Hello $1
    return 5
}

print_it User
print_it Reader
echo The previous function returned a value of $?
```

#### OUTPUT

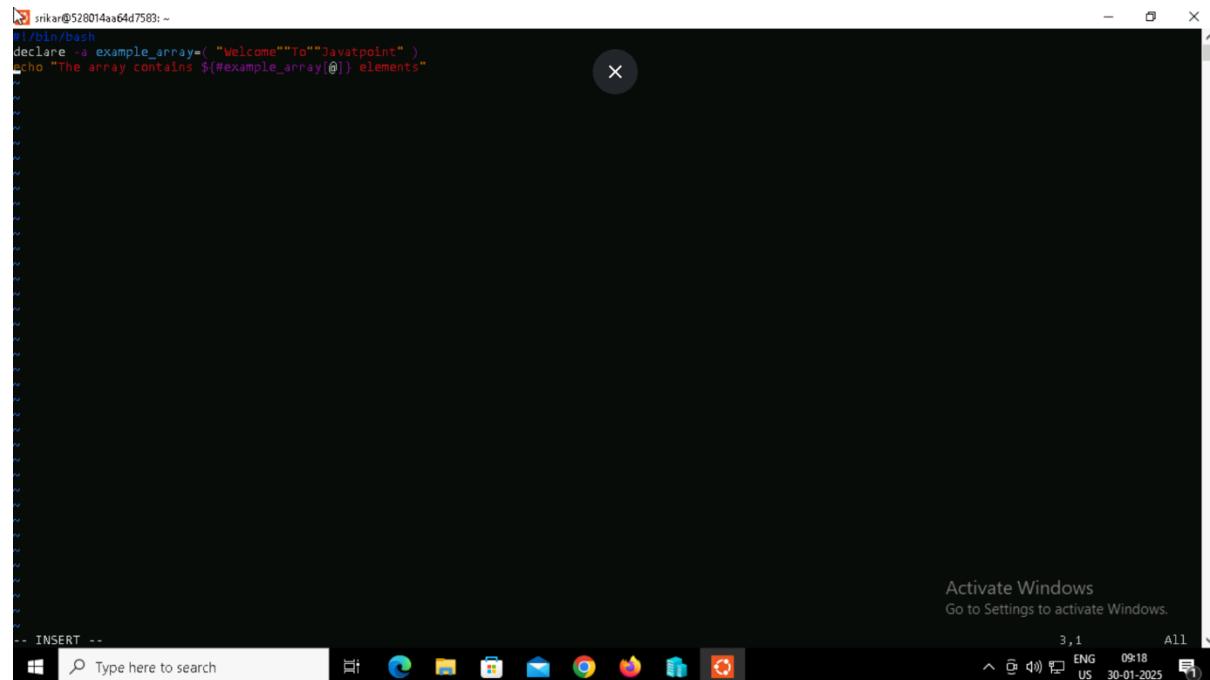


```
srikan@528014aa64d7583: ~
srikan@528014aa64d7583:~$ vi f50.sh
srikan@528014aa64d7583:~$ chmod +x f50.sh
srikan@528014aa64d7583:~$ ./f50.sh
Hello User
Hello Reader
The previous function returned a value of 5
srikan@528014aa64d7583:~$
```

## BASH ARRAYS

### 1. Finding the length of Array

#### CODE



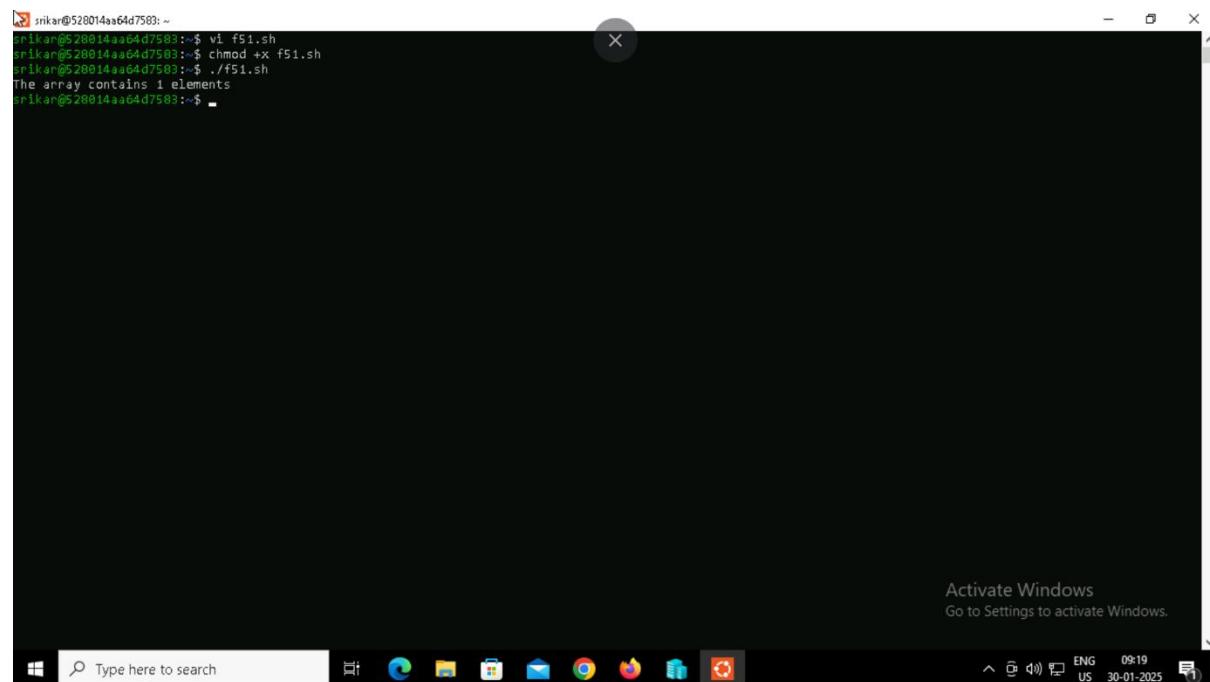
```
srikan@528014aa64d7583: ~
$ /bin/bash
declare -a example_array=( "Welcome""To""Javaatpoint" )
echo "The array contains ${#example_array[@]} elements"
-- TINSERT --
Windows Type here to search 3,1 ENG 09:18 All
US 30-01-2025
```

The screenshot shows a Windows desktop environment with a terminal window open. The terminal window has a dark background and displays the following Bash script:

```
#!/bin/bash
declare -a example_array=( "Welcome""To""Javaatpoint" )
echo "The array contains ${#example_array[@]} elements"
```

The terminal window is titled with its command line prompt. At the bottom of the screen, there is a taskbar with various icons for applications like File Explorer, Edge, and Google Chrome. The system tray shows the date and time as 30-01-2025 at 09:18, and the language and region settings as English (US).

#### OUTPUT



```
srikan@528014aa64d7583: ~
$ vi f51.sh
srikan@528014aa64d7583: ~$ chmod +x f51.sh
srikan@528014aa64d7583: ~$ ./f51.sh
The array contains 1 elements
srikan@528014aa64d7583: ~
Windows Type here to search 3,1 ENG 09:19 All
US 30-01-2025
```

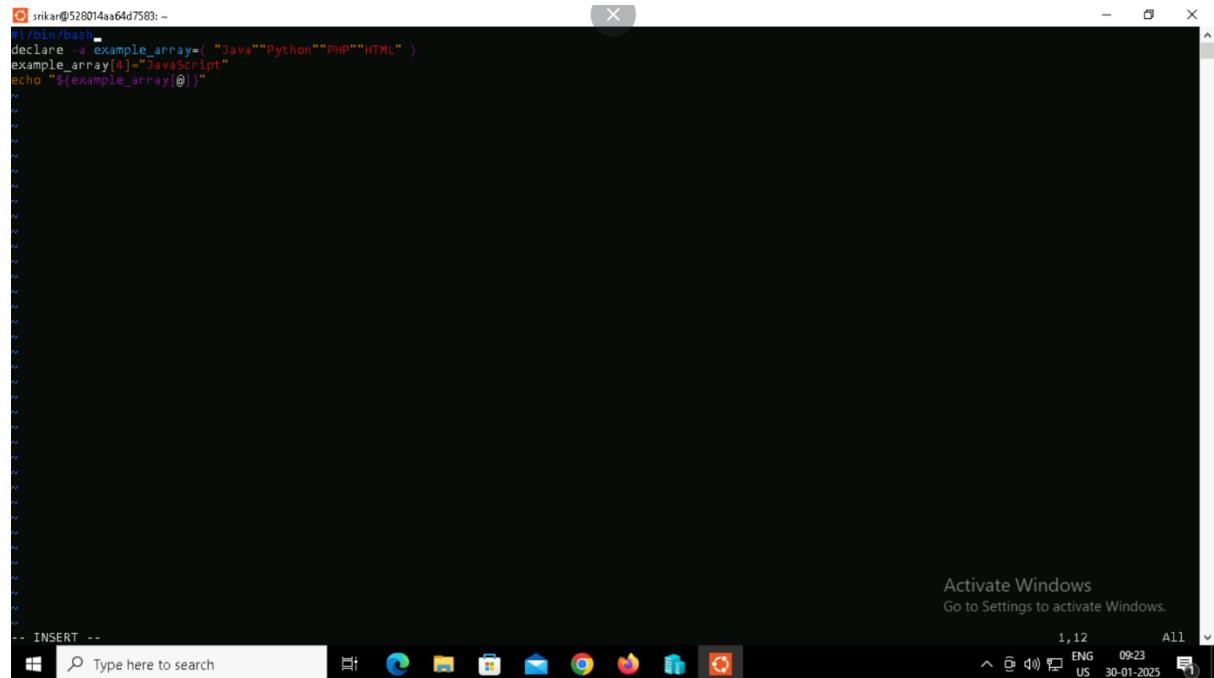
The screenshot shows the same Windows desktop environment with the terminal window now displaying the output of the script. The output is:

```
The array contains 1 elements
```

The terminal window is titled with its command line prompt. At the bottom of the screen, there is a taskbar with various icons for applications like File Explorer, Edge, and Google Chrome. The system tray shows the date and time as 30-01-2025 at 09:19, and the language and region settings as English (US).

## 2. Adding the elements in the array

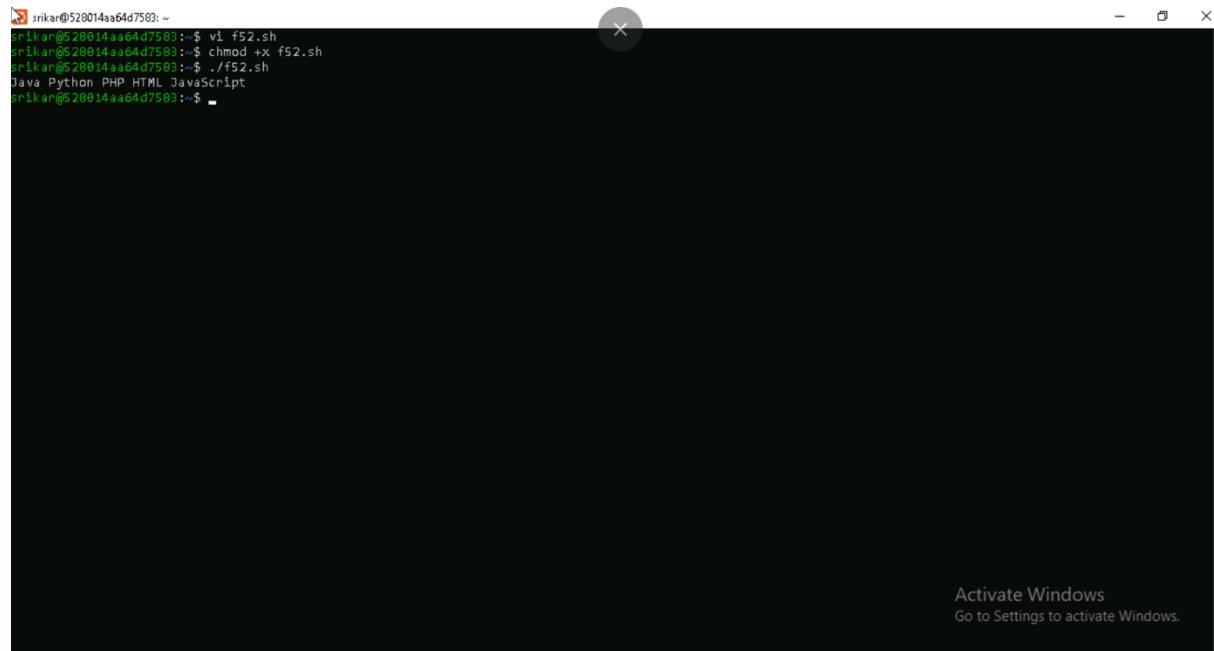
### CODE



```
srikan@528014aa64d7583: ~
$ /bin/bash
declare -a example_array=( "Java""Python""PHP""HTML" )
example_array[4]="JavaScript"
echo "${example_array[@]}"

-- INSERT --
Windows Type here to search 1,12 ENG 09:23 All
Go to Settings to activate Windows.
US 30-01-2025
```

### OUTPUT

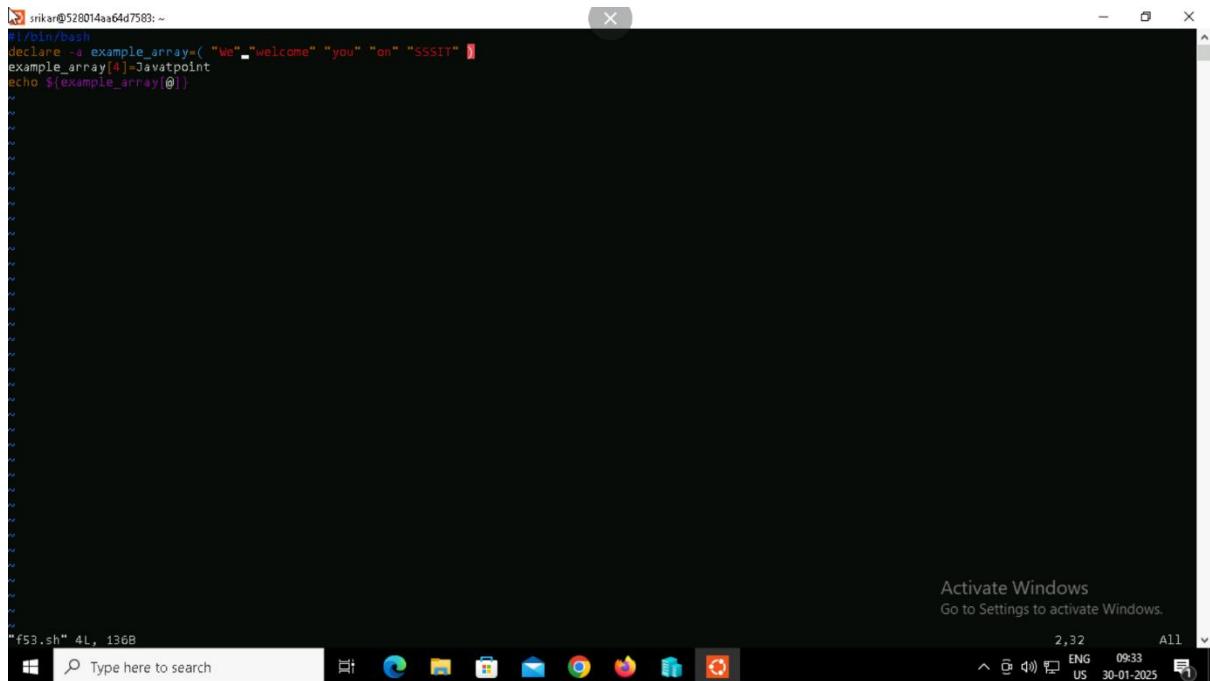


```
srikan@528014aa64d7583: ~
$ vi f52.sh
srikan@528014aa64d7583: ~$ chmod +x f52.sh
srikan@528014aa64d7583: ~$ ./f52.sh
Java Python PHP HTML JavaScript
srikan@528014aa64d7583: ~
```

Activate Windows  
Go to Settings to activate Windows.

### 3. Updating the elements in array

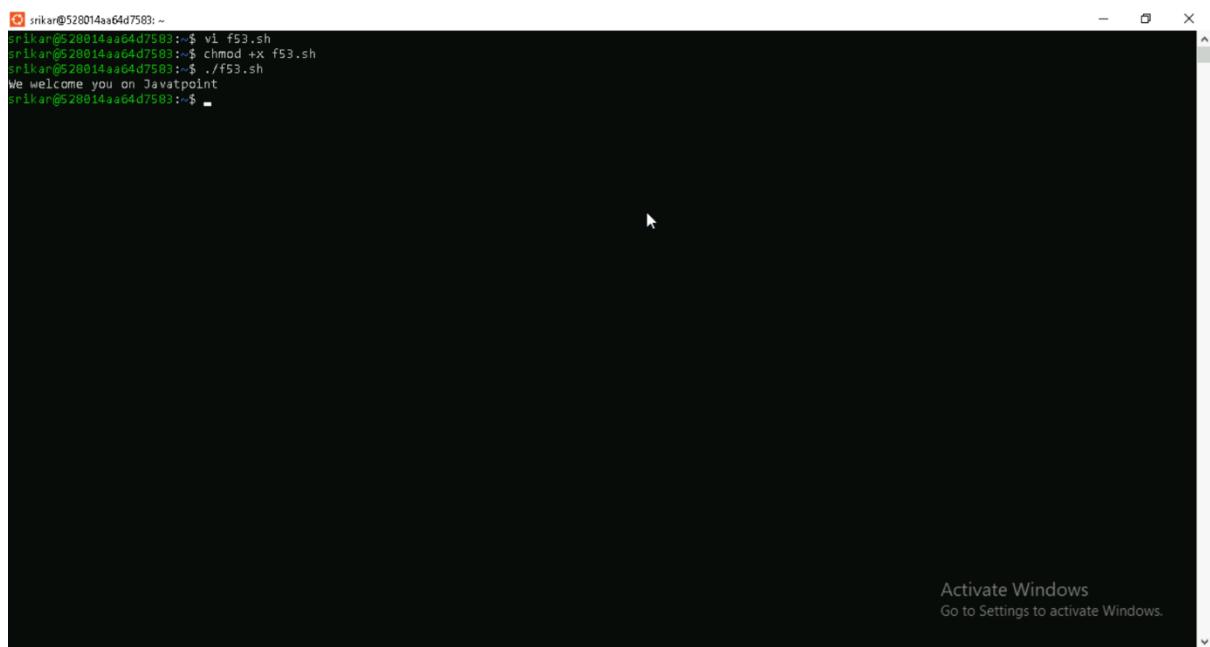
#### CODE



```
#!/bin/bash
declare -a example_array=( "We" "welcome" "you" "on" "SSSIT" )
example_array[4]=Javatpoint
echo ${example_array[@]}
```

The screenshot shows a Windows terminal window with a black background. At the top, it displays the command line: `srikanth@520014aa64d7583: ~`. Below this, the terminal content is shown in white text. The code above is pasted into the terminal. The terminal window has a standard Windows title bar with minimize, maximize, and close buttons. At the bottom of the window, there is a status bar showing system information: `Activate Windows Go to Settings to activate Windows.`, `2,32 All`, `ENG 09:33`, and `US 30-01-2025`.

#### OUTPUT

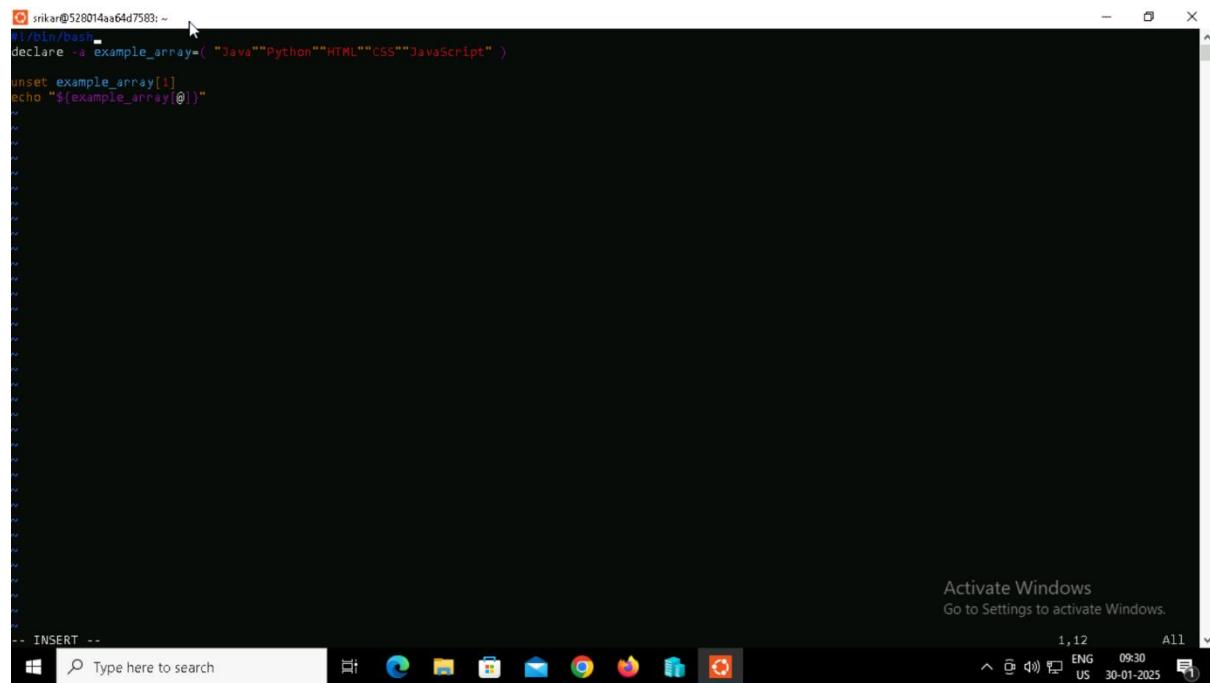


```
srikanth@520014aa64d7583: ~
srikanth@520014aa64d7583: ~$ vi f53.sh
srikanth@520014aa64d7583: ~$ chmod +x f53.sh
srikanth@520014aa64d7583: ~$ ./f53.sh
We welcome you on Javatpoint
srikanth@520014aa64d7583: ~$
```

The screenshot shows a Windows terminal window with a black background. At the top, it displays the command line: `srikanth@520014aa64d7583: ~`. Below this, the terminal content is shown in white text. The command `./f53.sh` is executed, and the output "We welcome you on Javatpoint" is displayed. The terminal window has a standard Windows title bar with minimize, maximize, and close buttons. At the bottom of the window, there is a status bar showing system information: `Activate Windows Go to Settings to activate Windows.`, `2,32 All`, `ENG 09:33`, and `US 30-01-2025`.

#### 4. Deleting the elements in array

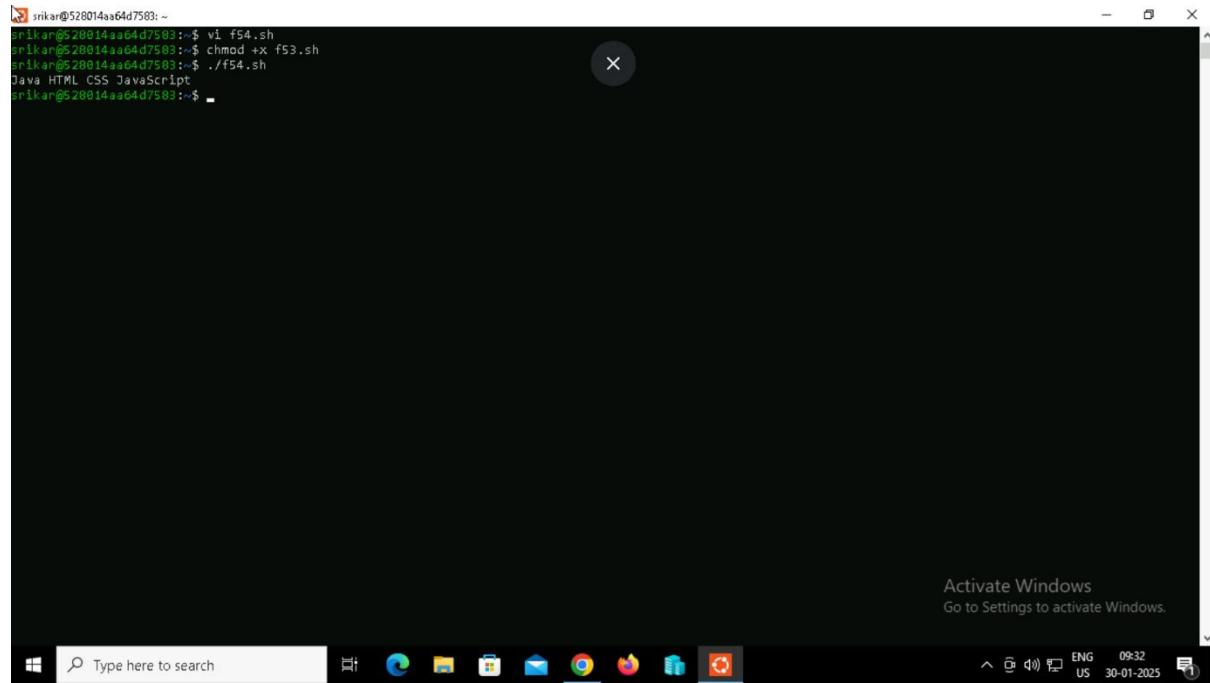
##### CODE



```
srikan@528014aae64d7583:~  
srikan@528014aae64d7583:~$ cd /bin/bash  
declare -a example_array=( "Java" "Python" "HTML" "CSS" "JavaScript" )  
unset example_array[1]  
echo "${example_array[@]}"  
Java HTML CSS JavaScript
```

The terminal window shows a Bash session. The user runs a command to declare an array named 'example\_array' containing several programming languages. Then, they use the 'unset' command to remove the second element ('Python') from the array. Finally, they print the array's contents again, which now only includes 'Java', 'HTML', 'CSS', and 'JavaScript'.

##### OUTPUT



```
srikan@528014aae64d7583:~$ vi f54.sh  
srikan@528014aae64d7583:~$ chmod +x f53.sh  
srikan@528014aae64d7583:~$ ./f54.sh  
Java HTML CSS JavaScript  
srikan@528014aae64d7583:~$
```

The terminal window shows the execution of a shell script named 'f54.sh'. The user first edits the script with 'vi', then changes its mode to executable with 'chmod +x'. Finally, they run the script with './f54.sh'. The output is identical to the code shown in the previous screenshot, displaying the array elements 'Java', 'HTML', 'CSS', and 'JavaScript'.