FOR LOOP

1.Basic example of “for loop”.

Step 1: Creates a new empty file named for_loop.sh in the current directory.

Step 2: Gives execute permission to the for_loop.sh file, making it runnable.

Step 3: Opens the for_loop.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the for_loop.sh script in the current directory after saving it.

Output:

2.The basic example to print a series of numbers from 1 to 10.

Step 1: Creates a new empty file named for_loop1.sh in the current directory.

Step 2: Gives execute permission to the for_loop1.sh file, making it runnable.

Step 3: Opens the for_loop1.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the for_loop1.sh script in the current directory after saving it.

Output:

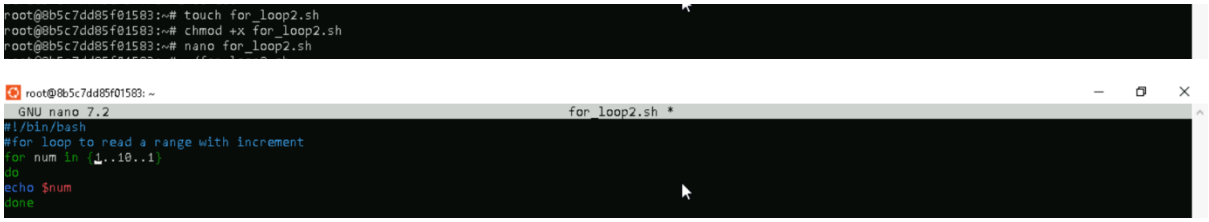
3. For loop to read a range with increment.

Step 1: Creates a new empty file named for_loop2.sh in the current directory.

Step 2: Gives execute permission to the for_loop2.sh file, making it runnable.

Step 3: Opens the for_loop2.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the for_loop2.sh script in the current directory after saving it.



Output:

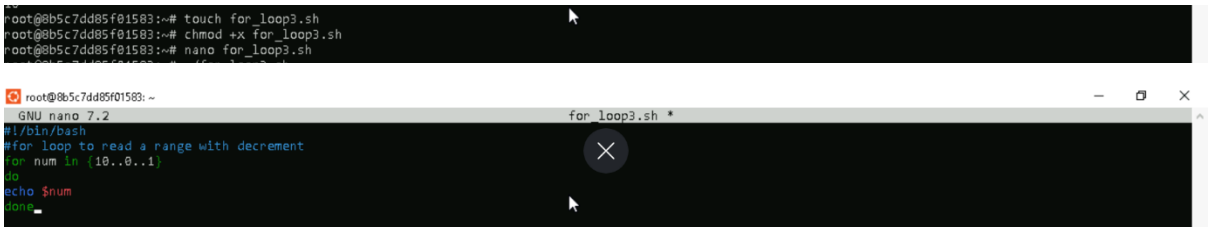
4. For loop to read a range with decrement.

Step 1: Creates a new empty file named for_loop3.sh in the current directory.

Step 2: Gives execute permission to the for_loop3.sh file, making it runnable.

Step 3: Opens the for_loop3.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the for_loop3.sh script in the current directory after saving it.



Output:

5. Array Declaration

Step 1: Creates a new empty file named for_loop4.sh in the current directory.

Step 2: Gives execute permission to the for_loop4.sh file, making it runnable.

Step 3: Opens the for_loop4.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the for_loop4.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch for_loop4.sh
root@8b5c7dd85f01583:~# chmod +x for_loop4.sh
root@8b5c7dd85f01583:~# nano for_loop4.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  for_loop4.sh *
#!/bin/bash
array=( "Welcome""to""JavaTpoint" )
for i in "${arr[@]}"
do
echo $i
done
```

Output:

```
root@8b5c7dd85f01583:~# ./for_loop4.sh
WelcometoJavaTpoint
root@8b5c7dd85f01583:~#
```

6. For loop to read white space in string as word separators

Step 1: Creates a new empty file named for_loop5.sh in the current directory.

Step 2: Gives execute permission to the for_loop5.sh file, making it runnable.

Step 3: Opens the for_loop5.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the for_loop5.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch for_loop5.sh
root@8b5c7dd85f01583:~# chmod +x for_loop5.sh
root@8b5c7dd85f01583:~# nano for_loop5.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  for_loop5.sh *
#!/bin/bash
#For Loop to Read white spaces in String as word separators

str="Let's start learning from Javatpoint."
for i in $str;
do
echo "$i"
done
```

Output:

```
root@8b5c7dd85f01583:~# nano for_loop5.sh
root@8b5c7dd85f01583:~# ./for_loop5.sh
Let's
start
learning
from
Javatpoint.
root@8b5c7dd85f01583:~#
```

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

Step 1: Creates a new empty file named for_loop6.sh in the current directory.

Step 2: Gives execute permission to the for_loop6.sh file, making it runnable.

Step 3: Opens the for_loop6.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the for_loop6.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch for_loop6.sh
root@8b5c7dd85f01583:~# chmod +x for_loop6.sh
root@8b5c7dd85f01583:~# nano for_loop6.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  for_loop6.sh *
#!/bin/bash
#For loop to read each line in string as a word
str="Let's start learning from Javatpoint."
for i in "$str";
do
echo "$i"
done
```

Output:

```
root@8b5c7dd85f01583:~# ./for_loop6.sh
Let's start learning from Javatpoint.
root@8b5c7dd85f01583:~#
```

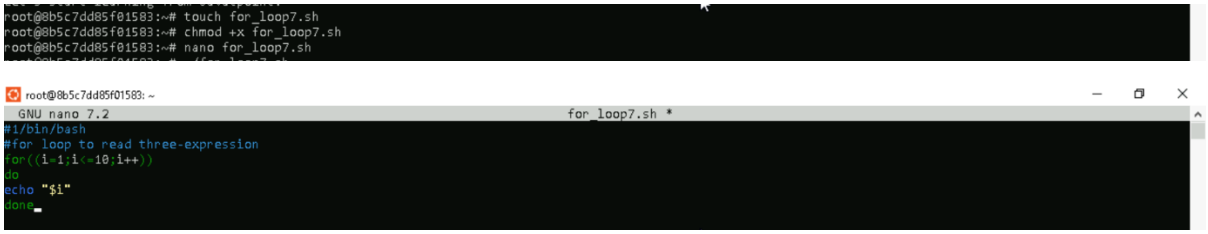
8. For loop to read three-expression

Step 1: Creates a new empty file named for_loop7.sh in the current directory.

Step 2: Gives execute permission to the for_loop7.sh file, making it runnable.

Step 3: Opens the for_loop7.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the for_loop7.sh script in the current directory after saving it.



Output:

9. Script for tables of 2

Step 1: Creates a new empty file named for_loop8.sh in the current directory.

Step 2: Gives execute permission to the for_loop8.sh file, making it runnable.

Step 3: Opens the for_loop8.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the for_loop8.sh script in the current directory after saving it.

Output:

10. Script for printing current number.

Step 1: Creates a new empty file named for_loop9.sh in the current directory.

Step 2: Gives execute permission to the for_loop9.sh file, making it runnable.

Step 3: Opens the for_loop9.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the for_loop9.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch for_loop9.sh
root@8b5c7dd85f01583:~# chmod +x for_loop9.sh
root@8b5c7dd85f01583:~# nano for_loop9.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  for_loop10.sh *
#!/bin/bash
i=1;
for (( ; ; ))
do
sleep 1s
echo "Current Number: $((i++))"
done
```

Output:

```
root@8b5c7dd85f01583:~# ./for_loop9.sh
1
2
3
4
5
16
17
18
19
20
```

11. Script for printing current number.

Step 1: Creates a new empty file named for_loop10.sh in the current directory.

Step 2: Gives execute permission to the for_loop10.sh file, making it runnable.

Step 3: Opens the for_loop10.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the for_loop10.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch for_loop10.sh
root@8b5c7dd85f01583:~# chmod +x for_loop10.sh
root@8b5c7dd85f01583:~# nano for_loop10.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  for_loop10.sh *
#!/bin/bash
i=1;
for (( ; ; ))
do
sleep 1s
echo "Current Number: $((i++))"
done
```

Output:

```
root@8b5c7dd85f01583:~# ./for_loop10.sh
Current Number: 1
Current Number: 2
Current Number: 3
Current Number: 4
Current Number: 5
Current Number: 6
Current Number: 7
Current Number: 8
Current Number: 9
Current Number: 10
Current Number: 11
Current Number: 12
Current Number: 13
Current Number: 14
Current Number: 15
Current Number: 16
Current Number: 17
```

CASE STATEMENT

12. Simple scenario to demonstrate the use of the case statement.

Step 1: Creates a new empty file named case.sh in the current directory.

Step 2: Gives execute permission to the case.sh file, making it runnable.

Step 3: Opens the case.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the case.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch case.sh
root@8b5c7dd85f01583:~# chmod +x case.sh
root@8b5c7dd85f01583:~# nano case.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  case.sh *
#!/bin/bash

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

Output:

```
root@8b5c7dd85f01583:~# ./case.sh
Do you know Java Programming?
Yes/No? :yes
That's amazing.
```

13. Defined a combined scenario where there is also a default case when no previous matched case is found.

Step 1: Creates a new empty file named case1.sh in the current directory.

Step 2: Gives execute permission to the case1.sh file, making it runnable.

Step 3: Opens the case1.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the case1.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch case1.sh
root@8b5c7dd85f01583:~# chmod +x case1.sh
root@8b5c7dd85f01583:~# nano case1.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                 case1.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:

```
root@8b5c7dd85f01583:~# ./case1.sh
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name:Android
This is my favorite. It has lots of applications.
```

WHILE LOOP

14.Basic example of while loop which will print series of numbers as per user input.

Step 1: Creates a new empty file named while1.sh in the current directory.

Step 2: Gives execute permission to the while1.sh file, making it runnable.

Step 3: Opens the while1.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the while1.shscript in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch while1.sh
root@8b5c7dd85f01583:~# chmod +x while1.sh
root@8b5c7dd85f01583:~# nano while1.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  while1.sh *
#1/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."
```

Output:

```
root@8b5c7dd85f01583:~# ./while1.sh
Enter starting number: 2
Enter ending number: 6
2
3
4
5
6
This is the sequence that you wanted.
root@8b5c7dd85f01583:~#
```

15. Example of while loop with multiple conditions in the expression.

Step 1: Creates a new empty file named while2.sh in the current directory.

Step 2: Gives execute permission to the while2.sh file, making it runnable.

Step 3: Opens the while2.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the while2.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch while2.sh
root@8b5c7dd85f01583:~# chmod +x while2.sh
root@8b5c7dd85f01583:~# nano while2.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  while2.sh *
#!/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:

```
root@8b5c7dd85f01583:~# ./while2.sh
Enter starting number: 1
Enter ending number: 10
1
2
3
4
5
6
7
8
9
10
This is the sequence that you wanted.
```

16. An infinite while loop. The loop will execute continuously until it is forcefully stopped using CRTL + C.

Step 1: Creates a new empty file named while3.sh in the current directory.

Step 2: Gives execute permission to the while3.sh file, making it runnable.

Step 3: Opens the while3.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the while3.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch while3.sh
root@8b5c7dd85f01583:~# chmod +x while3.sh
root@8b5c7dd85f01583:~# nano while3.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                while3.sh *
#!/bin/bash
#An infinite while loop

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

Output:

```
root@8b5c7dd85f01583: ~
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome t^C
```

17. An infinite while loop. The loop will execute continuously until it is forcefully stopped using CRTL + C.

Step 1: Creates a new empty file named while3.sh in the current directory.

Step 2: Gives execute permission to the while3.sh file, making it runnable.

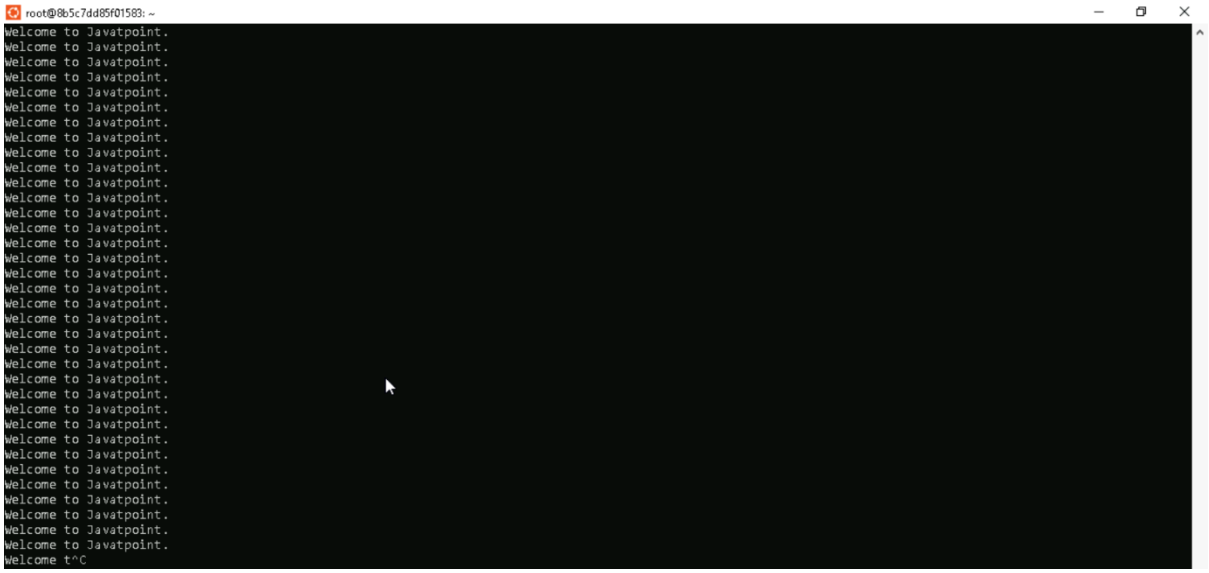
Step 3: Opens the while3.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the while3.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch while3.sh
root@8b5c7dd85f01583:~# chmod +x while3.sh
root@8b5c7dd85f01583:~# nano while3.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                while4.sh *
#!/bin/bash
#An infinite while loop
while :; do echo "Welcome to Javatpoint."; done
```

Output:



18. A break statement can be used to stop the loop as per the applied condition.

Step 1: Creates a new empty file named while5.sh in the current directory.

Step 2: Gives execute permission to the while5.sh file, making it runnable.

Step 3: Opens the while5.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the while5.sh script in the current directory after saving it.

Output:

19. A continue statement can be used to skip the iteration for a specific condition inside the while loop.

Step 1: Creates a new empty file named while6.sh in the current directory.

Step 2: Gives execute permission to the while6.sh file, making it runnable.

Step 3: Opens the while6.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the while6.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch while6.sh
root@8b5c7dd85f01583:~# chmod +x while6.sh
root@8b5c7dd85f01583:~# nano while6.sh
```

```
root@8b5c7dd85f01583: ~
  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:

```
root@8b5c7dd85f01583:~# ./while6.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.
root@8b5c7dd85f01583:~#
```

20. Write a while loop in bash script as similar as a while loop in C programming language.

Step 1: Creates a new empty file named while7.sh in the current directory.

Step 2: Gives execute permission to the while7.sh file, making it runnable.

Step 3: Opens the while7.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the while7.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch while7.sh
root@8b5c7dd85f01583:~# chmod +x while7.sh
root@8b5c7dd85f01583:~# nano while7.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  while7.sh *
#!/bin/bash
#While loop example in C style

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

Output:

```
root@8b5c7dd85f01583:~# ./while7.sh
1
2
3
4
5
6
7
8
9
10
root@8b5c7dd85f01583:~#
```

UNTIL LOOP

21. Basic example of until loop which will print series of numbers from 1 to 10.

Step 1: Creates a new empty file named until1.sh in the current directory.

Step 2: Gives execute permission to the until1.sh file, making it runnable.

Step 3: Opens the until1.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the until1.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch until1.sh
root@8b5c7dd85f01583:~# chmod +x until1.sh
root@8b5c7dd85f01583:~# nano until1.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  until1.sh *
#!/bin/bash
#Bash Until Loop example with a single condition

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

Output:

```
root@8b5c7dd85f01583:~# nano until1.sh
root@8b5c7dd85f01583:~# ./until1.sh
1
2
3
4
5
6
7
8
9
10
root@8b5c7dd85f01583:~#
```

22. Following is an example with multiple condition in an expression

Step 1: Creates a new empty file named until2.sh in the current directory.

Step 2: Gives execute permission to the until2.sh file, making it runnable.

Step 3: Opens the until2.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the until2.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch until2.sh
root@8b5c7dd85f01583:~# chmod +x until2.sh
root@8b5c7dd85f01583:~# nano until2.sh
root@8b5c7dd85f01583:~# ./until2.sh
a = 1 & b = 0.
a = 2 & b = 1.
a = 3 & b = 2.
a = 4 & b = 3.
a = 5 & b = 4.
root@8b5c7dd85f01583:~#
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  until2.sh *
#!/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:

```
root@8b5c7dd85f01583:~# ./until2.sh
a = 1 & b = 0.
a = 2 & b = 1.
a = 3 & b = 2.
a = 4 & b = 3.
a = 5 & b = 4.
root@8b5c7dd85f01583:~#
```

STRING

23. Script to check whether two strings are equal.

Step 1: Creates a new empty file named string.sh in the current directory.

Step 2: Gives execute permission to the string.sh file, making it runnable.

Step 3: Opens the string.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the string.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch string.sh
root@8b5c7dd85f01583:~# chmod +x string.sh
root@8b5c7dd85f01583:~# nano string.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  string.sh *
#!/bin/bash
#Script to check whether two strings are equal.
str1="Welcometojavatpoint."
str2="javatpoint"

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

Output:

```
root@8b5c7dd85f01583:~# ./string.sh
Strings are not equal.
root@8b5c7dd85f01583:~#
```

24. Script to check whether two strings are not equal.

Step 1: Creates a new empty file named string1.sh in the current directory.

Step 2: Gives execute permission to the string1.sh file, making it runnable.

Step 3: Opens the string1.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the string1.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch string1.sh
root@8b5c7dd85f01583:~# chmod +x string1.sh
root@8b5c7dd85f01583:~# nano string1.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  string1.sh *
#!/bin/bash
#Script to check whether two strings are equal.
str1="Welcometojavatpoint."
str2="javatpoint"
if [[ $str1 != $str2 ]];
then
echo "Strings are not equal."
else
echo "Strings are equal."
fi
```

Output:

```
root@8b5c7dd85f01583:~# ./string1.sh
Strings are not equal.
root@8b5c7dd85f01583:~#
```

25. The 'less than operator (\<)' is used to check if string1 is greater than string2.

Step 1: Creates a new empty file named string2.sh in the current directory.

Step 2: Gives execute permission to the string2.sh file, making it runnable.

Step 3: Opens the string2.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the string2.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch string2.sh
root@8b5c7dd85f01583:~# chmod +x string2.sh
root@8b5c7dd85f01583:~# nano string2.sh
```

```
root@8b5c7dd85f01583: ~
 GNU nano 7.2                                string2.sh *
#!/bin/sh
str1="WelcometoJavatpoint"
str2="Javatpoint"
if [ $str1 \< $str2 ];
then
echo "$str1 is less then $str2"
else
echo "$str1 is not less then $str2"
fi
```

Output:

```
root@8b5c7dd85f01583:~# ./string2.sh
WelcometoJavatpoint is not less then Javatpoint
root@8b5c7dd85f01583:~#
```

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

Step 1: Creates a new empty file named string3.sh in the current directory.

Step 2: Gives execute permission to the string3.sh file, making it runnable.

Step 3: Opens the string3.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the string3.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch string3.sh
root@8b5c7dd85f01583:~# chmod +x string3.sh
root@8b5c7dd85f01583:~# nano string3.sh
```

```
root@8b5c7dd85f01583: ~
 GNU nano 7.2                                string3.sh *
#!/bin/sh
str1="WelcometoJavatpoint"
str2="Javatpoint"
if [ $str1 \> $str2 ];

then
echo "$str1 is greater then $str2"
else
echo "$str1 is less then $str2"
fi
```

Output:

```
root@8b5c7dd85f01583:~# ./string3.sh
WelcometoJavatpoint is greater then Javatpoint
root@8b5c7dd85f01583:~#
```

27. To check if the string length is greater than to zero.

Step 1: Creates a new empty file named string4.sh in the current directory.

Step 2: Gives execute permission to the string4.sh file, making it runnable.

Step 3: Opens the string4.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the string4.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch string4.sh
root@8b5c7dd85f01583:~# chmod +x string4.sh
root@8b5c7dd85f01583:~# nano string4.sh
```

```
root@8b5c7dd85f01583: ~
 GNU nano 7.2                                string4.sh *
#!/bin/sh
str="WelcometoJavatpoint"
if [ -n $str ];
then
echo "String is not empty"
else
echo "String is empty"
fi
```

Output:

```
root@8b5c7dd85f01583:~# ./string4.sh
String is not empty
root@8b5c7dd85f01583:~#
```

28. To check if the string length is equal to zero.

Step 1: Creates a new empty file named string5.sh in the current directory.

Step 2: Gives execute permission to the string5.sh file, making it runnable.

Step 3: Opens the string5.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the string5.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch string5.sh
root@8b5c7dd85f01583:~# chmod +x string5.sh
root@8b5c7dd85f01583:~# nano string5.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                    string5.sh *
#!/bin/sh
str=""
if [ -z $str ];
then
echo "String is empty."
else
echo "String is non-empty."
fi
```

Output:

```
root@8b5c7dd85f01583:~# ./string5.sh
String is empty.
root@8b5c7dd85f01583:~#
```

FIND STRING

29. Find the length of string using ${#string_variable}

Step 1: Creates a new empty file named findstring1.sh in the current directory.

Step 2: Gives execute permission to the findstring1.sh file, making it runnable.

Step 3: Opens the findstring1.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the findstring1.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch findstring1.sh
root@8b5c7dd85f01583:~# chmod +x findstring1.sh
root@8b5c7dd85f01583:~# nano findstring1.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  findstring1.sh *
#!/bin/bash
#Bash program to find the length of a string

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

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

Output:

```
root@8b5c7dd85f01583:~# ./findstring1.sh
Length of 'Welcome to Javatpoint' is 21
root@8b5c7dd85f01583:~#
```

30. Find the length of string using 'expr length "$str"'

Step 1: Creates a new empty file named findstring2.sh in the current directory.

Step 2: Gives execute permission to the findstring2.sh, making it runnable.

Step 3: Opens the findstring2.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the findstring2.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch findstring2.sh
root@8b5c7dd85f01583:~# chmod +x findstring2.sh
root@8b5c7dd85f01583:~# nano findstring2.sh
```

```
root@8b5c7dd85f01583: ~
 GNU nano 7.2                                findstring2.sh *
#!/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:

```
root@8b5c7dd85f01583:~# ./findstring2.sh
Length of 'Welcome to Javatpoint' is 21
root@8b5c7dd85f01583:~#
```

31. Find the length of string using 'expr "$str"'

Step 1: Creates a new empty file named findstring3.sh in the current directory.

Step 2: Gives execute permission to the findstring3.sh file, making it runnable.

Step 3: Opens the findstring3.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the findstring3.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch findstring3.sh
root@8b5c7dd85f01583:~# chmod +x findstring3.sh
root@8b5c7dd85f01583:~# nano findstring3.sh
```

```
root@8b5c7dd85f01583: ~
 GNU nano 7.2                                findstring3.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:

```
root@8b5c7dd85f01583:~# ./findstring3.sh
Length of 'Welcome to Javatpoint' is 21
root@8b5c7dd85f01583:~#
```

32. Find the length of string using "wc" command

Step 1: Creates a new empty file named findstring4.sh in the current directory.

Step 2: Gives execute permission to the findstring4.sh file, making it runnable.

Step 3: Opens the findstring4.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the findstring4.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch findstring4.sh
root@8b5c7dd85f01583:~# chmod +x findstring4.sh
root@8b5c7dd85f01583:~# nano findstring4.sh
```

```
root@8b5c7dd85f01583: ~
 GNU nano 7.2                                findstring4.sh *
#!/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:

```
root@8b5c7dd85f01583:~# ./findstring4.sh
Length of 'Welcome to Javatpoint' is 22
root@8b5c7dd85f01583:~#
```

33. Find the length of string using "awk" command.

Step 1: Creates a new empty file named findstring5.sh in the current directory.

Step 2: Gives execute permission to the findstring5.sh file, making it runnable.

Step 3: Opens the findstring5.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the findstring5.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch findstring5.sh
root@8b5c7dd85f01583:~# chmod +x findstring5.sh
root@8b5c7dd85f01583:~# nano findstring5.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  findstring5.sh *
#!/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:

```
root@8b5c7dd85f01583:~# ./findstring5.sh
Length of 'Welcome to Javatpoint' is 21
root@8b5c7dd85f01583:~#
```

STRING SPLIT

34. A string is split using a space character delimiter.

Step 1: Creates a new empty file named split1.sh in the current directory.

Step 2: Gives execute permission to the split1.sh file, making it runnable.

Step 3: Opens the split1.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the split1.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch split1.sh
root@8b5c7dd85f01583:~# chmod +x split1.sh
root@8b5c7dd85f01583:~# nano split1.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                    split1.sh *
#!/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 -a 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:

```
root@8b5c7dd85f01583:~# ./split1.sh
Enter any string separated by space: we welcome you
we
welcome
you
root@8b5c7dd85f01583:~#
```

35. A string is split using a symbol comma (,) character as a delimiter.

Step 1: Creates a new empty file named split3.sh in the current directory.

Step 2: Gives execute permission to the split3.sh file, making it runnable.

Step 3: Opens the split3.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the split3.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch split3.sh
root@8b5c7dd85f01583:~# chmod +x split3.sh
root@8b5c7dd85f01583:~# nano split3.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                    split3.sh
#!/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:

```
root@8b5c7dd85f01583:~# ./split3.sh
Enter Name, State and Age separated by a comma: pradhi, tamilnadu, 22
Name : pradhi
State :  tamilnadu
Age :  22
root@8b5c7dd85f01583:~#
```

36. A string is split using a symbol colon(:) character as a delimiter.

Step 1: Creates a new empty file named split2.sh in the current directory.

Step 2: Gives execute permission to the split2.sh file, making it runnable.

Step 3: Opens the split2.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes split2.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch split2.sh
root@8b5c7dd85f01583:~# chmod +x split2.sh
root@8b5c7dd85f01583:~# nano split2.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                    split2.sh *
#!/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:

```
root@8b5c7dd85f01583:~# ./split2.sh
Enter any string separated by colon(:) we:welcome:you.

we
welcome
you.
```

37. Split string by another string.

Step 1: Creates a new empty file named split4.sh in the current directory.

Step 2: Gives execute permission to the split4.sh file, making it runnable.

Step 3: Opens the split4.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the split4.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch split4.sh
root@8b5c7dd85f01583:~# chmod +x split4.sh
root@8b5c7dd85f01583:~# nano split4.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                    split4.sh *
#!/bin/bash
#example for bash split string by another string
Str="WeLearnWelcomeLearnYouLearnOnLearnJavatpoint"
delimiter=Learn
s=$str$delimiter
array=();
while [[ $s ]];

do
array+=( "${s%%"$delimiter"*}" );
s=${s#*"$delimiter"};
done;
declare -p arrays
```

Output:

```
root@8b5c7dd85f01583:~# ./split4.sh
declare -a array=([0]="We" [1]="Welcome" [2]="You" [3]="On" [4]="Javatpoint")
root@8b5c7dd85f01583:~#
```

38. Split a string using trim command.

Step 1: Creates a new empty file named split5.sh in the current directory.

Step 2: Gives execute permission to the split5.sh file, making it runnable.

Step 3: Opens the split5.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the split5.sh script in the current directory after saving it.

Output:

SUB STRING

39. To extract till specific characters from starting.

Step 1: Creates a new empty file named sub1.sh in the current directory.

Step 2: Gives execute permission to the sub1.sh file, making it runnable.

Step 3: Opens the sub1.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the sub1.sh script in the current directory after saving it.

Output:

40. To extract till specific characters onwards.

Step 1: Creates a new empty file named sub2.sh in the current directory.

Step 2: Gives execute permission to the sub2.sh file, making it runnable.

Step 3: Opens the sub2.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the sub2.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch sub2.sh
root@8b5c7dd85f01583:~# chmod +x sub2.sh
root@8b5c7dd85f01583:~# nano sub2.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  sub2.sh *
#!/bin/bash
#Script to print from 11th character onwards
str="We welcome you on Javatpoint."
substr="${str:11}"
echo "$substr"
```

Output:

```
root@8b5c7dd85f01583:~# ./sub2.sh
you on Javatpoint.
root@8b5c7dd85f01583:~#
```

41. To extract a single character.

Step 1: Creates a new empty file named sub3.sh in the current directory.

Step 2: Gives execute permission to the sub3.sh file, making it runnable.

Step 3: Opens the sub3.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the sub3.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch sub3.sh
root@8b5c7dd85f01583:~# chmod +x sub3.sh
root@8b5c7dd85f01583:~# nano sub3.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  sub3.sh *
#!/bin/bash
#Script to print 11th character of a String
str="We welcome you on Javatpoint."
substr="${str:11:1}"
echo "$substr"
```

Output:

```
root@8b5c7dd85f01583:~# ./sub3.sh
y
root@8b5c7dd85f01583:~#
```

42. To extract the specific characters from last.

Step 1: Creates a new empty file named sub4.sh in the current directory.

Step 2: Gives execute permission to the sub4.sh file, making it runnable.

Step 3: Opens the sub4.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the sub4.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch sub4.sh
root@8b5c7dd85f01583:~# chmod +x sub4.sh
root@8b5c7dd85f01583:~# nano sub4.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  sub4.sh *
#!/bin/bash
#Script to extract 11 characters from last
str="We welcome you on Javatpoint."
substr="${str:(-11)}"
echo "$substr"
```

Output:

```
root@8b5c7dd85f01583:~# ./sub4.sh
Javatpoint.
root@8b5c7dd85f01583:~#
```

CONCATENATION

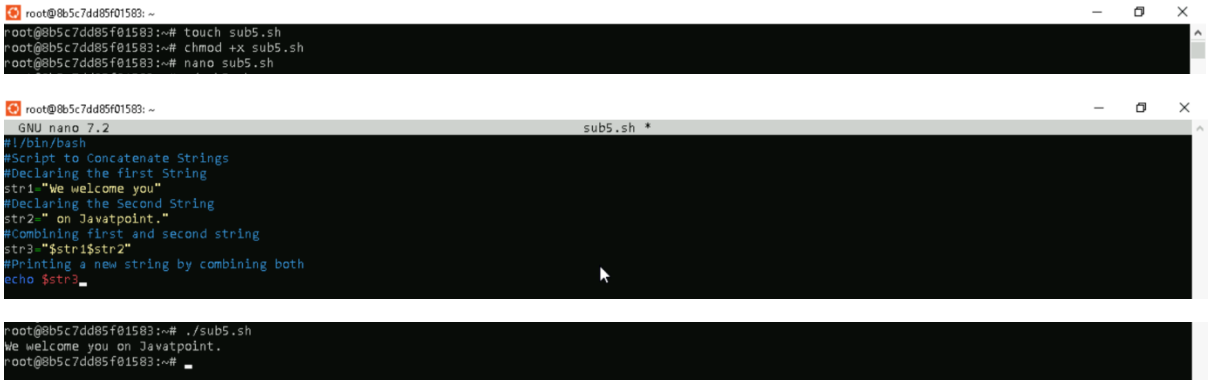
43. Script to concatenate string.

Step 1: Creates a new empty file named sub5.sh in the current directory.

Step 2: Gives execute permission to the sub5.sh file, making it runnable.

Step 3: Opens the sub5.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the sub5.sh script in the current directory after saving it.



44. Another easy method is to use variables inside the string, which is defined with double-quotes.

Step 1: Creates a new empty file named con2.sh in the current directory.

Step 2: Gives execute permission to the con2.sh file, making it runnable.

Step 3: Opens the con2.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the con2.sh script in the current directory after saving it.

Output:

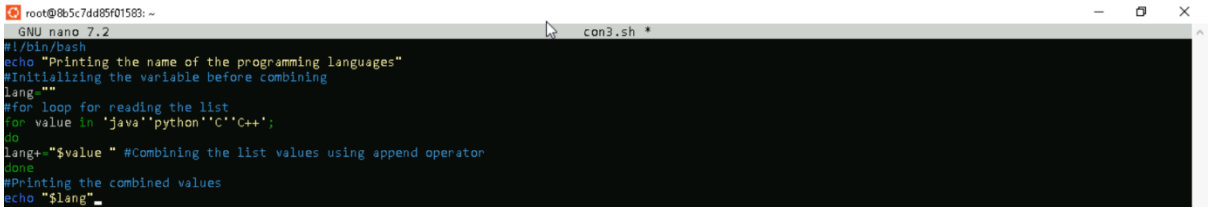
45. Using append operator with loop.

Step 1: Creates a new empty file named con3.sh in the current directory.

Step 2: Gives execute permission to the con3.sh file, making it runnable.

Step 3: Opens the con3.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the con3.sh script in the current directory after saving it.



Output:

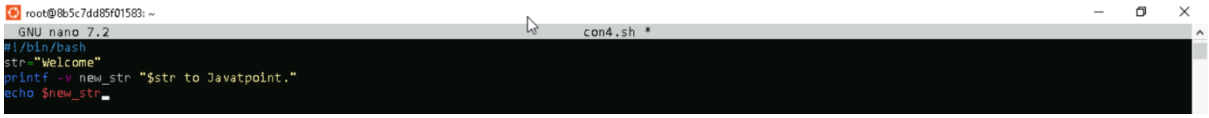
46. Using printf function.

Step 1: Creates a new empty file named con4.sh in the current directory.

Step 2: Gives execute permission to the con4.sh file, making it runnable.

Step 3: Opens the con4.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the con4.sh script in the current directory after saving it.



Output:

47. Using Literal strings.

Step 1: Creates a new empty file named con5.sh in the current directory.

Step 2: Gives execute permission to the con5.sh file, making it runnable.

Step 3: Opens the con5.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the con5.sh script in the current directory after saving it.

Output:

48. Using underscore

Step 1: Creates a new empty file named con6.sh in the current directory.

Step 2: Gives execute permission to the con6.sh file, making it runnable.

Step 3: Opens the con6.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the con6.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch con6.sh
root@8b5c7dd85f01583:~# chmod +x con6.sh
root@8b5c7dd85f01583:~# nano con6.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                con6.sh *
#!/bin/bash
str1="Hello"
str2="World!"
echo "${str1}_${str2}"
```

Output:

```
root@8b5c7dd85f01583:~# ./con5.sh
Welcome to Javatpoint.
```

49. Using any character

Step 1: Creates a new empty file named con7.sh in the current directory.

Step 2: Gives execute permission to the con7.sh file, making it runnable.

Step 3: Opens the con7.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the con7.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch con7.sh
root@8b5c7dd85f01583:~# chmod +x con7.sh
root@8b5c7dd85f01583:~# nano con7.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                con7.sh *
#!/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:

```
root@8b5c7dd85f01583:~# ./con7.sh
Enter First Name: pradhi
Enter State: tamilnadu
Enter Age: 22
Name, State, Age: pradhi,tamilnadu,22
root@8b5c7dd85f01583:~#
```

FUNCTION

50. Example for function – Method 1

Step 1: Creates a new empty file named func1.sh in the current directory.

Step 2: Gives execute permission to the func1.sh file, making it runnable.

Step 3: Opens the func1.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the func1.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch func1.sh
root@8b5c7dd85f01583:~# chmod +x func1.sh
root@8b5c7dd85f01583:~# nano func1.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                               func1.sh *
#!/bin/bash
JTP () {
echo 'Welcome to Javatpoint.'
}
JTP
```

Output:

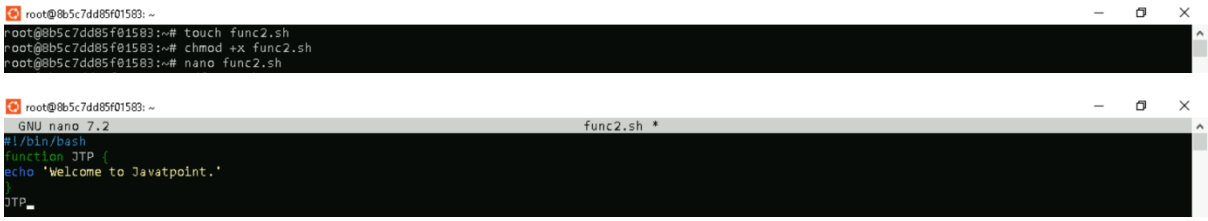
51. Example for function – Method 2

Step 1: Creates a new empty file named func2.sh in the current directory.

Step 2: Gives execute permission to the func2.sh file, making it runnable.

Step 3: Opens the func2.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the func2.sh script in the current directory after saving it.



Output:

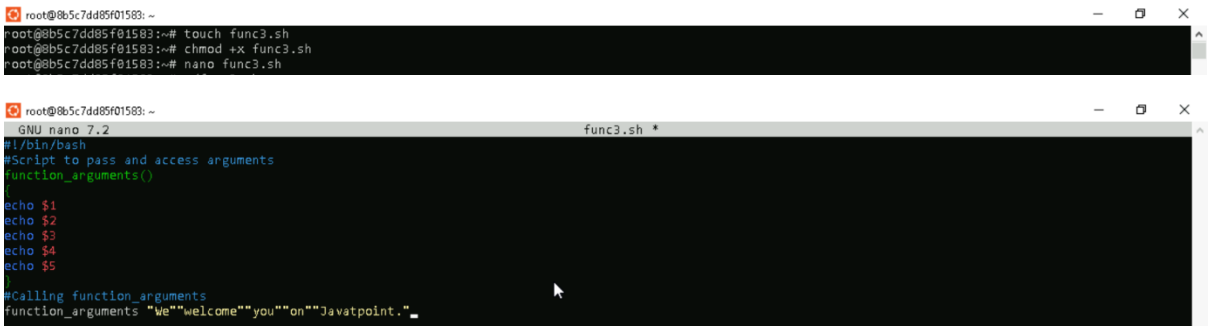
52. Script to pass and access arguments.

Step 1: Creates a new empty file named func3.sh in the current directory.

Step 2: Gives execute permission to the func3.sh file, making it runnable.

Step 3: Opens the func3.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the func3.sh script in the current directory after saving it.



Output:

53.Example for variable scope.

Step 1: Creates a new empty file named func4.sh in the current directory.

Step 2: Gives execute permission to the func4.sh file, making it runnable.

Step 3: Opens the func4.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the func4.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583:~# touch func4.sh
root@8b5c7dd85f01583:~# chmod +x func4.sh
root@8b5c7dd85f01583:~# nano func4.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  func4.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:

```
root@8b5c7dd85f01583:~# ./func4.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.
root@8b5c7dd85f01583:~#
```

54. Example for return values.

Step 1: Creates a new empty file named func5.sh in the current directory.

Step 2: Gives execute permission to the func5.sh file, making it runnable.

Step 3: Opens the func5.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the func5.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch func5.sh
root@8b5c7dd85f01583:~# chmod +x func5.sh
root@8b5c7dd85f01583:~# nano func5.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  func5.sh *
#!/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 $?
```

Output:

```
root@8b5c7dd85f01583:~# ./func5.sh
Hello User
Hello Reader
The previous function returned a value of 5
root@8b5c7dd85f01583:~#
```

55. Example for return values.

Step 1: Creates a new empty file named func6.sh in the current directory.

Step 2: Gives execute permission to the func6.sh file, making it runnable.

Step 3: Opens the func6.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the func6.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch func6.sh
root@8b5c7dd85f01583:~# chmod +x func6.sh
root@8b5c7dd85f01583:~# nano func6.sh
```

```
root@8b5c7dd85f01583: ~
 GNU nano 7.2                                  func6.sh *
#!/bin/bash

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

my_greet="$(print_it)"
echo $my_greet
```

Output:

```
root@8b5c7dd85f01583:~# ./func6.sh
Welcome to Javatpoint.
root@8b5c7dd85f01583:~#
```

56. Example for overriding commands.

Step 1: Creates a new empty file named func7.sh in the current directory.

Step 2: Gives execute permission to the func7.sh file, making it runnable.

Step 3: Opens the func7.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the func7.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch func7.sh
root@8b5c7dd85f01583:~# chmod +x func7.sh
root@8b5c7dd85f01583:~# nano func7.sh
```

```
root@8b5c7dd85f01583: ~
 GNU nano 7.2                                  func7.sh *
#!/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."
```

Output:

```
root@8b5c7dd85f01583:~# ./func7.sh
[01-29 12:35:19] : Welcome to Javatpoint.
root@8b5c7dd85f01583:~#
```

ARRAY

57.Script to print an element of an array with an index of 2.

Step 1: Creates a new empty file named arr1.sh in the current directory.

Step 2: Gives execute permission to the arr1.sh file, making it runnable.

Step 3: Opens the arr1.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the arr1.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch arr1.sh
root@8b5c7dd85f01583:~# chmod +x arr1.sh
root@8b5c7dd85f01583:~# nano arr1.sh
```

```
root@8b5c7dd85f01583: ~
 GNU nano 7.2                                  arr1.sh
#!/bin/bash
declare -a example_array=( "Welcome" "to" "javatpoint" )
echo ${example_array[2]}
```

Output:

```
root@8b5c7dd85f01583:~# ./arr1.sh
javatpoint
root@8b5c7dd85f01583:~#
```

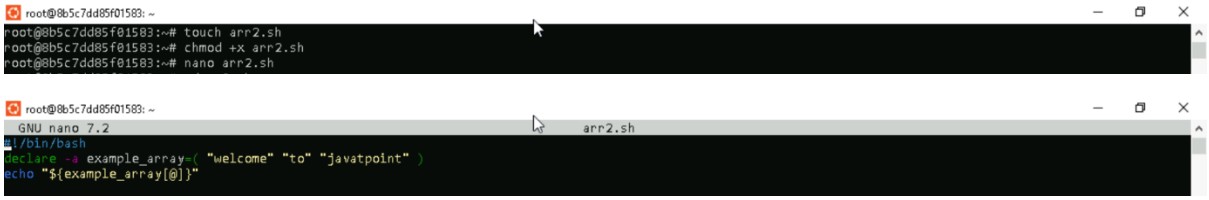
58. Script to print all the elements of the array.

Step 1: Creates a new empty file named arr2.sh in the current directory.

Step 2: Gives execute permission to the arr2.sh file, making it runnable.

Step 3: Opens the arr2.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the arr2.sh script in the current directory after saving it.



Output:

59. Script to print the keys of the array.

Step 1: Creates a new empty file named arr3.sh in the current directory.

Step 2: Gives execute permission to the arr3.sh file, making it runnable.

Step 3: Opens the arr3.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the arr3.sh script in the current directory after saving it.

Output:

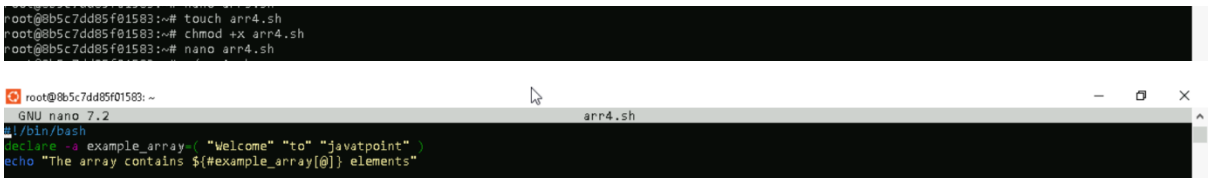
60. Count the number of elements contained in the array.

Step 1: Creates a new empty file named arr4.sh in the current directory.

Step 2: Gives execute permission to the arr4.sh file, making it runnable.

Step 3: Opens the arr4.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the arr4.sh script in the current directory after saving it.



Output:

```
root@8b5c7dd85f01583:~# ./arr4.sh
The array contains 3 elements
root@8b5c7dd85f01583:~#
```

61.Script to print all keys and values using loop through the array.

Step 1: Creates a new empty file named arr5.sh in the current directory.

Step 2: Gives execute permission to the arr5.sh file, making it runnable.

Step 3: Opens the arr5.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the arr5.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch arr5.sh
root@8b5c7dd85f01583:~# chmod +x arr5.sh
root@8b5c7dd85f01583:~# nano arr5.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                arr5.sh
#!/bin/bash
declare -a example_array=( "Welcome" "to" "javatpoint")
for i in "${!example_array[@]}"
do
echo the key value of element "${example_array[$i]}" is "$i"
done
```

Output:

```
root@8b5c7dd85f01583:~# ./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
root@8b5c7dd85f01583:~#
```

62. Script to loop through an array in C-Style.

Step 1: Creates a new empty file named arr6.sh in the current directory.

Step 2: Gives execute permission to the arr6.sh file, making it runnable.

Step 3: Opens the arr6.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the arr6.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch arr6.sh
root@8b5c7dd85f01583:~# chmod +x arr6.sh
root@8b5c7dd85f01583:~# nano arr6.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                arr6.sh
#!/bin/bash
declare -a example_array=( "Welcome" "to" "javatpointt" )
length=${#example_array[@]}
for((i=0;i<${length};i++))
do
echo $i ${example_array[$i]}
done
```

Output:

```
root@8b5c7dd85f01583:~# ./arr6.sh
0 Welcome
1 to
2 javatpointt
root@8b5c7dd85f01583:~#
```

63. Script to add new element to the array using index values.

Step 1: Creates a new empty file named arr7.sh in the current directory.

Step 2: Gives execute permission to the arr7.sh file, making it runnable.

Step 3: Opens the arr7.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the arr7.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch arr7.sh
root@8b5c7dd85f01583:~# chmod +x arr7.sh
root@8b5c7dd85f01583:~# nano arr7.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  arr7.sh *
#!/bin/bash
declare -a example_array=( "Java""Python""PHP""HTML" )
example_array[4]="JavaScript"
echo "${example_array[@]}"
```

Output:

```
root@8b5c7dd85f01583:~# ./arr7.sh
JavaPythonPHPHTML JavaScript
root@8b5c7dd85f01583:~#
```

64. Script to add new element to the array using operator.

Step 1: Creates a new empty file named arr8.sh in the current directory.

Step 2: Gives execute permission to the arr8.sh file, making it runnable.

Step 3: Opens the arr8.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the arr8.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch arr8.sh
root@8b5c7dd85f01583:~# chmod +x arr8.sh
root@8b5c7dd85f01583:~# nano arr8.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  arr8.sh *
#!/bin/bash
declare -a example_array=( "Java" "Python" " PHP" )
example_array+=( JavaScript CSS SQL )
echo "${example_array[@]}"
```

Output:

```
root@8b5c7dd85f01583:~# ./arr8.sh
Java Python  PHP JavaScript CSS SQL
root@8b5c7dd85f01583:~#
```

65.Script to update the array.

Step 1: Creates a new empty file named arr9.sh in the current directory.

Step 2: Gives execute permission to the arr9.sh file, making it runnable.

Step 3: Opens the arr9.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the arr9.sh script in the current directory after saving it.

```
root@8b5c7dd85f01583: ~
root@8b5c7dd85f01583:~# touch arr9.sh
root@8b5c7dd85f01583:~# chmod +x arr9.sh
root@8b5c7dd85f01583:~# nano arr9.sh
```

```
root@8b5c7dd85f01583: ~
  GNU nano 7.2                                  arr9.sh *
#!/bin/bash
declare -a example_array=( "We""welcome""you""on""SSSIT" )
example_array[4]=Javatpoint
echo ${example_array[@]}
```

Output:

```
root@8b5c7dd85f01583:~# ./arr9.sh
WewelcomeyouonSSSIT Javatpoint
root@8b5c7dd85f01583:~#
```

66. Script to delete an element from an array.

Step 1: Creates a new empty file named arr10.sh in the current directory.

Step 2: Gives execute permission to the arr10.sh file, making it runnable.

Step 3: Opens the arr10.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the arr10.sh script in the current directory after saving it.

Output:

67. Script to delete a entire array.

Step 1: Creates a new empty file named arr11.sh in the current directory.8.

Step 2: Gives execute permission to the arr11.sh file, making it runnable.

Step 3: Opens the arr11.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the arr11.sh script in the current directory after saving it.

Output:

68. Script to slice array elements.

Step 1: Creates a new empty file named arr12.sh in the current directory.

Step 2: Gives execute permission to the arr12.sh file, making it runnable.

Step 3: Opens the arr12.sh file in the Nano text editor to allow you to write or edit the script.

Step 4: Executes the arr12.sh script in the current directory after saving it.

Output:


root@8b5c7dd85f01583:~# ./arr12.sh
Python
HTML
CSS
root@8b5c7dd85f01583:~#