Bash For Loop

In this topic, we will understand the usage of **for loop** in Bash scripts.

Like any other programming language, bash shell scripting also supports 'for loops' to perform repetitive tasks. It helps us to iterate a particular set of statements over a series of words in a string, or elements in an array. For example, you can either run UNIX command (or task) many times or just read and process the list of commands using a 'for loop'.

Syntax of For Loop

We can apply 'for loop' on bash script in two ways. One way is 'for-in' and another way is the c-style syntax. Following is the syntax of 'for loop' in bash shell scripting:

1. **for** variable in list
2. **do**
3. commands
4. done

Or

1. **for** ((expression1; expression2; expression3))
2. **do**
3. commands
4. done

There are some key points of 'for loop' statement:

- Each block of 'for loop' in bash starts with 'do' keyword followed by the commands inside the block. The 'for loop' statement is closed by 'done' keyword.
- The number of time for which a 'for loop' will iterate depends on the declared list variables.
- The loop will select one item from the list and assign the value on a variable which will be used within the loop.

- After the execution of commands between 'do' and 'done', the loop goes back to the top and select the next item from the list and repeat the whole process.
- The list can contain numbers or string etc. separated by spaces.

Some of the 'for loop' examples are given below to illustrate how do they work:

Basic 'For Loop' Example

Bash Script

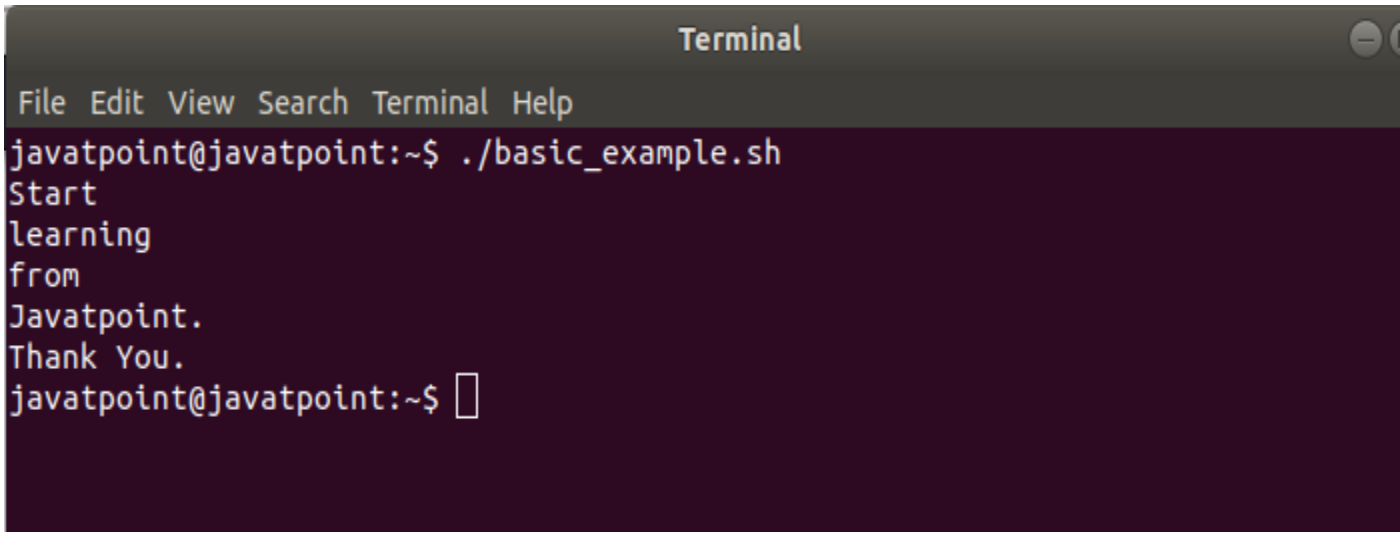
```
#!/bin/bash
#This is the basic example of 'for loop'.

learn="Start learning from Javatpoint."

for learn in $learn
do
echo $learn
done

        echo "Thank You."
```

Output

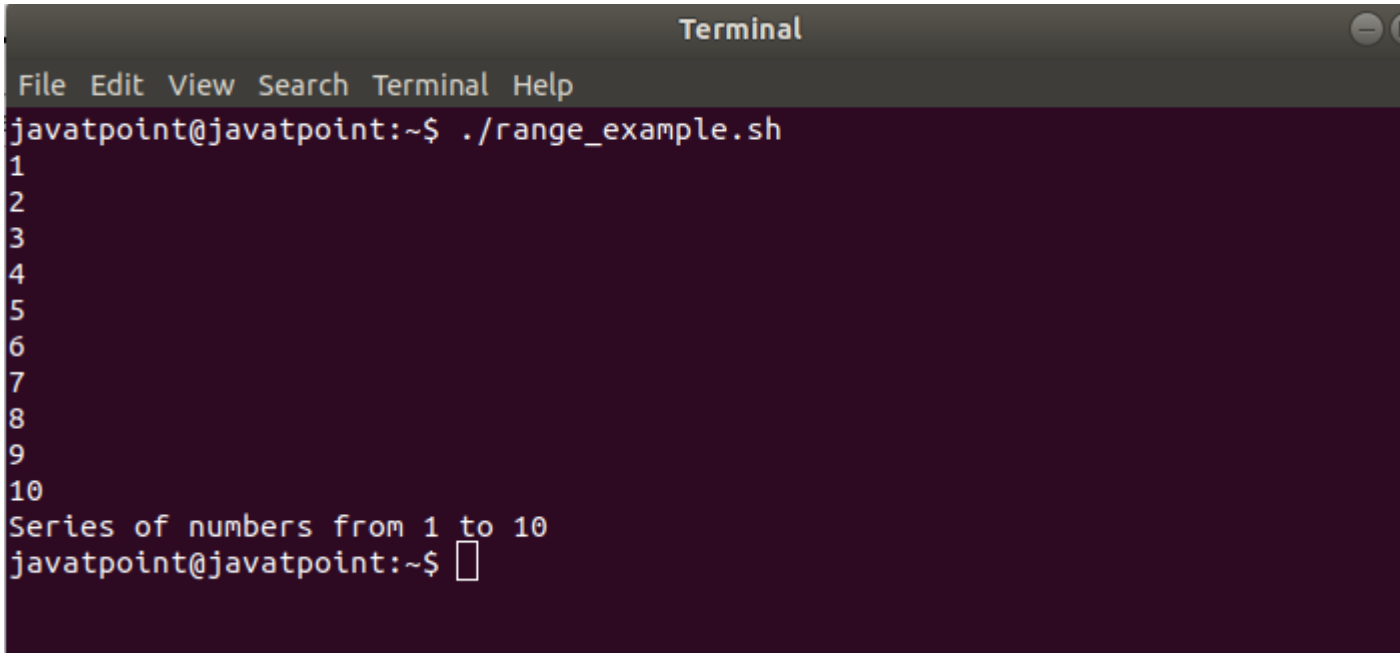


For Loop to Read a Range

Bash Script

```
1. #!/bin/bash
2. #This is the basic example to print a series of numbers from 1 t
   o 10.
3.
4. for num in {1..10}
5. do
6. echo $num
7. done
8.
9. echo "Series of numbers from 1 to 10."
```

Output



For Loop to Read a Range with Increment/Decrement

We can increase or decrease a specified value by adding two another dots (..) and the value to step by, e.g., {START..END..INCREMENT}. Check out the example below:

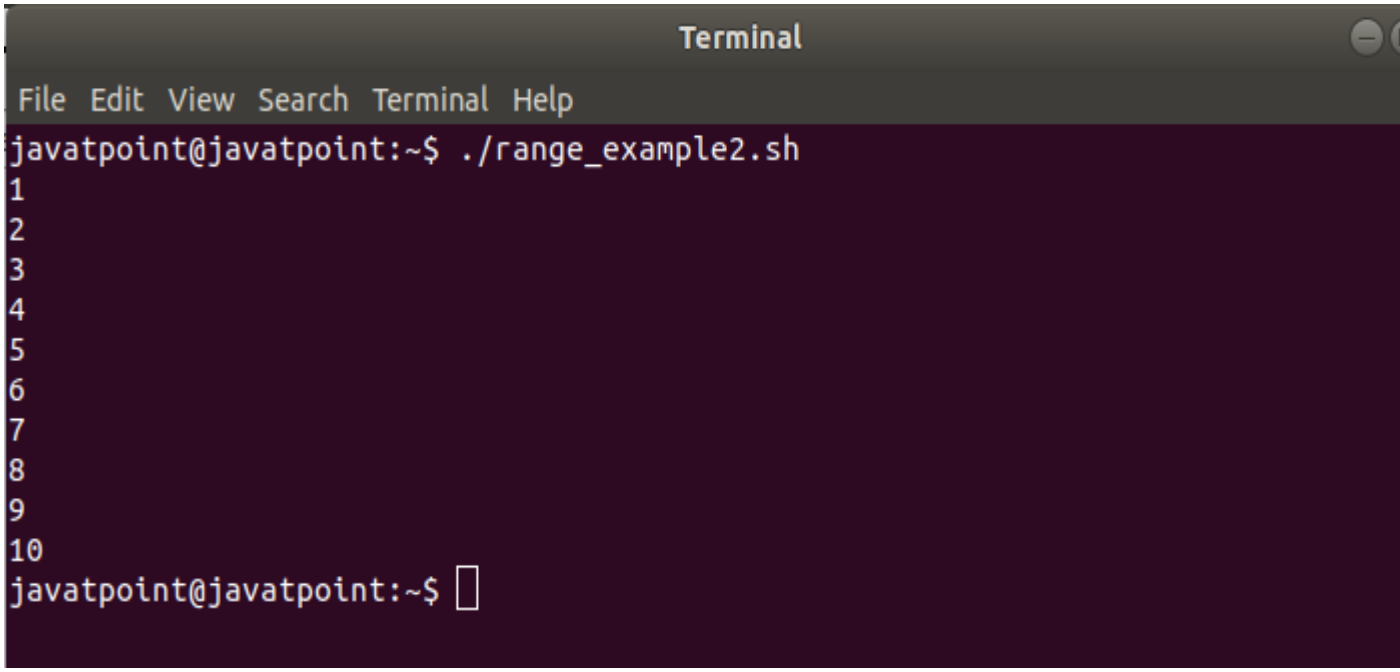
For Increment

```
1. #!/bin/bash
2.
```

```
#For Loop to Read a Range with Increment

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

Output



For Decrement

```
#!/bin/bash

#For Loop to Read a Range with Decrement

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

Output

```
Terminal
File  Edit  View  Search  Terminal  Help
javatpoint@javatpoint:~$ ./range_example2.2.sh
10
9
8
7
6
5
4
3
2
1
0
javatpoint@javatpoint:~$
```

For Loop to Read Array Variables

We can use 'for loop' to iterate the values of an array.

The syntax can be defined as:

```
array=(  "element1" "element 2" .  .  "elementN" )

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

Output

For each element in 'array', the statements or set of commands from 'do' till 'done' are executed. Each element could be accessed as 'i' within the loop for the respective iteration. Check out the example below explaining the use of 'for loop' to iterate over elements of an array:

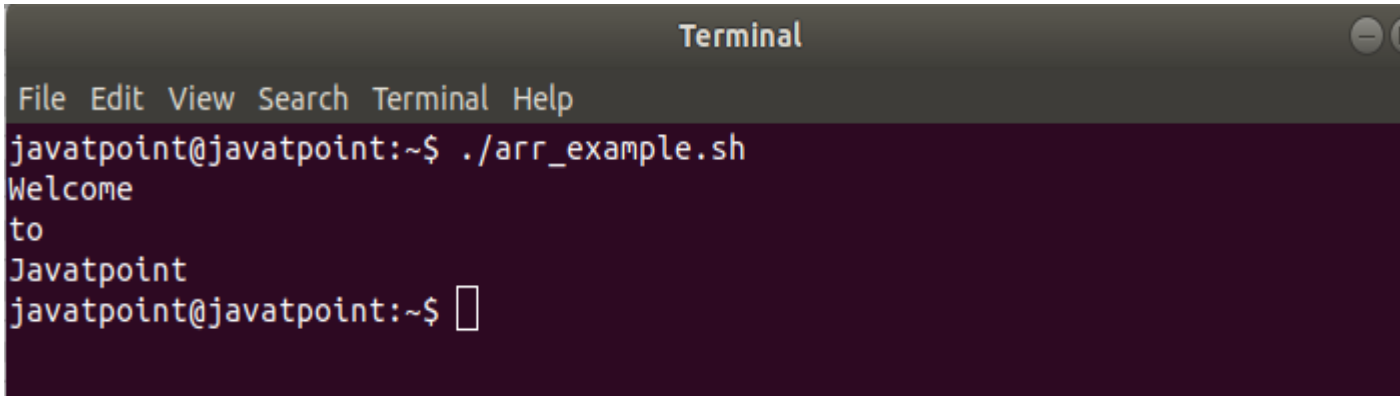
Bash Script

```
#!/bin/bash

```

```
3. #Array Declaration
4. arr=( "Welcome""to""Javatpoint" )
5.
6. for i in "${arr[@]}"
7. do
8. echo $i
9. done
```

Output



For Loop to Read white spaces in String as word separators

The syntax can be defined as below:

```
1. #!/bin/bash
2.
3. for word in $str;
4. do
5. <Statements>
6. done
```

Here, **str** refers to a string.

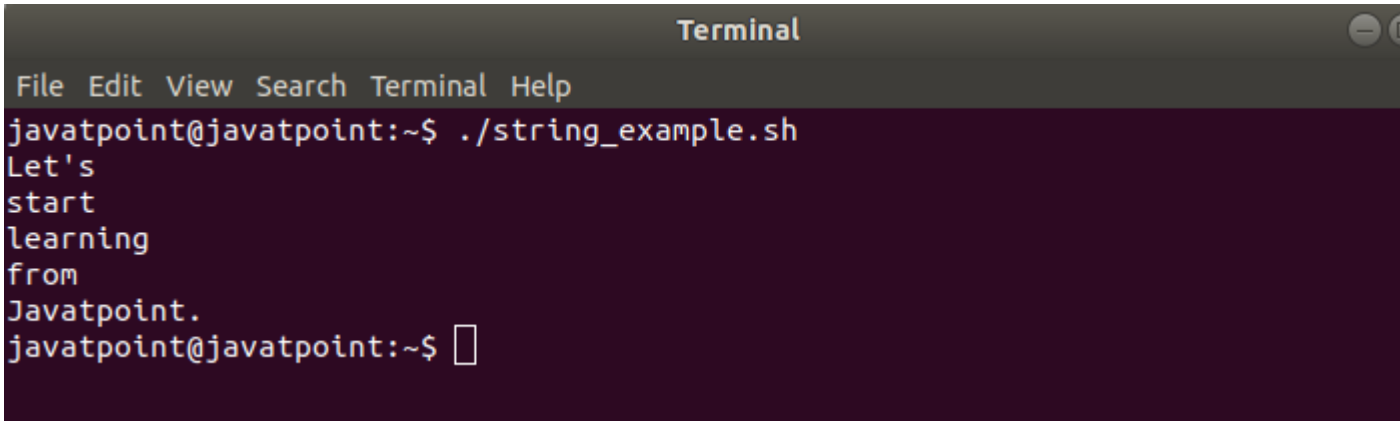
The statements from 'do' till 'done' are executed for each 'word' of a string. Check out the example below:

Bash Script

```
1. #!/bin/bash
2. #For Loop to Read white spaces in String as word separators
3.
```

```
4. str="Let's start
5. learning from Javatpoint."
6.
7. for i in $str;
8. do
9. echo "$i"
10.        done
```

Output



For Loop to Read each line in String as a word

The syntax can be defined as below:

```
1. #!/bin/bash
2.
3. for word in "$str";
4. do
5. <Statements>
6. done
```

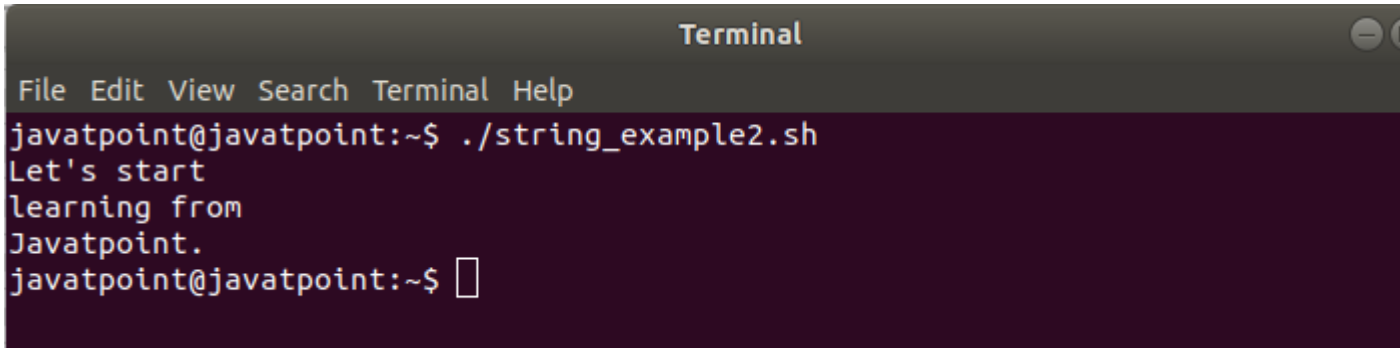
Here, the statements from 'do' till 'done' are executed for each 'line' of a string. Check out the example below:

Bash Script

```
1. #!/bin/bash
2. #For Loop to Read each line in String as a word
3.
4. str="Let's start
```

```
5. learning from
6. Javatpoint."
7.
8. for i in "$str";
9. do
10.     echo "$i"
11.     done
```

Output



Note: The only difference between 'For Loop to Read white spaces in String as word separators' and 'For Loop to Read each line in String as a word' is the double quotes around string variable.

For Loop to Read Three-expression

Three expression syntax is the most common syntax of 'for loop'. The first expression refers to the process of initialization, the second expression refers to the termination, and the third expression refers to the increment or decrement.

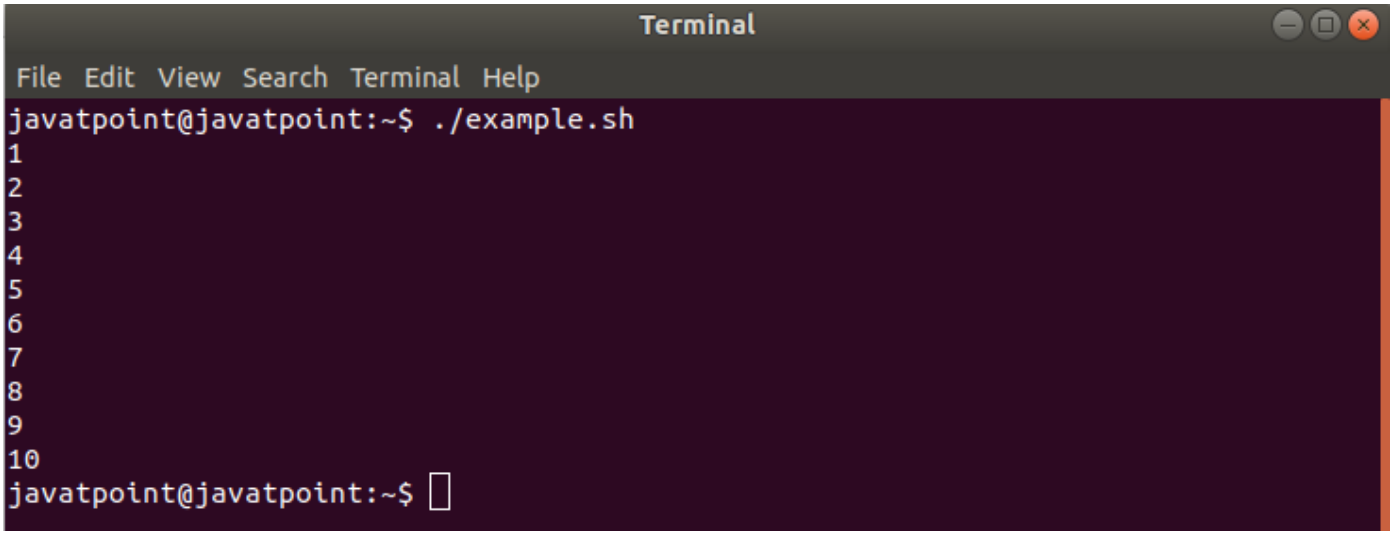
Check out the example below to print 1 to 10 numbers using three expressions with for loop:

Bash Script

```
1. #!/bin/bash
2. #For Loop to Read Three-expression
3.
4. for ((i=1; i<=10; i++))
5. do
6. echo "$i"
```

```
7. done
```

Output



For Loop with a Break Statement

A 'break' statement can be used inside 'for' loop to terminate from the loop.

Bash Script

```
1. #!/bin/bash
2. #Table of 2
3.
4. for table in {2..100..2}
5. do
6. echo $table
7. if [ $table == 20 ]; then
8. break
9. fi
10.        done
```

Output

```
Terminal
File  Edit  View  Search  Terminal  Help
javatpoint@javatpoint:~$ ./break_example.sh
2
4
6
8
10
12
14
16
18
20
javatpoint@javatpoint:~$
```

For Loop with a Continue Statement

We can use the 'continue' statement inside the 'for' loop to skip any specific statement on a particular condition. It tells Bash to stop executing that particular iteration of the loop and process the next iteration.

Bash Script

```
#!/bin/bash
#Numbers from 1 to 20, ignoring from 6 to 15 using continue st
atement"

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

Output

```
Terminal
File  Edit  View  Search  Terminal  Help
javatpoint@javatpoint:~$ ./continue_example.sh
1
2
3
4
5
16
17
18
19
20
javatpoint@javatpoint:~$
```

Infinite Bash For Loop

When there is no 'start, condition, and increment' in the bash three expressions for loop, it becomes an infinite loop. To terminate the infinite loop in Bash, we can press Ctrl+C.

Bash Script

```
1. #!/bin/bash
2.
3. i=1;
4. for (( ; ; ))
5. do
6. sleep 1s
7. echo "Current Number: $((i++))"
8. done
```

Output

```
Terminal
File  Edit  View  Search  Terminal  Help
javatpoint@javatpoint:~$ ./infinite.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
^C
javatpoint@javatpoint:~$
```