

BASH CASE

- 1) Simple scenario to demonstrate the use of the case statement

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch bash1.sh  
root@f304ea52daec59c:~# chmod bash1.sh  
chmod: missing operand after 'bash1.sh'  
Try 'chmod --help' for more information.  
root@f304ea52daec59c:~# chmod +x bash1.sh  
root@f304ea52daec59c:~# vi bash1.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
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
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./bash1.sh  
Do you know java programming?  
yes/no?:n  
it's easy.let's start learning from javatpoint.
```

- 2) Combined scenario where there is also a default case when no previous matched case is found.

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch bash2.sh  
root@f304ea52daec59c:~# chmod +x bash2.sh  
root@f304ea52daec59c:~# vi bash2.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
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 favourite.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
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./bash2.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.
```

BASH FOR LOOP

- 3) Basic For loop

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch for1.sh  
root@f304ea52daec59c:~# chmod +x for1.sh  
root@f304ea52daec59c:~# vi for1.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
learn="Start learning from javatpoint"  
for learn in $learn  
do  
    echo $learn  
done  
echo "thank you"
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./for1.sh  
Start  
learning  
from  
javatpoint  
thank you
```

4) For Loop to Read a Range

Step1: Create a Bash Script

```
root@f304ea52daec59c:~# touch for2.sh  
root@f304ea52daec59c:~# chmod +x for2.sh  
root@f304ea52daec59c:~# vi for2.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
for num in {1..10}  
do  
    echo $num  
done  
echo "series of number from 1 to 10."
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./for2.sh  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
series of number from 1 to 10.
```

5) For Loop to Read a Range with Increment.

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch for3.sh  
root@f304ea52daec59c:~# chmod +x for3.sh  
root@f304ea52daec59c:~# vi for3.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
for num in {1..10..1}  
do  
    echo $num  
done
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./for3.sh  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

6) For Loop to Read a Range with Decrement

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch for3.sh  
root@f304ea52daec59c:~# chmod +x for3.sh  
root@f304ea52daec59c:~# vi for3.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
for num in {1..10..1}  
do  
    echo $num  
done
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./for3.sh  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1  
0
```

7) For Loop to Read Array Variables

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch for5.sh  
root@f304ea52daec59c:~# chmod +x for5.sh  
root@f304ea52daec59c:~# vi for5.sh  
root@f304ea52daec59c:~# ./for5.sh  
./for5.sh: line 1: syntax error near unexpected token `)`  
./for5.sh: line 1: `arr=(welcome""to""javatpoint)"'  
root@f304ea52daec59c:~# vi for5.sh  
root@f304ea52daec59c:~# ./for5.sh  
./for5.sh: line 2: syntax error near unexpected token `in'`${arr[@]}"'  
./for5.sh: line 2: `for i in "${arr[@]}"'  
root@f304ea52daec59c:~# vi for5.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
arr=(welcome""to""javatpoint)  
for i in "${arr[@]}"  
do  
    echo $i  
done
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./for5.sh  
welcometojavatpoint
```

8) For Loop to Read white spaces in String as word separators.

Step1: Create a Bash Script

```
root@f304ea52daec59c:~# touch for6.sh  
root@f304ea52daec59c:~# chmod +x for6.sh  
root@f304ea52daec59c:~# vi for6.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
str="let's start  
learning from javatpoint"  
for i in $str;  
do  
    echo "$i"  
done
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./for6.sh  
let's  
start  
learning  
from  
javatpoint
```

9) For Loop to Read each line in String as a word

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch for7.sh  
root@f304ea52daec59c:~# vi for7.sh  
root@f304ea52daec59c:~# chmod +x for7.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
str="Let's start  
learning from  
javatpoint."  
for i in "$str";  
do  
    echo "$i"  
done
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./for7.sh  
Let's start  
learning from  
javatpoint.
```

10) For Loop to Read Three-expression

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch for8.sh  
root@f304ea52daec59c:~# chmod +x for8.sh  
root@f304ea52daec59c:~# vi for8.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
for ((i=1; i<=10; i++))  
do  
    echo "$i"  
done
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./for8.sh  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

11) For Loop with a Break Statement

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch for9.sh  
root@f304ea52daec59c:~# chmod +x for9.sh  
root@f304ea52daec59c:~# vi for9.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
for table in {2..100..2}  
do  
    echo $table  
    if [ $table == 20 ];  
    then  
        break  
    fi  
done
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./for9.sh  
2  
4  
6  
8  
10  
12  
14  
16  
18  
20
```

12) For Loop with a Continue Statement

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch for10.sh  
root@f304ea52daec59c:~# chmod +x for10.sh  
root@f304ea52daec59c:~# vi for10.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
for((i=1; i<=20; i++));  
do  
    if [[ $i -gt 5 && $i -lt 16 ]];  
    then  
        continue  
    fi  
    echo $i  
done
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./for10.sh  
1  
2  
3  
4  
5  
16  
17  
18  
19  
20
```

13) Infinite Bash For Loop

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch for11.sh  
root@f304ea52daec59c:~# chmod +x for11.sh  
root@f304ea52daec59c:~# vi for11.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
i=1;  
for(;;)  
do  
    sleep 1s  
    echo "current number: $((i++))"  
done
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./for11.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
```

BASH WHILE LOOP

14) While Loop with Single Condition

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch while1.sh  
root@f304ea52daec59c:~# chmod +x while1.sh  
root@f304ea52daec59c:~# vi while1.sh
```

Step 2: Write the script in the editor.

```

root@f304ea52daec59c: ~
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."

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./while1.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.

```

15) While Loop with Multiple Conditions

Step1: Create a Bash Script

```

root@f304ea52daec59c:~# touch while2.sh
root@f304ea52daec59c:~# vi while2.sh

```

Step 2: Write the script in the editor.

```

root@f304ea52daec59c: ~
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."

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./while2.sh
Enter starting number:2
Enter ending number:30
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
This is the sequence that you wanted.

```

16) Infinite While Loop

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch while4.sh
root@f304ea52daec59c:~# chmod +x while4.sh
root@f304ea52daec59c:~# vi while4.sh
root@f304ea52daec59c:~# ./while4.sh

```

Step 2: Write the script in the editor.

Step 3: Run the script

17) Infinite While Loop

Step 2: Write the script in the editor.

Step 3: Run the script


```

root@f304ea52daec59c: ~
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 statements."

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./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 statements.

```

20) While Loop with C-Style.

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch while7.sh
root@f304ea52daec59c:~# chmod +x while7.sh
root@f304ea52daec59c:~# vi while7.sh

```

Step 2: Write the script in the editor.

```

root@f304ea52daec59c: ~
i=1
while ((i<=10))
do
    echo $i
    let i++
done

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./while7.sh
1
2
3
4
5
6
7
8
9
10

```

BASH UNTIL

21) Until loop which will print series of numbers from 1 to 10

Step1: Create a Bash Script

```

root@f304ea52daec59c:~# touch until1.sh
root@f304ea52daec59c:~# chmod until1.sh
chmod: missing operand after 'until1.sh'
Try 'chmod --help' for more information.
root@f304ea52daec59c:~# chmod +x until1.sh
root@f304ea52daec59c:~# vi until1.sh

```

Step 2: Write the script in the editor.

```

root@f304ea52daec59c: ~
i=1
until [ $i -gt 10 ]
do
    echo $i
    ((i++))
done

```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./until1.sh
1
2
3
4
5
6
7
8
9
10
```

22) Multiple conditions in an expression

Step1: Create a Bash Script

```
root@f304ea52daec59c:~# touch until2.sh
root@f304ea52daec59c:~# chmod +x until2.sh
root@f304ea52daec59c:~# vi until2.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c:~#
max=5
a=1
b=0
until [[ $a -gt $max || $b -gt $max ]];
do
    echo "a = $a &b = $b"
    ((a++))
    ((b++))
done
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./until2.sh
a = 1 &b = 0
a = 2 &b = 1
a = 3 &b = 2
a = 4 &b = 3
a = 5 &b = 4
```

BASH STRING

23) Equal Operator

Step1: Create a Bash Script

```
root@f304ea52daec59c:~# touch string1.sh
root@f304ea52daec59c:~# chmod +x string1.sh
root@f304ea52daec59c:~# vi string1.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c:~#
str1="welcome to javatpoint"
str2="javatpoint"
if [ $str1 = $str2 ];
then
    echo "both the strings are equal."
else
    echo "strings are not equal."
fi
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./string1.sh
strings are not equal.
```

24) Not Equal Operator

Step1: Create a Bash Script

```
root@f304ea52daec59c:~# touch string2.sh
root@f304ea52daec59c:~# chmod +x string2.sh
root@f304ea52daec59c:~# vi string2.sh
```

Step 2: Write the script in the editor.

```

root@f304ea52daec59c: ~
str1="welcometojavatpoint"
str2="javatpoint"
if [ $str1 = $str2 ];
then
    echo "both the strings are equal."
else
    echo "strings are not equal."
fi

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./string2.sh
both the strings are equal.
root@f304ea52daec59c:~#

```

25) Less than Operator

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch string3.sh
root@f304ea52daec59c:~# chmod +x string3.sh
root@f304ea52daec59c:~# vi string3.sh

```

Step 2: Write the script in the editor.

```

root@f304ea52daec59c: ~
str1="welcometojavatpoint"
str2="javatpoint"
if [ $str1 < $str2 ];
then
    echo "$str1 is less then $str2."
else
    echo "$str1 is not less then $str2."
fi

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./string3.sh
Welcometojavatpoint is not less then Javatpoint.

```

26) Greater than Operator

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch string4.sh
root@f304ea52daec59c:~# chmod +x string4.sh
root@f304ea52daec59c:~# vi string4.sh

```

Step 2: Write the script in the editor.

```

root@f304ea52daec59c: ~
str1="welcometojavatpoint"
str2="javatpoint"
if [ $str1 > $str2 ];
then
    echo "$str1 is greater then $str2."
else
    echo "$str1 is less then $str2."
fi

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./string4.sh
Welcometojavatpoint is greater then Javatpoint.

```

27) To check if the string length is greater than Zero:

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch string5.sh
root@f304ea52daec59c:~# chmod +x string5.sh
root@f304ea52daec59c:~# vi string5.sh

```

Step 2: Write the script in the editor.

```

root@f304ea52daec59c: ~
str="Welcome to javatpoint"
if [ -n $str ];
then
    echo "String is not empty"
else
    echo "String is empty"
fi

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./string5.sh
String is not empty

```

28) To check if the string length is equal to Zero

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch string6.sh
root@f304ea52daec59c:~# chmod +x string6.sh
root@f304ea52daec59c:~# vi string6.sh

```

Step 2: Write the script in the editor.

```

root@f304ea52daec59c: ~
str=""
if [ -z $str ];
then
    echo "String is empty"
else
    echo "String is not empty"
fi

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./string6.sh
String is empty

```

BASH FIND STRING

29) The simplest way to calculate the length of a string is to use '#' symbol.

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch fstring1.sh
root@f304ea52daec59c:~# chmod +x fstring1.sh
root@f304ea52daec59c:~# vi fstring1.sh

```

Step 2: Write the script in the editor.

```

root@f304ea52daec59c: ~
str="welcome to javatpoint"
length=${#str}
echo "length of '$str' is $length"

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./fstring1.sh
length of 'welcome to javatpoint' is 21

```

30) Calculate the length of a string is to use `expr` command with the 'length' keyword

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch fstring2.sh
root@f304ea52daec59c:~# chmod +x fstring2.sh
root@f304ea52daec59c:~# vi fstring2.sh

```

Step 2: Write the script in the editor.

```

root@f304ea52daec59c: ~
str="welcome to javatpoint"
length=`expr length "$str"`
echo "length of '$str' is $length"

```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./fstring2.sh
length of 'Welcome to javatpoint' is 21
root@f304ea52daec59c:~#
```

31) Use `expr "\$str":'.'*` to find the length of a string

Step1: Create a Bash Script

```
root@f304ea52daec59c:~# touch fstring3.sh
root@f304ea52daec59c:~# chmod +x fstring3.sh
root@f304ea52daec59c:~# vi fstring3.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c:~#
str="Welcome to javatpoint"
length=$(expr length "$str")
echo "length of '$str' is $length"
root@f304ea52daec59c:~#
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./fstring3.sh
length of 'Welcome to javatpoint' is 21
root@f304ea52daec59c:~#
```

32) Use `wc` command to find the length of a string

Step1: Create a Bash Script

```
root@f304ea52daec59c:~# touch fstring4.sh
root@f304ea52daec59c:~# chmod +x fstring4.sh
root@f304ea52daec59c:~# vi fstring4.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c:~#
str="Welcome to Javatpoint"
length=$(echo $str | wc -c)
echo "Length of '$str' is $length"
root@f304ea52daec59c:~#
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./fstring4.sh
length of 'Welcome to Javatpoint' is 22
root@f304ea52daec59c:~#
```

33) Use `awk` command to find the length of a string.

Step1: Create a Bash Script

```
root@f304ea52daec59c:~# touch fstring5.sh
root@f304ea52daec59c:~# chmod +x fstring5.sh
root@f304ea52daec59c:~# vi fstring5.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c:~#
str="Welcome to Javatpoint"
length=$(echo $str | awk '{print length}')
echo "Length of '$str' is $length"
root@f304ea52daec59c:~#
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./fstring5.sh
length of 'Welcome to Javatpoint' is 21
root@f304ea52daec59c:~#
```

BASH SPLIT STRING

34) Bash Split String by Space

Step1: Create a Bash Script

```
root@f304ea52daec59c:~# touch split1.sh
root@f304ea52daec59c:~# chmod +x split1.sh
root@f304ea52daec59c:~# vi split1.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
read -p "Enter any string separated by space: " str #reading string value  
IFS=" " #setting space as delimiter  
read -ra ADDR <<"$str" #reading str as an array as tokens separated by IFS  
for i in "${ADDR[@]}"; #accessing each element of array  
do  
echo "$i"  
done
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./split1.sh  
Enter any string separated by space: i am ankita  
i  
am  
ankita  
root@f304ea52daec59c:~#
```

35) Bash Split String by Symbol

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch split2.sh  
root@f304ea52daec59c:~# chmod +x split2.sh  
root@f304ea52daec59c:~# vi split2.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
read -p "Enter Name, State and Age separated by a comma: " entry #reading string value  
IFS="," #setting comma as delimiter  
read -ra strarr <<"$entry" #reading str as an array as tokens separated by IFS  
echo "Name : ${strarr[0]}"  
echo "State : ${strarr[1]}"  
echo "Age : ${strarr[2]}"
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./split2.sh  
Enter Name, State and Age separated by a comma: ankita , kerala , 22  
Name : ankita  
State : kerala  
Age : 22  
root@f304ea52daec59c:~#
```

36) Bash Split String by Symbol

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch split3.sh  
root@f304ea52daec59c:~# chmod +x split3.sh  
root@f304ea52daec59c:~# vi split3.sh
```

Step 2: Write the script in the editor.

```
root@f304ea52daec59c: ~  
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
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./split3.sh  
Enter any string separated by colon(:) i am ankita: from bangalore: employee in ust:  
i am ankita  
from bangalore  
employee in ust  
root@f304ea52daec59c:~#
```

37) Bash Split String by another string

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch split4.sh  
root@f304ea52daec59c:~# chmod +x split4.sh  
root@f304ea52daec59c:~# vi split4.sh
```

Step 2: Write the script in the editor.

```

root@f304ea52daec59c: ~
str="WeLearnWelcomeLearnYouLearnOnLearnJavatpoint"
delimiter=Learn
s=${str}$delimiter
array=();
while [[ $s ]];
do
    array+=(" ${s%%$delimiter} ");
    s=${s#*$delimiter};
done;
declare -p array

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./split4.sh
declare -a array=([0]="We" [1]="Welcome" [2]="You" [3]="On" [4]="Javatpoint")

```

38) Bash Split String using Trim Command

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch split5.sh
root@f304ea52daec59c:~# chmod +x split5.sh
root@f304ea52daec59c:~# vi split5.sh

```

Step 2: Write the script in the editor.

```

root@f304ea52daec59c: ~
my_str="We;welcome;you;on;javatpoint."
my_arr=($(echo $my_str | tr ";" "\n"))
for i in "${my_arr[@]}"
do
    echo $i
done

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./split5.sh
We
welcome
you
on
javatpoint.

```

BASH SUBSTRING

39) To Extract till Specific Characters from Starting

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch sub1.sh
root@f304ea52daec59c:~# chmod sub1.sh
chmod: missing operand after 'sub1.sh'
Try 'chmod --help' for more information.
root@f304ea52daec59c:~# chmod +x sub1.sh
root@f304ea52daec59c:~# vi sub1.sh

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
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}"

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./sub1.sh
String: We welcome you on Javatpoint.
Total characters in a String: 29
Substring: We welcome
Total characters in Substring: 10

```

40) To Extract from Specific Character onwards

Step1: Create a Bash Script

```

root@f304ea52daec59c:~# touch sub2.sh
root@f304ea52daec59c:~# chmod +x sub2.sh
root@f304ea52daec59c:~# vi sub2.sh

```

Step 2: Write the script in the editor

```
root@f304ea52daec59c: ~  
str="we welcome you on Javatpoint."  
substr="${str:11}"  
echo "$substr"
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./sub2.sh  
you on Javatpoint.
```

41) To Extract a Single Character

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch sub3.sh  
root@f304ea52daec59c:~# chmod +x sub3.sh  
root@f304ea52daec59c:~# vi sub3.sh  
root@f304ea52daec59c:~# ./sub3.sh  
y
```

Step 2: Write the script in the editor

```
root@f304ea52daec59c: ~  
str="we welcome you on Javatpoint."  
substr="${str:11:1}"  
echo "$substr"
```

Step 3: Run the script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch sub3.sh  
root@f304ea52daec59c:~# chmod +x sub3.sh  
root@f304ea52daec59c:~# vi sub3.sh
```

42) To Extract the specific characters from last

Step1: Create a Bash Script

```
Select root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch sub4.sh  
root@f304ea52daec59c:~# chmod +x sub4.sh  
root@f304ea52daec59c:~# vi sub4.sh
```

Step 2: Write the script in the editor

```
root@f304ea52daec59c: ~  
str="we welcome you on Javatpoint."  
substr="${str:(-11)}"  
echo "$substr"
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./sub4.sh  
Javatpoint.
```

BASH CONCATENATE

43) Write Variables Side by Side

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch cat1.sh  
root@f304ea52daec59c:~# chmod +x cat1.sh  
root@f304ea52daec59c:~# vi cat1.sh
```

Step 2: Write the script in the editor


```

root@f304ea52daec59c: ~
str1="we welcome you"
#Declaring the Second String
str2=" on Javatpoint."
#Combining first and second string
str3="$str1$str2"
##Printing a new string by combining both
echo $str3
~

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./cat1.sh
we welcome you on Javatpoint.

```

44) Using Double Quotes

Step1: Create a Bash Script

```

root@f304ea52daec59c:~# touch cat2.sh
root@f304ea52daec59c:~# chmod +x cat2.sh
root@f304ea52daec59c:~# vi cat2.sh

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
str="we welcome you"
#Add the variable within the string
echo "$str on Javatpoint."

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./cat2.sh
we welcome you on Javatpoint.
root@f304ea52daec59c:~#

```

45) Using Append Operator with Loop

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch cat3.sh
root@f304ea52daec59c:~# chmod +x cat3.sh
root@f304ea52daec59c:~# vi cat3.sh

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
#!/bin/bash
echo "Printing the name of the programming languages"
lang=""
for loop for reading the list
do
  for value in "java" "python" "C" "C++";
  do
    lang+="$value " #Combining the list values using append operator
  done
done
##Printing the combined values
echo "$lang"

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./cat3.sh
Printing the name of the programming languages
java python C C++

```

46) Using the Printf Function

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch cat4.sh
root@f304ea52daec59c:~# chmod +x cat4.sh
root@f304ea52daec59c:~# vi cat4.sh

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
str="welcome"
printf -v new_str "$str to Javatpoint."
echo $new_str

```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./cat4.sh
Welcome to Javatpoint.
root@f304ea52daec59c:~#
```

47) Using Literal Strings

Step1: Create a Bash Script

```
root@f304ea52daec59c:~# touch cat5.sh
root@f304ea52daec59c:~# chmod +x cat5.sh
root@f304ea52daec59c:~# vi cat5.sh
```

Step 2: Write the script in the editor

```
root@f304ea52daec59c:~#
str="Welcome to"
newstr="${str} Javatpoint."
echo "$newstr"
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./cat5.sh
Welcome to Javatpoint.
```

48) Using Underscore

Step1: Create a Bash Script

```
root@f304ea52daec59c:~# touch cat6.sh
root@f304ea52daec59c:~# chmod +x cat6.sh
root@f304ea52daec59c:~# vi cat6.sh
```

Step 2: Write the script in the editor

```
root@f304ea52daec59c:~#
str1="Hello"
str2="World!"
echo "${str1}_${str2}"
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./cat6.sh
Hello World!
```

49) Using any Character

Step1: Create a Bash Script

```
root@f304ea52daec59c:~# touch cat7.sh
root@f304ea52daec59c:~# chmod +x cat7.sh
root@f304ea52daec59c:~# vi cat7.sh
```

Step 2: Write the script in the editor

```
root@f304ea52daec59c:~#
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"
```

Step 3: Run the script

```
root@f304ea52daec59c:~# ./cat7.sh
Enter First Name: ankita
Enter State: karnataka
Enter Age: 22
Name, State, Age: ankita,karnataka,22
```

BASH FUNCTIONS

50) Function Name

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch function1.sh
root@f304ea52daec59c:~# chmod +x function1.sh
chmod: cannot access 'function1.sh': No such file or directory
root@f304ea52daec59c:~# chmod +x function1.sh
root@f304ea52daec59c:~# vi function1.sh
root@f304ea52daec59c:~# ./function1.sh
Welcome to Javatpoint.

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
JTP()
echo 'Welcome to Javatpoint.'
JTP

```

Step 3: Run the script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch function1.sh
root@f304ea52daec59c:~# chmod +x function1.sh
chmod: cannot access 'function1.sh': No such file or directory
root@f304ea52daec59c:~# chmod +x function1.sh
root@f304ea52daec59c:~# vi function1.sh
root@f304ea52daec59c:~# ./function1.sh
Welcome to Javatpoint.

```

51) Function Name

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch function2.sh
root@f304ea52daec59c:~# chmod +x function2.sh
root@f304ea52daec59c:~# vi function2.sh
root@f304ea52daec59c:~# ./function2.sh
Welcome to Javatpoint.

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
function JTP {
echo 'Welcome to Javatpoint.'
}
JTP

```

Step 3: Run the script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch function2.sh
root@f304ea52daec59c:~# chmod +x function2.sh
root@f304ea52daec59c:~# vi function2.sh
root@f304ea52daec59c:~# ./function2.sh
Welcome to Javatpoint.

```

52) Script to pass and access arguments

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch function3.sh
root@f304ea52daec59c:~# chmod +x function3.sh
root@f304ea52daec59c:~# vi function3.sh
root@f304ea52daec59c:~# ./function3.sh
root@f304ea52daec59c:~# vi function3.sh
root@f304ea52daec59c:~# ./function3.sh
WelcomeyouonJavatpoint.

```

```

root@f304ea52daec59c:~# vi function3.sh
root@f304ea52daec59c:~# ./function3.sh
We
welcome
you
on
Javatpoint.

```

Step 2: Write the script in the editor

```
root@f304ea52daec59c: ~  
function_arguments()  
{  
    echo $1  
    echo $2  
    echo $3  
    echo $4  
    echo $5  
}  
function_arguments "We" "welcome" "you" "on" "Javatpoint."
```

Step 3: Run the script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch function3.sh  
root@f304ea52daec59c:~# chmod +x function3.sh  
root@f304ea52daec59c:~# vi function3.sh  
root@f304ea52daec59c:~# ./function3.sh  
root@f304ea52daec59c:~# vi function3.sh  
root@f304ea52daec59c:~# ./function3.sh  
We  
welcome  
you  
on  
Javatpoint.
```

53) Variable Scope

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch function4.sh  
root@f304ea52daec59c:~# chmod +x function4.sh  
root@f304ea52daec59c:~# vi function4.sh  
root@f304ea52daec59c:~# ./function4.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.
```

Step 2: Write the script in the editor

```
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."
```

Step 3: Run the script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch function4.sh  
root@f304ea52daec59c:~# chmod +x function4.sh  
root@f304ea52daec59c:~# vi function4.sh  
root@f304ea52daec59c:~# ./function4.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.
```

54) Return Values

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch function5.sh  
root@f304ea52daec59c:~# chmod +x function5.sh  
root@f304ea52daec59c:~# vi function5.sh  
root@f304ea52daec59c:~# ./function5.sh  
Hello Use  
Hello Reader  
The previous function returned a value of 5
```

Step 2: Write the script in the editor

```
root@f304ea52daec59c: ~  
print_it() {  
    echo Hello $1  
    return 5  
}  
  
print_it Use  
print_it Reader  
echo The previous function returned a value of $?
```

Step 3: Run the script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch function5.sh  
root@f304ea52daec59c:~# chmod +x function5.sh  
root@f304ea52daec59c:~# vi function5.sh  
root@f304ea52daec59c:~# ./function5.sh  
Hello Use  
Hello Reader  
The previous function returned a value of 5
```

55) Return Values

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch function6.sh  
root@f304ea52daec59c:~# chmod +x function6.sh  
root@f304ea52daec59c:~# vi function6.sh  
root@f304ea52daec59c:~# ./function6.sh  
Welcome to Javatpoint.
```

Step 2: Write the script in the editor

```
root@f304ea52daec59c: ~  
print_it() {  
    local my_greet="Welcome to Javatpoint."  
    echo "$my_greet"  
}  
  
my_greet="$(print_it)"  
echo $my_greet
```

Step 3: Run the script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch function6.sh  
root@f304ea52daec59c:~# chmod +x function6.sh  
root@f304ea52daec59c:~# vi function6.sh  
root@f304ea52daec59c:~# ./function6.sh  
Welcome to Javatpoint.
```

56) Overriding Commands

Step1: Create a Bash Script

```
root@f304ea52daec59c: ~  
root@f304ea52daec59c:~# touch function7.sh  
root@f304ea52daec59c:~# chmod +x function7.sh  
root@f304ea52daec59c:~# vi function7.sh  
root@f304ea52daec59c:~# ./function7.sh  
[01-29 12:31:33] : Welcome to Javatpoint.
```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
echo () {
    builtin echo -n `date +%m-%d %H:%M:%S` " "
    builtin echo $1
}

echo "Welcome to Javatpoint."

```

Step 3: Run the script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~#
root@f304ea52daec59c:~# touch function7.sh
root@f304ea52daec59c:~# chmod +x function7.sh
root@f304ea52daec59c:~# vi function7.sh
root@f304ea52daec59c:~# ./function7.sh
[01-29 12:31:33] : Welcome to Javatpoint.

```

BASH ARRAY

57) Reference Elements

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch array1.sh
root@f304ea52daec59c:~# chmod +x array1.sh
root@f304ea52daec59c:~# vi array1.sh
root@f304ea52daec59c:~# ./array1.sh

root@f304ea52daec59c:~# vi array1.sh
root@f304ea52daec59c:~# ./array1.sh
Javatpoint

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
declare -a example_array=( "Welcome" "To" "Javatpoint" )

#Printing the element with index of 2
echo ${example_array[2]}

```

Step 3: Run the script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch array1.sh
root@f304ea52daec59c:~# chmod +x array1.sh
root@f304ea52daec59c:~# vi array1.sh
root@f304ea52daec59c:~# ./array1.sh

root@f304ea52daec59c:~# vi array1.sh
root@f304ea52daec59c:~# ./array1.sh
Javatpoint

```

58) Reference Elements

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch array1.sh
root@f304ea52daec59c:~# touch array2.sh
root@f304ea52daec59c:~# chmod +x array2.sh
root@f304ea52daec59c:~# vi array2.sh
root@f304ea52daec59c:~# ./array2.sh
WelcomeToJavatpoint

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
declare -a example_array=( "Welcome""To""Javatpoint" )

#Printing all the elements
echo "${example_array[@]}"

```

Step 3: Run the script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch array1.sh
root@f304ea52daec59c:~# touch array2.sh
root@f304ea52daec59c:~# chmod +x array2.sh
root@f304ea52daec59c:~# vi array2.sh
root@f304ea52daec59c:~# ./array2.sh
WelcomeToJavatpoint

```

59) Printing the Keys of an Array

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch array3.sh
root@f304ea52daec59c:~# chmod +x array3.sh
root@f304ea52daec59c:~# vi array3.sh
root@f304ea52daec59c:~# ./array3.sh
0 1 2

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
declare -a example_array=( "Welcome" "To" "Javatpoint" )

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

```

Step 3: Run the script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch array3.sh
root@f304ea52daec59c:~# chmod +x array3.sh
root@f304ea52daec59c:~# vi array3.sh
root@f304ea52daec59c:~# ./array3.sh
0 1 2

```

60) Finding Array Length

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch array4.sh
root@f304ea52daec59c:~# chmod +x array4.sh
root@f304ea52daec59c:~# vi array4.sh
root@f304ea52daec59c:~# ./array4.sh
The array contains 1 elements
root@f304ea52daec59c:~# vi array4.sh
root@f304ea52daec59c:~# ./array4.sh
The array contains 3 elements

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
declare -a example_array=( "Welcome" "To" "Javatpoint" )

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

```

Step 3: Run the script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch array4.sh
root@f304ea52daec59c:~# chmod +x array4.sh
root@f304ea52daec59c:~# vi array4.sh
root@f304ea52daec59c:~# ./array4.sh
The array contains 1 elements
root@f304ea52daec59c:~# vi array4.sh
root@f304ea52daec59c:~# ./array4.sh
The array contains 3 elements

```

61) Loop through the Array

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch arrays5.sh
root@f304ea52daec59c:~# chmod +x arrays5.sh
root@f304ea52daec59c:~# vi arrays5.sh
root@f304ea52daec59c:~# ./arrays5.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

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
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

```

Step 3: Run the script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch arrays5.sh
root@f304ea52daec59c:~# chmod +x arrays5.sh
root@f304ea52daec59c:~# vi arrays5.sh
root@f304ea52daec59c:~# ./arrays5.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

```

62) Loop through the Array

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch array6.sh
root@f304ea52daec59c:~# chmod +x array6.sh
root@f304ea52daec59c:~# vi array6.sh
root@f304ea52daec59c:~# ./array6.sh
0 Welcome
1 To
2 Javatpoint

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
declare -a example_array=( "Welcome" "To" "Javatpoint" )

#length of the Array
length=${#example_array[@]}
#
# #Array Loop
for (( i=0; i < ${length}; i++ ))
do
echo $i ${example_array[$i]}
done

```

Step 3: Run the script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch array6.sh
root@f304ea52daec59c:~# chmod +x array6.sh
root@f304ea52daec59c:~# vi array6.sh
root@f304ea52daec59c:~# ./array6.sh
0 Welcome
1 To
2 Javatpoint

```

63) Adding Elements to an Array

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch array7.sh
root@f304ea52daec59c:~# chmod +x array7.sh
root@f304ea52daec59c:~# vi array7.sh

```

Step 2: Write the script in the editor


```

root@f304ea52daec59c: ~
declare -a example_array=( "Java" "Python" "PHP" "HTML" )

#Adding new element
example_array[4]="JavaScript"
#
# #Printing all the elements
echo "${example_array[@]}"
~

```

Step 3: Run the script

```

root@f304ea52daec59c: ~# vi array7.sh
root@f304ea52daec59c: ~# ./array7.sh
Java Python PHP HTML JavaScript

```

64) Adding Elements to an Array

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c: ~# touch array8.sh
root@f304ea52daec59c: ~# chmod +x array8.sh
root@f304ea52daec59c: ~# vi array8.sh
~

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
declare -a example_array=( "Java" "Python" "PHP" )

#Adding new elements
example_array+=( "JavaScript" "CSS" "SQL" )
#
# #Printing all the elements
echo "${example_array[@]}"
~

```

Step 3: Run the script

```

root@f304ea52daec59c: ~# ./array8.sh
Java Python PHP JavaScript CSS SQL

```

65) Updating Array Element

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c: ~# touch array9.sh
root@f304ea52daec59c: ~# chmod +x array9.sh
root@f304ea52daec59c: ~# vi array9.sh
~

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
declare -a example_array=( "We" "welcome" "you" "on" "JSSSIT" )

#Updating the Array Element
example_array[4]=Javatpoint
#
# #Printintg all the elements of the Array
echo "${example_array[@]}"
~

```

Step 3: Run the script

```

root@f304ea52daec59c: ~# vi array9.sh
root@f304ea52daec59c: ~# ./array9.sh
We welcome you on Javatpoint

```

66) Deleting an Element from an Array

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch array10.sh
root@f304ea52daec59c:~# chmod +x array10.sh
root@f304ea52daec59c:~# vi array10.sh

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
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[@]}"

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./array10.sh
Java HTML CSS JavaScript

```

67) Deleting the Entire Array

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch array11.sh
root@f304ea52daec59c:~# vi array11.sh
root@f304ea52daec59c:~# chmod +x array11.sh

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
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[@]}"

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./array11.sh

```

68) Slice Array Elements

Step1: Create a Bash Script

```

root@f304ea52daec59c: ~
root@f304ea52daec59c:~# touch array12.sh
root@f304ea52daec59c:~# chmod +x array12.sh
root@f304ea52daec59c:~# vi array12.sh

```

Step 2: Write the script in the editor

```

root@f304ea52daec59c: ~
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

```

Step 3: Run the script

```

root@f304ea52daec59c:~# ./array12.sh
Python
HTML
CSS
root@f304ea52daec59c:~#

```

