

Bash Project

Bash Case :

Example 1 In this example, we have defined a simple scenario to demonstrate the use of the case statement.

Code :

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/bash

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

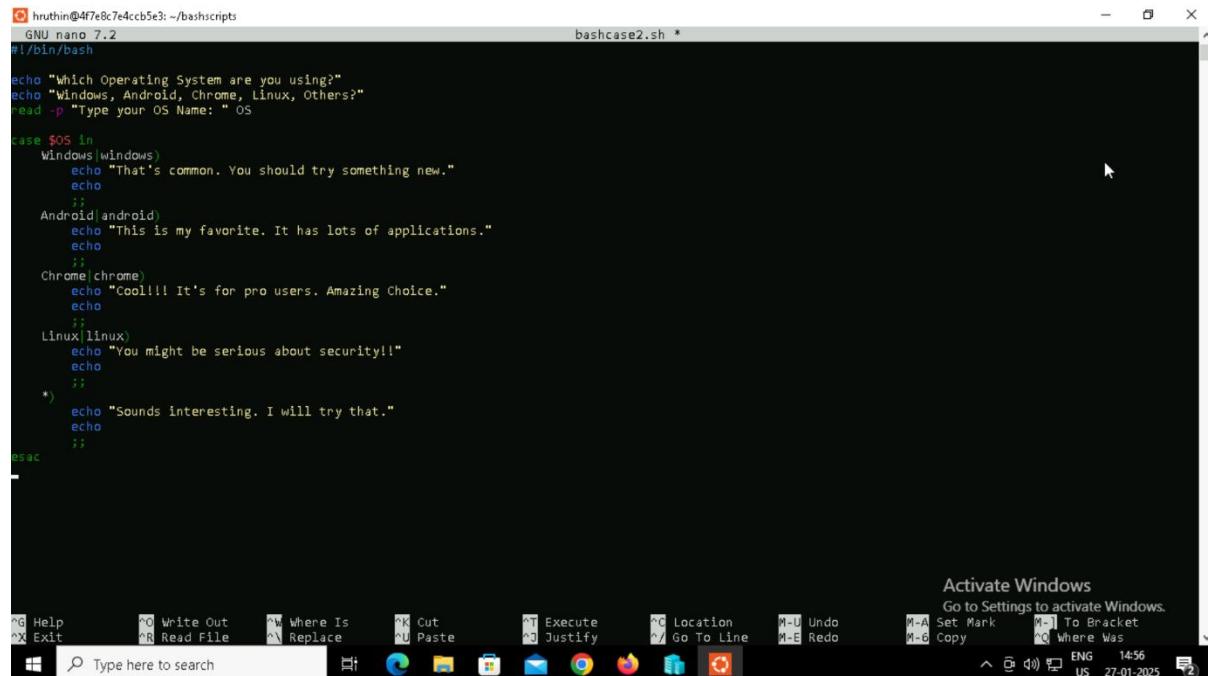
Output :

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x ifelse3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./ifelse3.sh
Enter a value: 57
The value you typed is greater than 9.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano ifelse4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x ifelse4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./ifelse4.sh
Enter a value: 58
The value you typed is greater than 9.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano elseif1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x elseif1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./elseif1.sh
Enter a number of quantity: 10
Eligible for 5% discount
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano elseif2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x elseif2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./elseif2.sh
Enter a number of quantity: 20
No discount
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano bashcase.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x bashcase.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./bashcase.sh
Do you know Java Programming?
Yes/No? : yes
That's amazing.

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 2 In this example, we have defined a combined scenario where there is also a default case when no previous matched case is found.

Code :

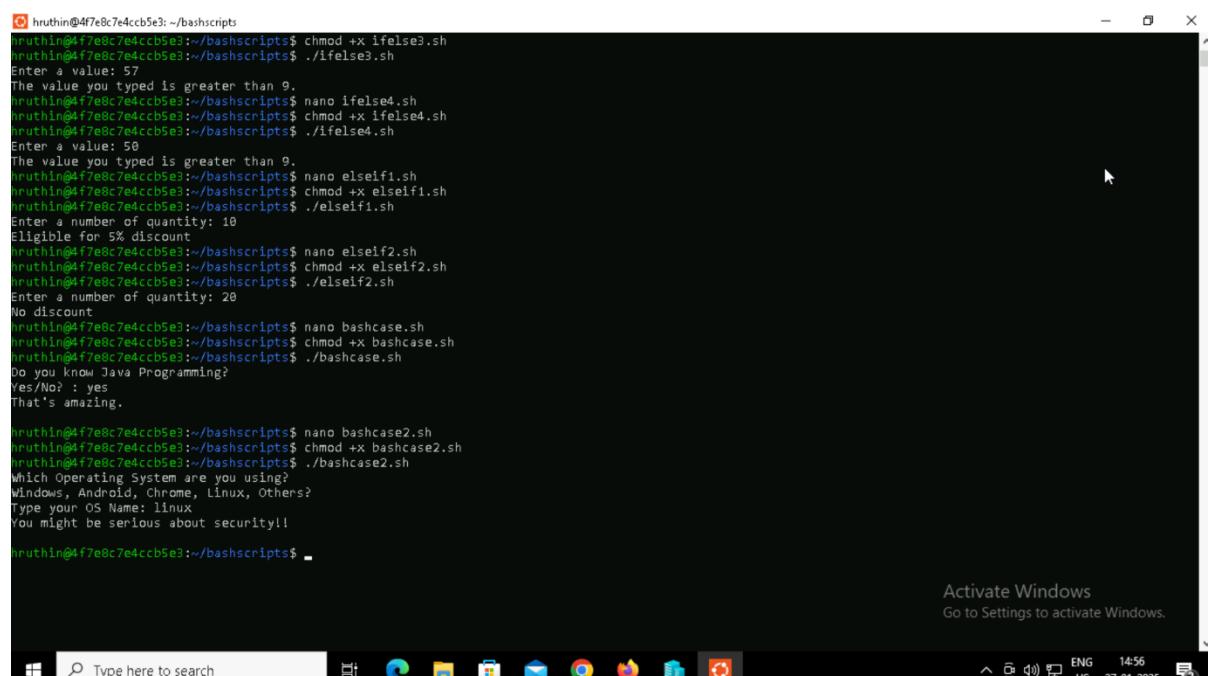


```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2                                bashcase2.sh *
#!/bin/bash

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 :



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x ifelse3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./ifelse3.sh
Enter a value: 57
The value you typed is greater than 9.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano ifelse4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x ifelse4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./ifelse4.sh
Enter a value: 58
The value you typed is greater than 9.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano elseif1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x elseif1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./elseif1.sh
Enter a number of quantity: 10
Eligible for 5% discount
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano elseif2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x elseif2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./elseif2.sh
Enter a number of quantity: 20
No discount
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano bashcase.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x bashcase.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./bashcase.sh
Do you know Java Programming?
Yes/No? : yes
That's amazing.

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano bashcase2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x bashcase2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./bashcase2.sh
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name: linux
You might be serious about security!!

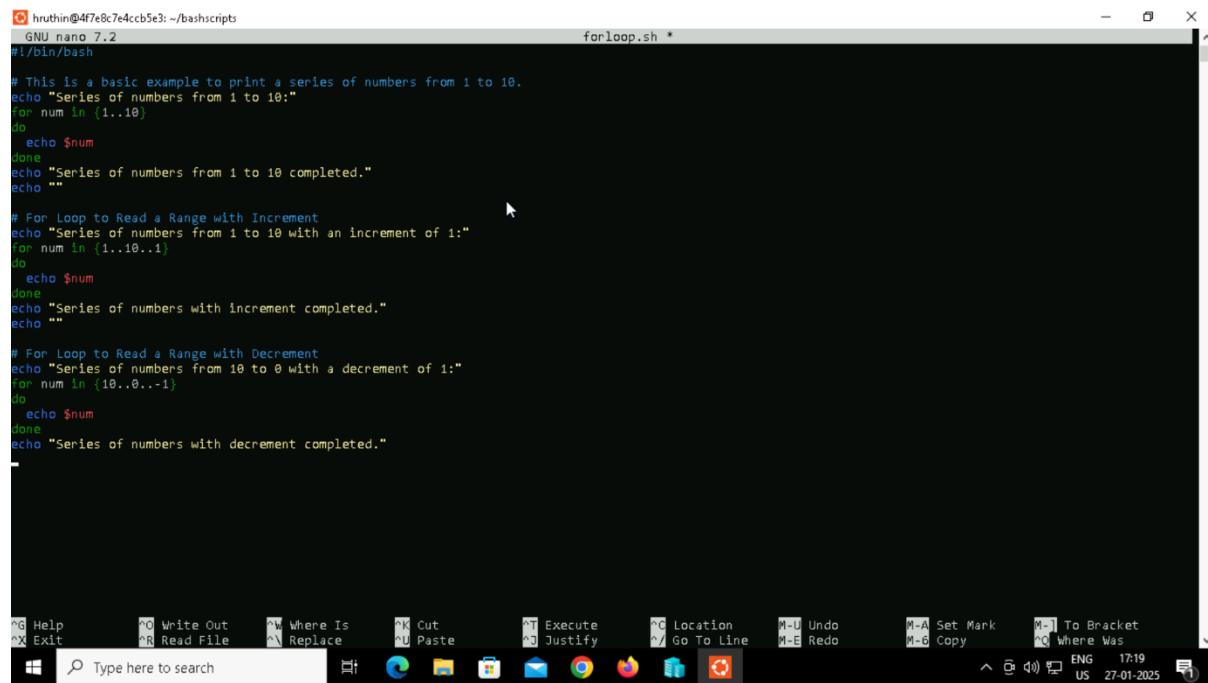
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

For loop:

Example 1:

For Loop to Read a Range (includes incrementing and decrementing):

Code:



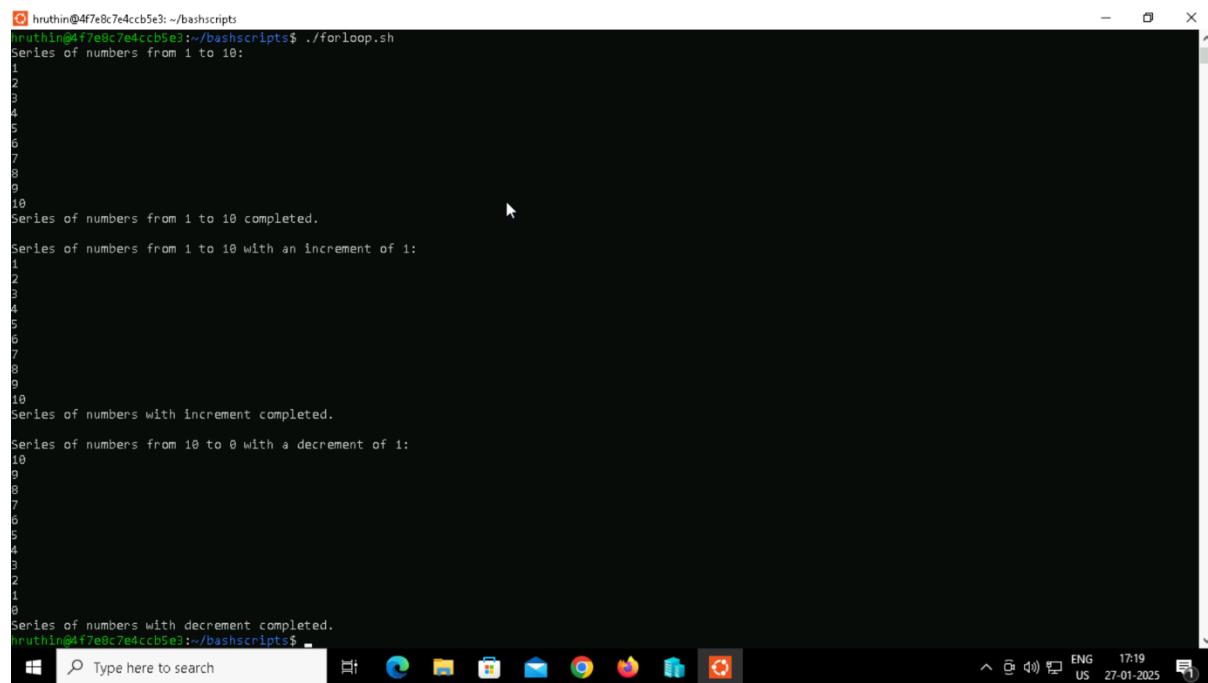
```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/bash

# This is a basic example to print a series of numbers from 1 to 10.
echo "Series of numbers from 1 to 10:"
for num in {1..10}
do
echo $num
done
echo "Series of numbers from 1 to 10 completed."
echo ""

# For Loop to Read a Range with Increment
echo "Series of numbers from 1 to 10 with an increment of 1:"
for num in {1..10..1}
do
echo $num
done
echo "Series of numbers with increment completed."
echo ""

# For Loop to Read a Range with Decrement
echo "Series of numbers from 10 to 0 with a decrement of 1:"
for num in {10..0..-1}
do
echo $num
done
echo "Series of numbers with decrement completed."
```

Output :



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./forloop.sh
Series of numbers from 1 to 10:
1
2
3
4
5
6
7
8
9
10
Series of numbers from 1 to 10 completed.

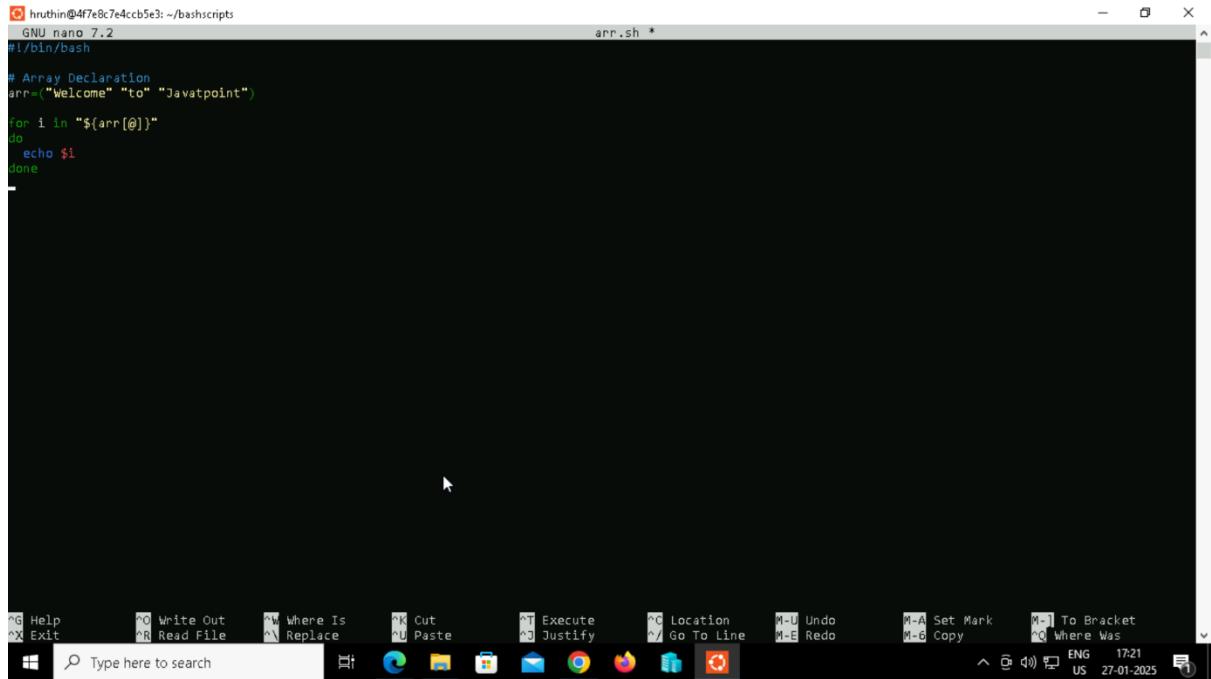
Series of numbers from 1 to 10 with an increment of 1:
1
2
3
4
5
6
7
8
9
10
Series of numbers with increment completed.

Series of numbers from 10 to 0 with a decrement of 1:
10
9
8
7
6
5
4
3
2
1
0
Series of numbers with decrement completed.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 2:

For Loop to Read Array Variables:

Code :

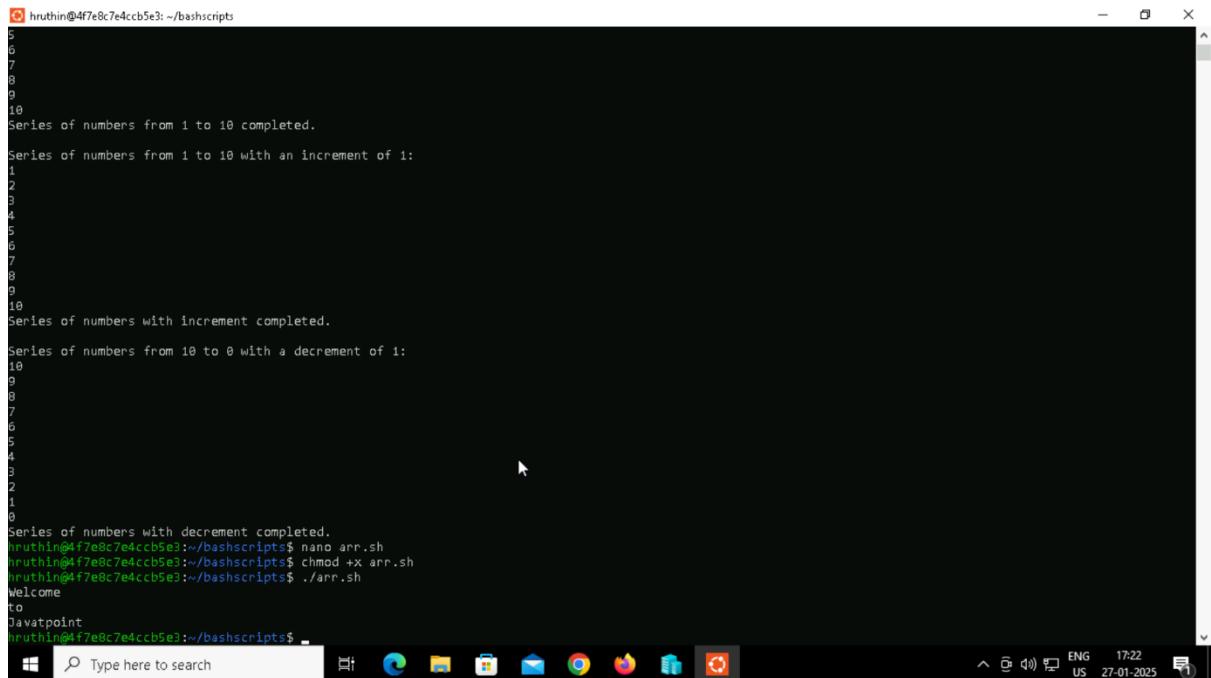


```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/bash

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

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

Output :



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
5
6
7
8
9
10
Series of numbers from 1 to 10 completed.
Series of numbers from 1 to 10 with an increment of 1:
1
2
3
4
5
6
7
8
9
10
Series of numbers with increment completed.
Series of numbers from 10 to 0 with a decrement of 1:
10
9
8
7
6
5
4
3
2
1
0
Series of numbers with decrement completed.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr.sh
Welcome
to
Javatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 3:

For Loop with a Break Statement A 'break' statement can be used inside 'for' loop to terminate from the loop.

Code:

The screenshot shows a terminal window titled "break.sh *". The code inside is:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/bash

# Table of 2
for table in {..100..2}
do
    echo $table
    if [ $table == 20 ]; then
        break
    fi
done
```

The terminal window has a menu bar with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, To Bracket, Copy, Where Was, and a search bar at the bottom. The system tray shows the date and time as 27-01-2025.

Output:

The screenshot shows a terminal window with the following output:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
5
6
7
8
9
10
Series of numbers with increment completed.

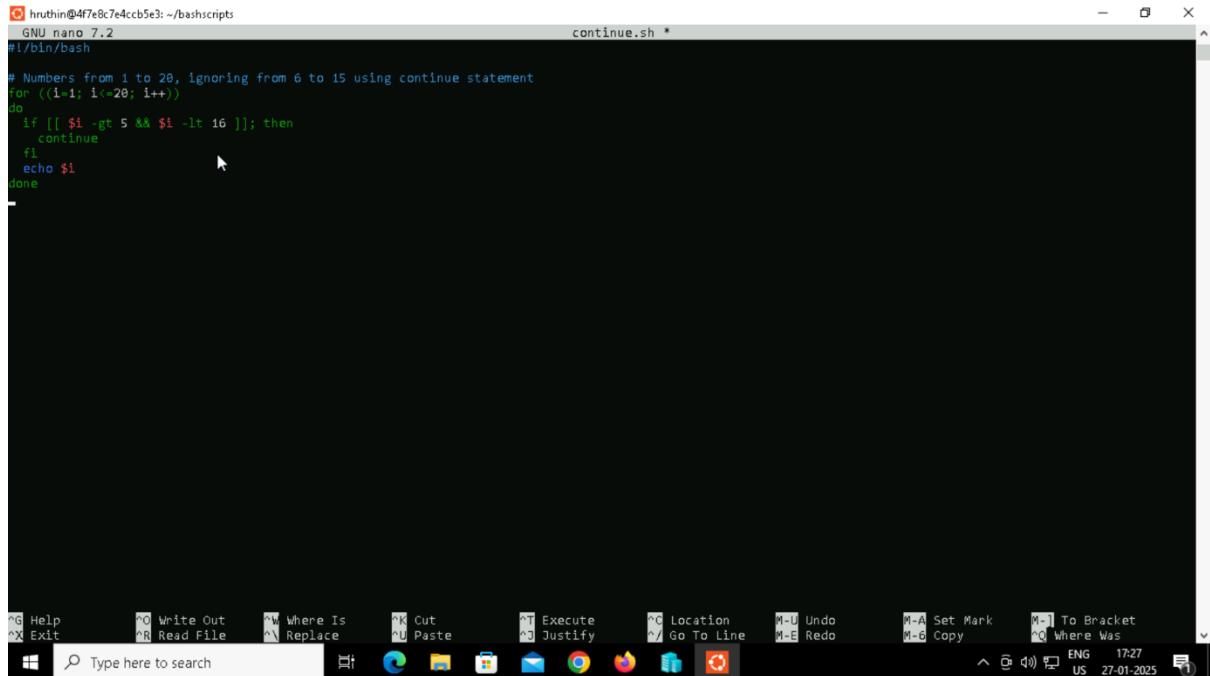
Series of numbers from 10 to 0 with a decrement of 1:
10
9
8
7
6
5
4
3
2
1
0
Series of numbers with decrement completed.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr.sh
Welcome
to
Davatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano break.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x break.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./break.sh
2
4
6
8
10
12
14
16
18
20
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

The terminal window has a menu bar with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, To Bracket, Copy, Where Was, and a search bar at the bottom. The system tray shows the date and time as 27-01-2025.

Example 4:

For Loop with a Continue Statement

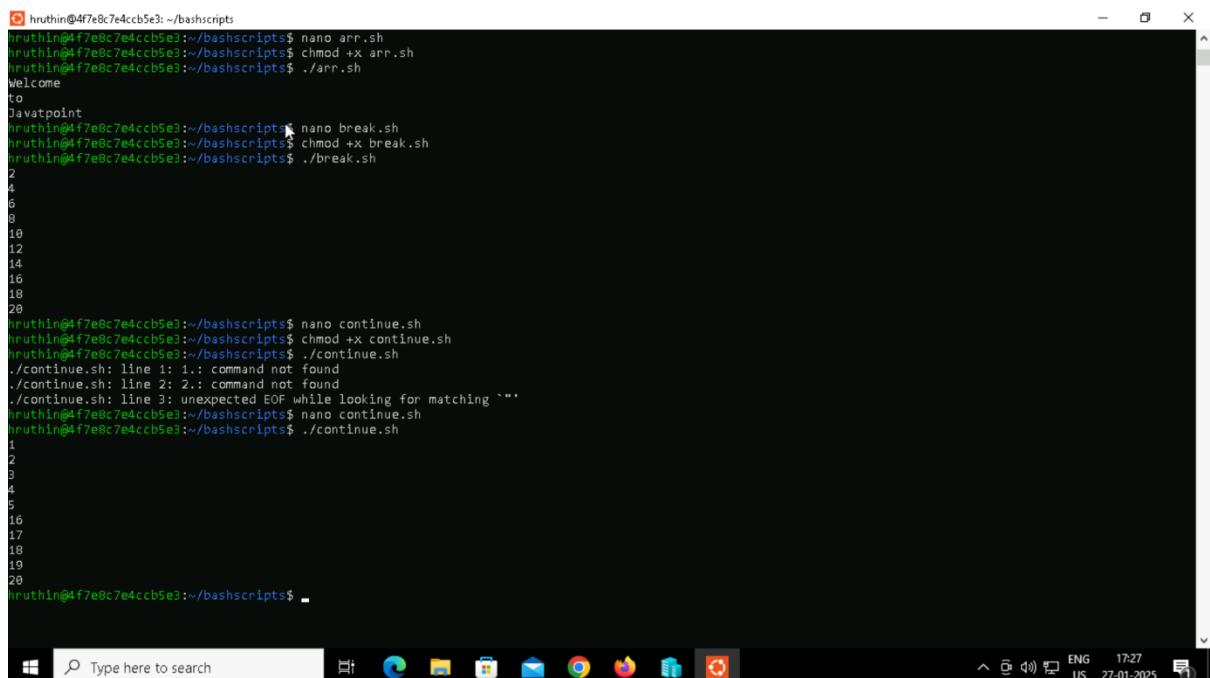
Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
$ nano nano.sh
GNU nano 7.2
#!/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
```

Output:

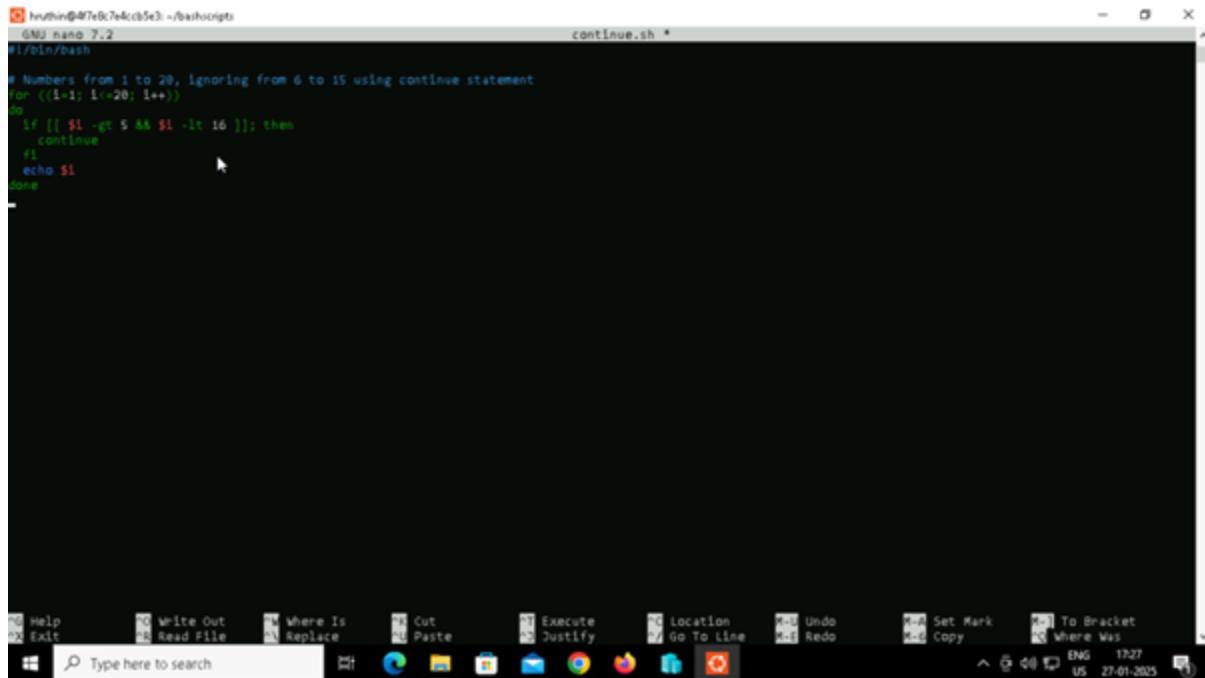


```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
$ nano arr.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr.sh
Welcome
to
Davatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano break.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x break.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./break.sh
2
4
6
8
10
12
14
16
18
20
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano continue.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x continue.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./continue.sh
./continue.sh: line 1: 1.: command not found
./continue.sh: line 2: 2.: command not found
./continue.sh: line 3: unexpected EOF while looking for matching `'''
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano continue.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./continue.sh
1
2
3
4
5
6
16
17
18
19
20
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 5:

Infinite Bash For Loop:

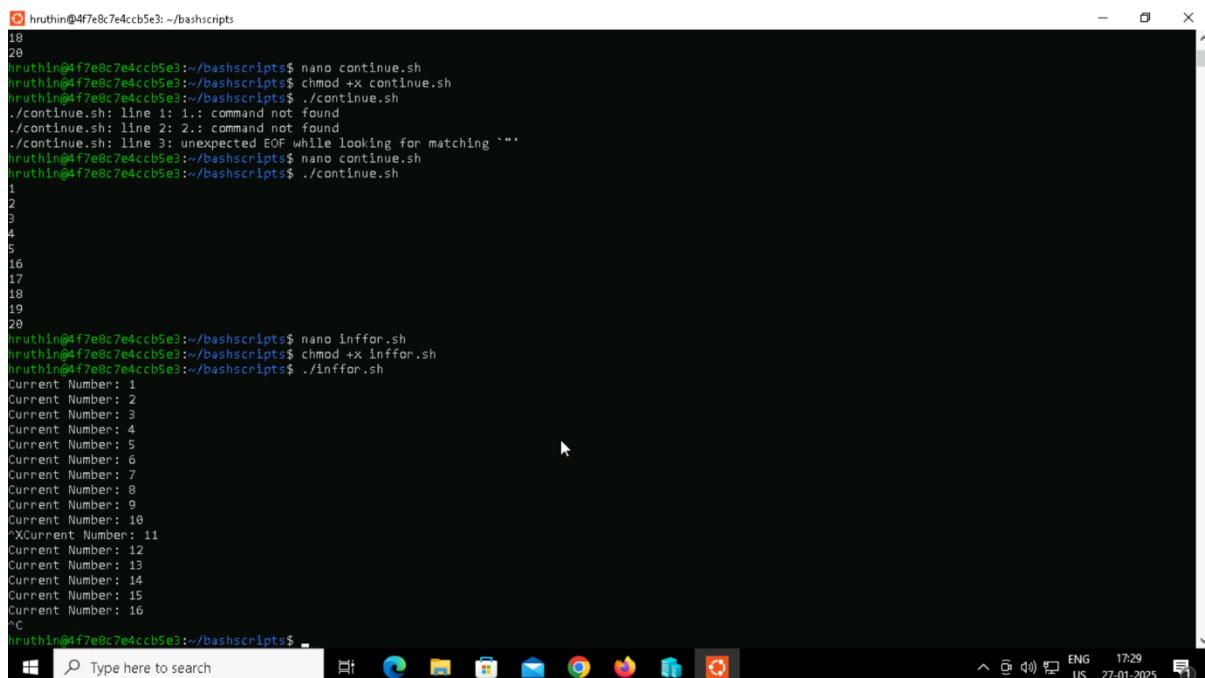
Code:



```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
GNU nano 2.2                               continue.sh *

# 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
```

Output :



```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
18
20
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano continue.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x continue.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./continue.sh
./continue.sh: line 1: 1:: command not found
./continue.sh: line 2: 2:: command not found
./continue.sh: line 3: unexpected EOF while looking for matching `'''
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano continue.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./continue.sh
1
2
3
4
5
16
17
18
19
20
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano inffor.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x inffor.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./inffor.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
^XCurrent Number: 11
Current Number: 12
Current Number: 13
Current Number: 14
Current Number: 15
Current Number: 16
^C
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Bash While:

Example 1:

In this example, the while loop is used with a single condition in expression. It is the basic example of while loop which will print series of numbers as per user input:

Code :

The screenshot shows a terminal window titled 'while1.sh *'. The script content is as follows:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 2.2
#!/bin/bash
# Script to get specified numbers

read -p "Enter starting number: " snum
read -p "Enter ending number: " enum

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

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

The terminal window has a menu bar at the top with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Undo, Redo, Set Mark, To Bracket, Where Was, and a search bar below it. At the bottom, there's a taskbar with icons for various applications and a system tray showing the date and time.

Output:

The screenshot shows a terminal window on an Ubuntu 24.04 LTS system. The output of the script is displayed:

```
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 5.15.167.4-microsoft-standard-WSL2 x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Tue Jan 28 04:05:31 UTC 2025

System load: 0.65      Processes:          72
Usage of /: 0.2% of 1006.85GB   Users logged in:     0
Memory usage: 5%        IPv4 address for eth0: 172.30.10.97
Swap usage: 0%         

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.
https://ubuntu.com/engage/secure-kubernetes-at-the-edge

This message is shown once a day. To disable it please create the
/home/hruthin/.hushlogin file.
hruthin@4f7e8c7e4ccb5e3:~$ cd bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano while1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x while1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./while1.sh
Enter starting number: 40
Enter ending number: 50
40
41
42
43
44
45
46
47
48
49
50
This is the sequence that you wanted.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

The terminal window has a menu bar at the top with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Undo, Redo, Set Mark, To Bracket, Where Was, and a search bar below it. At the bottom, there's a taskbar with icons for various applications and a system tray showing the date and time.

Example2:

While Loop with Multiple Conditions

Code:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/bash
# Script to get specified numbers

read -p "Enter starting number: " snum
read -p "Enter ending number: " enum

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

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

Output:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:~$ cd bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano while1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x while1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./while1.sh
Enter starting number: 40
Enter ending number: 50
40
41
42
43
44
45
46
47
48
49
50
This is the sequence that you wanted.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano while2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x while2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./while2.sh
Enter starting number: 20
Enter ending number: 35
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
This is the sequence that you wanted.
```

Example3:

Infinite Loop:

```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
  GNU nano 7.2                                         while3.sh *
#!/bin/bash
# An infinite while loop

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

Output:

Example 4:

Infinite loop 2:

Code:

```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
GNU nano 7.2                                         while4.sh *
#!/bin/bash
# An infinite while loop

while true
do
    echo "Welcome to Javatpoint"
done
```

Output:

Example 5:

While Loop with a Break Statement:

Code:

```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
$ nano 7.2
GNU nano 7.2
#!/bin/bash
# While Loop Example with a Break Statement

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

Output:

Example 6:

While Loop with a Continue Statement:

Code:

```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
GNU nano 7.2                                         while6.sh *
#!/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."
```

Output:

Until Loop:

Example 1:

Until Loop with Single Condition :

Code:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/bash
# Bash Until Loop example with a single condition

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

Output:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano until1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x until1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./until1.sh
Countdown for Website Launching...
10
9
8
7
6
5
4
3
2
1
Skipped number 5 using Continue Statement.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano while5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x while5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./while5.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
Mission Aborted, Some Technical Error Found.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano while6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x while6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./while6.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.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano until1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x until1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./until1.sh
1
2
3
4
5
6
7
8
9
10
```

Example 2:

Until Loop with Multiple Conditions:

Code:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
$ nano 7.2
#!/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
```

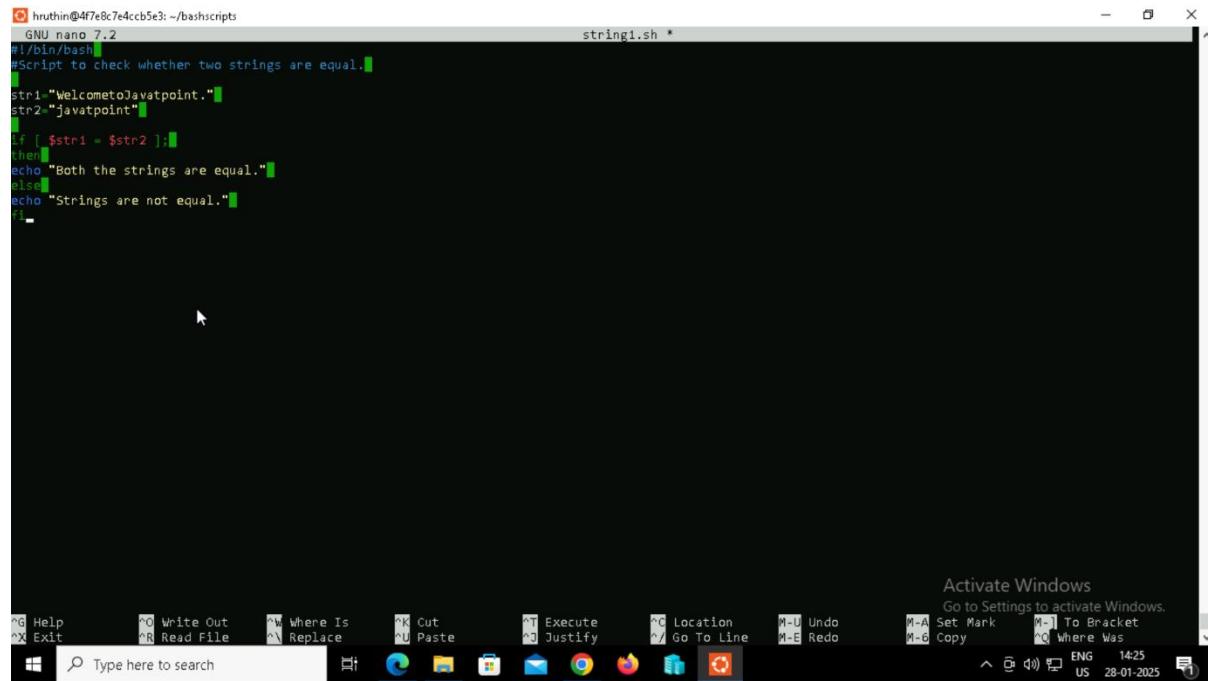
Output:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
$ Mission Aborted, Some Technical Error Found.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano while6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x while6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./while6.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.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano until1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x until1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./until1.sh
1
2
3
4
5
6
7
8
9
10
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano until2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x until2.sh
chmod: cannot access 'until2.ah': No such file or directory
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x until2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./until2.sh
a = 1 & b = 0,
a = 2 & b = 1,
a = 3 & b = 2,
a = 4 & b = 3,
a = 5 & b = 4,
```

Strings:

Example 1: An equal operator (=) is used to check whether two strings are equal.

Code:

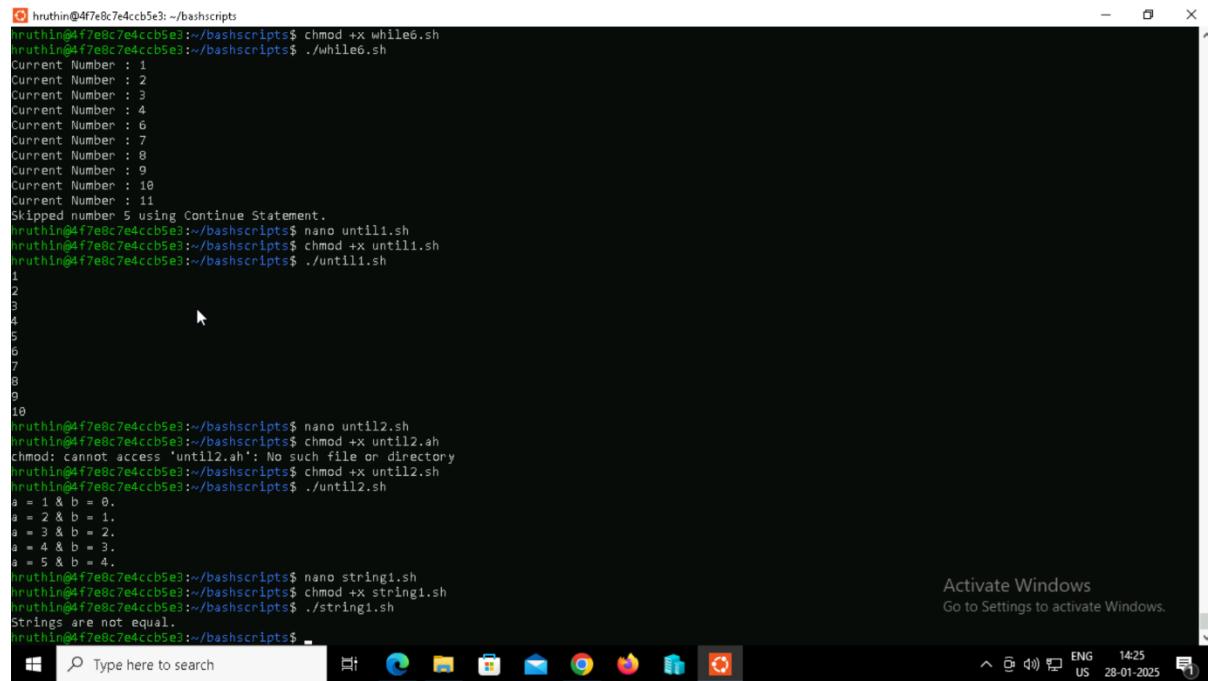


```
GNU nano 7.2                               string1.sh *
#!/bin/bash
#Script to check whether two strings are equal.

str1="WelcometoJavaPoint."
str2="javatpoint"

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

Output:

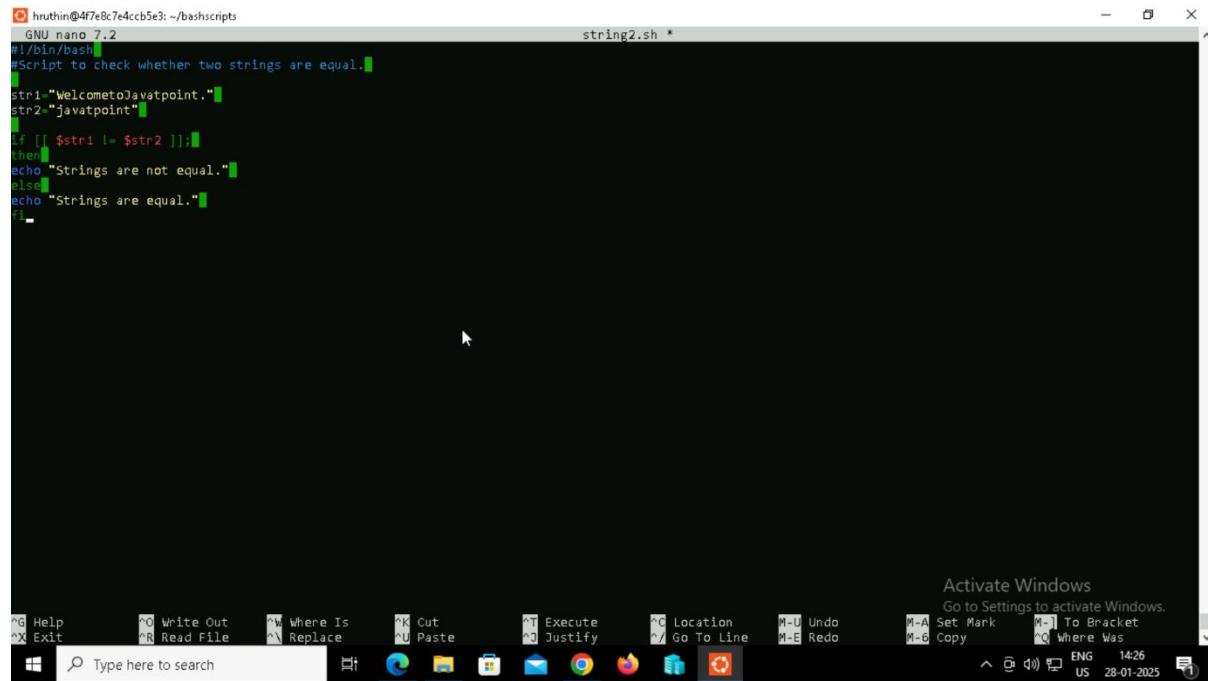


```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x while6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./while6.sh
Current Number : 1
Current Number : 2
Current Number : 3
Current Number : 4
Current Number : 5
Current Number : 6
Current Number : 7
Current Number : 8
Current Number : 9
Current Number : 10
Current Number : 11
Skipped number 5 using Continue Statement.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano until11.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x until11.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./until11.sh
1
2
3
4
5
6
7
8
9
10
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano until2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x until2.sh
chmod: cannot access 'until2.sh': No such file or directory
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x until2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./until2.sh
a = 1 & b = 0.
a = 2 & b = 1.
a = 3 & b = 2.
a = 4 & b = 3.
a = 5 & b = 4.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string1.sh
Strings are not equal.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example2:

Not equal operator (!=) is used to define that strings are not equal.

Code:

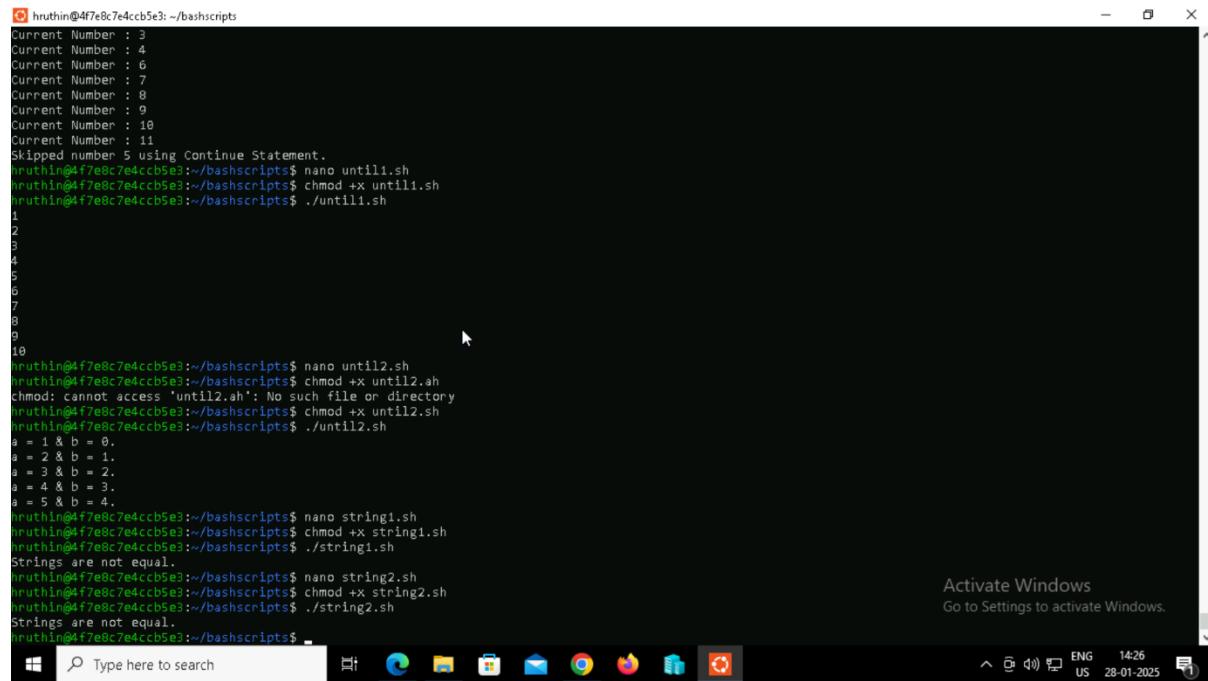


```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
$ nano string2.sh
GNU nano 7.2
#!/bin/bash
#Script to check whether two strings are equal.

str1="WelcometoJavaPoint."
str2="javatpoint"

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

Output:

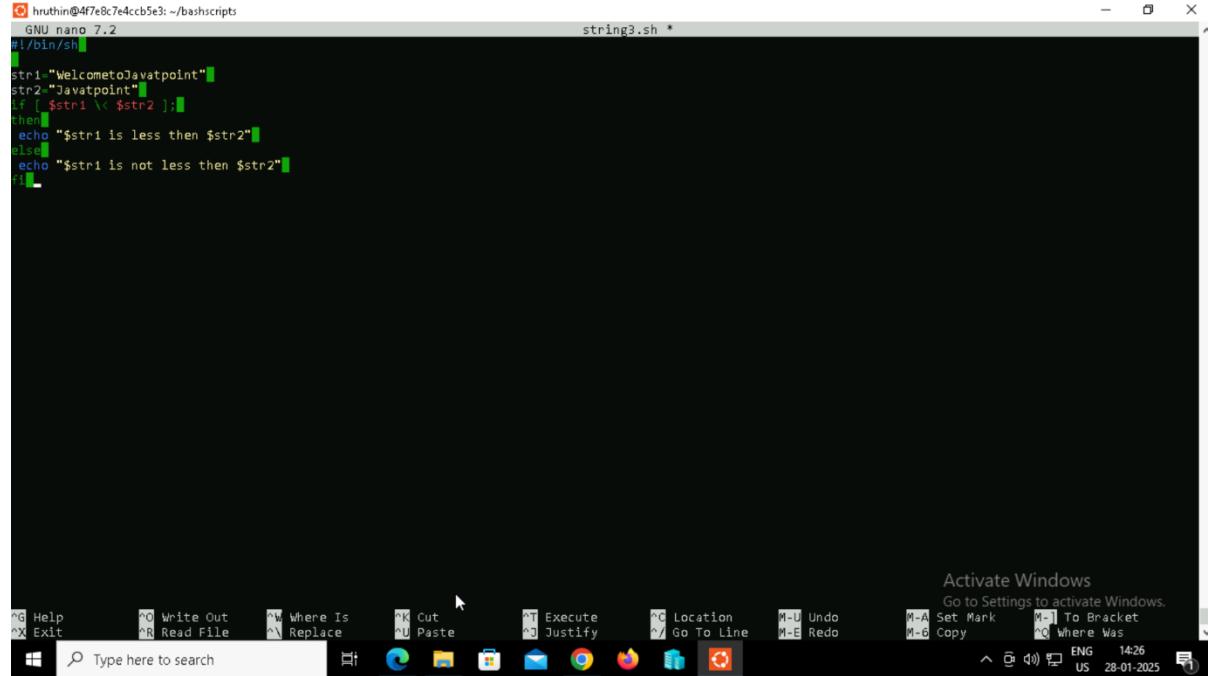


```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
$ ./string2.sh
Strings are not equal.
```

Example 3:

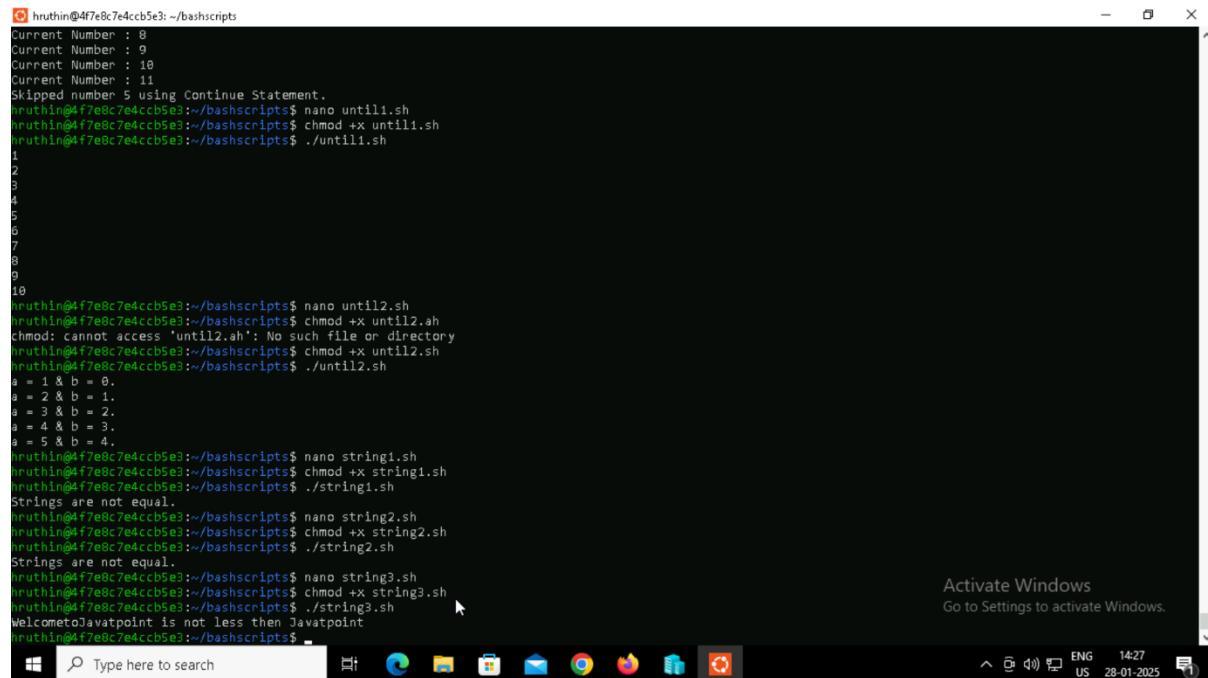
The less than operator

Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/sh
string3.sh *
str1 "Welcometolavatpoint"
str2 "Javatpoint"
if [ $str1 < $str2 ]; then
echo "$str1 is less than $str2"
else
echo "$str1 is not less than $str2"
fi
```

Output:

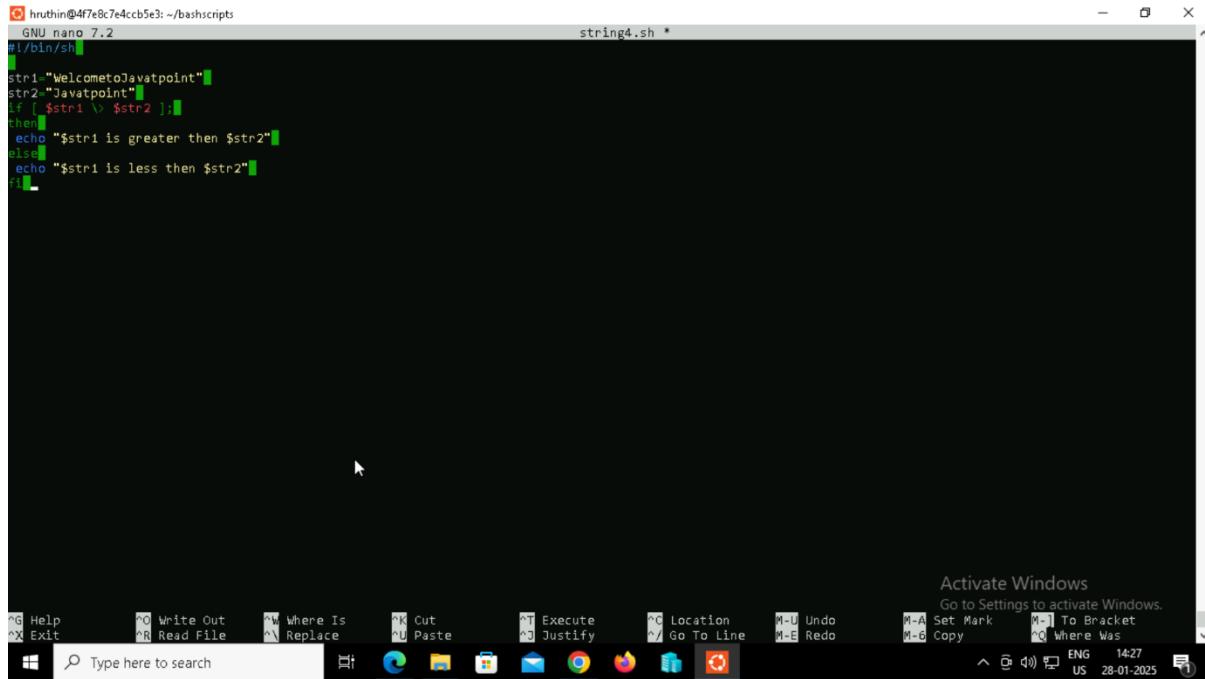


```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
Current Number : 8
Current Number : 9
Current Number : 10
Current Number : 11
Skipped number 5 using Continue Statement.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano until1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x until1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./until1.sh
1
2
3
4
5
6
7
8
9
10
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano until2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x until2.ah
chmod: cannot access 'until2.ah': No such file or directory
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x until2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./until2.sh
a = 1 & b = 0,
a = 2 & b = 1,
a = 3 & b = 2,
a = 4 & b = 3,
a = 5 & b = 4,
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string1.sh
Strings are not equal.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string2.sh
Strings are not equal.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string3.sh
Welcometolavatpoint is not less than Javatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 4:

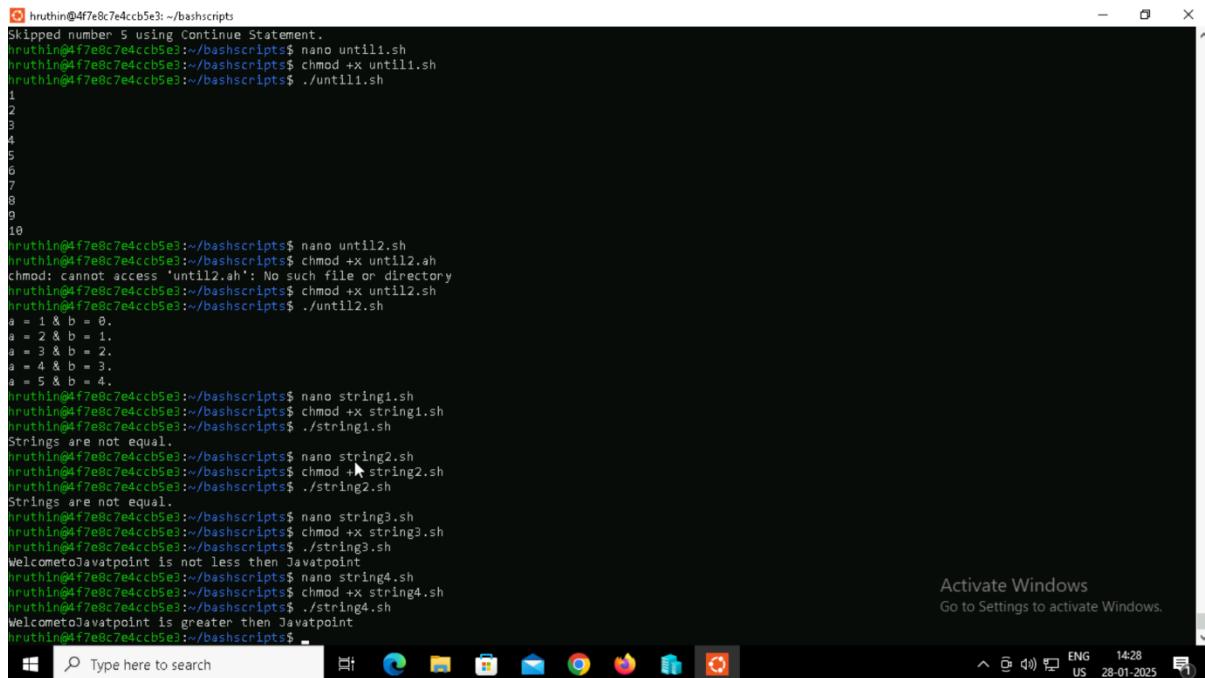
The 'greater than operator (>)' is used to check if string1 is greater than string2.

Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/sh
string4.sh *
str1 "WelcometoJavatpoint"
str2 "Javatpoint"
if [ $str1 > $str2 ]; then
echo "$str1 is greater than $str2"
else
echo "$str1 is less than $str2"
fi
```

Output:

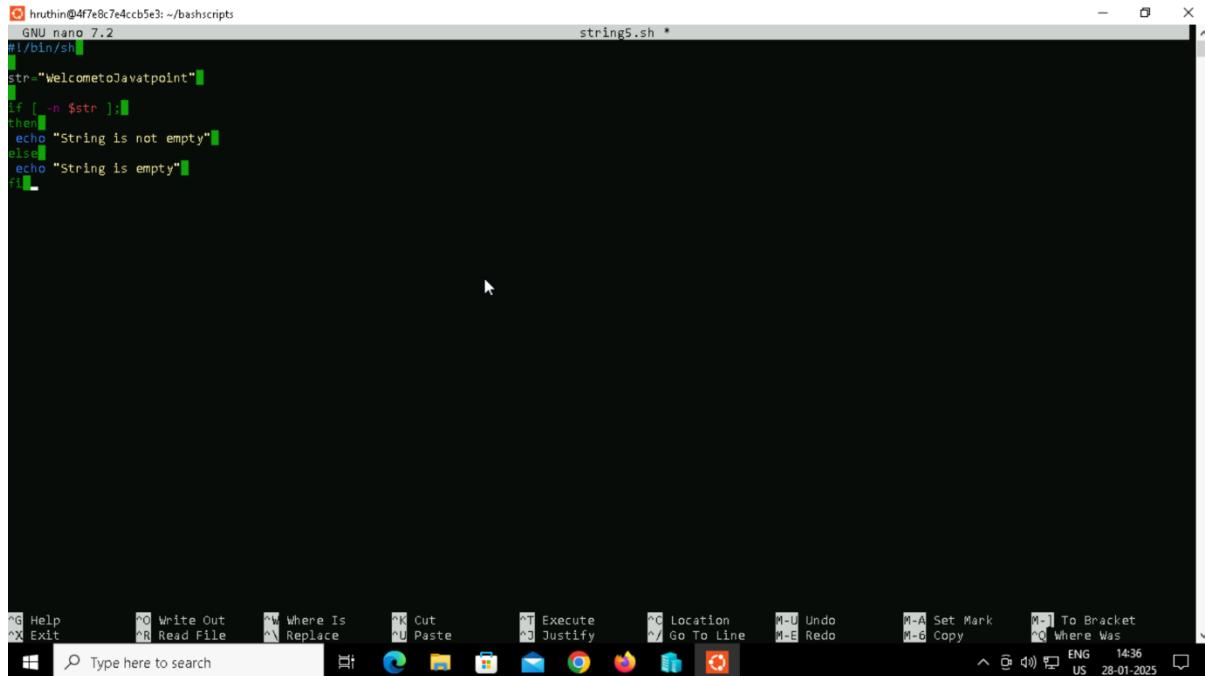


```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
Skipped number 5 using Continue Statement.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano untili.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x untili.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./untili.sh
1
2
3
4
5
6
7
8
9
10
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano until2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x until2.sh
chmod: cannot access 'until2.sh': No such file or directory
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x until2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./until2.sh
a = 1 & b = 0.
a = 2 & b = 1.
a = 3 & b = 2.
a = 4 & b = 3.
a = 5 & b = 4.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string1.sh
Strings are not equal.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string2.sh
Strings are not equal.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string3.sh
WelcometoJavatpoint is not less than Javatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string4.sh
WelcometoJavatpoint is greater than Javatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 5:

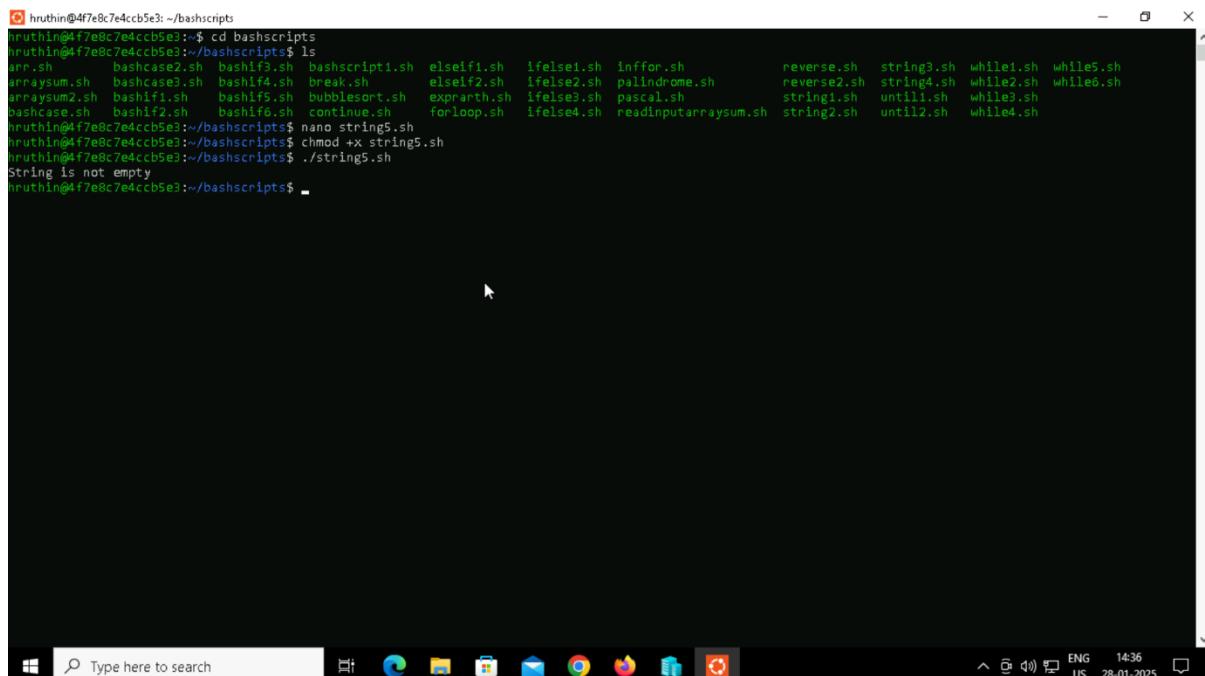
To check if the string length is greater than Zero:

Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/sh
str="WelcometoJavatpoint"
if [ -n $str ]; then
echo "String is not empty"
else
echo "String is empty"
fi
```

Output:

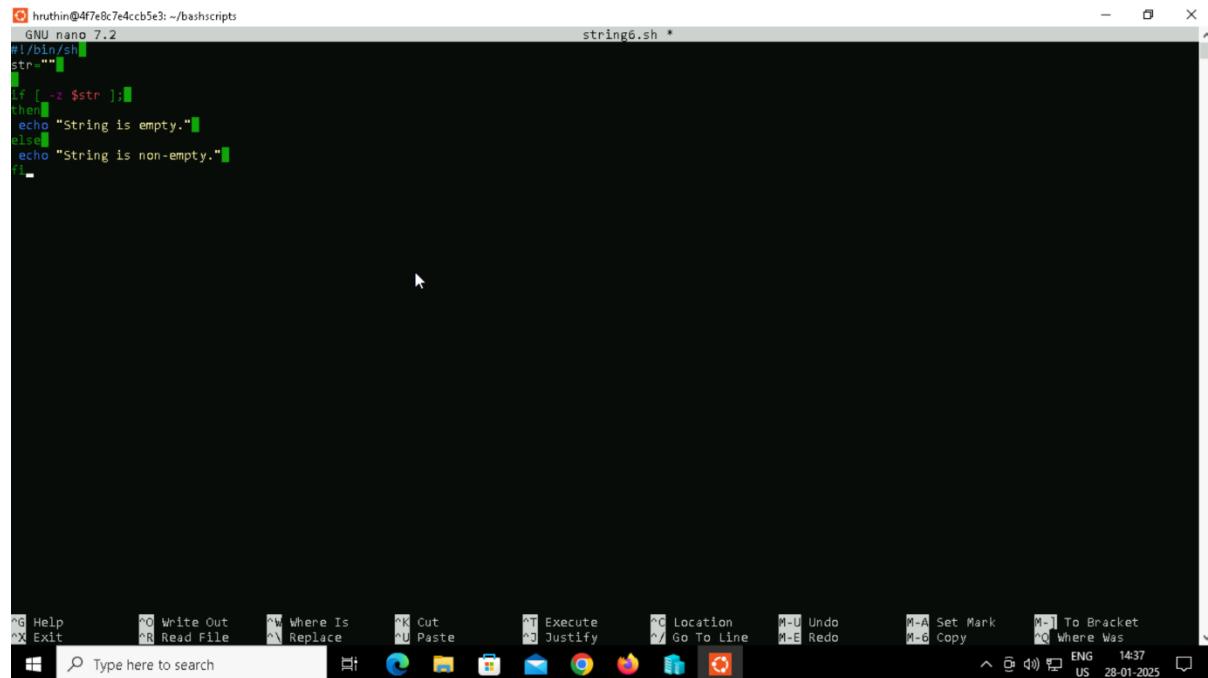


```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ cd bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ls
arr.sh           bashcase2.sh  bashif3.sh  bashscript1.sh  elseif1.sh   ifelse1.sh  inffor.sh      reverse.sh  string3.sh  while1.sh  while5.sh
arraysum.sh      bashcase3.sh  bashif4.sh  bashscript2.sh  elseif2.sh   ifelse2.sh  palindrome.sh  reverse2.sh  string4.sh  while2.sh  while6.sh
arraysum2.sh     bashif1.sh   bashif5.sh  break.sh       elseif3.sh   ifelse3.sh  pascal.sh     string1.sh  until1.sh  while3.sh
bashcase.sh      bashif2.sh   bashif6.sh  continue.sh    forloop.sh   ifelse4.sh  readingarraysum.sh  string2.sh  until2.sh  while4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string5.sh
String is not empty
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 6:

To check if the string length is equal to Zero

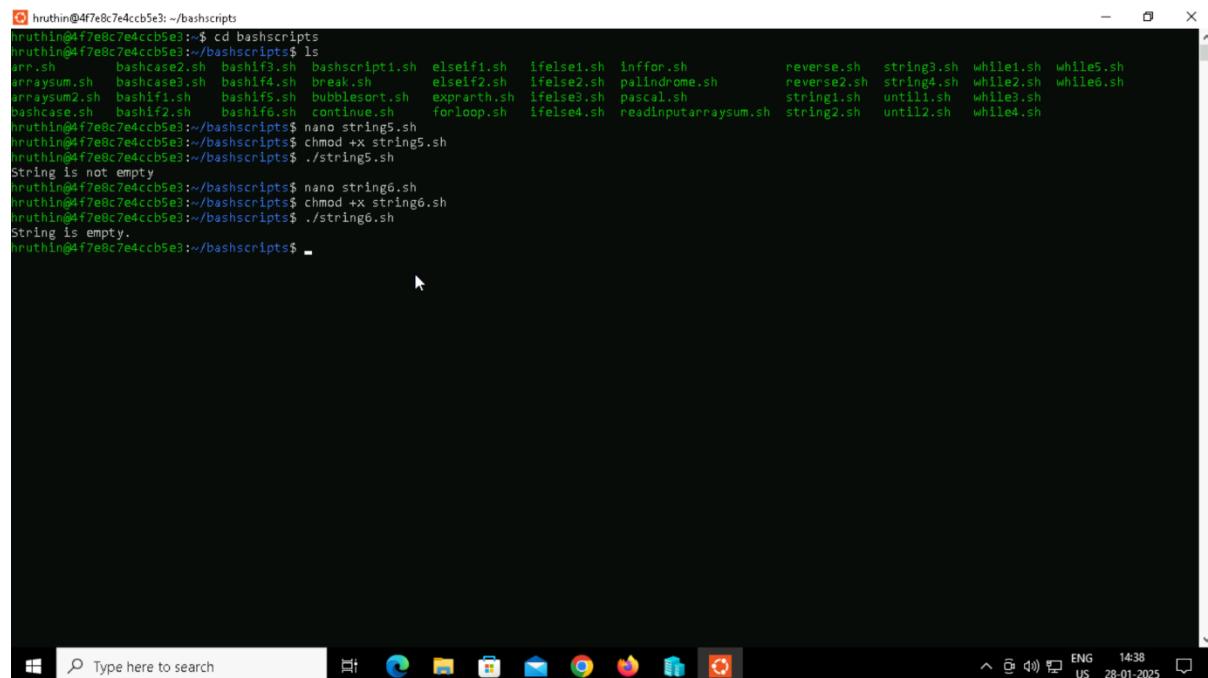
Code:



```
GNU nano 7.2                               string6.sh *
#!/bin/sh
str=""

if [ -z $str ]; then
    echo "String is empty."
else
    echo "String is non-empty."
fi
```

Output:

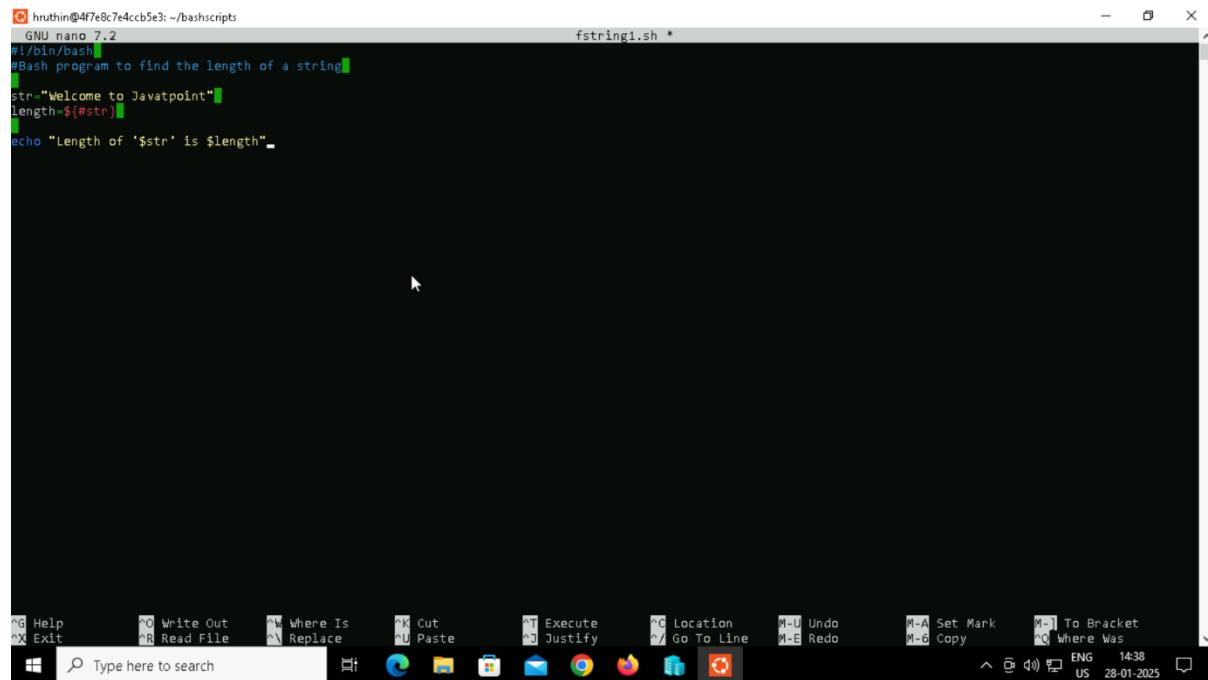


```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ cd bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ls
arr.sh          bashcase2.sh  bashif3.sh  elseif1.sh  ifelse1.sh  inffor.sh      reverse.sh  string3.sh  while1.sh  while5.sh
arraysum.sh     bashcase3.sh  bashif4.sh  break.sh   elseif2.sh  ifelse2.sh  palindrome.sh  reverse2.sh  string4.sh  while2.sh  while6.sh
arraysum2.sh    bashif1.sh   bashif5.sh  bubblesort.sh elseif3.sh  ifelse3.sh  pascal.sh   string1.sh  until1.sh  while3.sh
bashcase.sh     bashif2.sh   bashif6.sh  continue.sh  forloop.sh  ifelse4.sh  readingarraysum.sh string2.sh  until2.sh  while4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string5.sh
String is not empty
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string6.sh
String is empty.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Find String:

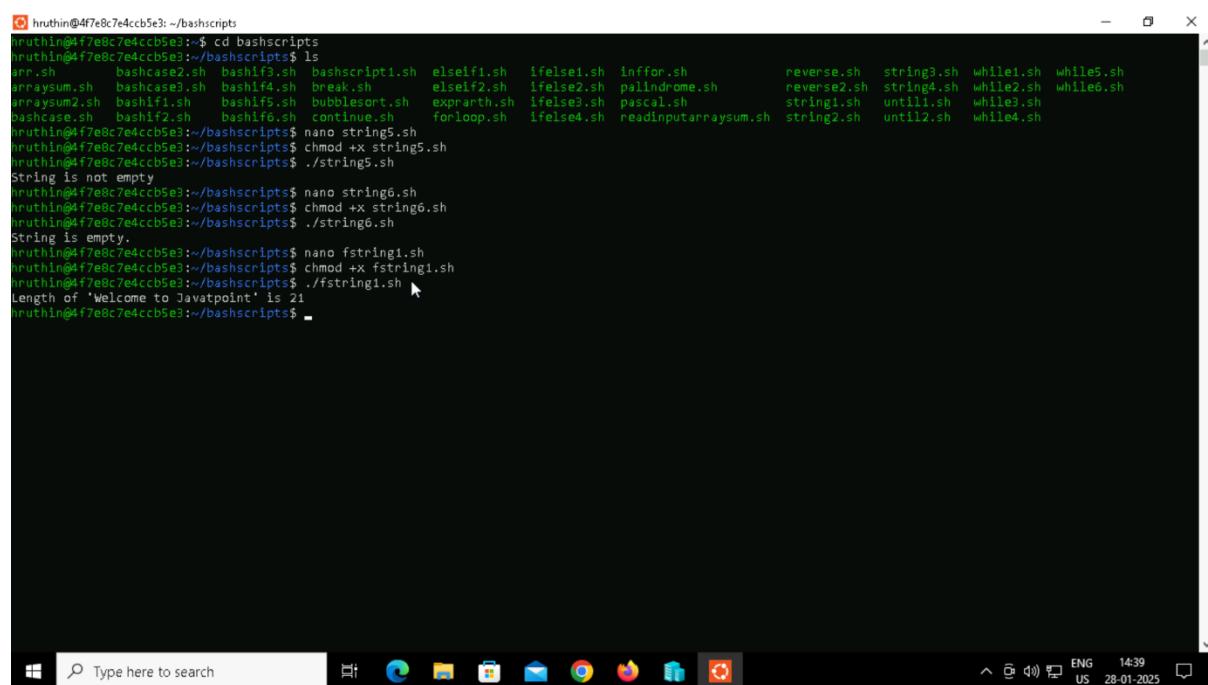
Example 1 The simplest way to calculate the length of a string is to use '#' symbol. In this example, we have used \${#string_variable_name} to find the length of a string.

Code:



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

Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ls
arr.sh           bashcase2.sh  bashif3.sh  bashscript1.sh  elseif1.sh  ifelse1.sh  inffor.sh      reverse.sh  string3.sh  while1.sh  while5.sh
arraysum.sh      bashcase3.sh  bashif4.sh  bashscript2.sh  elseif2.sh  ifelse2.sh  palindrome.sh  reverse2.sh  string4.sh  while2.sh  while6.sh
arraysum2.sh     bashif1.sh   bashif5.sh  bashscript3.sh  elseif3.sh  ifelse3.sh  expranth.sh   string1.sh  until1.sh  while3.sh
bashcase.sh      bashif2.sh   bashif6.sh  continue.sh    forloop.sh   ifelse4.sh  pascal.sh    string2.sh  until2.sh  while4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string5.sh
String is not empty.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string6.sh
String is empty.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fstring1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fstring1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fstring1.sh
Length of 'Welcome to Javatpoint' is 21
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 2 Another way to calculate the length of a string is to use `expr` command with the 'length' keyword. In this example, we have used `expr length "\$str"` to find the length of a string.

Code:

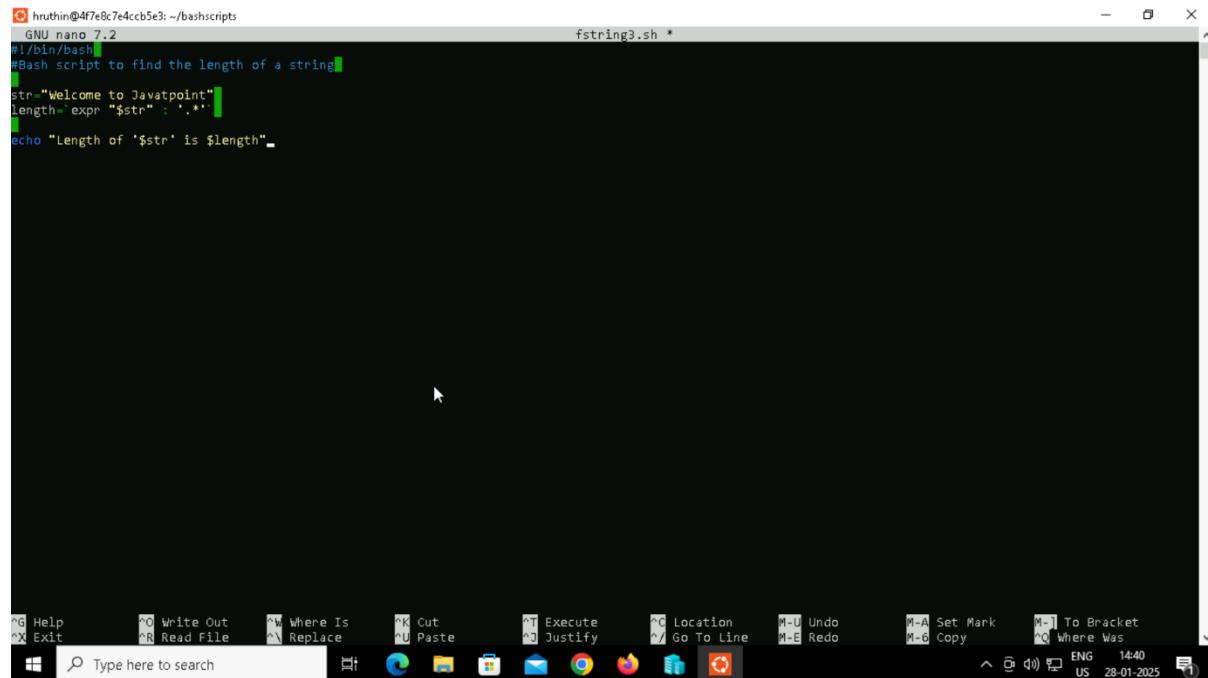
```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/bash
#Bash script to find the length of a string
str="Welcome to Javatpoint"
length= `expr length "$str"`
echo "Length of '$str' is $length"
```

Output:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ ls
array.sh      bashcase2.sh  bashif3.sh  bashscript1.sh  elseif1.sh  ifelse1.sh  inffor.sh      reverse.sh  string3.sh  while1.sh  while5.sh
arraysum.sh   bashcase3.sh  bashif4.sh  bashscript2.sh  elseif2.sh  ifelse2.sh  palindrome.sh  reverse2.sh  string4.sh  while2.sh  while6.sh
arraysum2.sh  bashif1.sh   bashif5.sh  bubblesort.sh  elseif3.sh  ifelse3.sh  pascal.sh    string1.sh  until1.sh  while3.sh
bashcase.sh   bashif2.sh   bashif6.sh  continue.sh   elseif4.sh  ifelse4.sh  readinputarraysum.sh string2.sh  until2.sh  while4.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ nano string5.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ chmod +x string5.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ ./string5.sh
String is not empty.
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ nano string6.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ chmod +x string6.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ ./string6.sh
String is empty.
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ nano fstring1.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ chmod +x fstring1.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ ./fstring1.sh
Length of 'Welcome to Javatpoint' is 21
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ nano fstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ chmod +x fstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ ./fstring2.sh
Length of 'Welcome to Javatpoint' is 21
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$
```

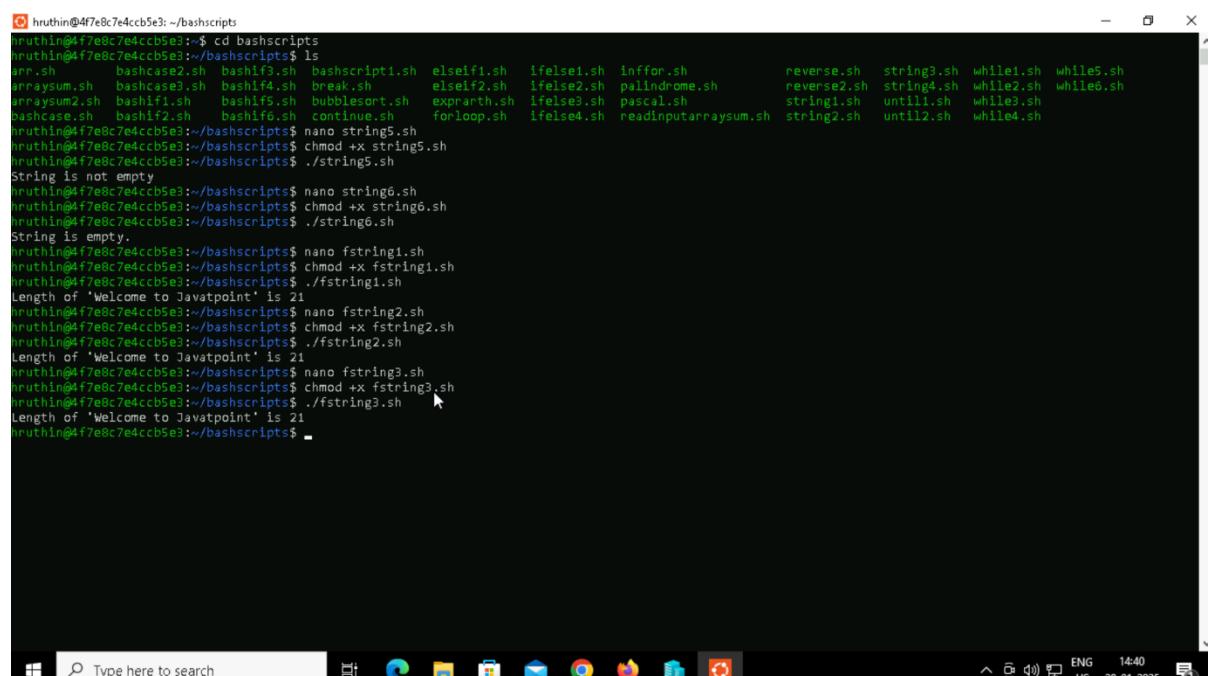
Example 3 In this example, we have used `expr "\$str": '.'` to find the length of a string. Here, str is a string variable.

Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2                               fstring3.sh *
#!/bin/bash
#Bash script to find the length of a string
str="Welcome to Javatpoint"
length= expr "$str": '.'
echo "Length of '$str' is $length"
```

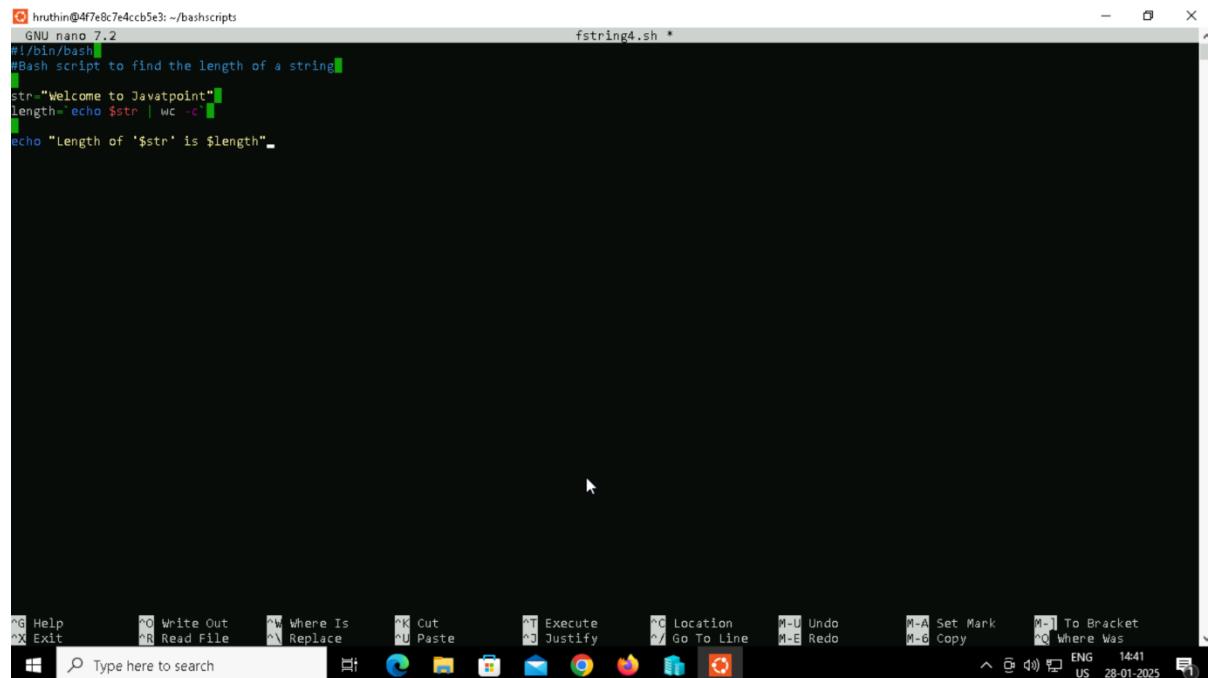
Output:



```
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ cd bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ls
arr.sh          bashcase2.sh  bashif3.sh  bashscript1.sh  elseif1.sh  ifelse1.sh  inffor.sh      reverse.sh  string3.sh  while1.sh  while5.sh
arraysum.sh     bashcase3.sh  bashif4.sh  break.sh       elseif2.sh  ifelse2.sh  palindrome.sh  reverse2.sh  string4.sh  while2.sh  while6.sh
arraysum2.sh    bashif1.sh   bashif5.sh  bubblesort.sh  exprarth.sh  ifelse3.sh  pascal.sh     string1.sh  until1.sh  while3.sh
bashcase.sh     bashif2.sh   bashif6.sh  continue.sh   forloop.sh   ifelse4.sh  readinputarraysum.sh  string2.sh  until2.sh  while4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string5.sh
String is not empty
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string6.sh
String is empty.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fstring1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fstring1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fstring1.sh
Length of 'Welcome to Javatpoint' is 21
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fstring2.sh
Length of 'Welcome to Javatpoint' is 21
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fstring3.sh
Length of 'Welcome to Javatpoint' is 21
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

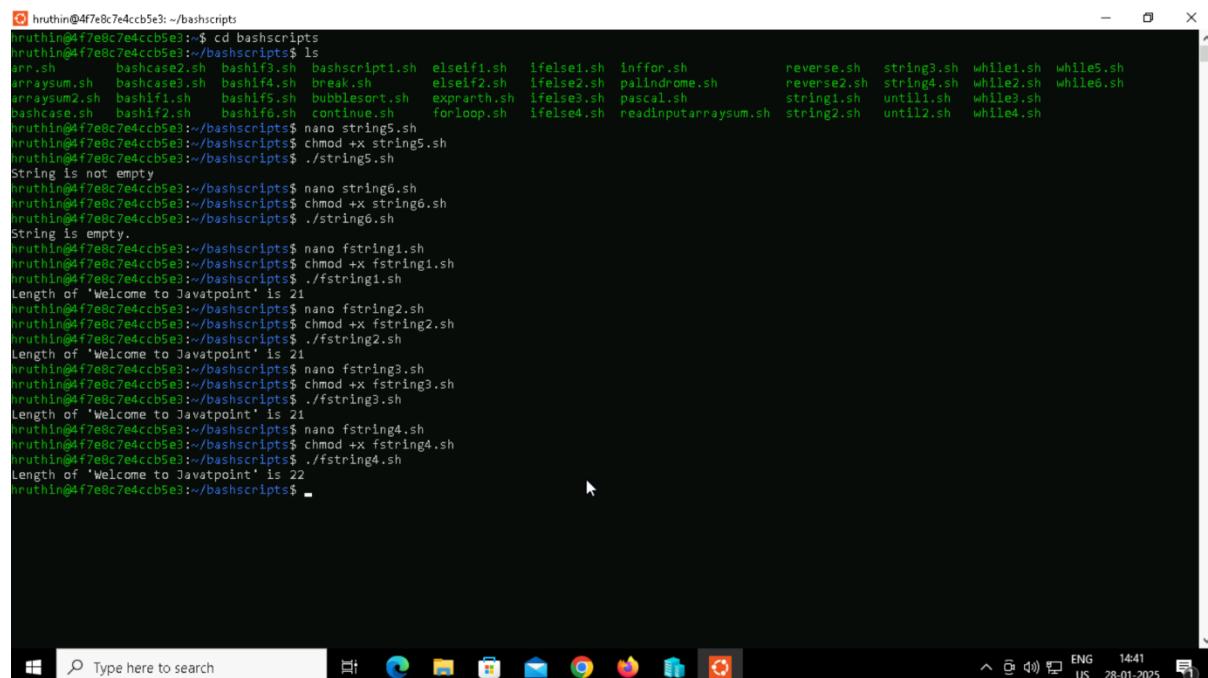
Example 4 In this example, we have used `wc` command to find the length of a string.

Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/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"
```

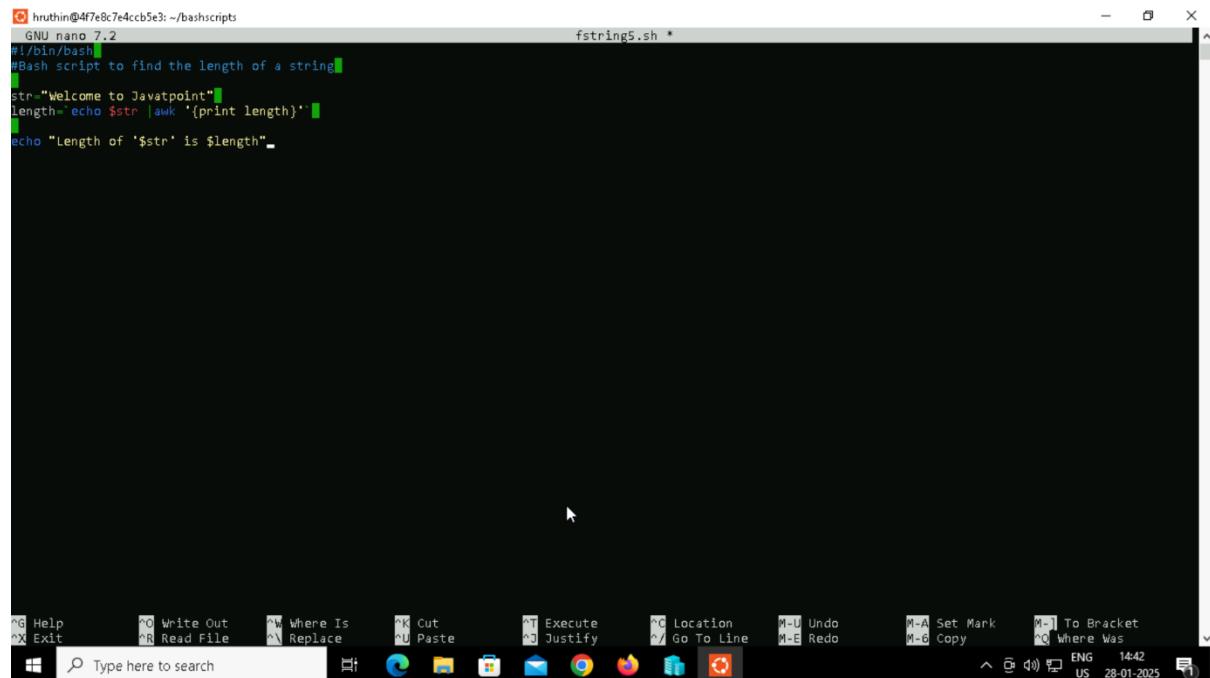
Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ls
arr.sh           bashccase2.sh  bashif3.sh  bashscript1.sh  elseif1.sh  ifelse1.sh  inffor.sh      reverse.sh   string3.sh  while1.sh  while5.sh
arraysum.sh      bashcase3.sh  bashif4.sh  bashscripti.sh  elseif2.sh  ifelse2.sh  palindrome.sh  reverse2.sh  string4.sh  while2.sh  while6.sh
arraysum2.sh     bashif1.sh   bashif5.sh  bubblesort.sh  elseif3.sh  ifelse3.sh  expranh.sh    reverse3.sh  string5.sh  while3.sh  while7.sh
bashccase.sh     bashif2.sh   bashif6.sh  continue.sh   forloop.sh  ifelse4.sh  pascal.sh     string1.sh  until1.sh  while4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string5.sh
String is not empty.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string6.sh
String is empty.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fstring1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fstring1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fstring1.sh
Length of 'Welcome to Javatpoint' is 21
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fstring2.sh
Length of 'Welcome to Javatpoint' is 21
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fstring3.sh
Length of 'Welcome to Javatpoint' is 21
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fstring4.sh
Length of 'Welcome to Javatpoint' is 22
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

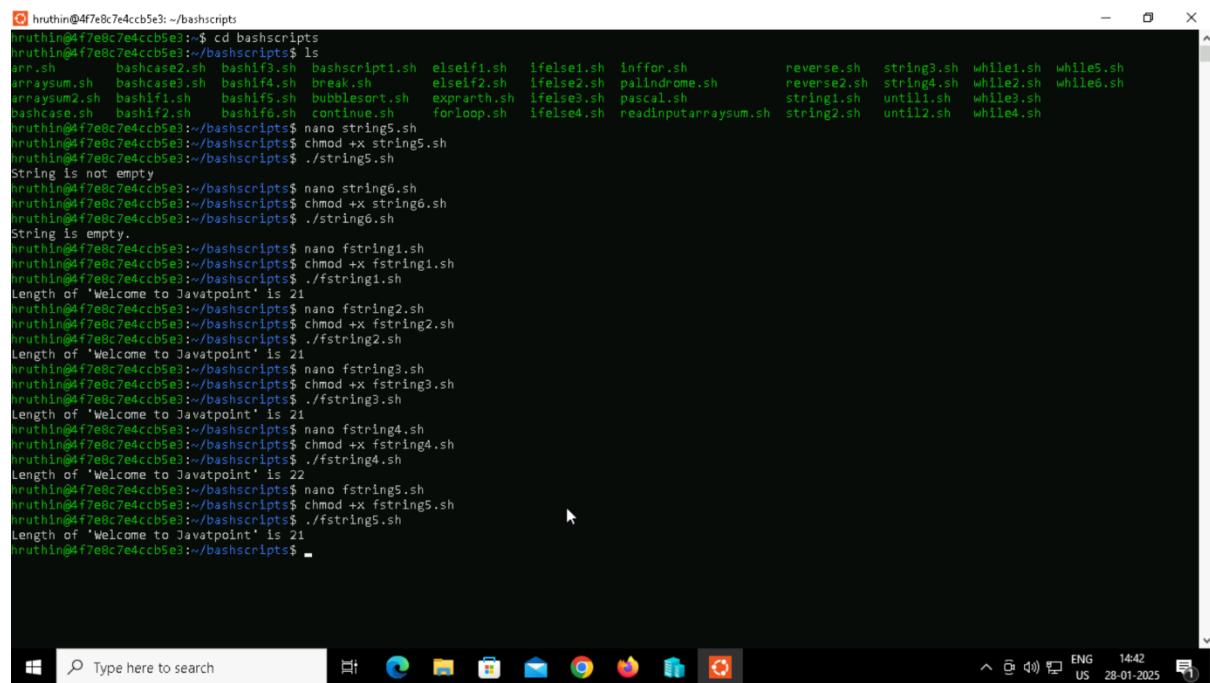
Example 5 In this example, we have used `awk` command to find the length of a string.

Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/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"-
```

Output:

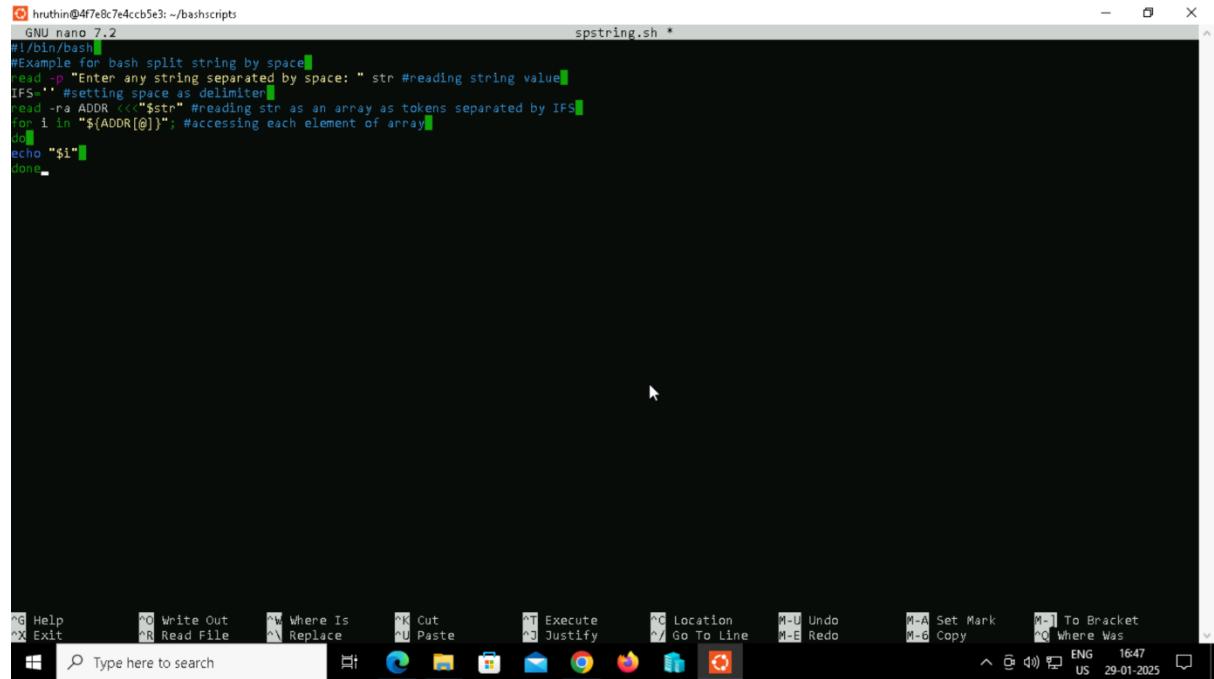


```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ls
arr.sh           bashif2.sh  bashif3.sh  bashscript1.sh  elseif1.sh  ifelse1.sh  inffor.sh      reverse.sh  string3.sh  while1.sh  while5.sh
arraysum.sh      bashcase2.sh  bashif4.sh  bashscripti.sh  elseif2.sh  ifelse2.sh  palindrome.sh  reverse2.sh  string4.sh  while2.sh  while6.sh
arraysum2.sh     bashif1.sh   bashif5.sh  bubblesort.sh  elseif3.sh  ifelse3.sh  expranh.sh    reverse3.sh  string5.sh  while3.sh
bashcase.sh      bashif2.sh   bashif6.sh  continue.sh   elseif4.sh  ifelse4.sh  ifelse4.sh    reverse4.sh  string6.sh  while4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string5.sh
String is not empty
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano string6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x string6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./string6.sh
String is empty.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fstring1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fstring1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fstring1.sh
Length of 'Welcome to Javatpoint' is 21
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fstring2.sh
Length of 'Welcome to Javatpoint' is 21
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fstring3.sh
Length of 'Welcome to Javatpoint' is 21
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fstring4.sh
Length of 'Welcome to Javatpoint' is 22
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fstring5.sh
Length of 'Welcome to Javatpoint' is 21
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ -
```

Split String:

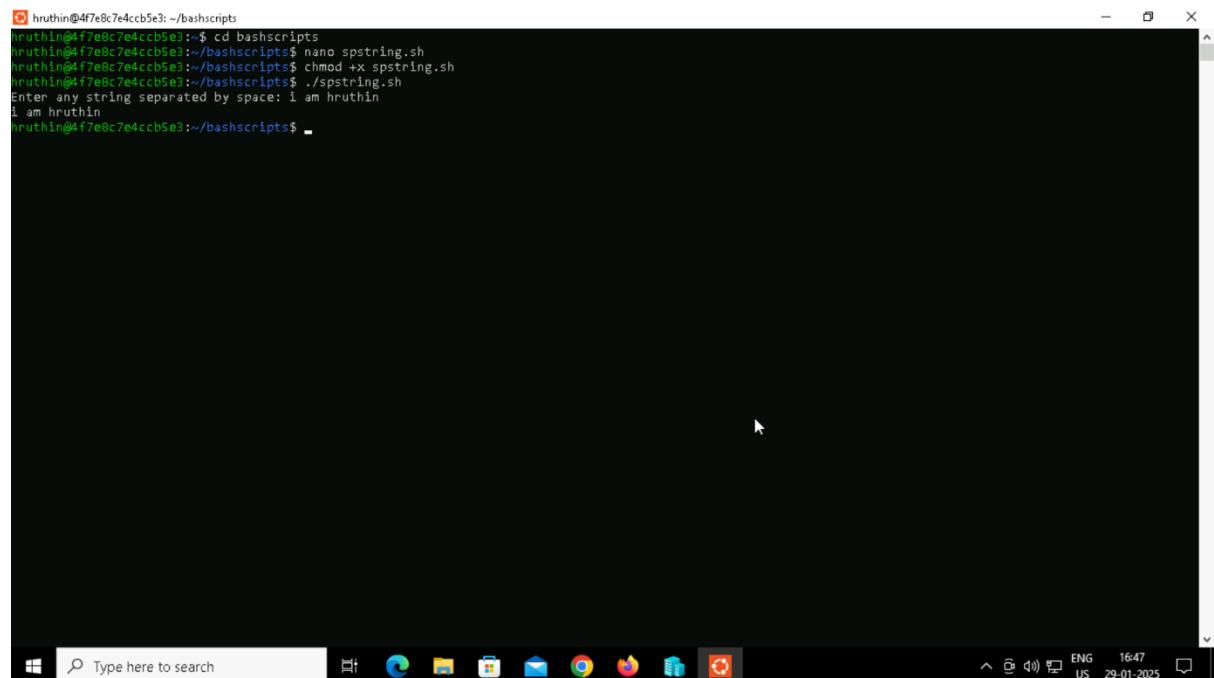
Example 1: Bash Split String by Space

Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/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
```

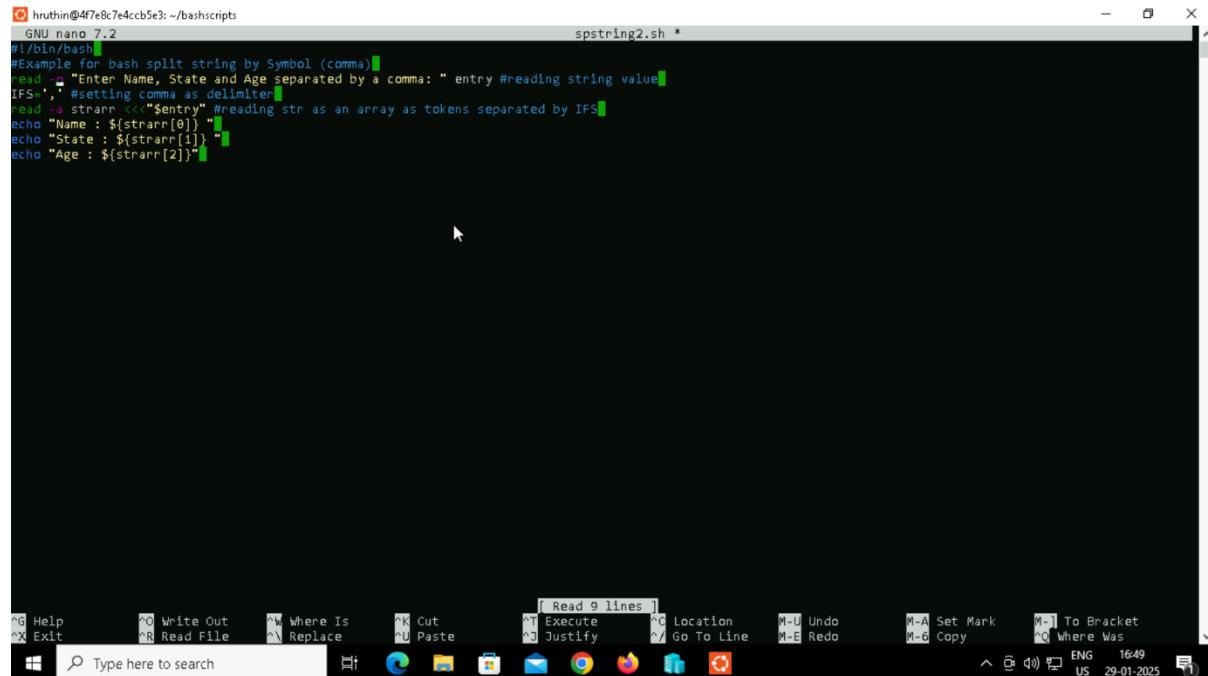
Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:~$ cd bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring.sh
Enter any string separated by space: i am hruthin
i am hruthin
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

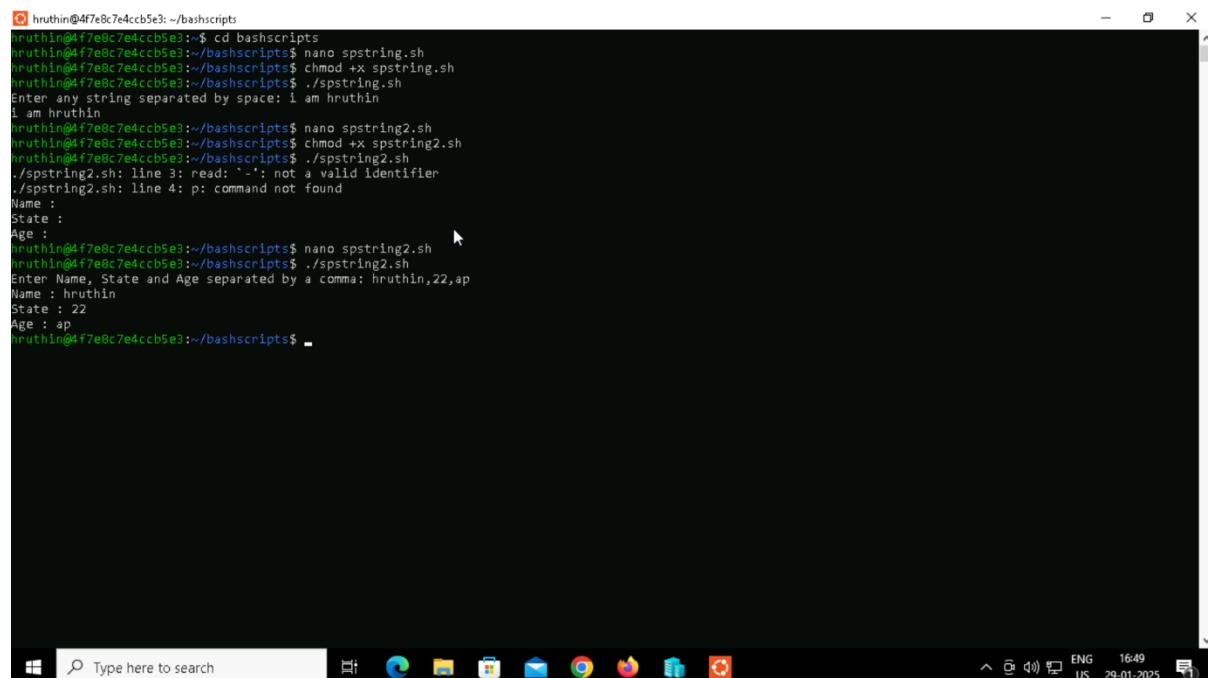
Example 2: Bash Split String by Symbol

Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/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]} "
```

Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:~$ cd bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring.sh
Enter any string separated by space: i am hruthin
i am hruthin
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring2.sh
./spstring2.sh: line 3: read: '-': not a valid identifier
./spstring2.sh: line 4: p: command not found
Name :
State :
Age :
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring2.sh
Enter Name, State and Age separated by a comma: hruthin,22,ap
Name : hruthin
State : 22
Age : ap
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 3: Bash Split String by Symbol

Code:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/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
```

Output:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ cd bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring.sh
Enter any string separated by space: i am hruthin
i am hruthin
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring2.sh
./spstring2.sh: line 3: read: '-' not a valid identifier
./spstring2.sh: line 4: p: command not found
Name :
State :
Age :
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring2.sh
Enter Name, State and Age separated by a comma: hruthin,22,ap
Name : hruthin
State : 22
Age : ap
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring3.sh
Enter any string separated by colon() ansjs:enes

ansjs
enes

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 4: Bash Split String by another string

Code:

```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
GNU nano 7.2                                         spstring4.sh *
#!/bin/bash
#Example for bash split string by another string
str="WeLearnWelcomeLearnYouLearnOnLearnJavaatpoint"
delimiter=Learn
s=$str$delimiter
array=()
while [[ $s ]]; do
array+=("$(s%%$delimiter*)")
s=${#s%$delimiter}
done
declare -p array
```

Output:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:$ cd bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring.sh
Enter any string separated by space: i am hruthin
i am hruthin
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring2.sh
./spstring2.sh: line 3: read: `-' not a valid identifier
./spstring2.sh: line 4: p: command not found
Name :
State :
Age :
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring2.sh
Enter Name, State and Age separated by a comma: hruthin,22,ap
Name : hruthin
State : 22
Age : ap
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring3.sh
Enter any string separated by colon() ansjs:enes

ansjs
enes

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring4.sh
declare -a array=([0]="We" [1]="Welcome" [2]="You" [3]="On" [4]="Javaatpoint")
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example: 5 Bash Split String using Trim Command

Code:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/bash
#Example to split a string using trim (tr) command
my_str="We;welcome;you;on;;Javaatpoint."
my_arr=(${echo $my_str | tr ";" "\n"})
for i in "${my_arr[@]}"
do
echo $i
done
```

The terminal window shows the code for splitting a string using the `trim` command. The code uses `tr` to replace semicolons with newlines, then splits the resulting string into an array `my_arr`. Finally, it loops through the array and prints each element.

Output:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ nano spstring.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ chmod +x spstring.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ ./spstring.sh
Enter any string separated by space: i am hruthin
i am hruthin
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ nano spstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ chmod +x spstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ ./spstring2.sh
./spstring2.sh: line 3: read: -: not a valid identifier
./spstring2.sh: line 4: p: command not found
Name :
State :
Age :
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ nano spstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ ./spstring2.sh
Enter Name, State and Age separated by a comma: hruthin,22,ap
Name : hruthin
State : 22
Age : ap
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ nano spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ nano spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ chmod +x spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ ./spstring3.sh
Enter any string separated by colon(): ansjs:enes
ansjs
enes

hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ nano spstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ chmod +x spstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ ./spstring4.sh
declare -a array=( [0]="We" [1]="Welcome" [2]="You" [3]="On" [4]="Javaatpoint")
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ nano spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ chmod +x spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ ./spstring5.sh
tr: missing operand after ':'
Two strings must be given when translating.
Try 'tr --help' for more information.
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$ nano spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/.bashscripts$
```

The terminal window shows the execution of the bash script. It first runs `spstring.sh`, which prompts for a string separated by spaces and prints it. Then it runs `spstring2.sh`, which fails because it expects a valid identifier for the variable `p`. Next, it runs `spstring3.sh`, which prompts for three values separated by commas and prints them. Finally, it runs `spstring4.sh` and `spstring5.sh`, both of which fail because they require two strings for translation.

Bash Sub String:

Example 1: To Extract till Specific Characters from Starting

Code:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/bash
#Script to extract first 10 characters of a string
echo "String: We welcome you on Javatpoint."
str="We welcome you on Javatpoint."
echo "Total characters in a String: ${#str} "
substr="${str:0:10}"
echo "Substring: $substr"
echo "Total characters in Substring: ${#substr}"
```

Output:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
./spstring2.sh: line 3: read: `-' not a valid identifier
./spstring2.sh: line 4: p: command not found
Name :
State :
Age :
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring2.sh
Enter Name, State and Age separated by a comma: hruthin,22,ap
Name : hruthin
state : 22
Age : ap
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring3.sh
Enter any string separated by colon(): ansjs:enes
ansjs
enes

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring4.sh
declare -a array=([0]=""[1]=""Welcome"[2]=""you"[3]=""On"[4]=""Javatpoint")
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring5.sh
tr: missing operand after `:':`\\n'
Two strings must be given when translating.
Try `tr --help' for more information.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ mv substring5.sh substring
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ mv substring substring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring.sh
String: We welcome you on Javatpoint.
Total characters in a String: 29
Substring: We welcome
Total characters in Substring: 10
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 2: To Extract from Specific Character onwards

Code:

```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
GNU nano 7.2
#!/bin/bash
#Script to print from 11th character onwards
str="We welcome you on Javatpoint."
substr="${str:11}"
echo "$substr"
```

The terminal window shows the code for extracting substrings. The script `substring2.sh` is created and contains the command `substr="\${str:11}"` to extract characters from index 11 onwards. The terminal window has a menu bar with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, To Bracket, Copy, and Where Was. The status bar at the bottom shows the date and time as 29-01-2025.

Output:

```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
Age :
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring2.sh
Enter Name, State and Age separated by a comma: hruthin,22,ap
Name : hruthin
State : 22
Age : ap
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring3.sh
Enter any string separated by colon() :ansjs:enes
ansjs
enes

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring4.sh
declare -a array[@]={"We" "[1]"="You" "[2]"="On" "[3]"="Javatpoint"}
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring5.sh
tr: missing operand after `;`\\n"
Two strings must be given when translating.
Try 'tr --help' for more information.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ mv substring5.sh substring
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ mv substring.substring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring.sh
String: We welcome you on Javatpoint.
Total characters in a String: 29
Substring: We welcome
Total characters in Substring: 10
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring2.sh
you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

The terminal window shows the execution of the bash scripts. It starts with `spstring2.sh` which prints the input as it is. Then it runs `spstring3.sh` which prints the input with a colon separator. Next, it runs `spstring4.sh` which prints the input with a space separator. Finally, it runs `spstring5.sh` which prints the input with a tab separator. The terminal window has a menu bar with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, To Bracket, Copy, and Where Was. The status bar at the bottom shows the date and time as 29-01-2025.

Example 3: To Extract a Single Character

Code:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/Bash
#Script to print 1ith character of a String
str="We welcome you on Javatpoint."
substr="${str:1:1}"
echo "$substr"
```

The screenshot shows a terminal window titled "hruthin@4f7e8c7e4ccb5e3:~/bashscripts". It contains a single line of Bash script code. The code defines a variable str with the value "We welcome you on Javatpoint.", then uses parameter expansion to assign the first character of str to substr, and finally prints substr. The terminal window has a standard Windows-style menu bar at the top and a taskbar at the bottom.

Output:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
Name : hruthin
State : 22
Age : ap
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring3.sh
Enter any string separated by colon(): ansj:enes
ansj:enes

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring4.sh
declare -a array=([0]="We" [1]="Welcome" [2]="You" [3]="On" [4]="Javatpoint")
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring5.sh
tr: missing operand after `;`\\n'
Two strings must be given when translating.
Try "tr --help" for more information.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ mv substring5.sh substring
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring.sh
String: We welcome you on Javatpoint.
Total characters in a String: 29
Substring: We welcome
Total characters in Substring: 10
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring2.sh
you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring3.sh
y
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

The screenshot shows a terminal window with the same title and environment as the previous one. It displays the execution of the script and its output. The script defines an array, creates a spstring5.sh file, and then creates a substring5.sh file which is renamed to substring. The substring command is then run with the string "We welcome you on Javatpoint.". The output shows the total length of the string and the extracted substring "We welcome". The terminal window has a standard Windows-style menu bar at the top and a taskbar at the bottom.

Example 4: To Extract the specific characters from last

Code:

```
GNU nano 2.2
#!/bin/bash
#Script to extract 11 characters from last
str="We welcome you on Javatpoint."
substr="${str: -11}"
echo "$substr"
```

The terminal window shows the command being typed and the resulting output. The output is the string "Javatpoint.". The terminal has a standard Windows-style interface with a taskbar at the bottom.

Output:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring3.sh
Enter any string separated by colon(:) ansjs:enes

ansjs
enes

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring4.sh
declare -a array=([0]=""[1]="We" [2]=" " [3]=""[4]=""[5]="On" [6]=""[7]=""[8]=""[9]=""[10]=""[11]="Javatpoint")
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring5.sh
tr: missing operand after ':\\n'
Two strings must be given when translating.
Try 'tr --help' for more information.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ mv substring5.sh substring
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ mv substring substring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring.sh
String: We welcome you on Javatpoint.
Total characters in a String: 29
Substring: We welcome
Total characters in Substring: 10
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring2.sh
you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring3.sh
y
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring4.sh
Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

The terminal window shows the execution of the script and its output. The output includes the original string, the total character count, the extracted substrings, and the results of various substring extraction attempts. The terminal has a standard Windows-style interface with a taskbar at the bottom.

Concatenating a String:

Example 1: Write Variables Side by Side

Code:

```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
GNU nano 7.2                                         cstring.sh *
#!/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
```

Output:

Example 2: Using Double Quotes

Code:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/Bash
#Script to Concatenate Strings
#Declaring String Variable
str="We welcome you"
#Add the variable within the string
echo "$str on Javatpoint." cstring2.sh *
```

The screenshot shows a terminal window titled "cstring2.sh *". It contains a Bash script with a single line of code: "echo \"\$str on Javatpoint.\"". The terminal interface includes a menu bar with options like Help, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, To Bracket, Copy, and Where Was. Below the menu is a toolbar with icons for file operations. At the bottom, there's a search bar labeled "Type here to search" and a status bar showing the date and time.

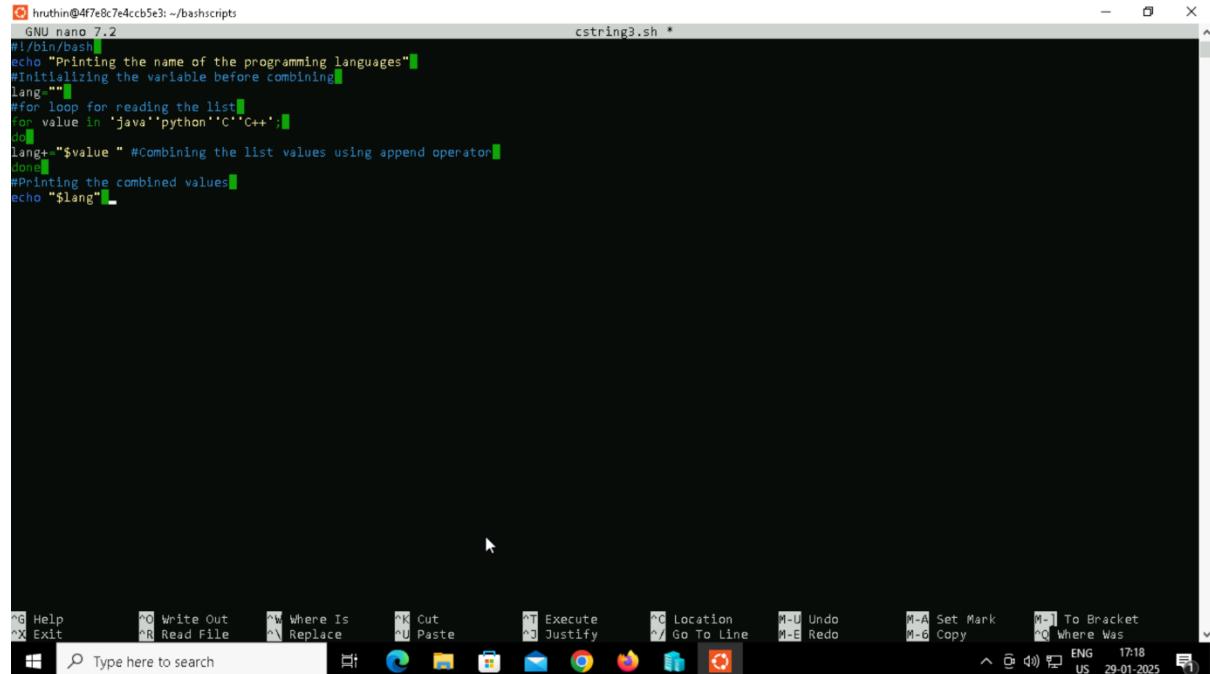
Output:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring4.sh
declare -a array=( [0]="We" [1]="Welcome" [2]="You" [3]="On" [4]="Javatpoint" )
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ tr -d '\n' > spstring5.sh
tr: missing operand after ':\'\\n'
Two strings must be given when translating.
Try 'tr --help' for more information.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ mv substring5.sh substring
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ mv substring substring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring.sh
String: We welcome you on Javatpoint.
Total characters in a String: 29
Substring: We welcome
Total characters in Substring: 10
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring2.sh
you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring3.sh
y
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring4.sh
Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring.sh
We welcome you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring2.sh
We welcome you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

The screenshot shows a terminal window with the same environment as the previous one. It displays the execution of the Bash script and its output. The terminal interface is identical, including the menu bar, toolbar, search bar, and status bar.

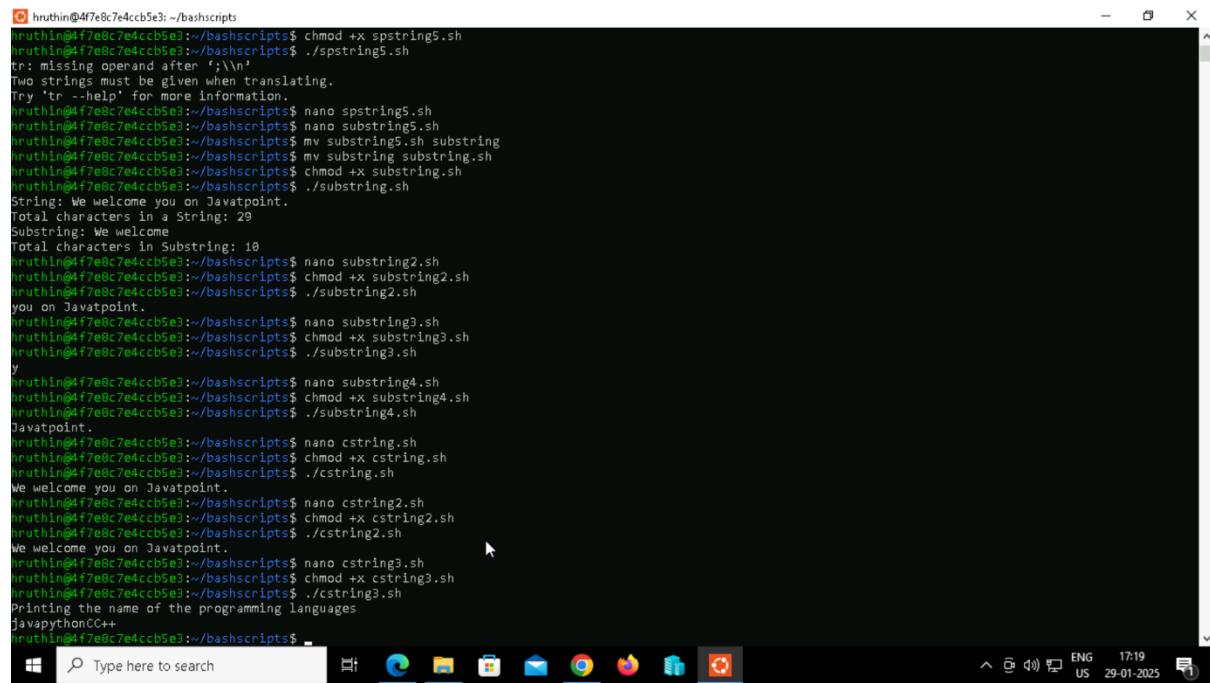
Example 3: Using Append Operator with Loop

Code:



```
GNU nano 7.2                                         cstring3.sh *
#!/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"
```

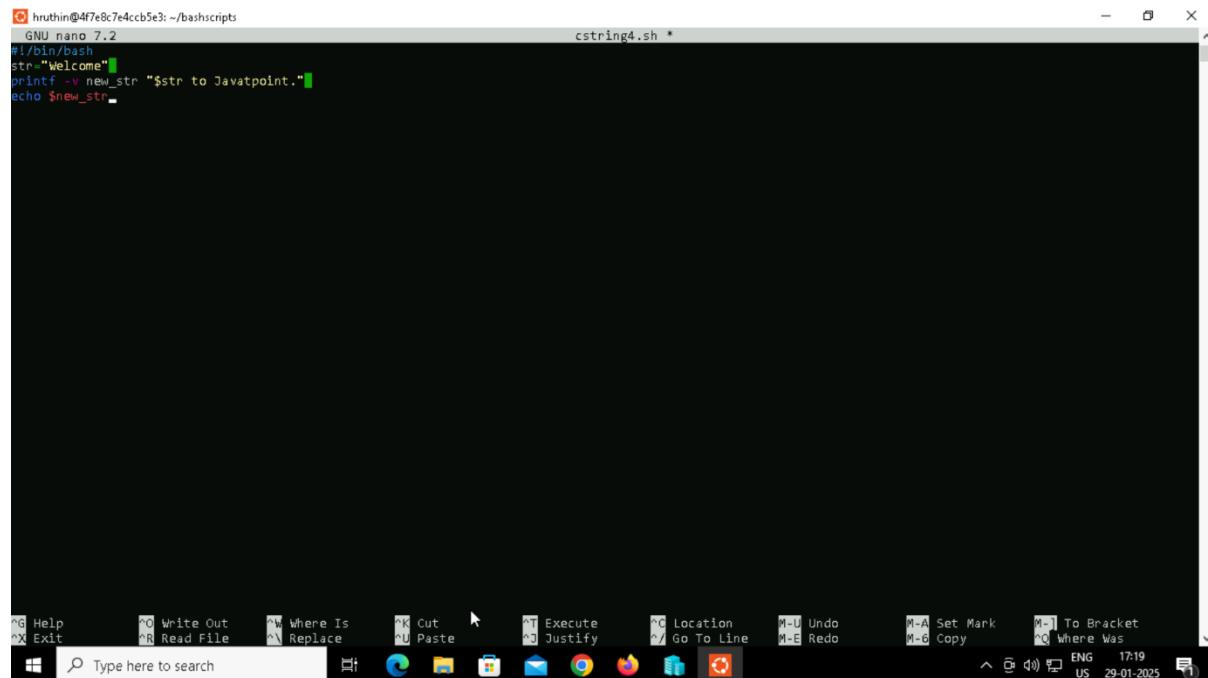
Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./spstring5.sh
tr: missing operand after ':\\n'
Two strings must be given when translating.
Try 'tr --help' for more information.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano spstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ mv substring5.sh substring
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring.sh
String: We welcome you on Javatpoint.
Total characters in a String: 29
Substring: We welcome
Total characters in Substring: 10
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring2.sh
you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring3.sh
y
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring4.sh
Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring.sh
We welcome you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring2.sh
Printing the name of the programming languages
javapythonC++
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

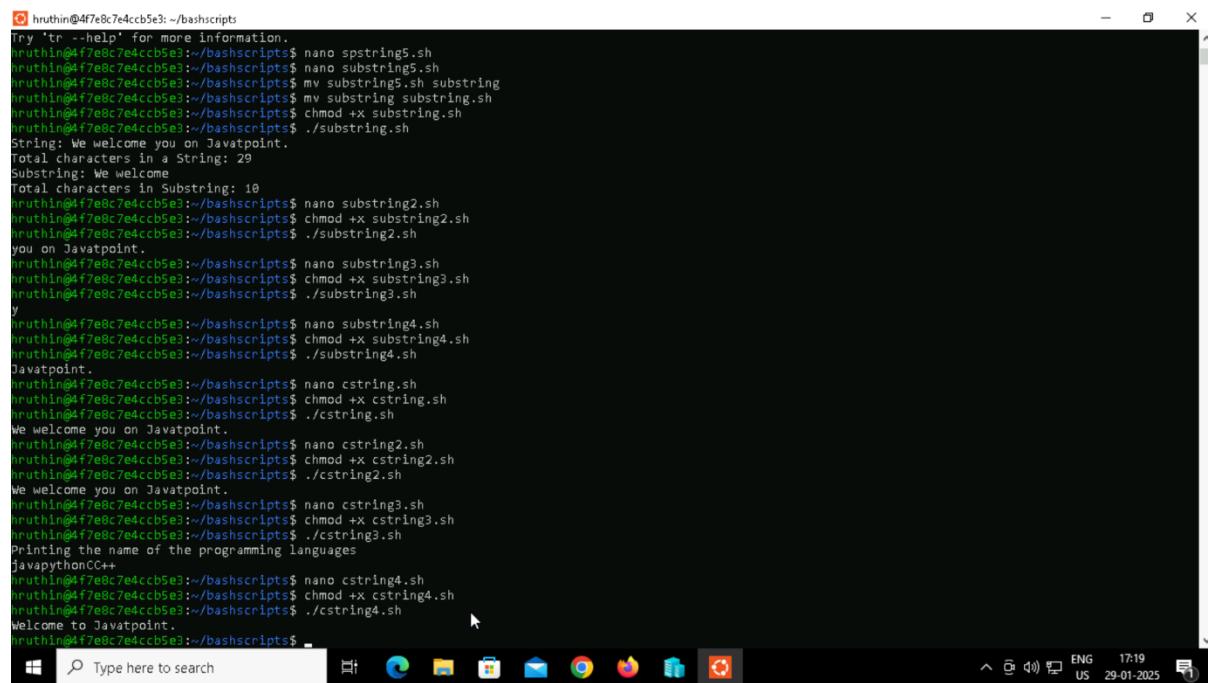
Example 4: Using the Printf Function

Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/bash
str="Welcome"
printf -v new_str "%s to Javatpoint." $str
echo $new_str
```

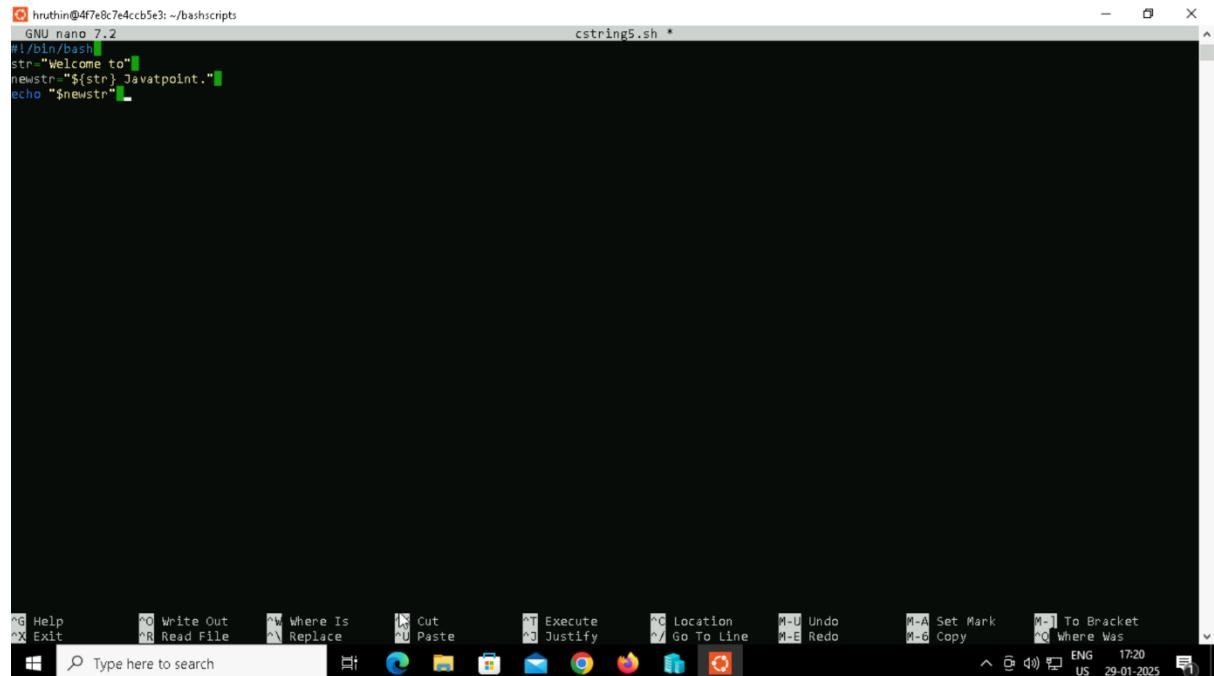
Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
Try 'tr --help' for more information.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ mv substring5.sh substring
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ mv substring substring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring.sh
String: We welcome you on Javatpoint.
Total characters in a String: 29
Substring: We welcome
Total characters in Substring: 10
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring2.sh
you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring3.sh
y
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring4.sh
Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring.sh
We welcome you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring2.sh
We welcome you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring3.sh
Printing the name of the programming languages
javaythonCC+
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring4.sh
Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

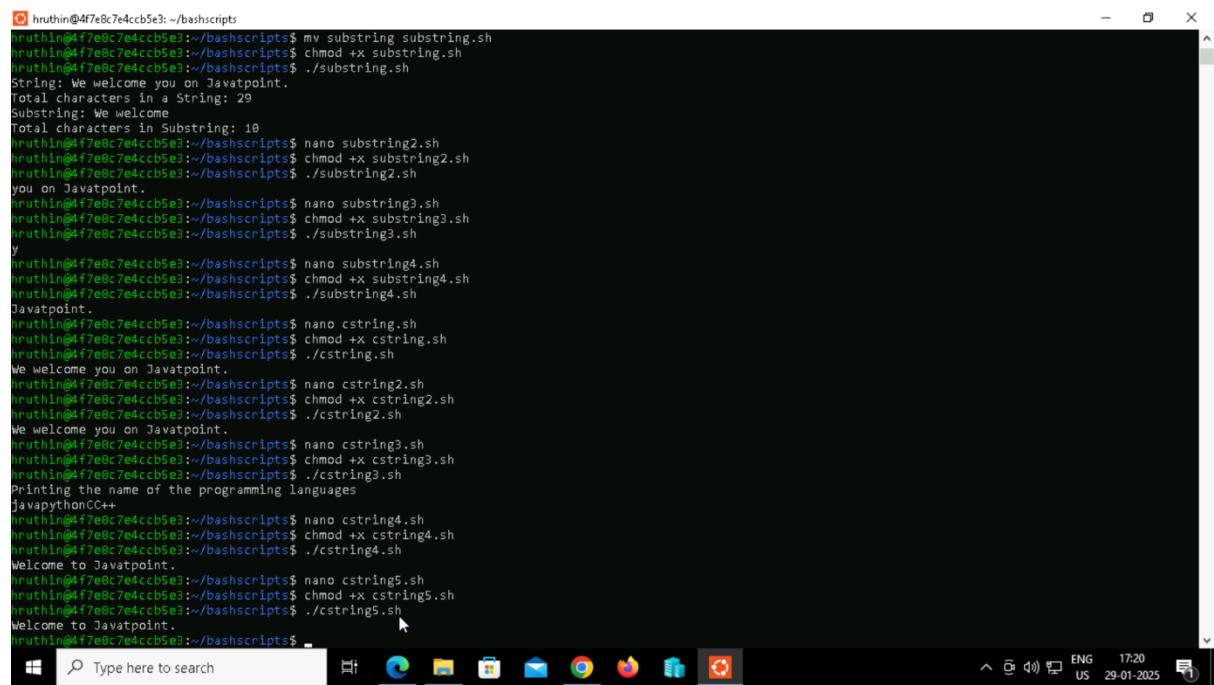
Example 5: Using Literal Strings

Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/bash
str="Welcome to Javatpoint."
newstr="${str} Javatpoint."
echo "$newstr"
```

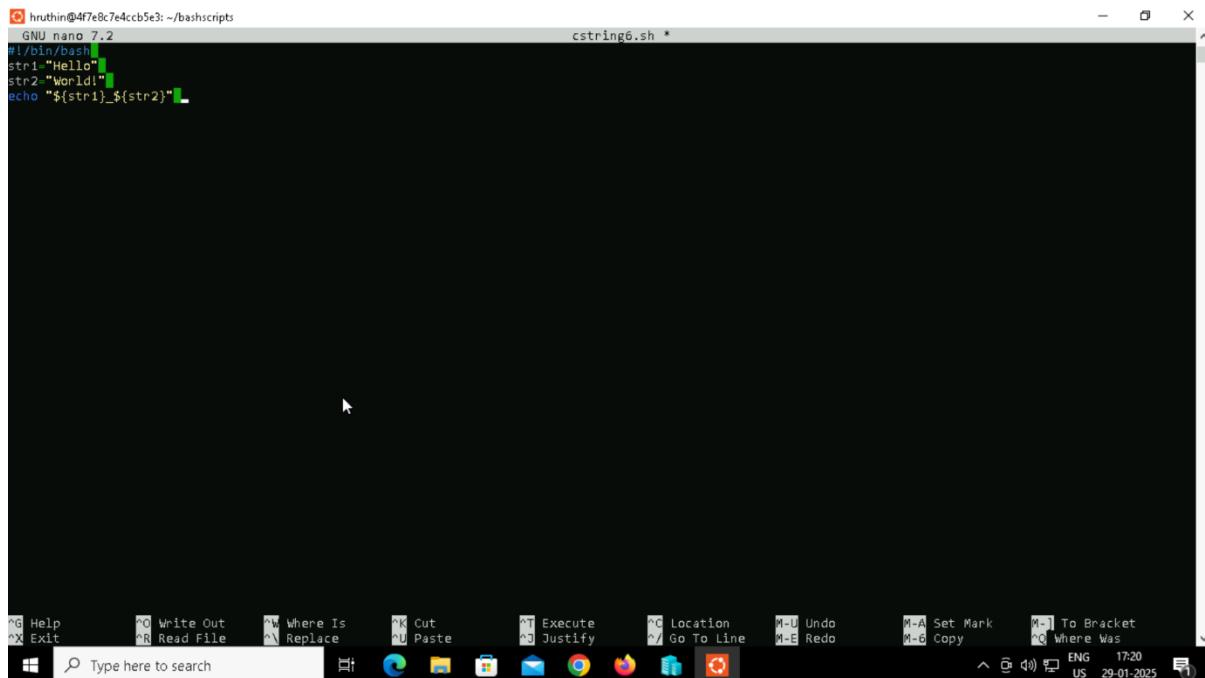
Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ mv substring substring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring.sh
String: We welcome you on Javatpoint.
Total characters in a String: 29
Substring: We welcome
Total characters in Substring: 10
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring2.sh
you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring3.sh
y
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring4.sh
Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring.sh
We welcome you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring2.sh
We welcome you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring3.sh
Printing the name of the programming languages
javapthonCC+
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring4.sh
Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring5.sh
Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

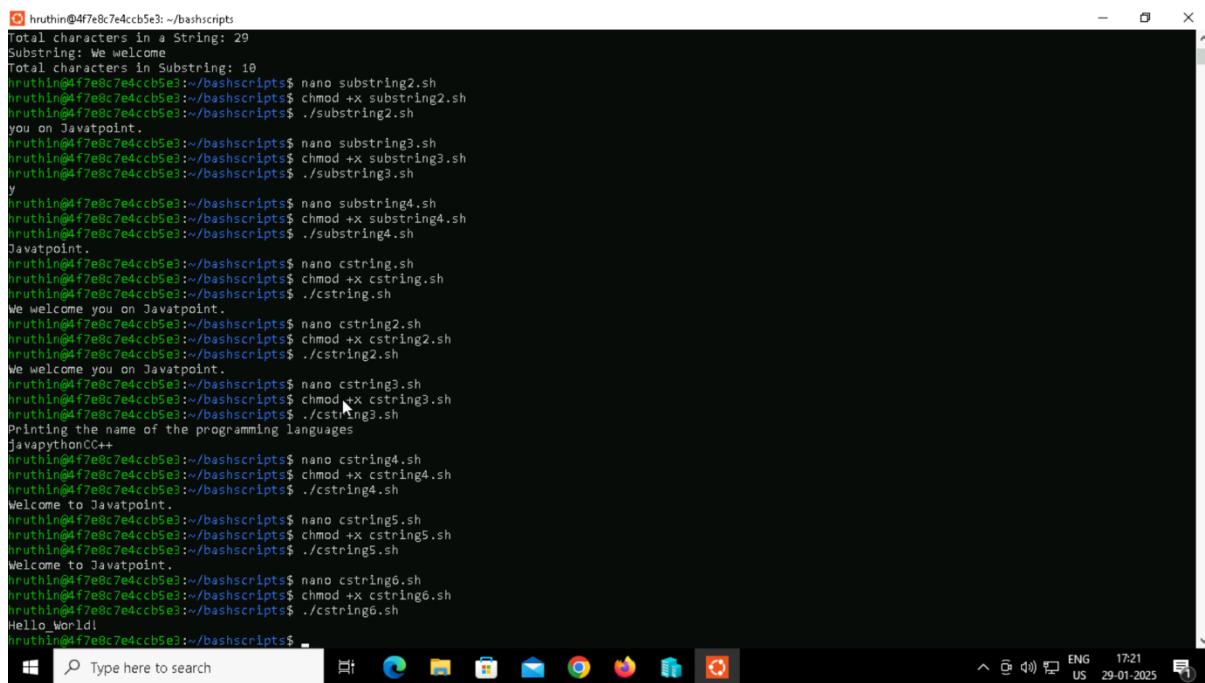
Example 6: Using Underscore

Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/Bash
str1="Hello"
str2="World!"
echo "${str1}_${str2}"
```

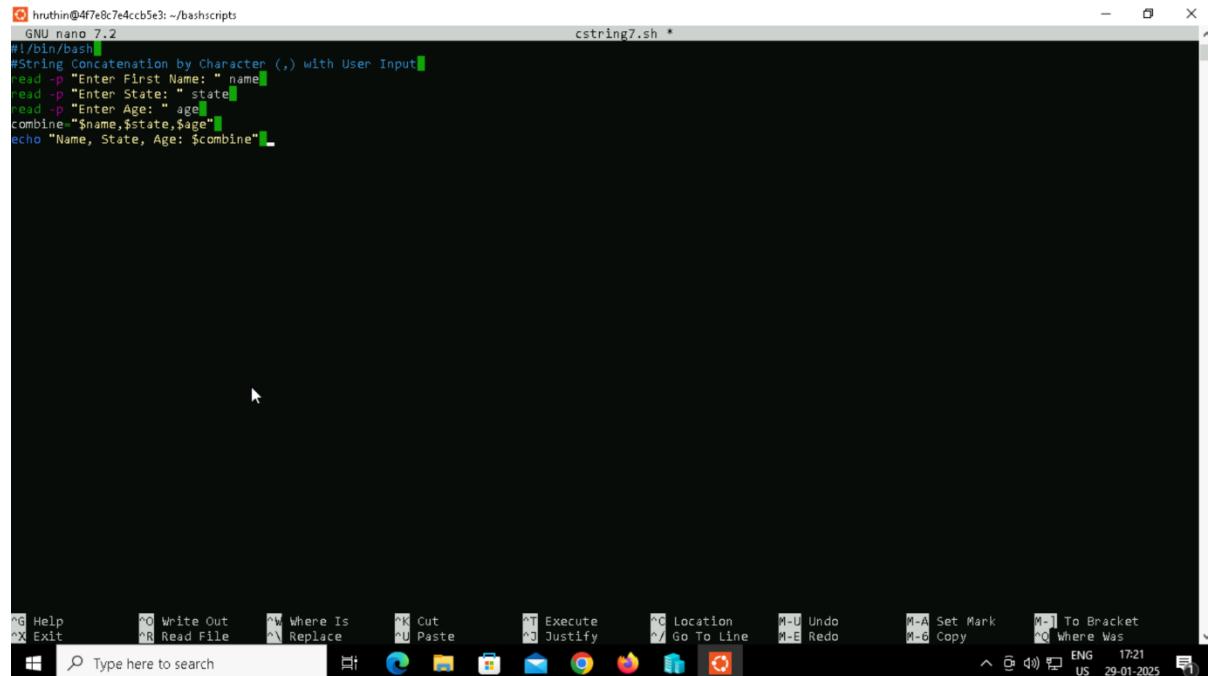
Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
Total characters in a String: 29
Substring: We welcome
Total characters in SubString: 10
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring2.sh
you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring3.sh
y
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring4.sh
Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring.sh
We welcome you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring2.sh
We welcome you on Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring3.sh
Printing the name of the programming languages
javapythonCC++
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring4.sh
Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring5.sh
Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring6.sh
Hello_World!
```

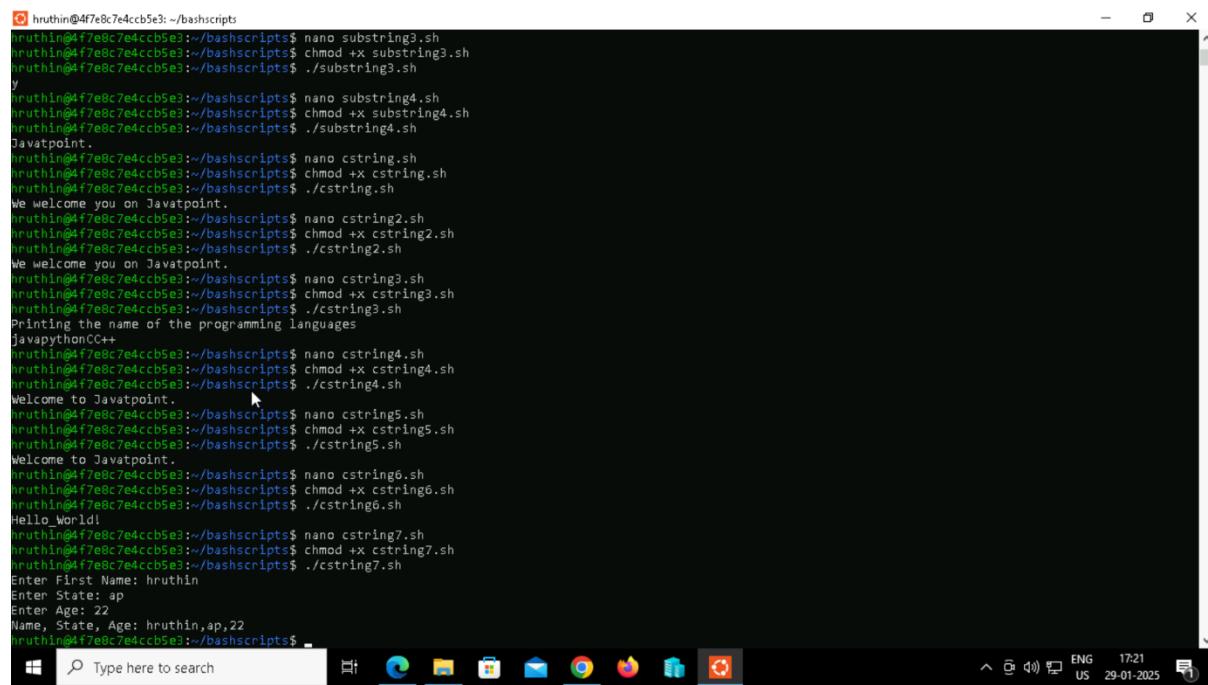
Example 7: Using any Character

Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/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"
```

Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring3.sh
y
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano substring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x substring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./substring4.sh
Javaatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring.sh
We welcome you on Javaatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring2.sh
We welcome you on Javaatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring3.sh
Printing the name of the programming languages
javaythonCC+
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring4.sh
Welcome to Javaatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring5.sh
Welcome to Javaatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring6.sh
Hello_World!
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano cstring7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x cstring7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./cstring7.sh
Enter First Name: hruthin
Enter State: ap
Enter Age: 22
Name, State, Age: hruthin,ap,22
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example: Method 1

Code:

The screenshot shows a terminal window titled "fun1.sh *". Inside the window, the following code is visible:

```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
GNU nano 7.2
#!/bin/bash
DTP () {
echo 'Welcome to Javatpoint.'
```

The terminal window has a dark background and light-colored text. The title bar says "fun1.sh *". The bottom of the window shows a menu bar with options like Help, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, To Bracket, Copy, and Where Was. Below the menu bar is a search bar with the placeholder "Type here to search". At the very bottom of the screen, there is a taskbar with icons for various applications.

Output:

The screenshot shows a terminal window with a large amount of text output from the "grep" command. The text includes detailed documentation for the "grep" command, listing various options and their descriptions. It also shows the command being run and its output:

```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
ACTION is 'read', 'recurse', or 'skip'
-D, --devices=ACTION      how to handle devices, FIFOs and sockets;
-ACTION is 'read' or 'skip'
-r, --recursive           like -d recursive-recurse
-R, --dereference-recursive likewise, but follow all symlinks
--include=GLOB            search only files that match GLOB (a file pattern)
--exclude=GLOB             skip files that match GLOB
--exclude-from=FILE       skip files that match any file pattern from FILE
--exclude-dir=GLOB        skip directories that match GLOB
-L, --files-without-match print only names of FILES with no selected lines
-l, --files-with-matches  print only names of FILES with selected lines
-c, --count                print only a count of selected lines per FILE
-T, --initial-tab         make tabs line up (if needed)
-Z, --null                 print 0 byte after FILE name

Context control:
-B, --before-context=NUM  print NUM lines of leading context
-A, --after-context=NUM   print NUM lines of trailing context
-C, --context=NUM          print NUM lines of output context
-NUM                      same as --context=NUM
--group-separator=SEP    print SEP on line between matches with context
--no-group-separator     do not print separator for matches with context
--color[=WHEN],           use markers to highlight the matching strings;
--colour[=WHEN]           WHEN is 'always', 'never', or 'auto'
-U, --binary               do not strip CR characters at EOL (MSDOS/Windows)

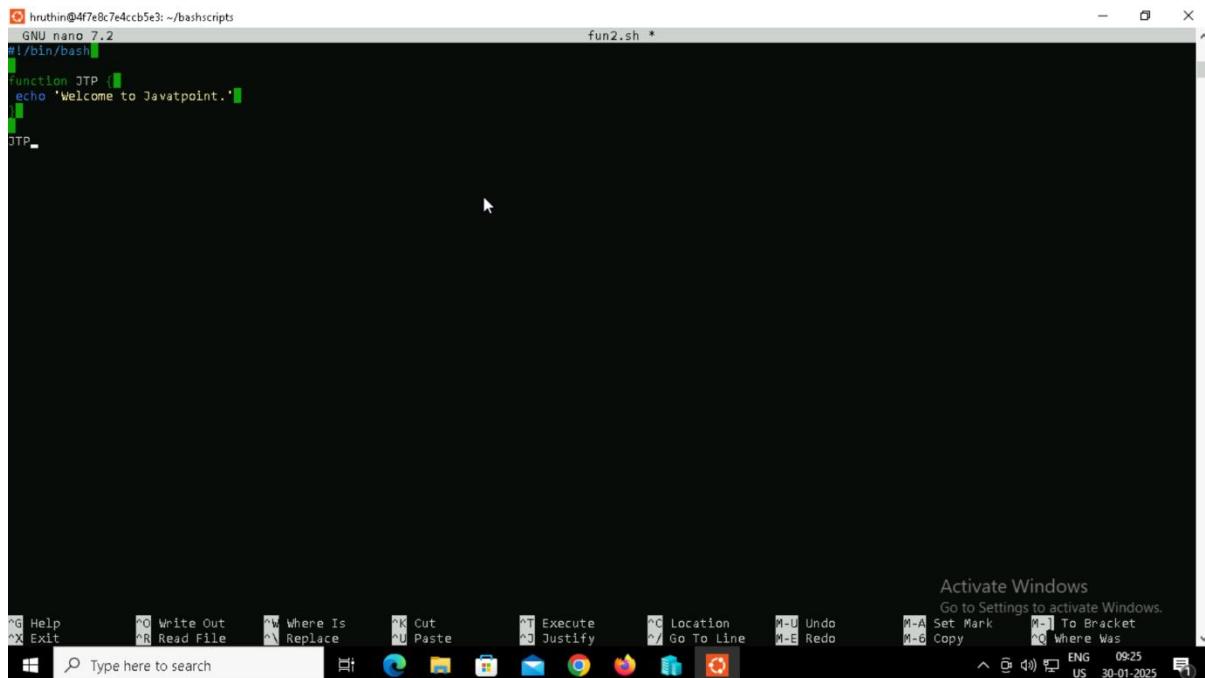
When FILE is '-', read standard input. With no FILE, read '.' if
recursive, '-' otherwise. With fewer than two FILES, assume -h.
Exit status is 0 if any line is selected, 1 otherwise;
if any error occurs and -q is not given, the exit status is 2.

Report bugs to: bug-grep@gnu.org
GNU grep home page: <https://www.gnu.org/software/grep/>
General help using GNU software: <https://www.gnu.org/gethelp/>
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ls | grep "it"
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun1.sh
Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

The terminal window has a dark background and light-colored text. The title bar says "Activate Windows" and "Go to Settings to activate Windows.". The bottom of the window shows a menu bar with options like Help, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, To Bracket, Copy, and Where Was. Below the menu bar is a search bar with the placeholder "Type here to search". At the very bottom of the screen, there is a taskbar with icons for various applications.

Example: Method 2

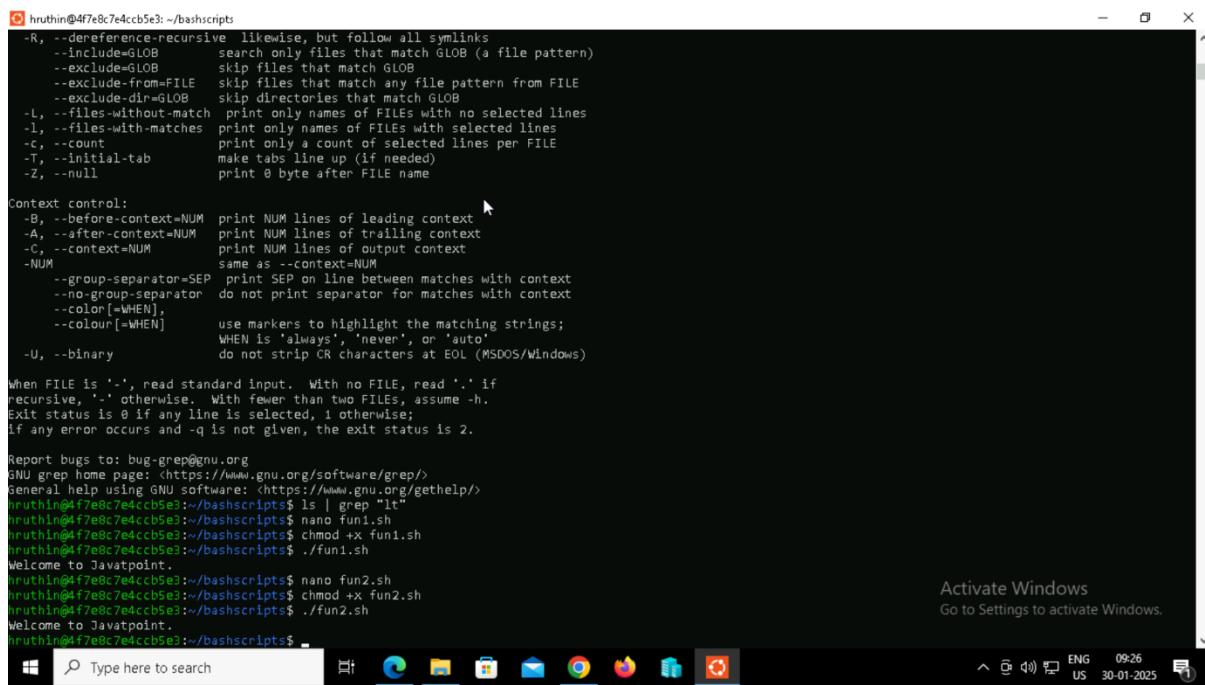
Code:



```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
GNU nano 7.2
#!/bin/bash
function JTP {
echo "Welcome to Javatpoint."
}
JTP
```

The screenshot shows a terminal window titled "fun2.sh *". It contains the code for a bash script named "fun2.sh". The script defines a function "JTP" that prints "Welcome to Javatpoint.". The terminal window has a dark background and a light-colored text area. At the bottom, there is a toolbar with various icons and a status bar showing "Activate Windows", "Go to Settings to activate Windows.", "ENG 09:25", "US 30-01-2025", and a battery icon.

Output:



```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
--R, --dereference-recursive likewise, but follow all symlinks
--include=GLOB search only files that match GLOB (a file pattern)
--exclude=GLOB skip files that match GLOB
--exclude-from=FILE skip files that match any file pattern from FILE
--exclude-dir=GLOB skip directories that match GLOB
-L, --files-without-match print only names of FILES with no selected lines
-l, --files-with-matches print only names of FILES with selected lines
-c, --count print only a count of selected lines per FILE
-t, --initial-tab make tabs line up (if needed)
-Z, --null print 0 byte after FILE name

Context control:
-B, --before-context=NUM print NUM lines of leading context
-A, --after-context=NUM print NUM lines of trailing context
-C, --context=NUM print NUM lines of output context
-NUM same as --context=NUM
--group-separator=SEP print SEP on line between matches with context
--no-group-separator do not print separator for matches with context
--color[=WHEN], --colour[=WHEN] use markers to highlight the matching strings;
WHEN is 'always', 'never', or 'auto'
-U, --binary do not strip CR characters at EOL (MSDOS/Windows)

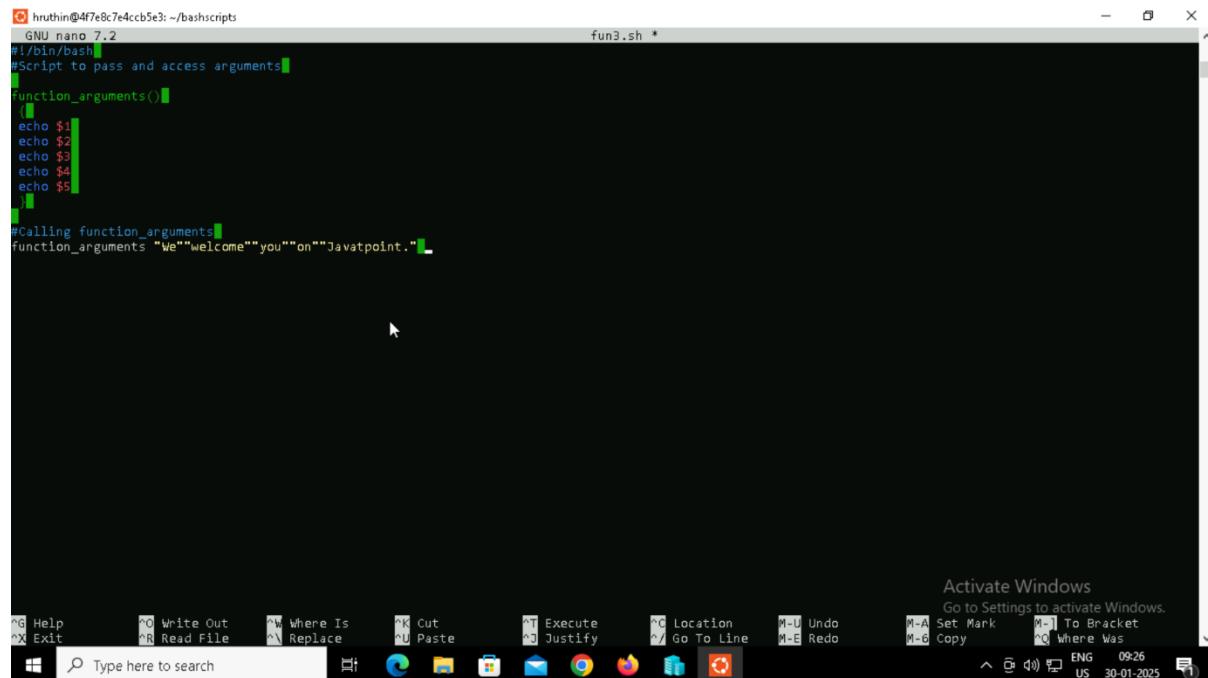
When FILE is '-', read standard input. With no FILE, read '.' if
recursive, '-' otherwise. With fewer than two FILES, assume -h.
Exit status is 0 if any line is selected, 1 otherwise;
if any error occurs and -q is not given, the exit status is 2.

Report bugs to: bug-grep@gnu.org
GNU grep home page: <https://www.gnu.org/software/grep/>
General help using GNU software: <https://www.gnu.org/gethelp/>
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ls | grep "lt"
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun1.sh
Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun2.sh
Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

The screenshot shows a terminal window with the title "Activate Windows" and "Go to Settings to activate Windows.". It displays the help output for the "grep" command and the execution of "fun2.sh", which prints "Welcome to Javatpoint.". The terminal window has a dark background and a light-colored text area. At the bottom, there is a toolbar with various icons and a status bar showing "Activate Windows", "Go to Settings to activate Windows.", "ENG 09:26", "US 30-01-2025", and a battery icon.

Example 3: passing arguments

Code:

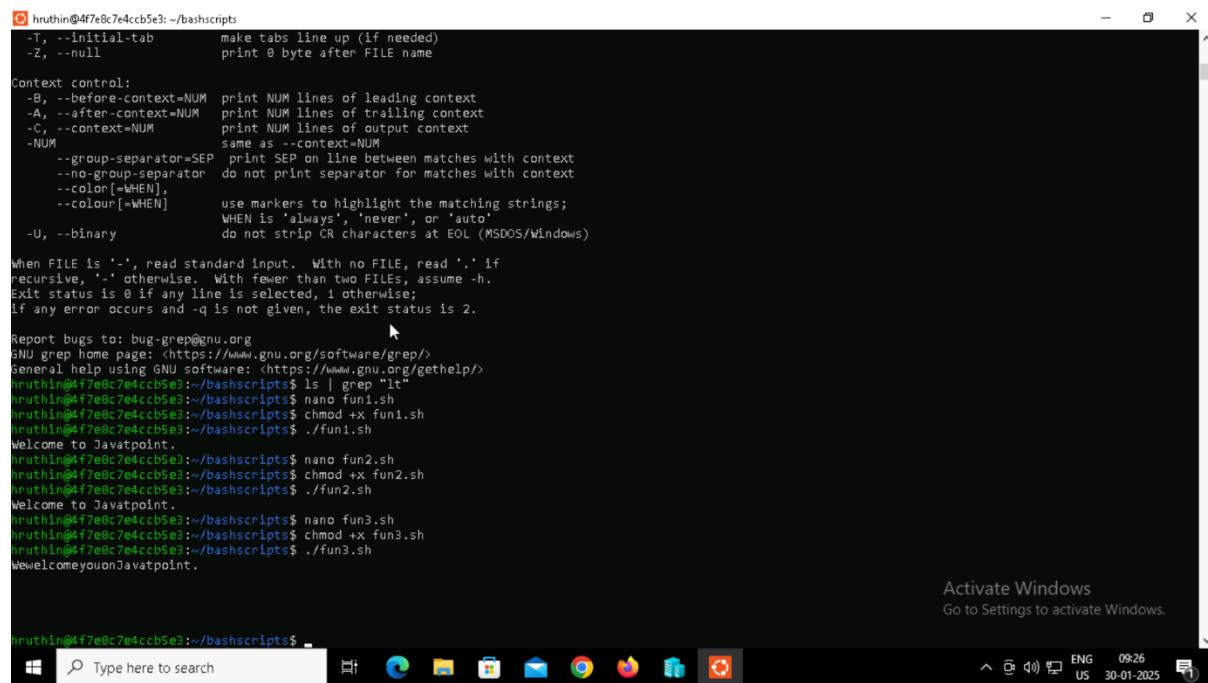


```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/Bash
#Script to pass and access arguments

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

calling function_arguments
function_arguments "We""welcome""you""on""Javatpoint."
```

Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
-T, --initial-tab      make tabs line up (if needed)
-Z, --null              print 0 byte after FILE name

Context control:
-B, --before-context=NUM  print NUM lines of leading context
-A, --after-context=NUM   print NUM lines of trailing context
-C, --context=NUM         print NUM lines of output context
-NUM                     same as --context=NUM
--group-separator=SEP    print SEP on line between matches with context
--no-group-separator     do not print separator for matches with context
--color[=WHEN]
--colour[=WHEN]           use markers to highlight the matching strings;
                           WHEN is 'always', 'never', or 'auto'.
-U, --binary             do not strip CR characters at EOL (MSDOS/Windows)

When FILE is '-', read standard input.  With no FILE, read '-' if
recursive, '--' otherwise.  With fewer than two FILES, assume -h,
Exit status is 0 if any line is selected, 1 otherwise;
If any error occurs and -q is not given, the exit status is 2.

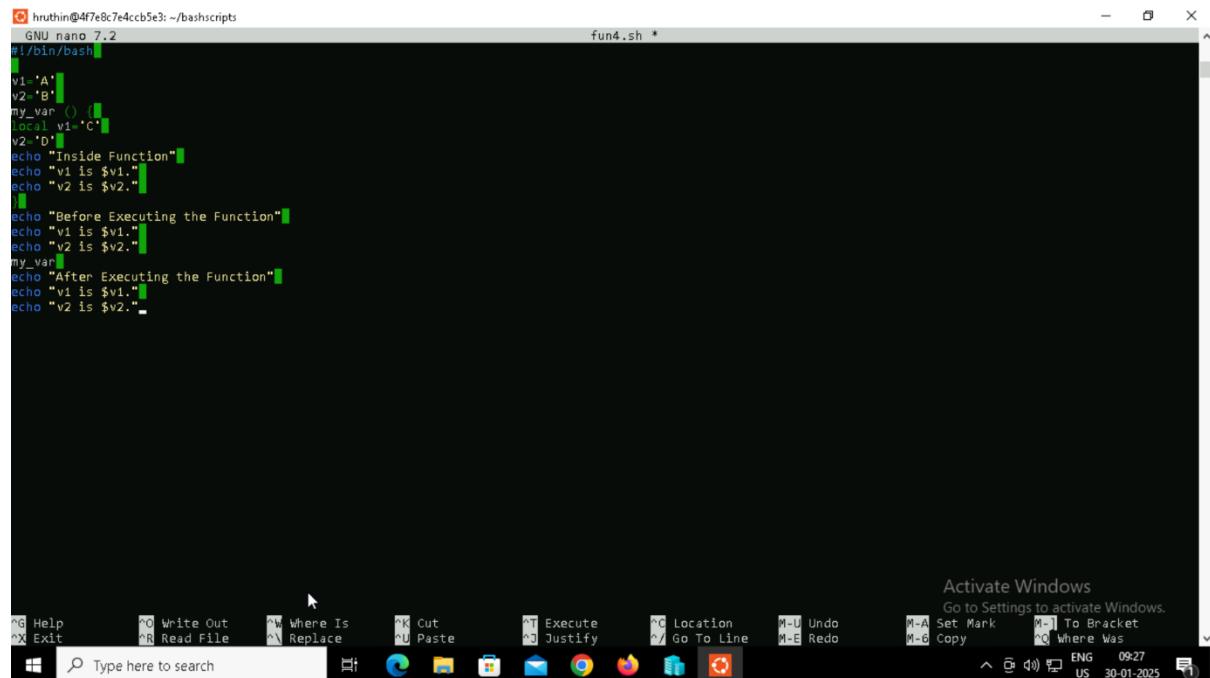
Report bugs to: bug-grep@gnu.org
GNU grep home page: <https://www.gnu.org/software/grep/>
General help using GNU software: <https://www.gnu.org/gethelp/>
hruthing@4f7e8c7e4ccb5e3:~/bashscripts$ ls | grep "it"
hruthing@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun1.sh
hruthing@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun1.sh
hruthing@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun1.sh
Welcome to Javatpoint.

hruthing@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun2.sh
hruthing@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun2.sh
hruthing@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun2.sh
Welcome to Javatpoint.

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun3.sh
hruthing@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun3.sh
hruthing@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun3.sh
We welcome you on Javatpoint.
```

Example 4: Variable Scope

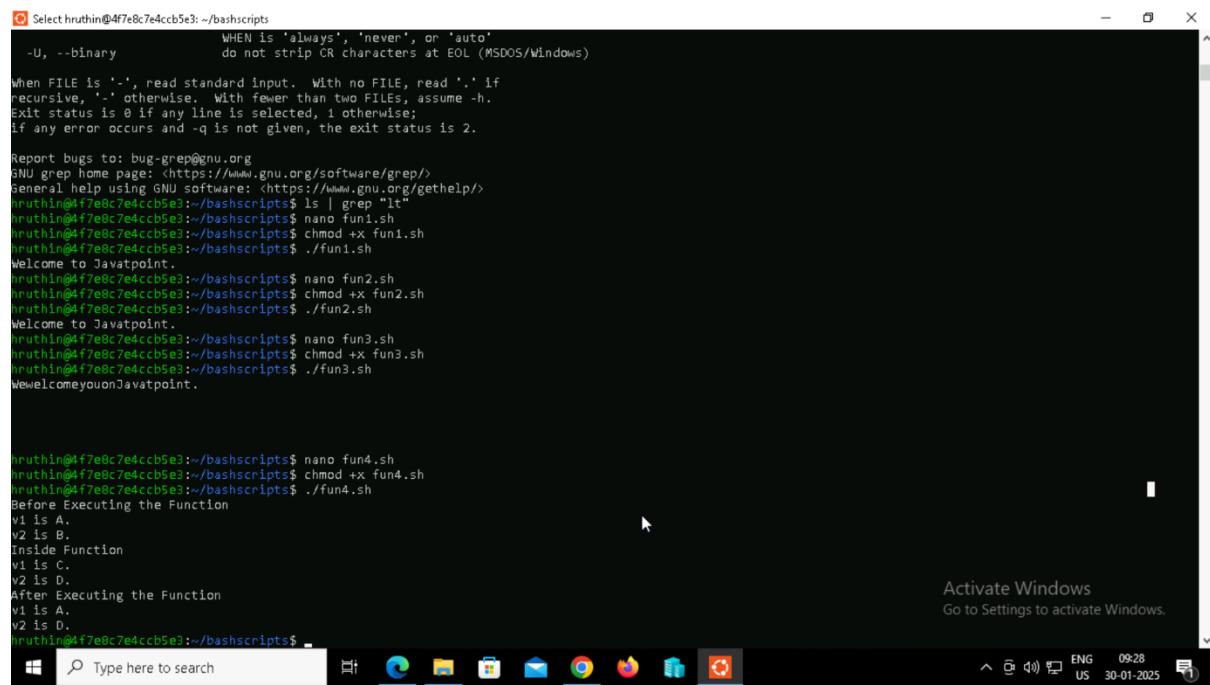
Code:



```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
$ nano fun4.sh
#!/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:



```
Select hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
-WHEN is 'always', 'never', or 'auto'
-U, --binary          do not strip CR characters at EOL (MSDOS/Windows)

When FILE is '-', read standard input.  With no FILE, read '-' if
recursive, '-' otherwise.  With fewer than two FILES, assume -h.
Exit status is 0 if any line is selected, 1 otherwise;
if any error occurs and -q is not given, the exit status is 2.

Report bugs to: bug-grep@gnu.org
GNU grep home page: <https://www.gnu.org/software/grep/>
General help using GNU software: <https://www.gnu.org/gethelp/>
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ls | grep "it"
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun1.sh
Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun2.sh
Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun3.sh
Welcome youonJavatpoint.

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun4.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.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 5: Return Values

A screenshot of a Windows desktop environment. In the center is a terminal window titled "fun5.sh". It contains the following code:

```
GNU nano 7.2
1. #!/bin/bash
2. #Setting up a return status for a function
3.
4. print_it () {
5.     echo Hello $1
6.     return 5
7. }
8.
9. print_it User
10. print_it Reader
11. echo The previous function returned a value of $?
```

The terminal window has a dark background with light-colored text. At the bottom of the window, there's a menu bar with options like Help, Write Out, Where Is, Cut, Execute, Location, Undo, Set Mark, To Bracket, Read File, Replace, Paste, Justify, Go To Line, Redo, Copy, and Where Was. Below the menu is a search bar with the placeholder "Type here to search". At the very bottom of the screen is a Windows taskbar with icons for various applications like File Explorer, Task View, and the Start button.

Output:

A screenshot of a Windows desktop environment, similar to the one above, showing the same terminal window. The terminal window now displays the output of running the script:

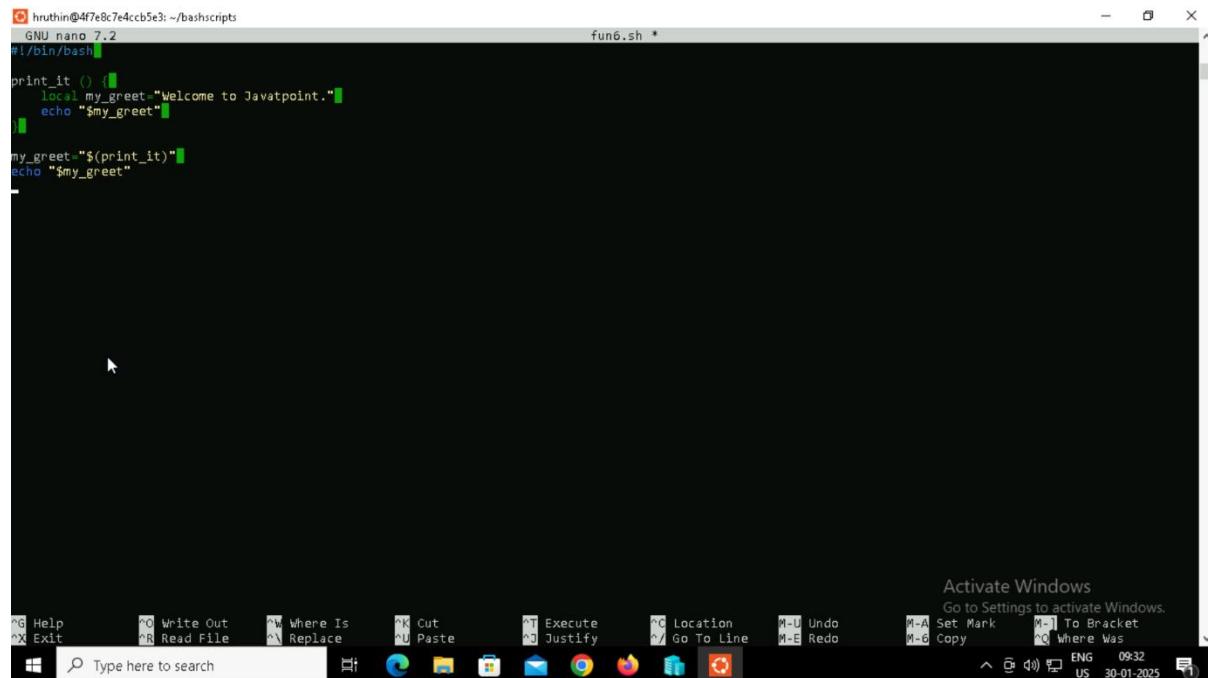
```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun1.sh
Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun2.sh
Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun3.sh
Welcome youonJavatpoint.

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun4.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.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun5.sh
./fun5.sh: line 1: 1: command not found
./fun5.sh: line 2: 2: command not found
./fun5.sh: line 3: 3: command not found
./fun5.sh: line 4: syntax error near unexpected token `('
./fun5.sh: line 4: 4. print_it () {
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun5.sh
Hello User
Hello Reader
The previous function returned a value of 5
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

The terminal window and taskbar are identical to the first screenshot, showing the same application icons and system status.

Example 6:

Code:



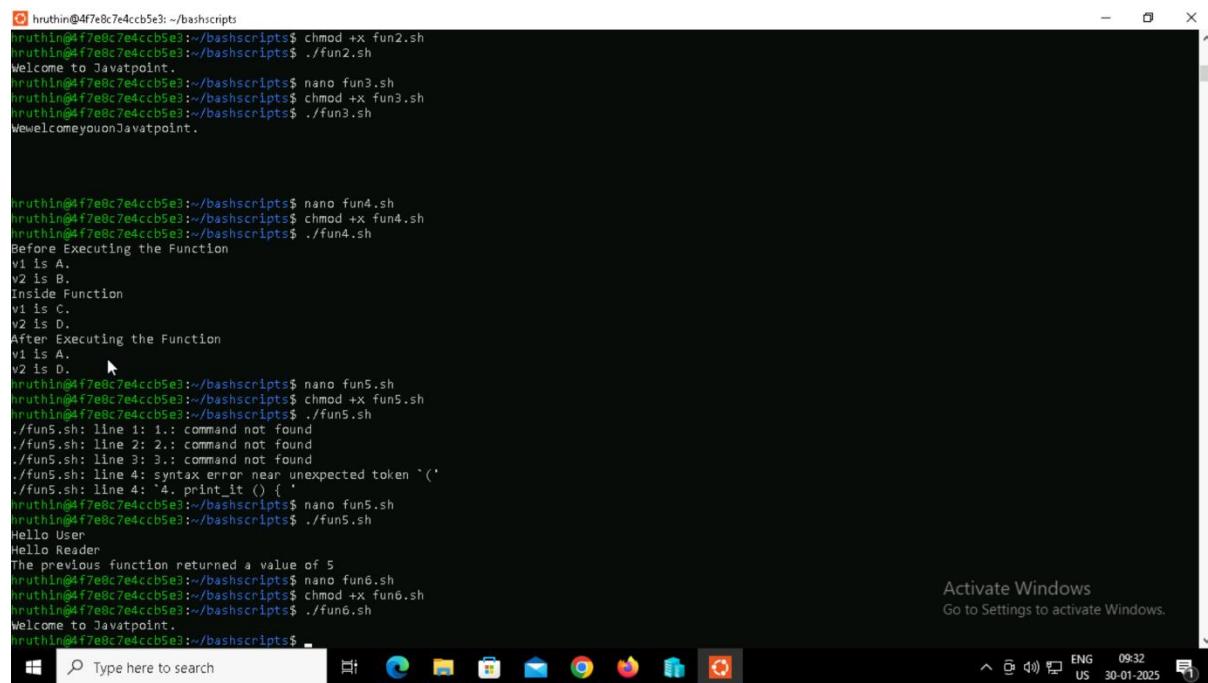
```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
$ nano fun6.sh
#!/bin/bash

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

my_greet=$(print_it)
echo "$my_greet"


```

Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
$ chmod +x fun6.sh
$ ./fun6.sh
Welcome to Javatpoint.

Activate Windows
Go to Settings to activate Windows.

HRUTHIN@4F7E8C7E4CCB5E3 ~ % Help Write Out Where Is Cut Execute Location Undo Set Mark To Bracket
HRUTHIN@4F7E8C7E4CCB5E3 ~ % Read File Replace Paste Justify Go To Line Redo Copy Where Was
HRUTHIN@4F7E8C7E4CCB5E3 ~ %
Windows Type here to search ENG 09:32
US 30-01-2025

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun2.sh
Welcome to Javatpoint.

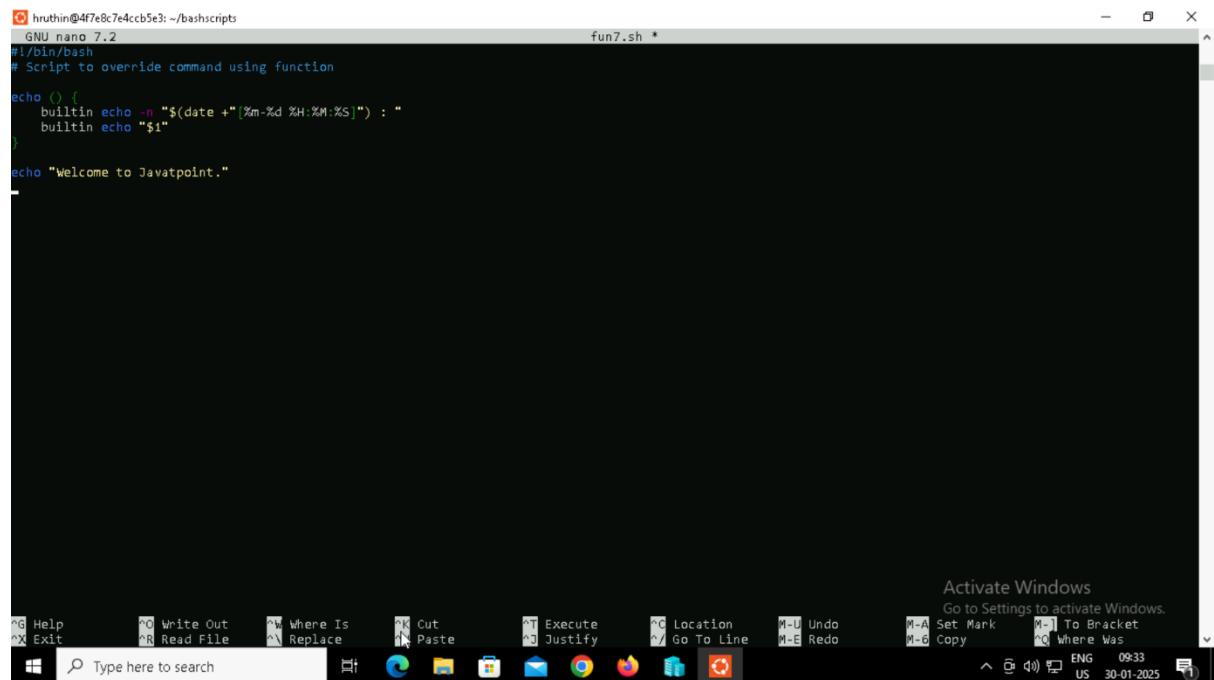
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun3.sh
Welcome you on Javatpoint.

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun4.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.

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun5.sh
./fun5.sh: line 1: command not found
./fun5.sh: line 2: command not found
./fun5.sh: line 3: command not found
./fun5.sh: line 4: syntax error near unexpected token `('
./fun5.sh: line 4: 4, print_it () { '
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun5.sh
Hello Reader
Hello Reader
The previous function returned a value of 5
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun6.sh
Welcome to Javatpoint.

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 6: Overriding commands



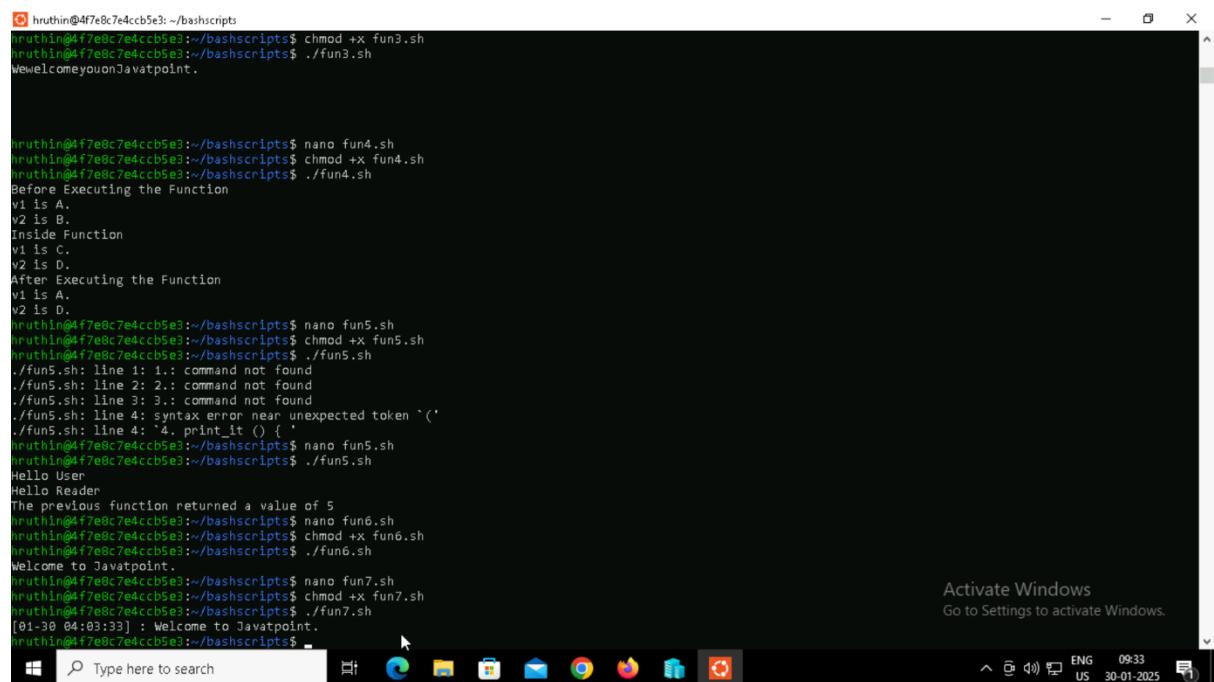
```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/bash
# Script to override command using function

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

echo "Welcome to Javatpoint."
```

The screenshot shows a Windows desktop environment. In the foreground, there is a terminal window titled "fun7.sh *". The window contains a bash script with a function that overrides the built-in echo command to include the current date and time. Below the terminal, the Windows taskbar is visible, featuring the Start button, a search bar, and icons for various applications like File Explorer, Edge, and Task View.

Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun3.sh
Welcome you on Javatpoint.

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun4.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.

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun5.sh
./fun5.sh: line 1: 1.: command not found
./fun5.sh: line 2: 2.: command not found
./fun5.sh: line 3: 3.: command not found
./fun5.sh: line 4: syntax error near unexpected token `('
./fun5.sh: line 4: 4. print_it () {
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun5.sh
Hello User
Hello Reader
The previous function returned a value of 5
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun6.sh
Welcome to Javatpoint.

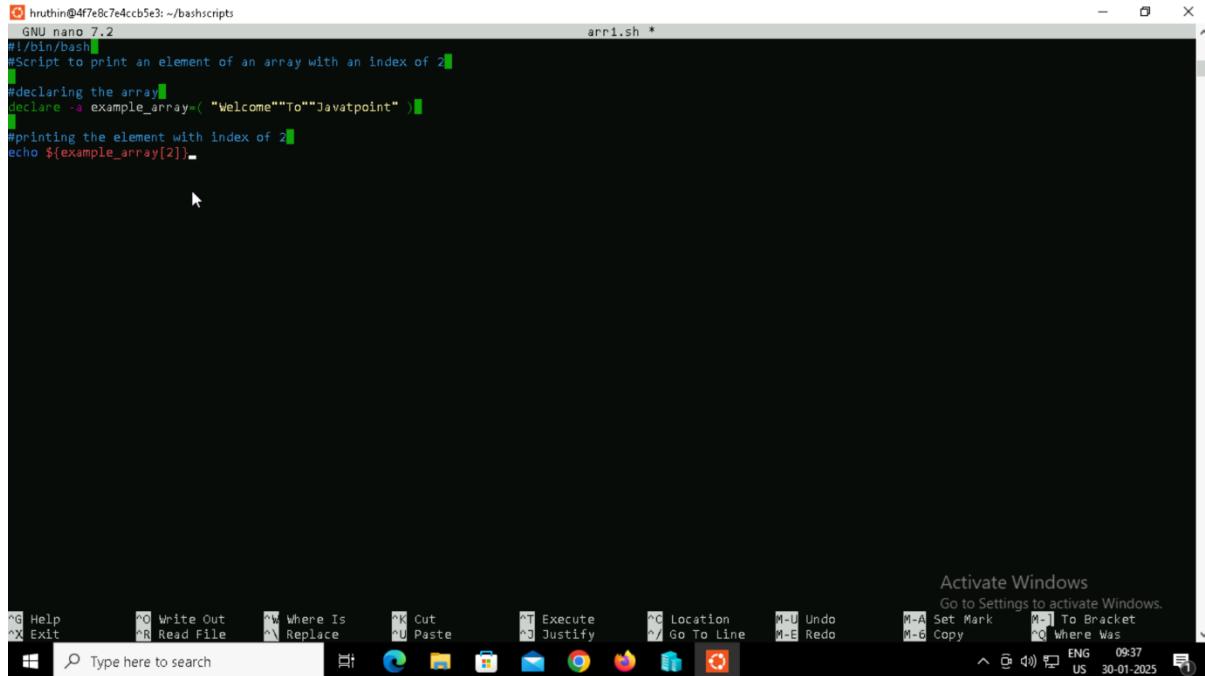
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun7.sh
[01-30 04:03:33] : Welcome to Javatpoint.

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

The screenshot shows the terminal output of the bash script execution. The script first creates and executes fun3.sh, then fun4.sh, which demonstrates function overriding. It then attempts to execute fun5.sh, which fails due to syntax errors. Finally, it executes fun6.sh and fun7.sh, both of which successfully run and display their respective messages. The Windows taskbar at the bottom is visible again.

Arrays:

Example 1: Array operations

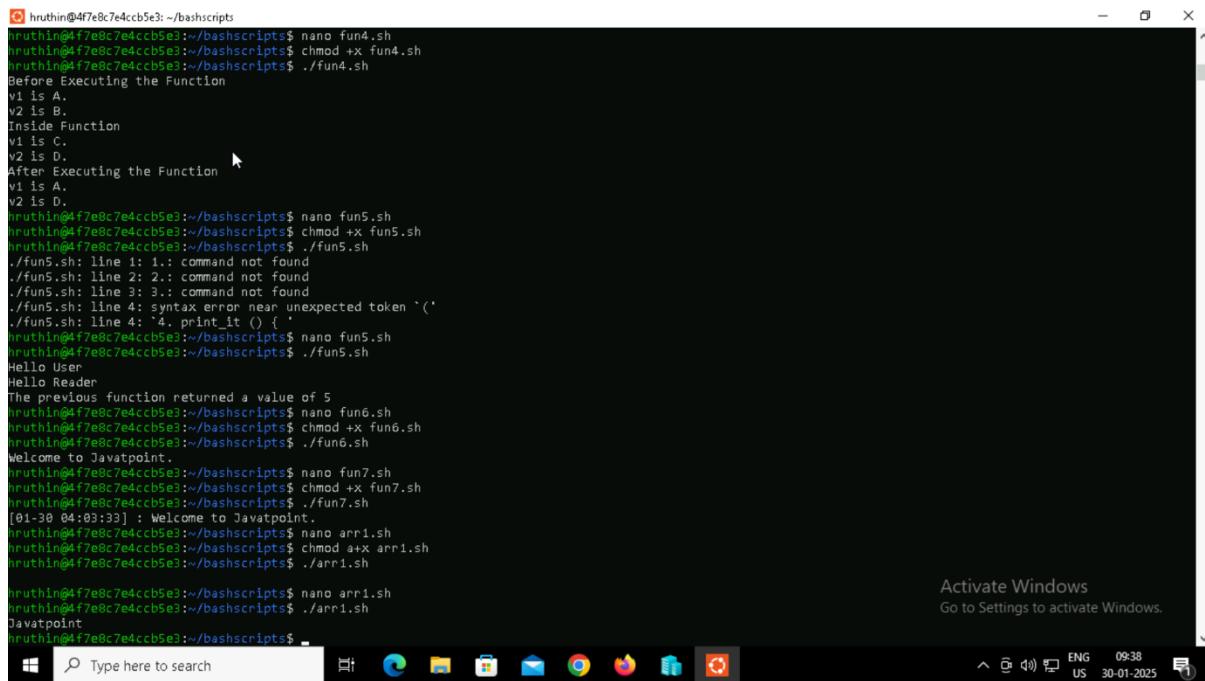


```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/Bash
#Script to print an element of an array with an index of 2
#declaring the array
declare -a example_array=( "Welcome""To""JavaPoint" )
#printing the element with index of 2
echo ${example_array[2]}


```

The screenshot shows a Windows terminal window titled "arr1.sh *". It contains the code for a Bash script named arr1.sh. The script declares an array called example_array with three elements: "Welcome", "To", and "JavaPoint". It then prints the third element of the array, which is "JavaPoint". The terminal window has a standard Windows-style menu bar at the top and a taskbar with various icons at the bottom.

Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun4.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.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun5.sh
./fun5.sh: line 1: command not found
./fun5.sh: line 2: 2.: command not found
./fun5.sh: line 3: 3.: command not found
./fun5.sh: line 4: syntax error near unexpected token `('
./fun5.sh: line 4: 4. printit () { '
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun5.sh
Hello User
Hello Reader
The previous function returned a value of 5
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun6.sh
Welcome to JavaPoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun7.sh
[01-30 04:03:33] : Welcome to JavaPoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod a+x arr1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr1.sh
JavaPoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

The screenshot shows a Windows terminal window titled "arr1.sh *". It displays the output of the arr1.sh script. The output includes the declaration of the array example_array, the printing of its third element "JavaPoint", and the execution of the fun4.sh, fun5.sh, fun6.sh, fun7.sh, and arr1.sh scripts. The terminal window has a standard Windows-style menu bar at the top and a taskbar with various icons at the bottom.

Example 2:

Code:

```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
GNU nano 7.2
#!/bin/Bash
#Script to print all the elements of the array
#declaring the array
declare -a example_array=( "Welcome" "To" "Javatpoint" )
#Printing all the elements
echo "${example_array[@]}"


```

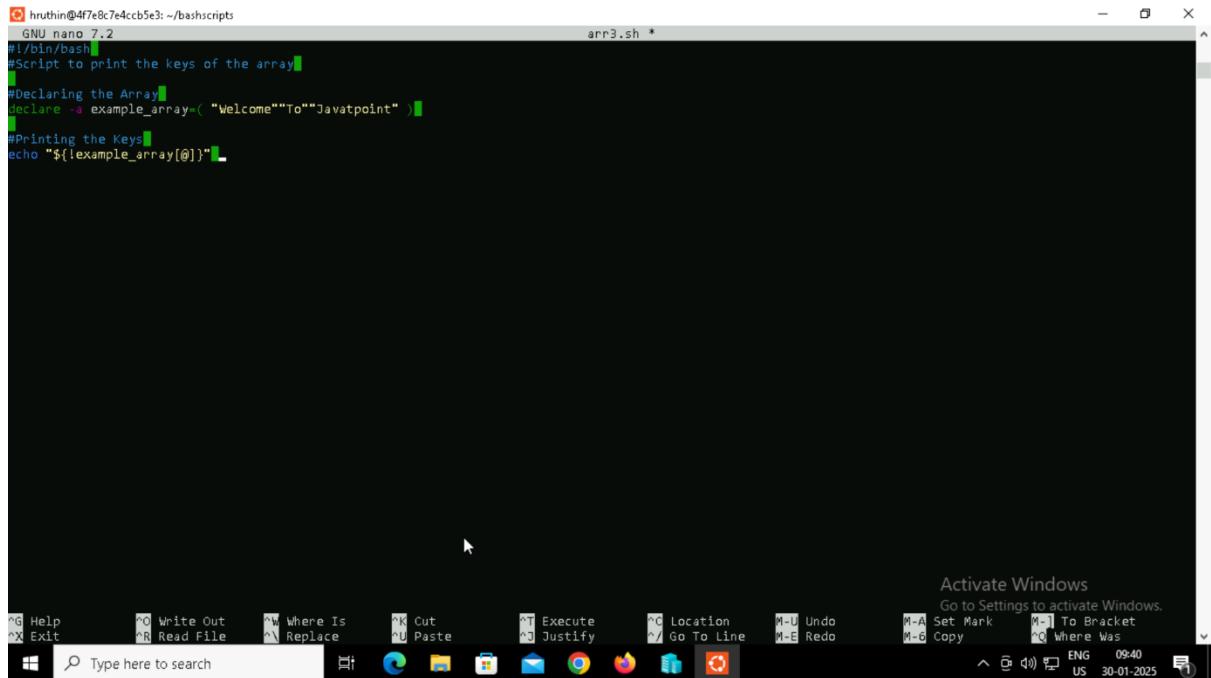
Output:

```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
v1 is A.
v2 is B.
Inside Function
v1 is C.
v2 is D.
After Executing the Function
v1 is A.
v2 is D.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun5.sh
./fun5.sh: line 1: 1: command not found
./fun5.sh: line 2: 2: command not found
./fun5.sh: line 3: 3: command not found
./fun5.sh: line 4: syntax error near unexpected token `('
./fun5.sh: line 4: 4: print_it () {
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun5.sh
Hello User
Hello Reader
The previous function returned a value of 5
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun6.sh
Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun7.sh
[01-30 04:03:33] : Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr1.sh

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr2.sh
Welcome To Javatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 3: Printing keys

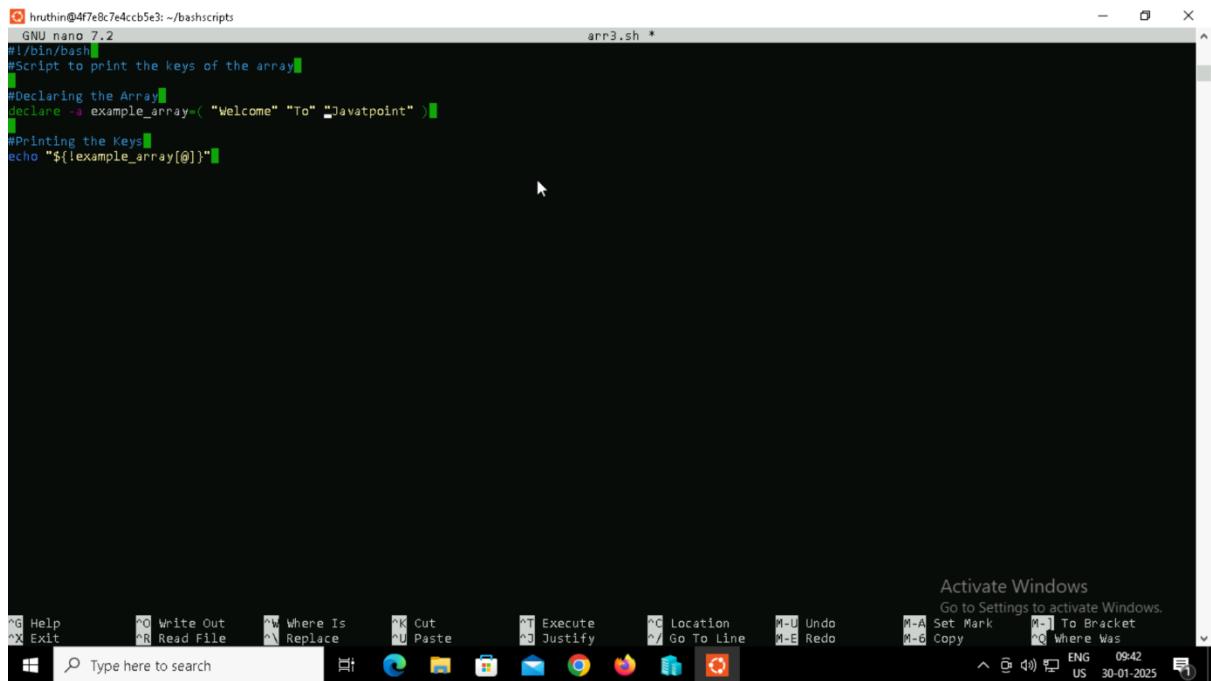
Code:



```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
GNU nano 7.2                                         arr3.sh *
#!/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[@]}"
```

The screenshot shows a terminal window titled "arr3.sh *". The window contains the provided Bash script. The script declares an array named "example_array" with three elements: "Welcome", "To", and "Javatpoint". It then prints the entire array using the command "echo \${example_array[@]}". The terminal window has a dark background and light-colored text. At the bottom, there is a Windows-style taskbar with icons for various applications like File Explorer, Edge, and Google Chrome. The system tray shows the date as 30-01-2025 and the time as 09:40.

Output:



```
hruthin@4f7e8c7e4ccb5e3: ~/bashscripts
GNU nano 7.2                                         arr3.sh *
#!/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[@]}"
```

The screenshot shows the same terminal window as before, but now it displays the output of the script. The output consists of the three elements of the array, each on a new line: "Welcome", "To", and "Javatpoint". The terminal window and its surroundings remain the same as in the previous screenshot.

Example 4: Finding array length

Code:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/bash
#Declaring the Array
declare -a example_array=( "Welcome" "To" "Javatpoint" )
#Printing Array Length
echo "The array contains ${#example_array[@]} elements"
```

The screenshot shows a terminal window titled "arr4.sh *". The window contains the provided Bash script code. The terminal interface includes a menu bar with options like Help, Write Out, Read File, Where Is, Cut, Paste, Execute, Justify, Location, Undo, Redo, Set Mark, To Bracket, Copy, and Where Was. Below the menu is a toolbar with icons for file operations. At the bottom, there's a search bar labeled "Type here to search" and a system tray showing the date and time as "30-01-2025".

Output:

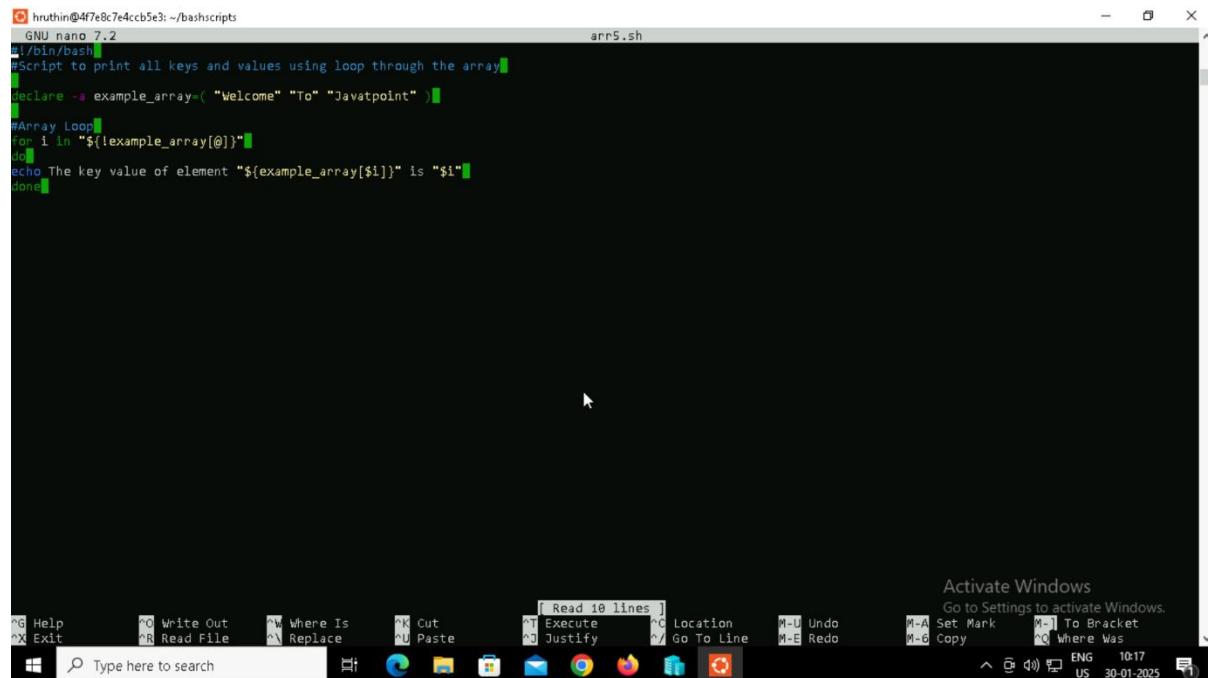
```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
./fun5.sh: line 2: 2.: command not found
./fun5.sh: line 3: 3.: command not found
./fun5.sh: line 4: syntax error near unexpected token `('
./fun5.sh: line 4: `4. print_it () { '
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun6.sh
Hello User
Hello Reader
The previous function returned a value of 5
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun6.sh
Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun7.sh
[01-30 04:03:33] : Welcome to Javatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod a+x arr1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr1.sh

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr1.sh
Javatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr2.sh
Welcome To Javatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr3.sh
0
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr3.sh
0 1 2
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr4.sh
The array contains 3 elements
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

The screenshot shows a terminal window titled "arr4.sh *". The window displays the execution of the Bash script. It shows various stages of script development, including failed attempts at defining functions and arrays, successful nano edits, chmod changes, and finally the execution of the script which prints the array length. The terminal interface is identical to the one in the code screenshot, including the menu bar, toolbar, and system tray.

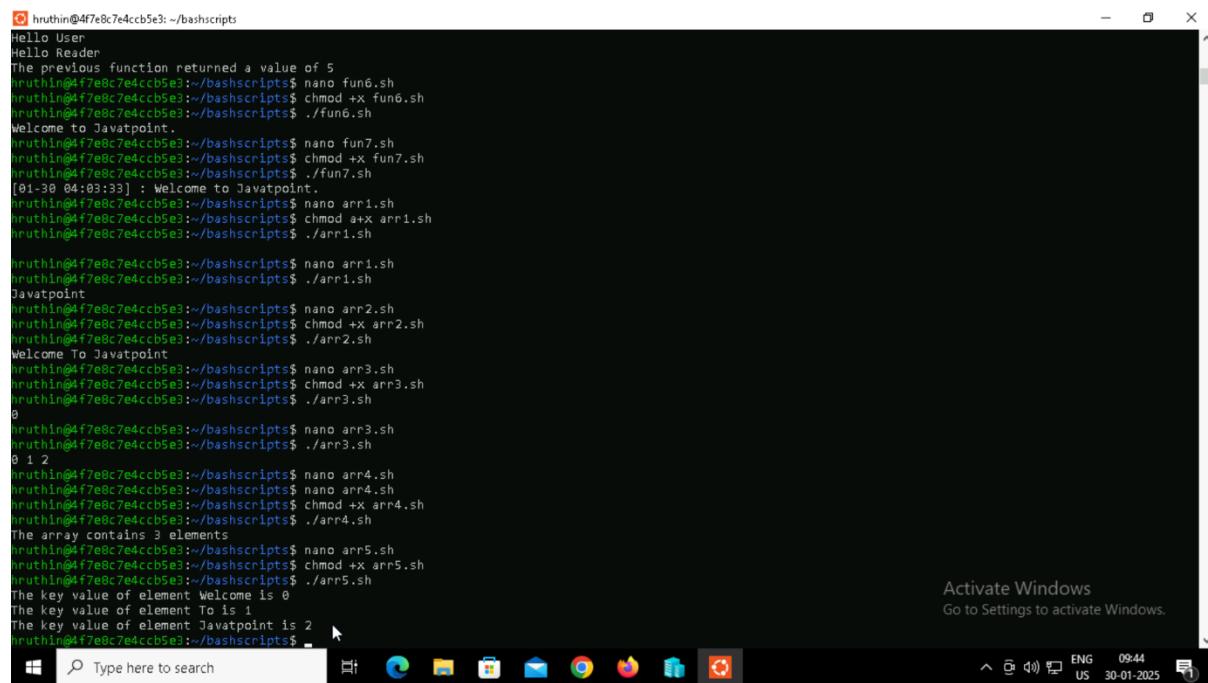
Example 5: Looping through the array

Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/bash
#Script to print all keys and values using loop through the array
declare -a example_array=( "Welcome" "To" "Javaatpoint" )
#Array Loop
for i in "${example_array[@]}"
do
echo The key value of element "${example_array[$i]}" is "$i"
done
```

Output:

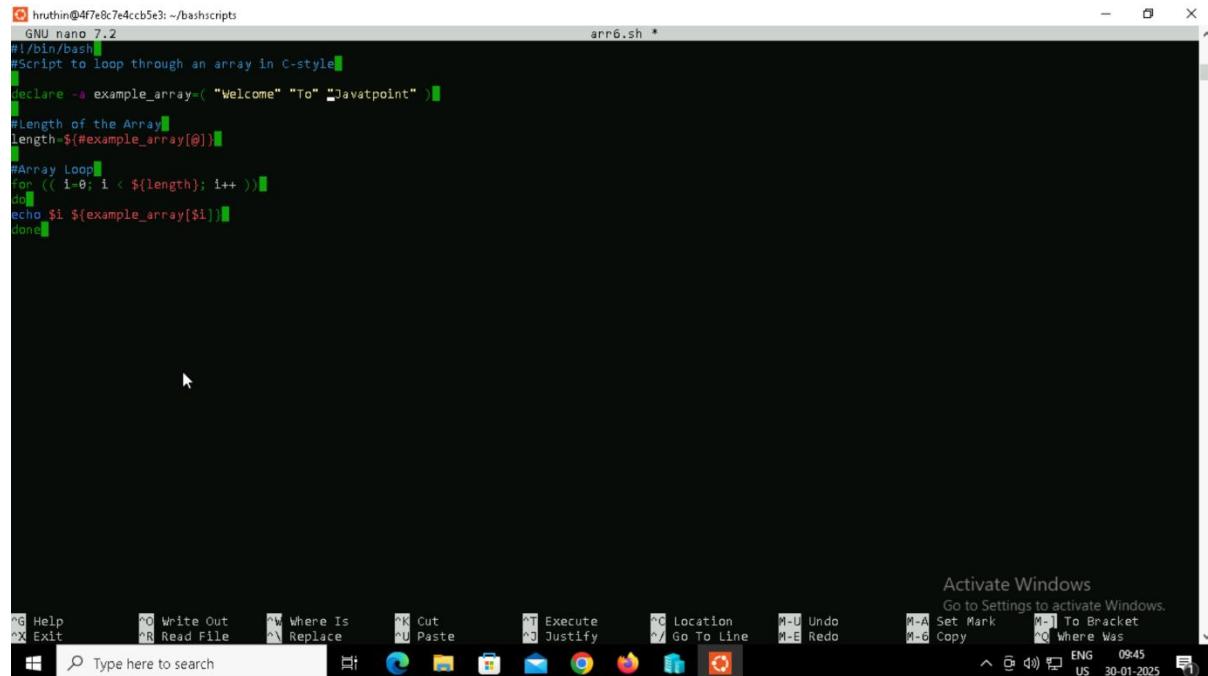


```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
Hello User
Hello Reader
The previous function returned a value of 5
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun6.sh
Welcome to Javaatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun7.sh
[01-30 04:03:33] : Welcome to Javaatpoint.
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr1.sh

hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr1.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr1.sh
Javaatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr2.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr2.sh
Welcome To Javaatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr3.sh
0
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr3.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr3.sh
0 1 2
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr4.sh
The array contains 3 elements
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr5.sh
The key value of element Welcome is 0
The key value of element To is 1
The key value of element Javaatpoint is 2
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

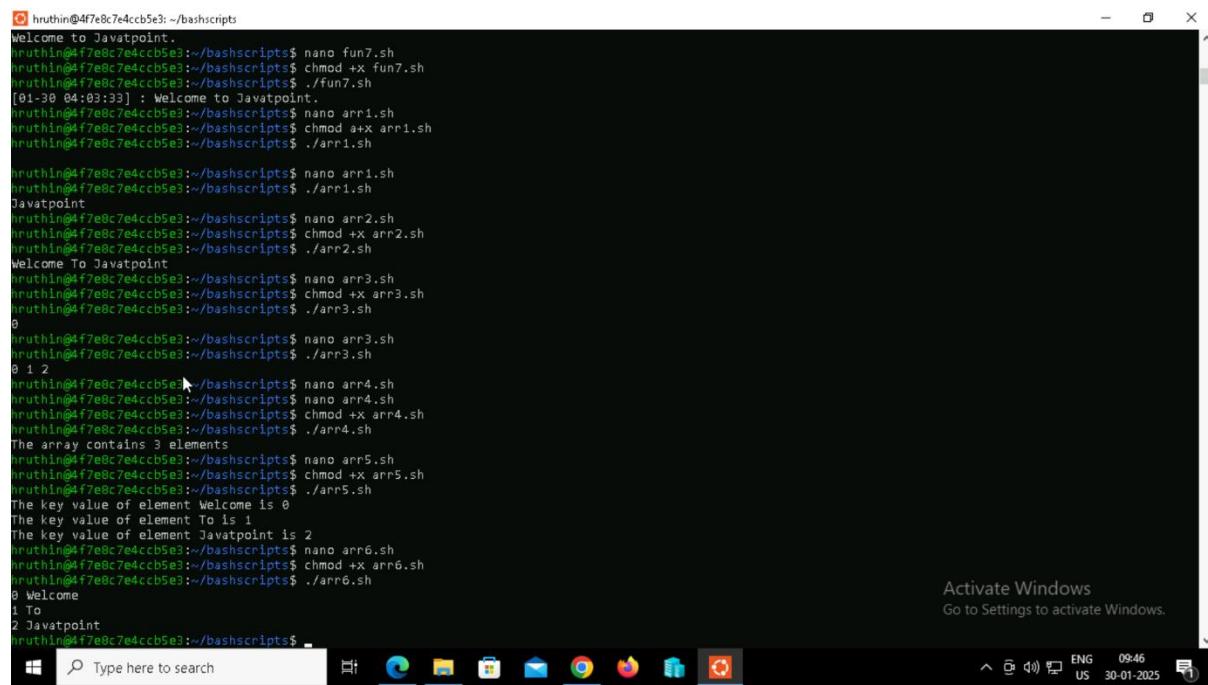
Example 6:

Code:



```
GNU nano 7.2
#!/bin/bash
#Script to loop through an array in C-style
declare -a example_array=( "Welcome" "To" "Javatpoint" )
length=${#example_array[@]}
#Array Loop
for (( i=0; i < ${length}; i++ )) ; do
echo ${example_array[$i]}
done
```

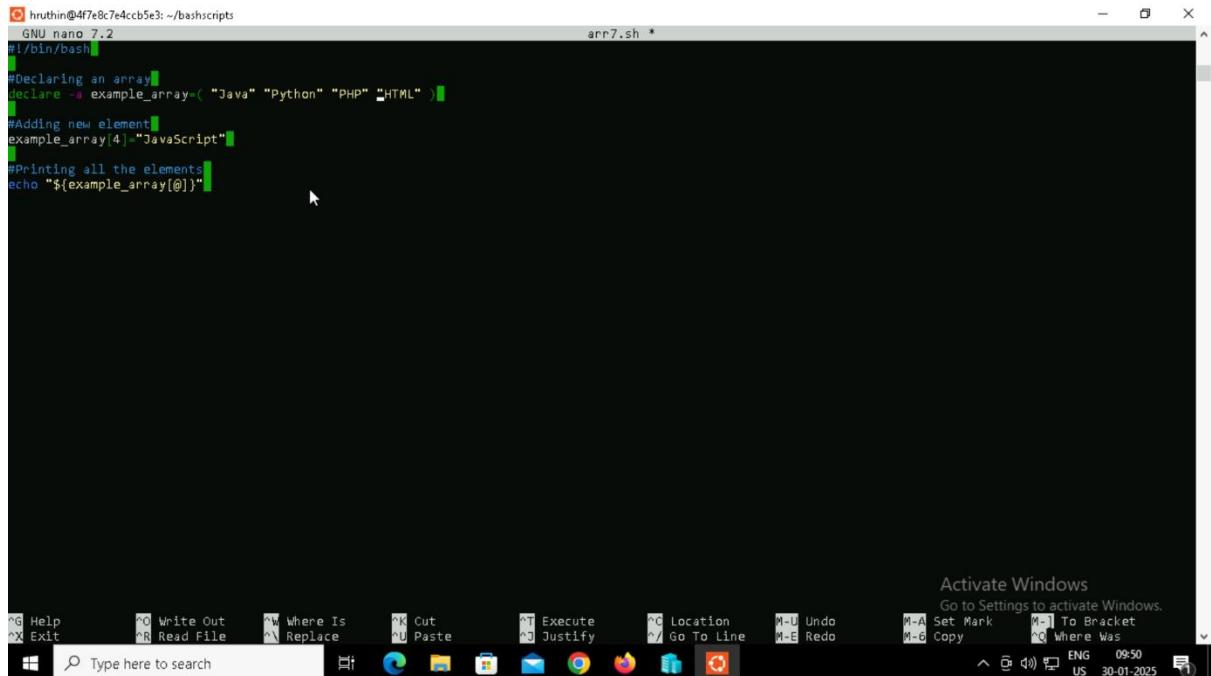
Output:



```
Welcome to Javatpoint.
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano fun7.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x fun7.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./fun7.sh
[01-30 04:03:33] : Welcome to Javatpoint.
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr1.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod a+x arr1.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr1.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr1.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr1.sh
Javatpoint
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr2.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr2.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr2.sh
Welcome To Javatpoint
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr3.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr3.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr3.sh
0
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr3.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr3.sh
0 1 2
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr4.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr4.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr4.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr4.sh
The array contains 3 elements
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr5.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr5.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr5.sh
The key value of element Welcome is 0
The key value of element To is 1
The key value of element Javatpoint is 2
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr6.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr6.sh
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr6.sh
0 Welcome
1 To
2 Javatpoint
nruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 7: Adding Elements

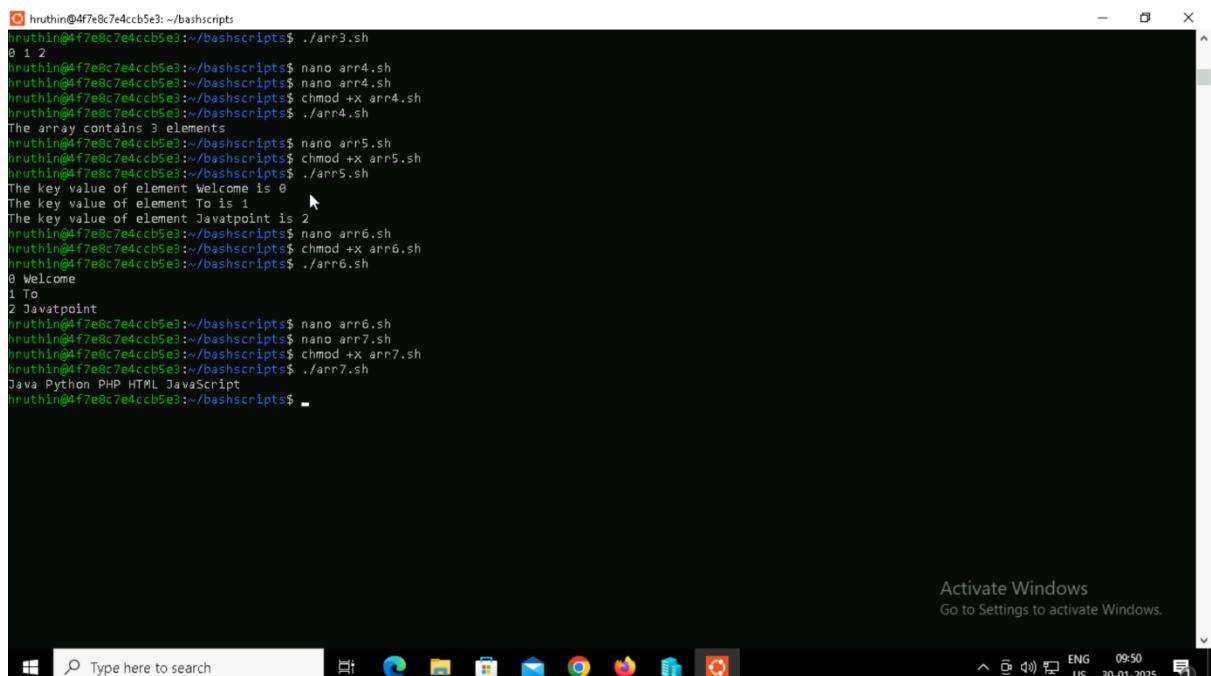
Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
$ nano arr7.sh
GNU nano 7.2
#!/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[@]}"


```

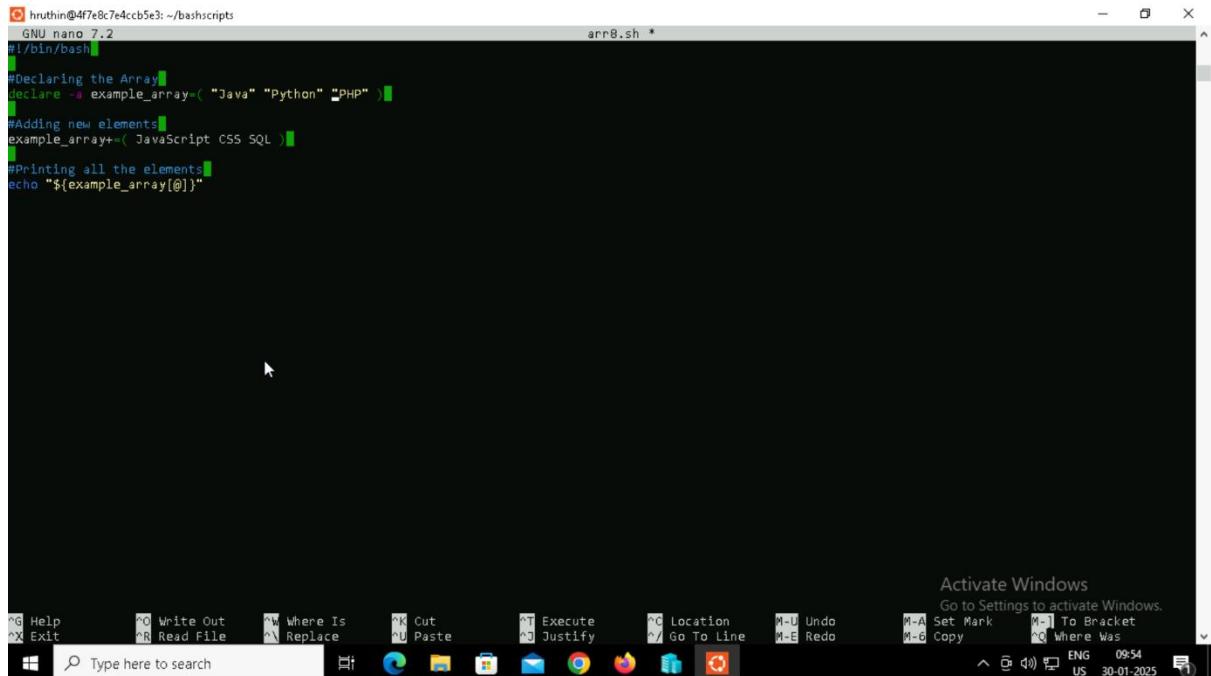
Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr7.sh
0 1 2
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr4.sh
The array contains 3 elements
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr5.sh
The key value of element Welcome is 0
The key value of element To is 1
The key value of element Javatpoint is 2
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr6.sh
0 Welcome
1 To
2 Javatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr7.sh
Java Python PHP HTML JavaScript
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 8:

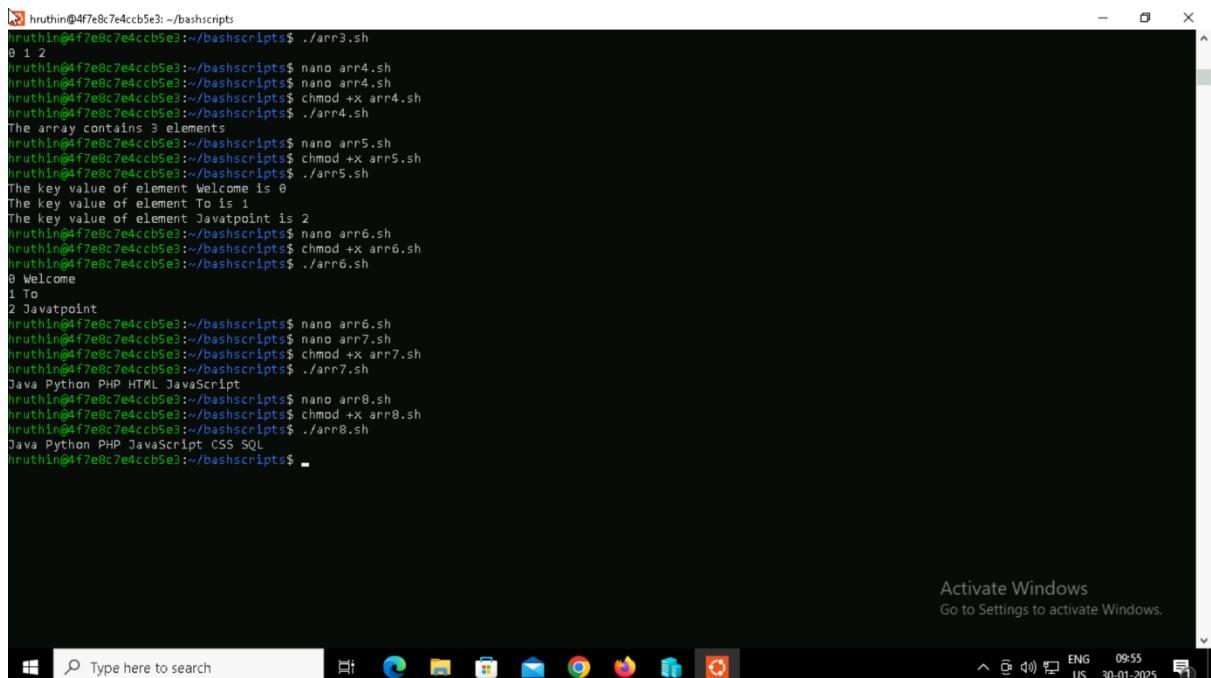
Code:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
$ nano arr8.sh
GNU nano 7.2
#!/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[@]}"


```

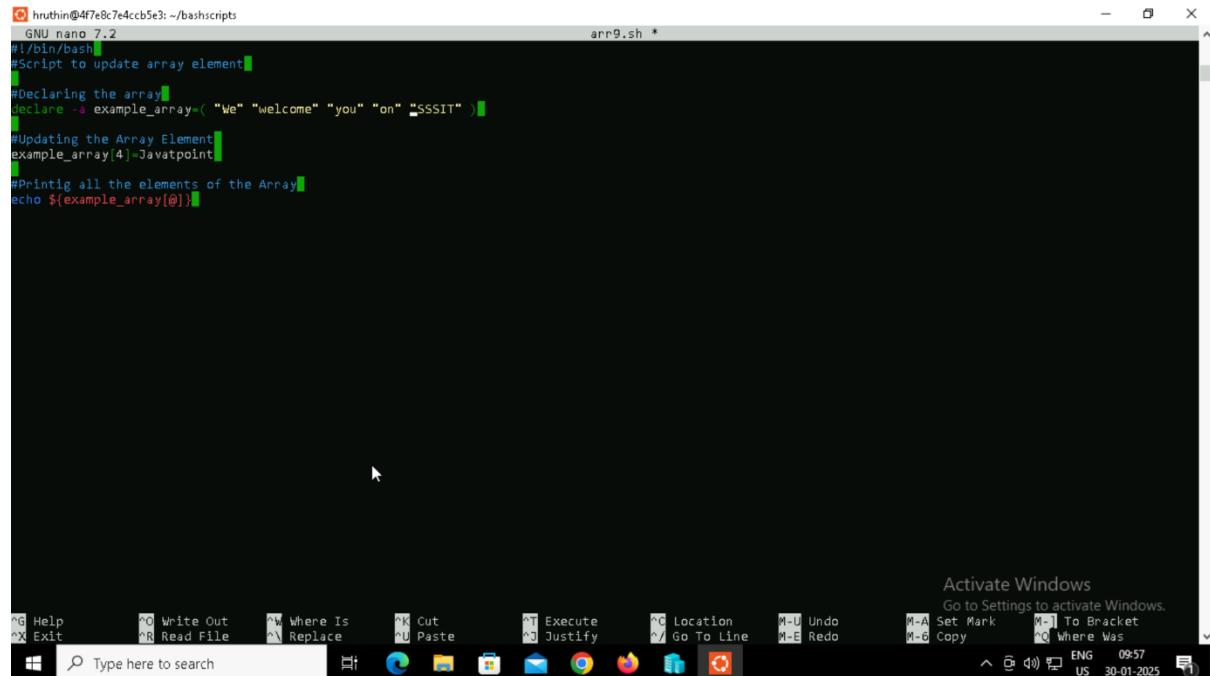
Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr8.sh
0 1 2
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr4.sh
The array contains 3 elements
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr5.sh
The key value of element Welcome is 0
The key value of element To is 1
The key value of element Javatpoint is 2
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr6.sh
0 Welcome
1 To
2 Javatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr7.sh
Java Python PHP HTML JavaScript
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr8.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr8.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr8.sh
Java Python PHP JavaScript CSS SQL
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 9: updating an element

Code:



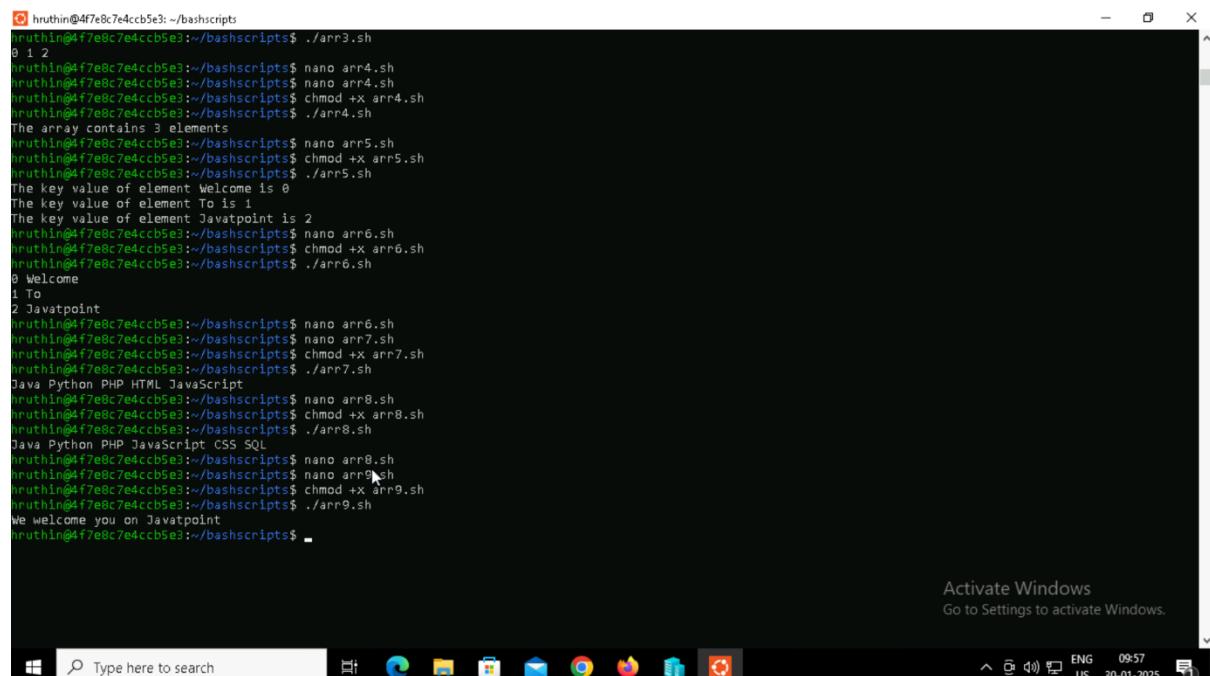
```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
GNU nano 7.2
#!/bin/Bash
#Script to update array element

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

#Updating the Array Element
example_array[4]=Javatpoint

#Printint all the elements of the Array
echo ${example_array[@]}
```

Output:



```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr3.sh
0 1 2
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr4.sh
The array contains 3 elements
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr5.sh
The key value of element Welcome is 0
The key value of element To is 1
The key value of element Javatpoint is 2
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr6.sh
0 Welcome
1 To
2 Javatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr7.sh
Java Python PHP HTML JavaScript
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr8.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr8.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr8.sh
Java Python PHP JavaScript CSS SQL
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr8.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr9.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr9.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr9.sh
We welcome you on Javatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

Example 10: Slicing an array

Code:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts
$ nano arr10.sh
#!/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
```

The terminal window shows the script content. The title bar says "arr10.sh". The status bar at the bottom right shows "Activate Windows", "Go to Settings to activate Windows.", "ENG 14:25", "US 30-01-2025", and a battery icon.

Output:

```
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr3.sh
0 1 2
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr4.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr4.sh
The array contains 3 elements
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr5.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr5.sh
The key value of element Welcome is 0
The key value of element To is 1
The key value of element Javatpoint is 2
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr6.sh
0 Welcome
1 To
2 Javatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr6.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr7.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr7.sh
Java Python PHP HTML JavaScript
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr8.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr8.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr8.sh
Java Python PHP JavaScript CSS SQL
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr8.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr9.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr9.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr9.sh
We welcome you on Javatpoint
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ nano arr10.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ chmod +x arr10.sh
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$ ./arr10.sh
Python
HTML
CSS
hruthin@4f7e8c7e4ccb5e3:~/bashscripts$
```

The terminal window shows the execution of the script and its output. The title bar says "arr10.sh". The status bar at the bottom right shows "Activate Windows", "Go to Settings to activate Windows.", "ENG 10:00", "US 30-01-2025", and a battery icon.

