Bash If

In this topic, we will understand how to use **if statements** in Bash scripts to get our automated tasks completed.

Bash if statements are beneficial. They are used to perform conditional tasks in the sequential flow of execution of statements. If statements usually allow us to make decisions in our Bash scripts. They help us to decide whether or not to run a piece of codes based upon the condition that we may set.

Basic if Statements

A basic if statement commands that if a particular condition is true, then only execute a given set of actions. If it is not true, then do not execute those actions. If statement is based on the following format:

Syntax

```
if [ expression ];
then
statements
fi
```

The statement between **then** and **fi** (If backwards) will be executed only if the expression (between the square brackets) is true.

Note: Observe the spaces used in the first line and a semicolon at the end of the first line; both are mandatory to use. If conditional statement ends with fi.

- For using multiple conditions with AND operator:

```
if [ expression_1 ] && [ expression_2 ];
then
statements
fi
```

- For using multiple conditions with OR operator:

1. if [expression_1] || [expression_2];
2. then
3. statements
4. fi

- For compound expressions with AND & OR operators, we can use the following syntax:

1. if [expression_1 && expression_2 || expression_3];
2. then
3. statements
4. fi

Following are some examples demonstrating the usage of if statement:

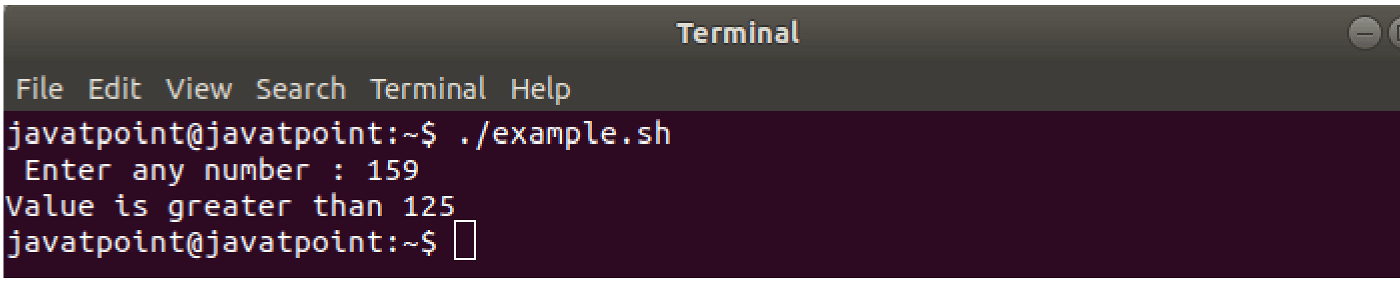
Example 1

In this example, take a user-input of any number and check if the value is greater than 125.

```
#!/bin/bash

read -p " Enter number : " number

if [ $number -gt 125 ]
then
echo "Value is greater than 125"
fi
```

Output

If we enter the number 159, then the output will look like:



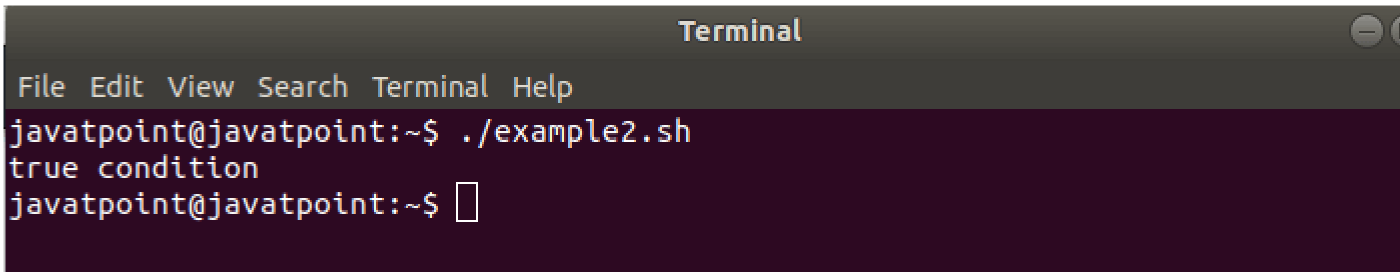
```
javatpoint@javatpoint:~$ ./example.sh
 Enter any number : 159
Value is greater than 125
javatpoint@javatpoint:~$
```

Example 2

In this example, we demonstrate the usage of **if statement** with a simple scenario of comparing two strings:

```
#!/bin/bash

# if condition is true
if [ "myfile" == "myfile" ];
then
echo "true condition"
fi

# if condition is false
if [ "myfile" == "yourfile" ];
then
echo "false condition"
fi
```

Output



```
javatpoint@javatpoint:~$ ./example2.sh
true condition
javatpoint@javatpoint:~$
```

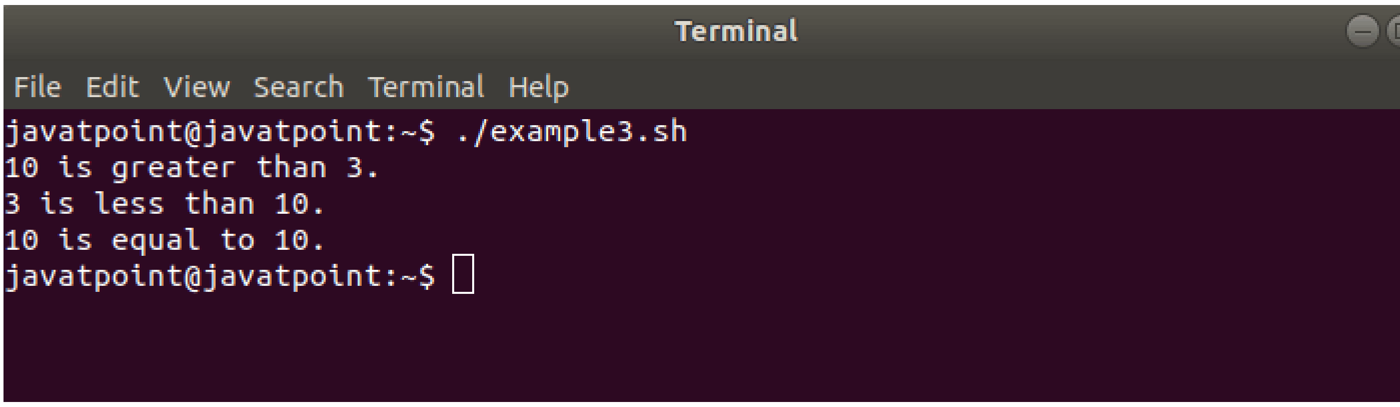
Example 3

In this example, we demonstrate how to compare numbers by using the if statement:

```
#!/bin/bash

#if condition (greater than) is true
if [ 10 -gt 3 ];
then
echo "10 is greater than 3."
fi

#if condition (greater than) is false
if [ 3 -gt 10 ];
then
echo "3 is not greater than 10."
fi

#if condition (lesser than) is true
if [ 3 -lt 10 ];
then
echo "3 is less than 10."
fi

#if condition (lesser than) is false
if [ 10 -lt 3 ];
then
echo "10 is not less than 3."
fi

#if condition (equal to) is true
if [ 10 -eq 10 ];
then
echo "10 is equal to 10."
fi
```

-
- #if condition (equal to) is false
- if [10 -eq 9];
- then
- echo "10 is not equal to 9"
- fi

Output



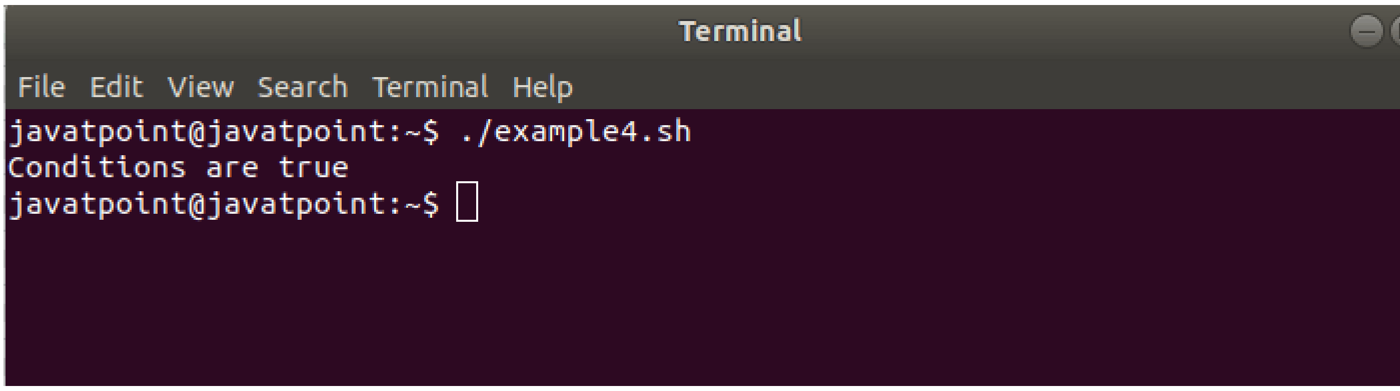
```
javatpoint@javatpoint:~$ ./example3.sh
10 is greater than 3.
3 is less than 10.
10 is equal to 10.
javatpoint@javatpoint:~$ 
```

Example 4

In this example, we will define how to use AND operator to include multiple conditions in the if expression:

- #!/bin/bash
-
- # TRUE && TRUE
- if [8 -gt 6] && [10 -eq 10];
- then
- echo "Conditions are true"
- fi
-
- # TRUE && FALSE
- if ["mylife" == "mylife"] && [3 -gt 10];
- then
- echo "Conditions are false"
- fi

Output



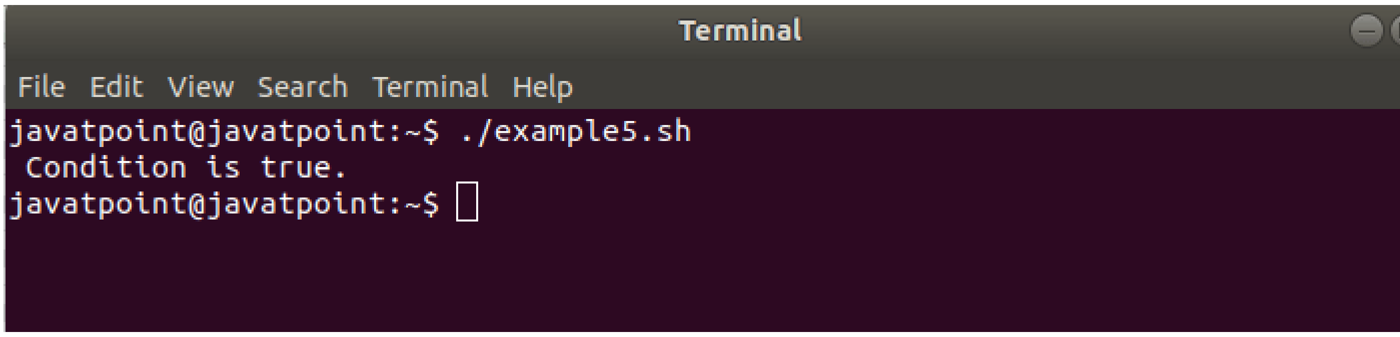
```
javatpoint@javatpoint:~$ ./example4.sh
Conditions are true
javatpoint@javatpoint:~$
```

Example 5

In this example, we will define how to use OR operator to include multiple conditions in the if expression:

```
#!/bin/bash

# TRUE || FALSE
if [ 8 -gt 7 ] || [ 10 -eq 3 ];
then
echo " Condition is true. "
fi

# FALSE || FALSE
if [ "mylife" == "yourlife" ] || [ 3 -gt 10 ];
then
echo " Condition is false. "
fi
```

Output



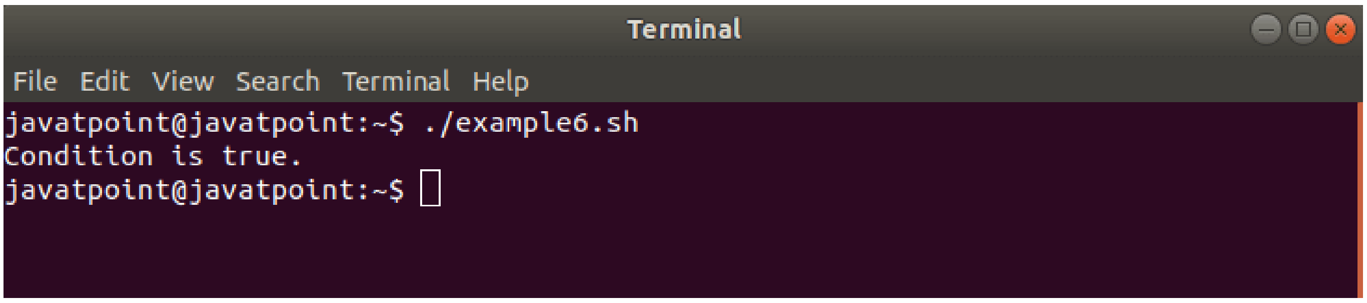
```
javatpoint@javatpoint:~$ ./example5.sh
 Condition is true.
javatpoint@javatpoint:~$
```

Example 6

In this example, we will define how to use AND and OR to include multiple conditions in the if expression:

```
#!/bin/bash

# TRUE && FALSE || FALSE || TRUE
if [[ 10 -eq 10 && 5 -gt 4 || 3 -eq 4 || 3 -lt 6 ]];
then
echo "Condition is true."
fi

# TRUE && FALSE || FALSE
if [[ 8 -eq 8 && 8 -gt 10 || 9 -lt 5 ]];
then
echo "Condition is false"
fi
```

Output



```
javatpoint@javatpoint:~$ ./example6.sh
Condition is true.
javatpoint@javatpoint:~$
```

Options for If statement in Bash Scripting

If statement contains many options to perform a specific task. These options can be used for file operations, string operations, etc. Following are the some mostly used options:

Options (Operators)	**Description**
! EXPRESSION	To check if EXPRESSION is false.
-n STRING	To check if the length of STRING is greater than zero.
-z STRING	To check if the length of STRING is zero (i.e., it is empty)
STRING1 == STRING2	To check if STRING1 is equal to STRING2.
STRING1 != STRING2	To check if STRING1 is not equal to STRING2.
INTEGER1 -eq INTEGER2	To check if INTEGER1 is numerically equal to INTEGER2.
INTEGER1 -gt INTEGER2	To check if INTEGER1 is numerically greater than INTEGER2.
INTEGER1 -lt INTEGER2	To check if INTEGER1 is numerically less than INTEGER2.
-d FILE	To check if FILE exists and it is a directory.
-e FILE	To check if FILE exists.
-r FILE	To check if FILE exists and the read permission is granted.
-s FILE	To check if FILE exists and its size is greater than zero (which means that it is not empty).
-w FILE	To check if FILE exists and the write permission is granted.
x FILE	To check if FILE exists and the execute permission is granted.

Nested If

You can apply as many 'if statements' as required inside your bash script. It is also possible to use an if statement inside another 'if statement'. It is known as Nested If Statement.

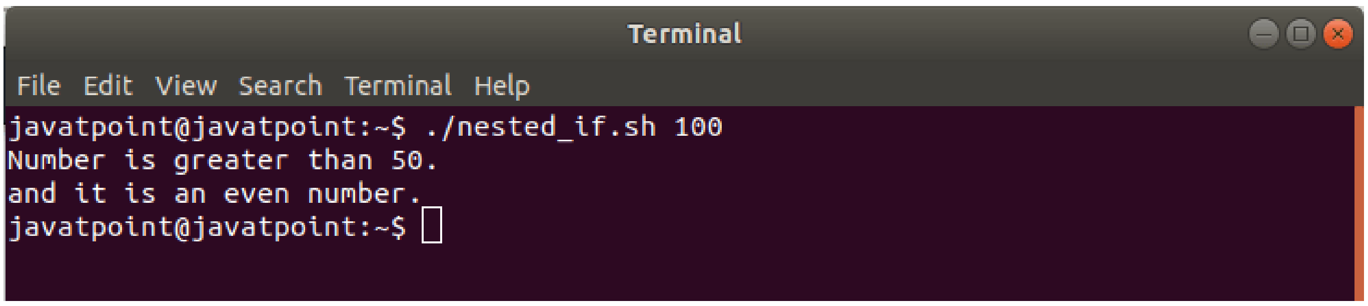
Example

In this example, we will find "if a given number is greater than 50 and if it is an even number" by using nested if expression.

```
#!/bin/bash
#Nested if statement

if [ $1 -gt 50 ]
then
  echo "Number is greater than 50."

  if (( $1 % 2 == 0 ))
  then
    echo "and it is an even number."
  fi
fi
```

Output

If we input an argument value as 100, then the output will look like:



```
javatpoint@javatpoint:~$ ./nested_if.sh 100
Number is greater than 50.
and it is an even number.
javatpoint@javatpoint:~$
```