

BASH Case

Q. Basic Case operation

The screenshot shows a Windows desktop environment with two terminal windows. The top window is a terminal session titled 'script1.sh *' showing the source code of a Bash script named 'script1.sh'. The script uses a 'case' statement to respond to user input ('Yes/No') about Java programming. The bottom window is another terminal session showing the execution of the script, where it asks if the user knows Java programming, and if 'Yes' is entered, it responds with 'That's amazing'.

```
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"
        ;;
    No|no|N|n)
        echo "It's easy. Let's start learning"
        ;;
esac
```

```
kpkm@c99afc4984265e2:~/scripting/bash-case$ nano script1.sh
kpkm@c99afc4984265e2:~/scripting/bash-case$ ./script1.sh
Do you know Java programming?
Yes/No?yes
That's amazing

kpkm@c99afc4984265e2:~/scripting/bash-case$ ./script1.sh
Do you know Java programming?
Yes/No?no
It's easy. Let's start learning
kpkm@c99afc4984265e2:~/scripting/bash-case$
```

Q. Case operation with default case

```
GNU nano 7.2                               script2.sh *
#!/bin/bash

#Script for case operation with default case

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 a lot of applications"
        echo ;;
    Linux|linux)
        echo "You might be serious about security!"
        echo ;;
    *)
        echo "Sounds interesting. I will try that"
        echo ;;
esac
```

```
kpkm@99afc4984265e2:~/scripting/bash-case$ nano script2.sh
kpkm@99afc4984265e2:~/scripting/bash-case$ chmod +x script2.sh
kpkm@99afc4984265e2:~/scripting/bash-case$ ./script2.sh
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name:Linux
You might be serious about security!

kpkm@99afc4984265e2:~/scripting/bash-case$ ./script2.sh
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name:Windows
That's common. You should try something new

kpkm@99afc4984265e2:~/scripting/bash-case$
```

BASH For Loop

Q. Basic for loop

The image shows a Windows desktop environment with two terminal windows open. The top window is a terminal session titled 'script1.sh *' showing the code for a Bash script. The bottom window is another terminal session showing the execution and output of the script.

Top Terminal Window (script1.sh):

```
GNU nano 7.2
#!/bin/bash
#Basic for loop
learn="Start learning from Javatpoint"
for learn in $learn
do
echo $learn
done
echo "Thank You"-
```

Bottom Terminal Window (Execution and Output):

```
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$ nano script1.sh
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$ chmod +x script1.sh
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$ ./script1.sh
Start
learning
from
Javatpoint
Thank You
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$
```

Q. For loop to print numbers from 1-10

```
pkmm@c99afc4984265e2:~/scripting/bash-for-loop
GNU nano 7.2
#To print a series of numbers from 1 to 10
#!/bin/bash

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

echo "Series of numbers from 1 to 10"-
```

```
pkmm@c99afc4984265e2:~/scripting/bash-for-loop$ nano script2.sh
pkmm@c99afc4984265e2:~/scripting/bash-for-loop$ chmod +x script2.sh
pkmm@c99afc4984265e2:~/scripting/bash-for-loop$ ./script2.sh
1
2
3
4
5
6
7
8
9
10
Series of numbers from 1 to 10
pkmm@c99afc4984265e2:~/scripting/bash-for-loop$
```

Q. For loop to read a range with increment

The image shows a Windows desktop environment with three terminal windows open, illustrating the creation and execution of a Bash script.

Terminal 1: A terminal window titled "script3.sh *". It displays the following code:

```
GNU nano 7.2
#!/bin/bash
#For loop to read a range with increment
for num in {1..10..1}
do
echo $num
done
```

Terminal 2: A terminal window showing the script being created and saved:kpkm@c99afc4984265e2:~/scripting/bash-for-loop\$ nano script3.sh
kpkm@c99afc4984265e2:~/scripting/bash-for-loop\$ chmod +x script3.sh
command 'chmod' not found, did you mean:
 command 'chmod' from deb coreutils (9.4-2ubuntu2)
Try: sudo apt install <deb name>
kpkm@c99afc4984265e2:~/scripting/bash-for-loop\$ chmod +x script3.sh
kpkm@c99afc4984265e2:~/scripting/bash-for-loop\$./script3.sh

Terminal 3: A terminal window showing the output of the script:1
2
3
4
5
6
7
8
9
10

Q. For loop to read a range with decrement

The image shows a Windows desktop environment with three terminal windows open, illustrating the execution of a bash script named `script4.sh`.

Terminal 1: A terminal window titled "script4.sh *". It displays the contents of the script file:

```
GNU nano 7.2
#For loop to read a range with decrement
#!/bin/bash

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

Terminal 2: A terminal window showing the command-line interface. It starts with the user's prompt, then runs the script, and finally shows the output of the for loop:

```
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$ nano script4.sh
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$ chmod +x script4.sh
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$ ./script4.sh
10
9
8
7
6
5
4
3
2
1
0
```

Terminal 3: A terminal window showing the command-line interface. It starts with the user's prompt, then runs the script, and finally shows the output of the for loop:

```
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$
```

Q. For loop to read array variables

```
GNU nano 7.2
#For loop to read array variables
#!/bin/bash

arr=( "Welcome""to""Javatpoint" )

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

```
kpkm@99afc4984265e2:~/scripting/bash-for-loop$ nano script5.sh
kpkm@99afc4984265e2:~/scripting/bash-for-loop$ chmod +x script5.sh
kpkm@99afc4984265e2:~/scripting/bash-for-loop$ ./script5.sh
Welcome to Javatpoint
kpkm@99afc4984265e2:~/scripting/bash-for-loop$
```

Q. For loop to read white spaces in string as word separators

```
GNU nano 7.2
#For loop to read white spaces in string as word separators
#!/bin/bash

str="Let's start learning from Javatpoint"

for i in $str;
do
echo "$i"
done
```

```
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$ nano script6.sh
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$ chmod +x script6.sh
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$ ./script6.sh
Let's
start
learning
from
Javatpoint
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$
```

Q. For loop to read each line in string as a word

```
pkm@99afc4984265e2:~/scripting/bash-for-loop
GNU nano 7.2
#For loop to read each line in string as a word
#!/bin/bash

str="Let's start learning from Javatpoint"

for i in "$str";
do
echo "$i"
done
```

```
pkm@99afc4984265e2:~/scripting/bash-for-loop$ nano script7.sh
pkm@99afc4984265e2:~/scripting/bash-for-loop$ chmod +x script7.sh
pkm@99afc4984265e2:~/scripting/bash-for-loop$ ./script7.sh
Let's start learning from Javatpoint
pkm@99afc4984265e2:~/scripting/bash-for-loop$
```

Q. For loop to read three expression

```
GNU nano 7.2
#For loop to read three expression
#!/bin/bash

for ((i=1;i<=10;i++))
do
echo "$i"
done
```

```
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$ nano script8.sh
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$ chmod +x script8.sh
kpkm@c99afc4984265e2:~/scripting/bash-for-loop$ ./script8.sh
1
2
3
4
5
6
7
8
9
10
```

Q. Table of 2 using for loop

```
GNU nano 7.2                                     script9.sh *
```

```
#Table of 2
#!/bin/bash

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

```
^G Help      ^O Write Out   ^W Where Is    ^K Cut        ^T Execute   ^C Location   M-U Undo   M-A Set Mark   M-J To Bracket
^X Exit      ^R Read File   ^P Replace    ^U Paste     ^J Justify   ^Y Go To Line M-E Redo   M-B Copy      M-Q Where Was
                                         Type here to search
```

```
[pkm@c99afc4984265e2:~/scripting/bash-for-loop]$ nano script9.sh
[pkm@c99afc4984265e2:~/scripting/bash-for-loop]$ chmod +x script9.sh
[pkm@c99afc4984265e2:~/scripting/bash-for-loop]$ ./script9.sh
```

```
2
4
6
8
10
12
14
16
18
20
```

```
[pkm@c99afc4984265e2:~/scripting/bash-for-loop]$
```

```
^G Help      ^O Write Out   ^W Where Is    ^K Cut        ^T Execute   ^C Location   M-U Undo   M-A Set Mark   M-J To Bracket
^X Exit      ^R Read File   ^P Replace    ^U Paste     ^J Justify   ^Y Go To Line M-E Redo   M-B Copy      M-Q Where Was
                                         Type here to search
```

```
[pkm@c99afc4984265e2:~/scripting/bash-for-loop]$
```

Q. Numbers from 1 to 20, ignoring from 6-15 using continue

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

Top Terminal Window:

```
GNU nano 7.2
#!/bin/bash
#Numbers from 1 to 20, ignoring from 6-15 using continue
for ((i=1;i<=20;i++));
do
if [[ $i -gt 5 && $i -lt 16 ]];
then
continue
fi
echo $i
done
```

Bottom Terminal Window:

```
pkm@99afc4984265e2:~/scripting/bash-for-loop$ nano script10.sh
pkm@99afc4984265e2:~/scripting/bash-for-loop$ chmod +x script10.sh
pkm@99afc4984265e2:~/scripting/bash-for-loop$ ./script10.sh
1
2
3
4
5
16
17
18
19
20
```

BASH While Loop

Q. Script to print series of numbers as per user input

The screenshot shows a Windows desktop environment with three terminal windows open, illustrating the execution of a Bash script to print a series of numbers based on user input.

Terminal 1: A nano editor window titled "script1.sh" containing the following code:

```
GNU nano 7.2
#!/bin/bash
#Script to print series of numbers as per user input
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 required sequence"
```

Terminal 2: A command-line window showing the script being created and executed:

```
kpkm@99afc4984265e2:~/scripting/bash-while-loop$ nano script1.sh
kpkm@99afc4984265e2:~/scripting/bash-while-loop$ chmod +x script1.sh
kpkm@99afc4984265e2:~/scripting/bash-while-loop$ ./script1.sh
Enter starting number:23
Enter ending number:32
23
24
25
26
27
28
29
30
31
32
This is the required sequence
```

Terminal 3: A command-line window showing the script being run again:

```
kpkm@99afc4984265e2:~/scripting/bash-while-loop$
```

Q. While loop with multiple conditions

The screenshot shows a Windows desktop environment with two terminal windows open. The top window displays a bash script named `script2.sh` containing code to read user input for starting and ending numbers, then print a sequence of numbers from the start to the end. The bottom window shows the execution of the script, including compilation with `chmod +x`, running with `./script2.sh`, and displaying the resulting sequence of numbers from 34 to 54.

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

#While loop with multiple conditions
#while loop with multiple conditions

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"_


```

```
kpkm@99afc4984265e2:~/scripting/bash-while-loop$ nano script2.sh
kpkm@99afc4984265e2:~/scripting/bash-while-loop$ chmod +x script2.sh
kpkm@99afc4984265e2:~/scripting/bash-while-loop$ ./script2.sh
Enter starting number:34
Enter ending number:54
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
This is the sequence that you wanted
kpkm@99afc4984265e2:~/scripting/bash-while-loop$
```

Q. Script for infinite while loop

The screenshot shows a Windows desktop environment. At the top, there is a taskbar with various icons including File Explorer, Microsoft Edge, and File Manager. Below the taskbar is a system tray with icons for battery, signal strength, and volume.

The main area of the screen is occupied by a terminal window titled "script3.sh *". The terminal displays the following content:

```
kpkm@c99afc4984265e2:~/scripting/bash-while-loop
# Script for infinite while loop
#!/bin/bash

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

Below the terminal window, the desktop background is visible, showing a dark-themed wallpaper. The bottom of the screen features a second taskbar with a search bar, a file icon, and several pinned application icons.

Q. While loop with break statement

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

#While loop with break statement
#while loop with 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
```

```
kpkm@c99afc4984265e2:~/scripting/bash-while-loop$ nano script4.sh
kpkm@c99afc4984265e2:~/scripting/bash-while-loop$ chmod +x script4.sh
kpkm@c99afc4984265e2:~/scripting/bash-while-loop$ ./script4.sh
countdown for website launching...
10
9
8
7
6
5
4
3
Mission Aborted, Some technical error found
kpkm@c99afc4984265e2:~/scripting/bash-while-loop$
```

Q. While loop with continue statement

The screenshot shows a Windows desktop environment with two terminal windows. The top window is a terminal session titled "script5.sh *". It displays a Bash script named "script5.sh" containing a while loop. The loop increments a variable \$i from 1 to 10. If \$i equals 5, it uses the "continue" command to skip the rest of the loop iteration. The script then prints the current value of \$i and continues the loop. The bottom window is another terminal session showing the execution of the script. It starts by nanoing the file, chmodding it to executable, and then running it. The output shows the numbers 1 through 4, then 6 through 11, with the number 5 omitted due to the "continue" statement.

```
GNU nano 7.2
#!/bin/bash
#While loop with continue statement
#while [ $i -le 10 ]
do
((i++))
if [[ "$i" == 5 ]];
then
continue
fi
echo "Current Number: $i"
done
echo "Skipped number 5 using continue statement"

kpkm@c99afc4984265e2:~/scripting/bash-while-loop$ nano script5.sh
kpkm@c99afc4984265e2:~/scripting/bash-while-loop$ chmod +x script5.sh
kpkm@c99afc4984265e2:~/scripting/bash-while-loop$ ./script5.sh
Current Number: 1
Current Number: 2
Current Number: 3
Current Number: 4
Current Number: 6
Current Number: 7
Current Number: 8
Current Number: 9
Current Number: 10
Current Number: 11
Skipped number 5 using continue statement
kpkm@c99afc4984265e2:~/scripting/bash-while-loop$
```

Q. While loop in C style

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

```
kpkm@c99afc4984265e2:~/scripting/bash-while-loop$ nano script6.sh
kpkm@c99afc4984265e2:~/scripting/bash-while-loop$ chmod +x script6.sh
kpkm@c99afc4984265e2:~/scripting/bash-while-loop$ ./script6
./script6: No such file or directory
kpkm@c99afc4984265e2:~/scripting/bash-while-loop$ ./script6.sh
1
2
3
4
5
6
7
8
9
10
```

BASH Until Loop

The image shows a Windows desktop environment with two terminal windows open. The top window is a terminal session titled "script1.sh *". It contains the following code:

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

The bottom window is another terminal session. It shows the command to create the file, change permissions, and run it:

```
pkm@99afc4984265e2:~/scripting/bash-until$ nano script1.sh
pkm@99afc4984265e2:~/scripting/bash-until$ chmod +x script1.sh
pkm@99afc4984265e2:~/scripting/bash-until$ ./script1.sh
```

The output of the script is displayed in the bottom terminal window, showing the numbers 1 through 10, each on a new line.

Q

```
GNU nano 7.2                                         script2.sh *
```

```
#Until loop with multiple conditions
#!/bin/bash

max=5
a=1
b=0

until [[ $a -gt $max || $b -gt $max ]];
do
echo "a=$a & b=$b"
((a++))
((b++))
done
```

```
^G Help      ^O Write Out   ^W Where Is    ^K Cut        ^T Execute   ^C Location   M-U Undo   M-A Set Mark   M-J To Bracket
^X Exit      ^R Read File   ^P Replace    ^U Paste     ^J Justify   ^Y Go To Line  M-E Redo   M-B Copy      M-Q Where Was
Windows Start Type here to search  File Explorer  Mail  Edge  Google Chrome  File History  Task View  Power User  Search  Taskbar Buttons  System Clock  ENG 12:32  US 28-01-2025
```

```
:~$ nano script2.sh
:~$ chmod +x script2.sh
:~$ ./script2.sh
a=1 & b=0
a=2 & b=1
a=3 & b=2
a=4 & b=3
a=5 & b=4
:~$
```

```
Windows Start Type here to search  File Explorer  Mail  Edge  Google Chrome  File History  Task View  Power User  Search  Taskbar Buttons  System Clock  ENG 12:33  US 28-01-2025
```

BASH String

Q. Script to check whether two strings are equal

The screenshot shows a Windows desktop environment with a terminal window open. The terminal window has two tabs: one for the nano editor showing a script and another for a command-line session.

Terminal Tab (Top):

```
kpkm@c99afc4984265e2:~/scripting/bash-string
GNU nano 7.2
#Script to check whether two strings are equal
#!/bin/bash

str1="WelcometoJavatpoint"
str2="javatpoint"

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

Command Line Tab (Bottom):

```
kpkm@c99afc4984265e2:~/scripting/bash-string$ nano script1.sh
kpkm@c99afc4984265e2:~/scripting/bash-string$ chmod +x script1.sh
kpkm@c99afc4984265e2:~/scripting/bash-string$ ./script1.sh
Strings are not equal
kpkm@c99afc4984265e2:~/scripting/bash-string$
```

Q. Script to check whether two strings are equal

```
pkm@c99afc4984265e2:~/scripting/bash-string
$ nano script2.sh
GNU nano 7.2
#Script to check whether two strings are equal
#!/bin/bash

str1="WelcometoJavatpoint"
str2="javatpoint"

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

pkm@c99afc4984265e2:~/scripting/bash-string
$ chmod +x script2.sh
pkm@c99afc4984265e2:~/scripting/bash-string
$ ./script2.sh
Strings are not equal
pkm@c99afc4984265e2:~/scripting/bash-string$
```

Q. Script to check whether two strings are equal

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

str1="WelcometoJavatpoint"
str2="Javatpoint"

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

```
kpkm@c99afc4984265e2:~/scripting/bash-string$ nano script3.sh
kpkm@c99afc4984265e2:~/scripting/bash-string$ chmod +x script3.sh
kpkm@c99afc4984265e2:~/scripting/bash-string$ ./script3.sh
WelcometoJavatpoint is not less than Javatpoint
kpkm@c99afc4984265e2:~/scripting/bash-string$
```

```
Type here to search
```

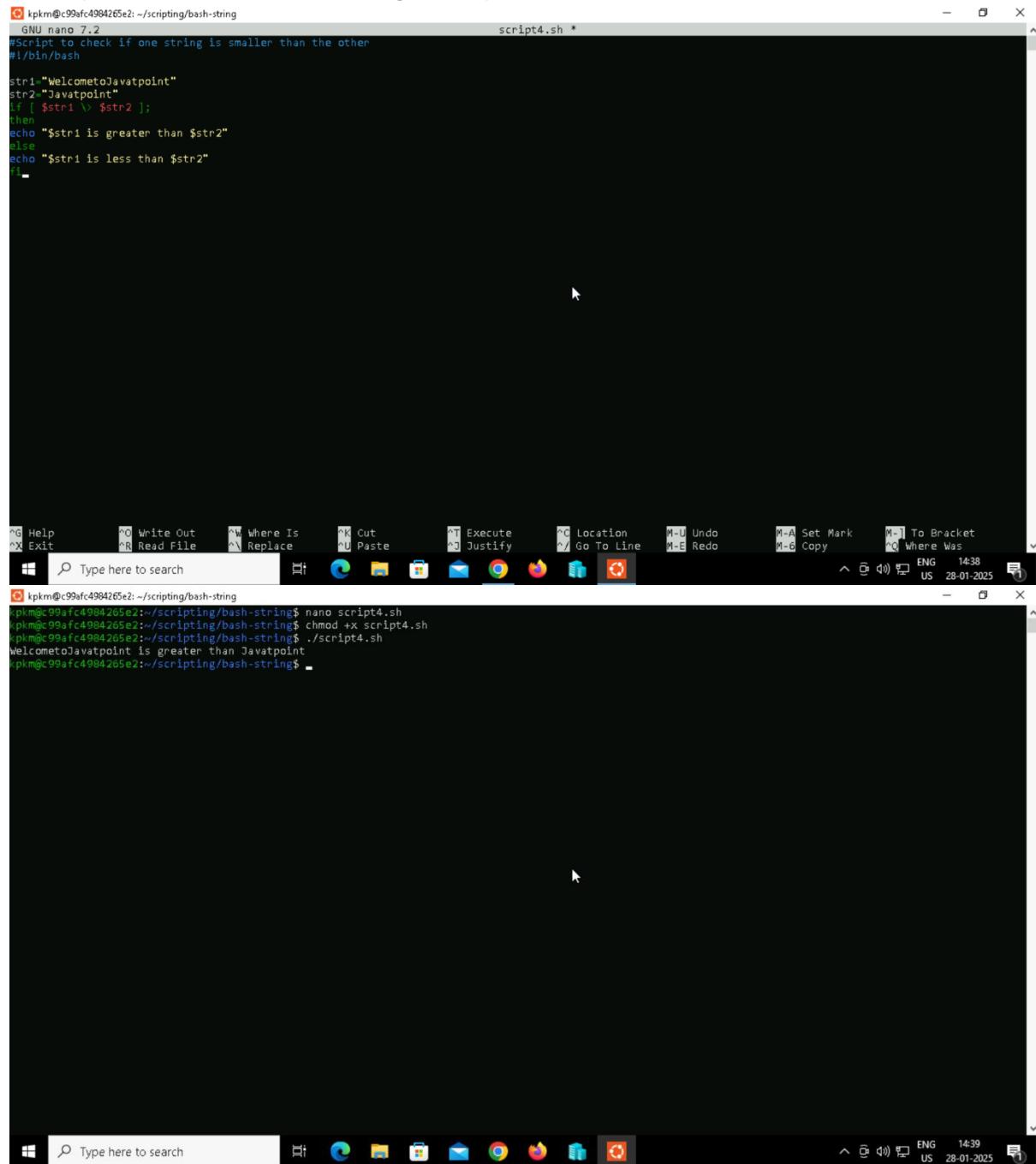
```
File Edit View Insert Cell Help
```

```
ENG 14:36
US 28-01-2025
```

```
File Edit View Insert Cell Help
```

```
ENG 14:37
US 28-01-2025
```

Q. Script to check whether two strings are equal



```
GNU nano 7.2                                     script4.sh *
#!/bin/bash

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

```
kpkm@c99afc4984265e2:~/scripting/bash-string$ nano script4.sh
kpkm@c99afc4984265e2:~/scripting/bash-string$ chmod +x script4.sh
kpkm@c99afc4984265e2:~/scripting/bash-string$ ./script4.sh
WelcometoJavatpoint is greater than Javatpoint
kpkm@c99afc4984265e2:~/scripting/bash-string$
```

```
Type here to search
```

```
PG Help      PO Write Out    PW Where Is    CK Cut          AT Execute    LC Location    MU Undo    MA Set Mark    MB To Bracket
EX Exit      PR Read File   PR Replace    CU Paste       JT Justify    GL Go To Line  M-U Redo    M-B Copy     M-W Where Was
Windows Start  Type here to search  Task View  Start  File  Favorites  Internet Explorer  Mail  Google Chrome  File Explorer  This PC  Recycle Bin  Taskbar Buttons  Search  System  Date/Time  ENG 14:38
US 28-01-2025
```

```
Type here to search
```

```
PKM@c99afc4984265e2:~/scripting/bash-string$ nano script4.sh
PKM@c99afc4984265e2:~/scripting/bash-string$ chmod +x script4.sh
PKM@c99afc4984265e2:~/scripting/bash-string$ ./script4.sh
WelcometoJavatpoint is greater than Javatpoint
PKM@c99afc4984265e2:~/scripting/bash-string$
```

```
Windows Start  Type here to search  Task View  Start  File  Favorites  Internet Explorer  Mail  Google Chrome  File Explorer  This PC  Recycle Bin  Taskbar Buttons  Search  System  Date/Time  ENG 14:39
US 28-01-2025
```

Q. Script to check whether two strings are equal

```
kpkm@c99afc4984265e2:~/scripting/bash-string
GNU nano 7.2
#!/bin/bash
#Script to check if the string length is greater than zero
str="WelcometoJavatpoint"
if [ -n $str ];
then
echo "String is not empty"
else
echo "String is empty"
fi
```

```
kpkm@c99afc4984265e2:~/scripting/bash-string$ nano script5.sh
kpkm@c99afc4984265e2:~/scripting/bash-string$ chmod +x script5.sh
kpkm@c99afc4984265e2:~/scripting/bash-string$ ./script5.sh
String is not empty
kpkm@c99afc4984265e2:~/scripting/bash-string$
```

```
kpkm@c99afc4984265e2:~/scripting/bash-string
Type here to search
Windows Start button  File  Home  Back  Forward  Address bar  Task View  Start  Taskbar  System tray
ENG 14:44
US 28-01-2025
```

```
kpkm@c99afc4984265e2:~/scripting/bash-string$ nano script5.sh
#!/bin/bash
#Script to check if the string length is greater than zero
str="WelcometoJavatpoint"
if [ -n $str ];
then
echo "String is not empty"
else
echo "String is empty"
fi
```

```
kpkm@c99afc4984265e2:~/scripting/bash-string$ chmod +x script5.sh
kpkm@c99afc4984265e2:~/scripting/bash-string$ ./script5.sh
String is not empty
kpkm@c99afc4984265e2:~/scripting/bash-string$
```

```
Type here to search
Windows Start button  File  Home  Back  Forward  Address bar  Task View  Start  Taskbar  System tray
ENG 14:44
US 28-01-2025
```

Q. Script to check whether two strings are equal

```
pkkm@c99afc4984265e2:~/scripting/bash-string
GNU nano 7.2
#To check if the string is equal to zero
#!/bin/bash

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



```
pkkm@c99afc4984265e2:~/scripting/bash-string$ nano script6.sh
pkkm@c99afc4984265e2:~/scripting/bash-string$ chmod +x script6.sh
pkkm@c99afc4984265e2:~/scripting/bash-string$ ./script6.sh
String is empty
pkkm@c99afc4984265e2:~/scripting/bash-string$
```



```
pkkm@c99afc4984265e2:~/scripting/bash-string$ nano script6.sh
GNU nano 7.2
#To check if the string is equal to zero
#!/bin/bash

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



```
pkkm@c99afc4984265e2:~/scripting/bash-string$ chmod +x script6.sh
pkkm@c99afc4984265e2:~/scripting/bash-string$ ./script6.sh
String is empty
pkkm@c99afc4984265e2:~/scripting/bash-string$
```

BASH Find

Q. Script to find the string length in bash

```
pkkm@c99afc4984265e2:~/scripting/bash-find
$ nano script1.sh
GNU nano 7.2
#!/bin/bash
str="Welcome to Javatpoint"
length=${#str}
echo "Length of '$str' is $length"

pkkm@c99afc4984265e2:~/scripting/bash-find$ chmod +x script1.sh
pkkm@c99afc4984265e2:~/scripting/bash-find$ ./script1.sh
Length of 'Welcome to Javatpoint' is 21
pkkm@c99afc4984265e2:~/scripting/bash-find$
```

Q. Script to find the string length in bash

The screenshot shows a Windows desktop environment with two terminal windows open. The top window is a terminal session titled 'script2.sh *' running in a terminal emulator. It displays the following code:

```
GNU nano 7.2
#Bash script to find the length of a string using expr
str="Welcome to Javatpoint"
length= `expr length "$str"`
echo "Length of '$str' is $length"
```

The bottom window is another terminal session showing the execution of the script. The command entered was:

```
pkm@99afc4984265e2:~/scripting/bash-find$ nano script2.sh
pkm@99afc4984265e2:~/scripting/bash-find$ chmod +x script2.sh
pkm@99afc4984265e2:~/scripting/bash-find$ ./script2.sh
Length of 'Welcome to Javatpoint' is 21
pkm@99afc4984265e2:~/scripting/bash-find$
```

The desktop interface includes a taskbar with various icons and a system tray at the bottom.

Q. Script to find the string length in bash

```
GNU nano 7.2                                script3.sh *
#Bash script to find the length of a string
#!/bin/bash
str="Welcome to Javatpoint"
length= `expr "$str":'.'`
echo "Length of '$str' is $length"

pkmm@c99afc4984265e2:~/scripting/bash-find$ nano script3.sh
pkmm@c99afc4984265e2:~/scripting/bash-find$ chmod +x script3.sh
pkmm@c99afc4984265e2:~/scripting/bash-find$ ./script3.sh
Length of 'Welcome to Javatpoint' is Welcome to Javatpoint..
pkmm@c99afc4984265e2:~/scripting/bash-find$ nano script3.sh
pkmm@c99afc4984265e2:~/scripting/bash-find$ ./script3.sh
Length of 'Welcome to Javatpoint' is 21
pkmm@c99afc4984265e2:~/scripting/bash-find$
```

Type here to search

ENG 14:53 28-01-2025

```
pkmm@c99afc4984265e2:~/scripting/bash-find$ nano script3.sh
pkmm@c99afc4984265e2:~/scripting/bash-find$ chmod +x script3.sh
pkmm@c99afc4984265e2:~/scripting/bash-find$ ./script3.sh
Length of 'Welcome to Javatpoint' is Welcome to Javatpoint..
pkmm@c99afc4984265e2:~/scripting/bash-find$ nano script3.sh
pkmm@c99afc4984265e2:~/scripting/bash-find$ ./script3.sh
Length of 'Welcome to Javatpoint' is 21
pkmm@c99afc4984265e2:~/scripting/bash-find$
```

Type here to search

ENG 14:54 28-01-2025

Q. Script to find the string length in bash

The screenshot shows a Windows desktop environment with two terminal windows open. The top window is a terminal session titled "script4.sh *". It contains the following Bash script:

```
GNU nano 7.2
#!/bin/bash
str="Welcome to Javatpoint"
length=$(echo $str | wc -c)
echo "Length of '$str' is $length"
```

The bottom window is another terminal session. The user runs the script and gets the output:

```
pkmm@99afc4984265e2:~/scripting/bash-find$ nano script4.sh
pkmm@99afc4984265e2:~/scripting/bash-find$ chmod +x script4.sh
pkmm@99afc4984265e2:~/scripting/bash-find$ ./script4.sh
Length of 'Welcome to Javatpoint' is 22
pkmm@99afc4984265e2:~/scripting/bash-find$
```

Q. Script to find the string length in bash

The screenshot shows a terminal window titled "script5.sh *". Inside the window, a Bash script is displayed:

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

Below the terminal window, the Windows taskbar is visible, showing various pinned icons and the system tray.

Terminal session details:

```
kpkm@c99afc4984265e2:~/scripting/bash-find$ nano script5.sh
kpkm@c99afc4984265e2:~/scripting/bash-find$ chmod +x script5.sh
kpkm@c99afc4984265e2:~/scripting/bash-find$ ./script5.sh
Length of 'Welcome to Javatpoint' is 21
kpkm@c99afc4984265e2:~/scripting/bash-find$
```

BASH String Split

Q. Split string using a space character delimiter

The screenshot shows a Windows desktop environment with three windows open:

- Top Window:** A terminal window titled "script1.sh *". It contains a Bash script named "script1.sh" which reads a string from standard input and splits it into words using a space as a delimiter. The script then prints each word on a new line.
- Middle Window:** A terminal window showing the output of the script. The user enters the string "welcome to devops basics" and the script outputs each word on a new line: "welcome", "to", "devops", and "basics".
- Bottom Window:** A search bar in the taskbar with the placeholder "Type here to search".

```
#!/usr/bin/bash
#Split string using a space character delimiter
read -p "Enter any string separated by space:" str
IFS=' '
read -ra ADDR <<< "$str"
for i in "${ADDR[@]}";
do
echo "$i"
done
```

```
kpkm@c99afc4984265e2:~/scripting/bash-split-string$ ./script1.sh
Enter any string separated by space:welcome to devops basics
welcome
to
devops
basics
kpkm@c99afc4984265e2:~/scripting/bash-split-string$
```

Q. Split string by symbol

The image shows a Windows desktop environment with two terminal windows open. The top window is a terminal session titled "script2.sh *". It displays the contents of a Bash script named "script2.sh" which reads input from the user, splits it by commas, and prints the name, state, and age separately. The bottom window is another terminal session showing the execution of the script. It starts with the command "nano script2.sh", saves the file, changes permissions with "chmod +x script2.sh", and runs the script with "./script2.sh". The output shows the user input "Enter name,state and age separated by comma:" followed by "parakram,jharkhand,22", and the script output "Name:parakram", "State:jharkhand", and "Age:22". The desktop taskbar at the bottom shows various pinned icons including File Explorer, Edge, and other utility programs.

```
GNU nano 7.2
#Split string by symbol
#!/bin/bash

read -p "Enter name,state and age separated by comma:" entry
IFS=','
read -a strarr<<< "$entry"
echo "Name:${strarr[0]}"
echo "State:${strarr[1]}"
echo "Age:${strarr[2]}"

^G Help      ^O Write Out    ^W Where Is     ^K Cut        ^T Execute     ^C Location    M-U Undo      M-A Set Mark   M-] To Bracket
^X Exit      ^R Read File    ^P Replace     ^U Paste      ^J Justify     ^Y Go To Line  M-E Redo      M-B Copy       M-@ Where Was
^H Type here to search
```

```
kpkm@99afc4984265e2:~/scripting/bash-split-string$ nano script2.sh
kpkm@99afc4984265e2:~/scripting/bash-split-string$ chmod +x script2.sh
kpkm@99afc4984265e2:~/scripting/bash-split-string$ ./script2.sh
Enter name,state and age separated by comma:parakram,jharkhand,22
Name:parakram
State:jharkhand
Age:22
kpkm@99afc4984265e2:~/scripting/bash-split-string$
```

Q. Split string without IFS variable

The screenshot shows a Windows desktop environment with three terminal windows open, illustrating a Bash script to split a string without using the IFS variable.

Top Terminal:

```
>Select kpkm@c99afc4984265e2:~/scripting/bash-split-string
GNU nano 7.2
#Split string without IFS variable
#!/bin/bash

read -p "Enter any string separated by colon()" str
readarray -d : -t strarr<< "$str"
printf"\n"
for (( n=0;n<#${strarr[*]};n++ ))
do
echo "${strarr[n]}"
done
```

Middle Terminal:

```
kpkm@c99afc4984265e2:~/scripting/bash-split-string$ nano script3.sh
kpkm@c99afc4984265e2:~/scripting/bash-split-string$ chmod +x script3.sh
kpkm@c99afc4984265e2:~/scripting/bash-split-string$ ./script3.sh
Enter any string separated by colon():welcome:to:devops:training
./script3.sh: line 6: printf\n: command not found
welcome
to
devops
training
```

Bottom Terminal:

```
kpkm@c99afc4984265e2:~/scripting/bash-split-string$
```

Q. Split string by another string

```
GNU nano 7.2                                     script4.sh *
```

```
#!/bin/bash
#Bash split string by another string
str="WelearnWelcomeLearnYouLearnOnLearnJavaatpoint"
delimiter=Learn
s=$str$delimiter
array=();
while [[ $s ]];
do
array+=("${s%%$delimiter*}");
s=${!#"$delimiter"};
done;
declare -p array
```

```
^G Help      ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo      M-A Set Mark   M-J To Bracket
^X Exit      ^R Read File    ^P Replace     ^U Paste        ^J Justify      ^Y Go To Line   M-E Redo      M-B Copy       M-O Where Was
Windows Start Type here to search  Taskbar icons: File Explorer, File Manager, Mail, Google Chrome, Firefox, Edge, OneDrive, Task View, Taskbar Search, Taskbar Clock, Taskbar Volume, Taskbar Network, Taskbar Battery, Taskbar Language, Taskbar Date and Time
ENG 17:58
US 28-01-2025
```

```
kpkm@e99afc4984265e2:~/scripting/bash-split-string$ nano script4.sh
kpkm@e99afc4984265e2:~/scripting/bash-split-string$ chmod +x script4.sh
kpkm@e99afc4984265e2:~/scripting/bash-split-string$ ./script4.sh
declare -a array=(["0"]="We" ["1"]="Welcome" ["2"]="You" ["3"]="On" ["4"]="Javaatpoint")
kpkm@e99afc4984265e2:~/scripting/bash-split-string$
```

```
Windows Start Type here to search  Taskbar icons: File Explorer, File Manager, Mail, Google Chrome, Firefox, Edge, OneDrive, Task View, Taskbar Search, Taskbar Clock, Taskbar Volume, Taskbar Network, Taskbar Battery, Taskbar Language, Taskbar Date and Time
ENG 17:58
US 28-01-2025
```

Q. String split using trim cmd

The image shows three separate terminal windows on a Windows desktop. Each window has a title bar with the text 'script5.sh *'. The first window shows the script being edited in nano, the second shows the script being run and failing, and the third shows the script running successfully.

```
GNU nano 7.2
#String split using trim cmd
#!/bin/bash

my_str="We;welcome;you;on;Javaatpoint"
my_arr=(${echo $my_str | tr ";" "\n"})
for i in "${my_arr[@]}"
do
echo $i
done
```

```
kpkm@99afc4984265e2:~/scripting/bash-split-string$ ./script5.sh
./script5.sh: line 5: tr:\n: command not found
kpkm@99afc4984265e2:~/scripting/bash-split-string$ nano script5.sh
kpkm@99afc4984265e2:~/scripting/bash-split-string$ ./script5.sh
tr: missing operand after ':'
Two strings must be given when translating.
Try 'tr --help' for more information.
kpkm@99afc4984265e2:~/scripting/bash-split-string$ nano script5.sh
kpkm@99afc4984265e2:~/scripting/bash-split-string$ ./script5.sh
tr: missing operand after ':'
Two strings must be given when translating.
Try 'tr --help' for more information.
kpkm@99afc4984265e2:~/scripting/bash-split-string$ nano script5.sh
kpkm@99afc4984265e2:~/scripting/bash-split-string$ ./script5.sh
We
welcome
you
on
Javaatpoint
```

BASH Substring

Q. Script to extract characters from list

```
#!/usr/bin/bash
#To extract till specific characters from starting
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}"
```

```
kpkm@c99afc4984265e2:~/scripting/bash-substr$ nano script1.sh
kpkm@c99afc4984265e2:~/scripting/bash-substr$ chmod +x script1.sh
kpkm@c99afc4984265e2:~/scripting/bash-substr$ ./script1.sh
String: We welcome you on Javatpoint
Total characters in a String: 28
Substring: We welcome
Total characters in Substring: 10
kpkm@c99afc4984265e2:~/scripting/bash-substr$
```

Q. Script to extract characters from list

The screenshot shows a Windows desktop environment with three terminal windows open, illustrating the creation and execution of a Bash script to extract characters from a string.

Top Terminal:

```
GNU nano 7.2
#!/bin/bash
#To extract from specific character onwards
#Script to print from 11th character onwards
str="We welcome you on Javatpoint."
substr="${str:11}"
echo "$substr"
```

Middle Terminal:

```
kpkm@99afc4984265e2:~/scripting/bash-substr$ nano script2.sh
kpkm@99afc4984265e2:~/scripting/bash-substr$ chmod +x script2.sh
kpkm@99afc4984265e2:~/scripting/bash-substr$ ./script2.sh
you on Javatpoint.
kpkm@99afc4984265e2:~/scripting/bash-substr$
```

Bottom Terminal:

```
kpkm@99afc4984265e2:~/scripting/bash-substr$
```

Q. . Script to extract characters from list

```
GNU nano 7.2                                     script3.sh
#!/bin/bash
#Script to print 11th character of a String
str="We welcome you on Javatpoint."
substr="${str:11:1}"
echo "$substr"

^G Help      ^O Write Out   ^W Where Is    ^K Cut        ^C Read 5 lines
^X Exit      ^R Read File   ^Y Replace    ^U Paste      ^T Execute
^F Find      ^L Location    ^A Undo       ^V Select     ^C Copy
^H Find Next ^J Justify    ^B Redo       ^D Delete    ^M Set Mark
^P Print     ^G Go To Line  ^S Save      ^E Cut/Copy  ^B To Bracket
^Q Quit      ^I Insert     ^Z Undo      ^X Cut/Copy  ^D Where Was

kpkm@k99afc4984265e2:~/scripting/bash-substr$ nano script3.sh
kpkm@k99afc4984265e2:~/scripting/bash-substr$ nano script3.sh
kpkm@k99afc4984265e2:~/scripting/bash-substr$ chmod +x script3.sh
kpkm@k99afc4984265e2:~/scripting/bash-substr$ ./script3.sh
y
kpkm@k99afc4984265e2:~/scripting/bash-substr$
```

Q. Script to extract characters from list

```
GNU nano 7.2                                     script4.sh *
#!/bin/bash
#Script to extract 11 characters from last
str="We welcome you on Javatpoint."
substr="${str: -11}"
echo "$substr"
```

```
kpkm@99afc4984265e2:~/scripting/bash-substr
$ nano script4.sh
$ chmod +x script4.sh
$ ./script4.sh
Javatpoint.
$
```

BASH String Concat

Q. Script to concatenate strings

The screenshot shows a terminal window with three distinct sections. The top section displays the contents of a file named 'script1.sh' created with nano editor. The middle section shows the terminal prompt and the command to make the script executable. The bottom section shows the output of running the script, which concatenates two strings into one.

```
pkkm@c99afc4984265e2:~/scripting/bash-concat
$ nano script1.sh
#!/bin/bash

str1="We welcome you"
str2="on Javatpoint"

str3="$str1$str2"

echo $str3

pkkm@c99afc4984265e2:~/scripting/bash-concat$ chmod +x script1.sh
pkkm@c99afc4984265e2:~/scripting/bash-concat$ ./script1.sh
We welcome youon Javatpoint
pkkm@c99afc4984265e2:~/scripting/bash-concat$
```

Q.

The image shows a Windows desktop environment with two terminal windows open. The top window is a terminal session titled 'script2.sh *' running in nano editor mode. The script content is:

```
#!/bin/bash
str="We welcome you"
echo "$str on Javatpoint"
```

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

```
pkm@99afc4984265e2:~/scripting/bash-concat$ nano script2.sh
pkm@99afc4984265e2:~/scripting/bash-concat$ chmod +x script2.sh
pkm@99afc4984265e2:~/scripting/bash-concat$ ./script2.sh
We welcome you on Javatpoint
pkm@99afc4984265e2:~/scripting/bash-concat$
```

The desktop taskbar at the bottom shows various pinned icons including File Explorer, Edge, File Manager, Mail, and others.

Q.

```
GNU nano 7.2                                         script3.sh *
#Using append operator
#!/bin/bash
echo "Printing the name of the programming languages"
lang ""
for value in 'java' 'python' 'C' 'C++';
do
lang+="$value"
done
echo "$lang"

^G Help      ^O Write Out   ^W Where Is    ^K Cut        ^T Execute   ^C Location   M-U Undo   M-A Set Mark   M-J To Bracket
^X Exit      ^R Read File   ^P Replace    ^U Paste     ^J Justify   ^Y Go To Line  M-E Redo   M-B Copy      M-D Where Was
^H Type here to search
```

```
[pkmc@99afc4984265e2:~/scripting/bash-concat]$ nano script3.sh
[pkmc@99afc4984265e2:~/scripting/bash-concat]$ chmod +x script3.sh
[pkmc@99afc4984265e2:~/scripting/bash-concat]$ ./script3.sh
Printing the name of the programming languages
javapythonC++
[pkmc@99afc4984265e2:~/scripting/bash-concat]$
```

```
^G Help      ^O Write Out   ^W Where Is    ^K Cut        ^T Execute   ^C Location   M-U Undo   M-A Set Mark   M-J To Bracket
^X Exit      ^R Read File   ^P Replace    ^U Paste     ^J Justify   ^Y Go To Line  M-E Redo   M-B Copy      M-D Where Was
^H Type here to search
```

```
[pkmc@99afc4984265e2:~/scripting/bash-concat]$
```

Q.

```
GNU nano 7.2
#!/bin/bash
#Using printf function
str="Welcome"
printf -v new_str "%s to Javatpoint"
echo $new_str
```

```
kpkm@99afc4984265e2:~/scripting/bash-concat$ nano script4.sh
kpkm@99afc4984265e2:~/scripting/bash-concat$ chmod +x script4.sh
kpkm@99afc4984265e2:~/scripting/bash-concat$ ./script4.sh
Welcome to Javatpoint
kpkm@99afc4984265e2:~/scripting/bash-concat$
```

```
Type here to search ENG 17:28
US 29-01-2025
```

```
Type here to search ENG 17:28
US 29-01-2025
```

Q.

```
GNU nano 7.2
#Using literal strings
#!/bin/bash

str="Welcome to"
newstr="${str} Javatpoint"
echo "$newstr"

pkm@99afc4984265e2:~/scripting/bash-concat$ nano script5.sh
pkm@99afc4984265e2:~/scripting/bash-concat$ chmod +x script5.sh
pkm@99afc4984265e2:~/scripting/bash-concat$ ./script5.sh
Welcome to Javatpoint
pkm@99afc4984265e2:~/scripting/bash-concat$
```

Q.

The image consists of three vertically stacked screenshots of a Windows desktop environment. Each screenshot shows a terminal window with a black background and white text, running on a Bash shell.

Screenshot 1: The terminal window title is "script6.sh". The command history shows:

```
pkm@C99AFC4984265E2:~/scripting/bash-concat$ nano script6.sh
pkm@C99AFC4984265E2:~/scripting/bash-concat$ chmod +x script6.sh
pkm@C99AFC4984265E2:~/scripting/bash-concat$ ./script6.sh
Hello_World!
```

Screenshot 2: The terminal window title is "script7.sh". The command history shows:

```
Select pkm@C99AFC4984265E2: ~/scripting/bash-concat
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"
```

Screenshot 3: The terminal window title is "script7.sh". The command history shows:

```
pkm@C99AFC4984265E2:~/scripting/bash-concat$ nano script7.sh
pkm@C99AFC4984265E2:~/scripting/bash-concat$ chmod +x script7.sh
pkm@C99AFC4984265E2:~/scripting/bash-concat$ ./script7.sh
Enter First Name: kpm
Enter State: MP
Enter Age: 21
Name, State, Age: kpm,MP,21
```

The desktop interface includes a taskbar at the bottom with icons for various applications like File Explorer, Edge, and Task View. The system tray shows the date and time as 29-01-2025 and 17:30. The system language is set to English (ENG) and the regional setting is US.

Q.

```
kpkm@kpkm-OptiPlex-5090:~/scripting/bash-concat$ nano script7.sh
kpkm@kpkm-OptiPlex-5090:~/scripting/bash-concat$ chmod +x script7.sh
kpkm@kpkm-OptiPlex-5090:~/scripting/bash-concat$ ./script7.sh
Enter First Name: Kumar
Enter State: Jharkhand
Enter Age: 22
Name, State, Age: Kumar,Jharkhand,22
kpkm@kpkm-OptiPlex-5090:~/scripting/bash-concat$
```

```
kpkm@kpkm-OptiPlex-5090:~/scripting/bash-functions$ nano script1.sh
kpkm@kpkm-OptiPlex-5090:~/scripting/bash-functions$ chmod +x script1.sh
kpkm@kpkm-OptiPlex-5090:~/scripting/bash-functions$ ./script1.sh
Welcome to Javatpoint.
kpkm@kpkm-OptiPlex-5090:~/scripting/bash-functions$
```

```
kpkm@kpkm-OptiPlex-5090:~/scripting/bash-concat$ nano script7.sh
kpkm@kpkm-OptiPlex-5090:~/scripting/bash-concat$ chmod +x script7.sh
kpkm@kpkm-OptiPlex-5090:~/scripting/bash-concat$ ./script7.sh
Enter First Name: Kumar
Enter State: Jharkhand
Enter Age: 22
Name, State, Age: Kumar,Jharkhand,22
kpkm@kpkm-OptiPlex-5090:~/scripting/bash-concat$
```

BASH Functions

Q.

The image shows a Windows desktop environment with two terminal windows open. The top window is titled "script1.sh" and contains the following code:

```
GNU nano 7.2
#!/bin/bash
JTP () {
    echo 'Welcome to Javatpoint.'
}
JTP
```

The bottom window shows the command-line interface with the following session:

```
kpkm@c99afc4984265e2:~/scripting/bash-functions$ nano script1.sh
kpkm@c99afc4984265e2:~/scripting/bash-functions$ chmod +x script1.sh
kpkm@c99afc4984265e2:~/scripting/bash-functions$ ./script1.sh
Welcome to Javatpoint.
kpkm@c99afc4984265e2:~/scripting/bash-functions$
```

The taskbar at the bottom of the screen includes icons for File Explorer, Task View, Edge, File Explorer, Mail, Google Chrome, and File Explorer.

Q.

```
GNU nano 7.2
#!/bin/bash
Function JTP () {
echo 'Welcome to Javatpoint.'
}
JTP
```

```
kpkm@e99afc4984265e2:~/scripting/bash-functions$ nano script2.sh
kpkm@e99afc4984265e2:~/scripting/bash-functions$ chmod +x script2.sh
kpkm@e99afc4984265e2:~/scripting/bash-functions$ ./script2.sh
Welcome to Javatpoint.
kpkm@e99afc4984265e2:~/scripting/bash-functions$
```

```
Type here to search ENG 17:32
US 29-01-2025
```

```
Type here to search ENG 17:33
US 29-01-2025
```

Q.

```
kpkm@kpkm-OptiPlex-5090:~/scripting/bash-functions
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."
```

A screenshot of a Windows desktop environment. At the top, there's a taskbar with various pinned icons including File Explorer, Edge, File Manager, Mail, Google Chrome, and others. The system tray shows the date as 29-01-2025 and the time as 17:33. The language is set to ENG US. Below the taskbar is a search bar with placeholder text "Type here to search". The main area of the screen is a terminal window titled "kpkm@c99afc4984265e2: ~/scripting/bash-functions". The terminal output shows the following commands being run:

```
kpkm@c99afc4984265e2:~/scripting/bash-functions$ nano script3.sh
kpkm@c99afc4984265e2:~/scripting/bash-functions$ chmod +x script3.sh
kpkm@c99afc4984265e2:~/scripting/bash-functions$ ./script3.sh
Welcome you on Javaatpoint.
```

The terminal window has a dark background and light-colored text. The cursor is visible at the bottom center of the screen.

Q.

GNU nano 7.2

```
#!/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."
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark M-J To Bracket
^X Exit ^R Read File ^Y Replace ^U Paste ^J Justify ^V Go To Line M-E Redo M-B Copy M-Q Where Was

Type here to search

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark M-J To Bracket
^X Exit ^R Read File ^Y Replace ^U Paste ^J Justify ^V Go To Line M-E Redo M-B Copy M-Q Where Was

ENG 17:34 US 29-01-2025

```
kpkm@c99afc4984265e2:~/scripting/bash-functions$ nano script4.sh
kpkm@c99afc4984265e2:~/scripting/bash-functions$ chmod +x script4.sh
kpkm@c99afc4984265e2:~/scripting/bash-functions$ ./script4
./script4: No such file or directory
kpkm@c99afc4984265e2:~/scripting/bash-functions$ ./script4.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.
kpkm@c99afc4984265e2:~/scripting/bash-functions$
```

Type here to search

ENG 17:34 US 29-01-2025

Q.

```
GNU nano 7.2
#!/bin/bash
#Setting up a return status for a function
print_it () {
    echo Hello $1
    return 5
}

print_it User
print_it Reader
echo The previous function returned a value of $1
```

```
kpkm@e99afc4984265e2:~/scripting/bash-functions$ nano script5.sh
kpkm@e99afc4984265e2:~/scripting/bash-functions$ chmod +x script5.sh
kpkm@e99afc4984265e2:~/scripting/bash-functions$ ./script5.sh
Hello User
Hello Reader
The previous function returned a value of 5
kpkm@e99afc4984265e2:~/scripting/bash-functions$
```

Q.

```
GNU nano 7.2                                     script6.sh *
#!/bin/bash

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

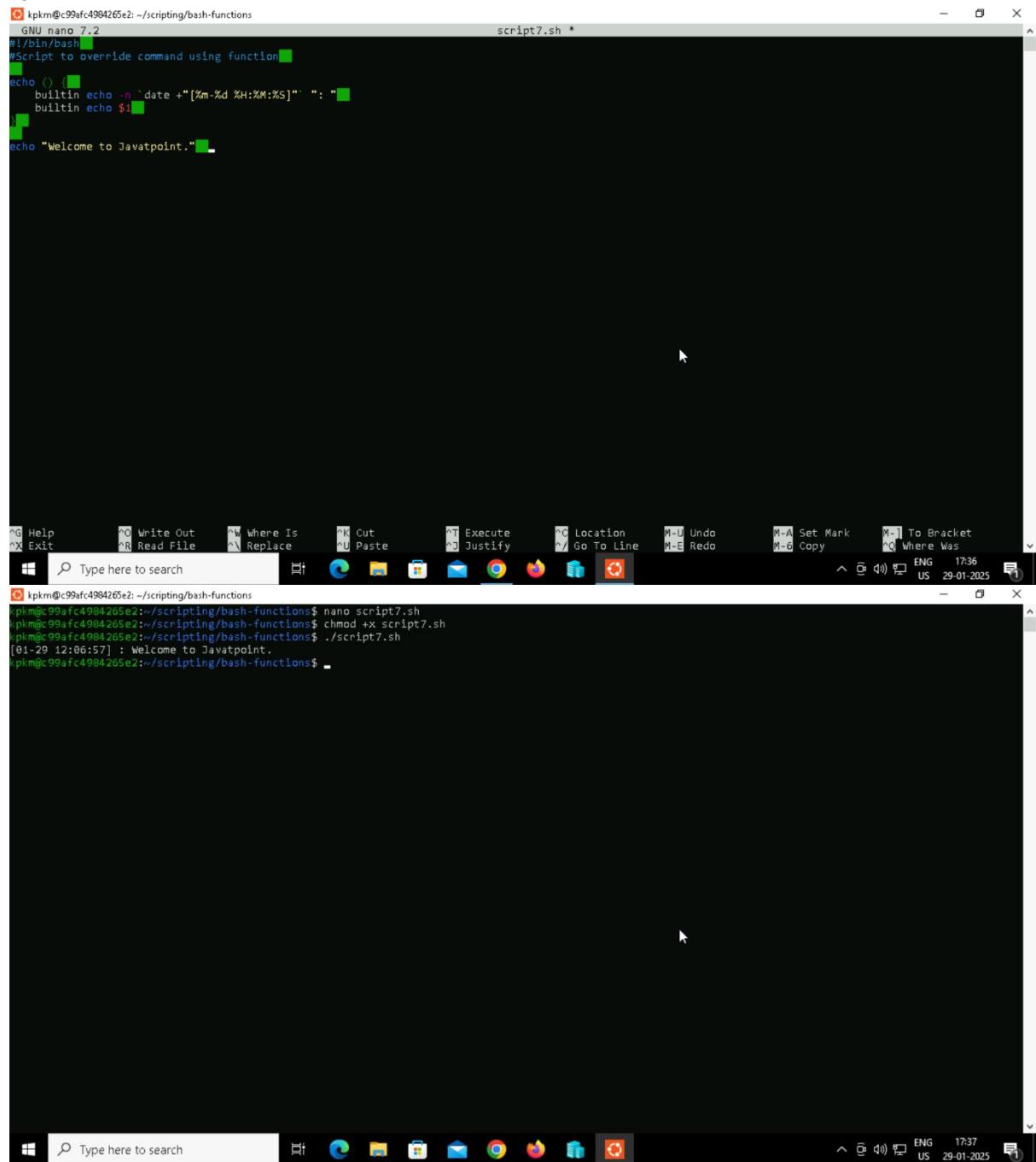
my_greet=$(print_it)
echo $my_greet

^G Help      ^O Write Out   ^W Where Is     ^K Cut        ^T Execute   ^C Location   M-U Undo   M-A Set Mark   M-J To Bracket
^X Exit      ^R Read File   ^P Replace     ^U Paste      ^J Justify   ^Y Go To Line  M-E Redo   M-B Copy      M-D Where Was
^F Type here to search
```

```
[pkmc@99afc4984265e2:~/scripting/bash-functions]$ nano script6.sh
[pkmc@99afc4984265e2:~/scripting/bash-functions]$ chmod +x script6.sh
[pkmc@99afc4984265e2:~/scripting/bash-functions]$ ./script6.sh
Welcome to Javatpoint.
[pkmc@99afc4984265e2:~/scripting/bash-functions]$
```

```
[pkmc@99afc4984265e2:~/scripting/bash-functions]$ nano script6.sh
[pkmc@99afc4984265e2:~/scripting/bash-functions]$ chmod +x script6.sh
[pkmc@99afc4984265e2:~/scripting/bash-functions]$ ./script6.sh
Welcome to Javatpoint.
[pkmc@99afc4984265e2:~/scripting/bash-functions]$
```

Q.



```
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."
```

```
kpkm@99afc4984265e2:~/scripting/bash-functions$ nano script7.sh
kpkm@99afc4984265e2:~/scripting/bash-functions$ chmod +x script7.sh
kpkm@99afc4984265e2:~/scripting/bash-functions$ ./script7.sh
[01-29 12:06:57] : Welcome to Javatpoint.
kpkm@99afc4984265e2:~/scripting/bash-functions$
```

```
kpkm@99afc4984265e2:~/scripting/bash-functions$ nano script7.sh
kpkm@99afc4984265e2:~/scripting/bash-functions$ chmod +x script7.sh
kpkm@99afc4984265e2:~/scripting/bash-functions$ ./script7.sh
[01-29 12:06:57] : Welcome to Javatpoint.
kpkm@99afc4984265e2:~/scripting/bash-functions$
```

```
Type here to search ENG 17:36
US 29-01-2025
```

```
Type here to search ENG 17:37
US 29-01-2025
```

BASH Arrays

Q.

```
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""Javatpoint" )
#printing the element with index of 2
echo ${example_array[2]}

kpkm@c99afc4984265e2:~/scripting/bash-array$ ./script1.sh
Javatpoint
kpkm@c99afc4984265e2:~/scripting/bash-array$
```

Q.

```
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[@]}"

^G Help      ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo      M-A Set Mark   M-J To Bracket
^X Exit      ^R Read File    ^P Replace     ^U Paste        ^J Justify      ^G Go To Line   M-E Redo      M-B Copy       M-D Where Was
^F Type here to search
```

```
:pkmc@99afc4984265e2:~/scripting/bash-array$ nano script2.sh
:pkmc@99afc4984265e2:~/scripting/bash-array$ chmod +x script2.sh
:pkmc@99afc4984265e2:~/scripting/bash-array$ ./script2.sh
Welcome to Javatpoint
:pkmc@99afc4984265e2:~/scripting/bash-array$
```

```
^G Help      ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo      M-A Set Mark   M-J To Bracket
^X Exit      ^R Read File    ^P Replace     ^U Paste        ^J Justify      ^G Go To Line   M-E Redo      M-B Copy       M-D Where Was
^F Type here to search
```

```
:pkmc@99afc4984265e2:~/scripting/bash-array$
```

Q.

```
kpkm@c99afc4984265e2:~/scripting/bash-functions
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."

```

The screenshot shows a Windows desktop environment. At the top, there is a taskbar with various pinned icons including File Explorer, Microsoft Edge, and File History. The system tray shows the date as 30-01-2025 and the time as 09:34. In the center, a terminal window titled 'script3.sh' is open in a dark-themed interface. The terminal displays the script code above. Below the terminal, the desktop background is visible, showing a dark landscape scene.

Q.

```
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"

[pkmc@99afc4984265e2:~/scripting/bash-array]$ nano script4.sh
[pkmc@99afc4984265e2:~/scripting/bash-array]$ chmod +x script4.sh
[pkmc@99afc4984265e2:~/scripting/bash-array]$ ./script4.sh
The array contains 3 elements
[pkmc@99afc4984265e2:~/scripting/bash-array]$
```

Q.

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

```
kpkm@99afc4984265e2:~/scripting/bash-array$ nano script5.sh
kpkm@99afc4984265e2:~/scripting/bash-array$ chmod +x script5.sh
kpkm@99afc4984265e2:~/scripting/bash-array$ ./script5.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
kpkm@99afc4984265e2:~/scripting/bash-array$
```

Q.

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

[Wrote 13 lines]

^T Execute ^C Location M-U Undo M-A Set Mark M-J To Bracket
^R Read File ^W Where Is ^K Cut M-D Redo M-B Copy M-Q Where Was
^X Exit ^O Write Out ^P Replace ^U Paste ENG 17:51
Type here to search US 29-01-2025

```
pkmc@99afc4984265e2:~/scripting/bash-array$ nano script6.sh
pkmc@99afc4984265e2:~/scripting/bash-array$ chmod +x script6.sh
pkmc@99afc4984265e2:~/scripting/bash-array$ ./script6.sh
0 Welcome
1 To
2 Javaatpoint
pkmc@99afc4984265e2:~/scripting/bash-array$
```

Type here to search ENG 17:52 US 29-01-2025

Q.

```
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[@]}"

pkm@99afc4984265e2:~/scripting/bash-array$ nano script7.sh
pkm@99afc4984265e2:~/scripting/bash-array$ chmod +x script7.sh
pkm@99afc4984265e2:~/scripting/bash-array$ ./script7.sh
Java Python PHP HTML JavaScript
pkm@99afc4984265e2:~/scripting/bash-array$
```

Q.

```
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[@]}"

kpkm@99afc4984265e2:~/scripting/bash-array$ ls
script1.sh  script3.sh  script5.sh  script7.sh
script2.sh  script4.sh  script6.sh
kpkm@99afc4984265e2:~/scripting/bash-array$ nano script8.sh
kpkm@99afc4984265e2:~/scripting/bash-array$ chmod +x script8.sh
kpkm@99afc4984265e2:~/scripting/bash-array$ ./script8.sh
Java Python PHP JavaScript CSS SQL
kpkm@99afc4984265e2:~/scripting/bash-array$
```

Q.

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

```
kpkm@99afc4984265e2:~/scripting/bash-array$ nano script9.sh
kpkm@99afc4984265e2:~/scripting/bash-array$ chmod +x script9.sh
kpkm@99afc4984265e2:~/scripting/bash-array$ ./script9.sh
Welcome you on SSSIT Javatpoint
kpkm@99afc4984265e2:~/scripting/bash-array$
```

Q.

```
GNU nano 7.2                                     script10.sh *
#!/bin/bash
#Script to delete the element from the array
#
#Declaring the array
declare -a example_array=( "Java" "Python" "HTML" "CSS" "JavaScript" )
#
#Removing the element
unset example_array[1]
#
#Printing all the elements after deletion
echo "${example_array[@]}"

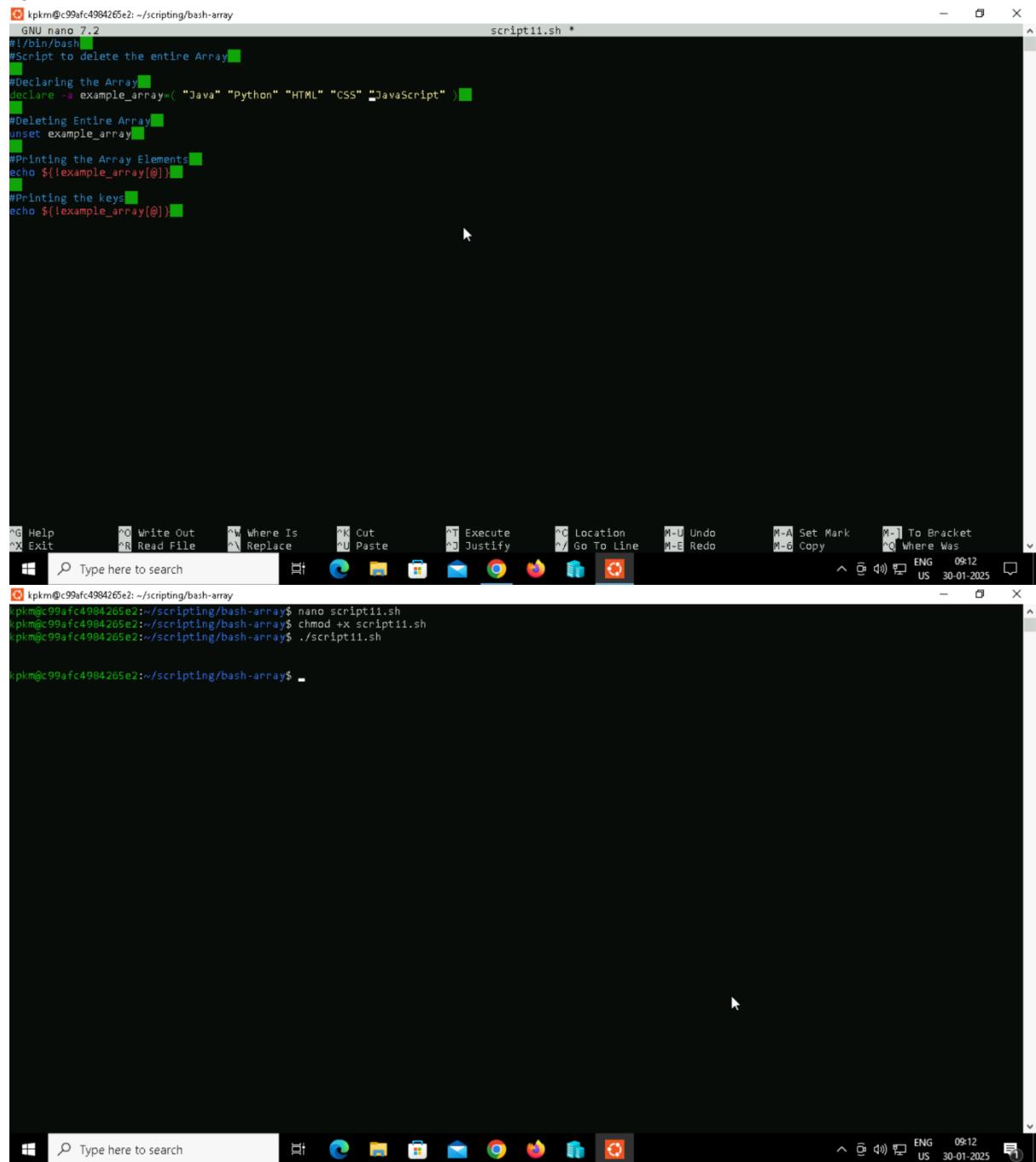
^G Help      ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo      M-A Set Mark   M-J To Bracket
^X Exit      ^R Read File    ^P Replace     ^U Paste        ^J Justify      ^G Go To Line   M-E Redo      M-B Copy       M-Q Where Was
^F Type here to search
```

```
[pkmc@99afc4984265e2:~/scripting/bash-array]$ nano script10.sh
[pkmc@99afc4984265e2:~/scripting/bash-array]$ chmod +x script10.sh
[pkmc@99afc4984265e2:~/scripting/bash-array]$ ./script10.sh
Java HTML CSS JavaScript
[pkmc@99afc4984265e2:~/scripting/bash-array]$
```

```
^G Help      ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo      M-A Set Mark   M-J To Bracket
^X Exit      ^R Read File    ^P Replace     ^U Paste        ^J Justify      ^G Go To Line   M-E Redo      M-B Copy       M-Q Where Was
^F Type here to search
```

```
[pkmc@99afc4984265e2:~/scripting/bash-array]$
```

Q.



```
GNU nano 7.2                                     script11.sh *
#!/bin/bash
#Script to delete the entire Array
#
#Declaring the Array
declare -a example_array=( "Java" "Python" "HTML" "CSS" "JavaScript" )
#
#Deleting Entire Array
unset example_array
#
#Printing the Array Elements
echo ${example_array[@]}
#
#Printing the keys
echo ${!example_array[@]}

^G Help      ^O Write Out   ^W Where Is    ^K Cut        ^T Execute   ^C Location   M-U Undo   M-A Set Mark   M-] To Bracket
^X Exit      ^R Read File   ^P Replace    ^U Paste     ^J Justify   ^Y Go To Line M-E Redo   M-B Copy      M-[ Where Was
^H Type here to search
^I
^L
^V
^D
^F
^S
^A
^B
^C
^E
^G
^H
^J
^K
^L
^M
^N
^O
^P
^R
^T
^U
^V
^W
^X
^Y
^Z
^_
```

```
kpkm@e99afc4984265e2:~/scripting/bash-array$ nano script11.sh
kpkm@e99afc4984265e2:~/scripting/bash-array$ chmod +x script11.sh
kpkm@e99afc4984265e2:~/scripting/bash-array$ ./script11.sh

kpkm@e99afc4984265e2:~/scripting/bash-array$
```

Q.

```
GNU nano 7.2
#!/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
```

```
kpkm@99afc4984265e2:~/scripting/bash-array$ nano script12.sh
kpkm@99afc4984265e2:~/scripting/bash-array$ chmod +x script12.sh
kpkm@99afc4984265e2:~/scripting/bash-array$ ./script12.sh
Python
HTML
CSS
kpkm@99afc4984265e2:~/scripting/bash-array$
```