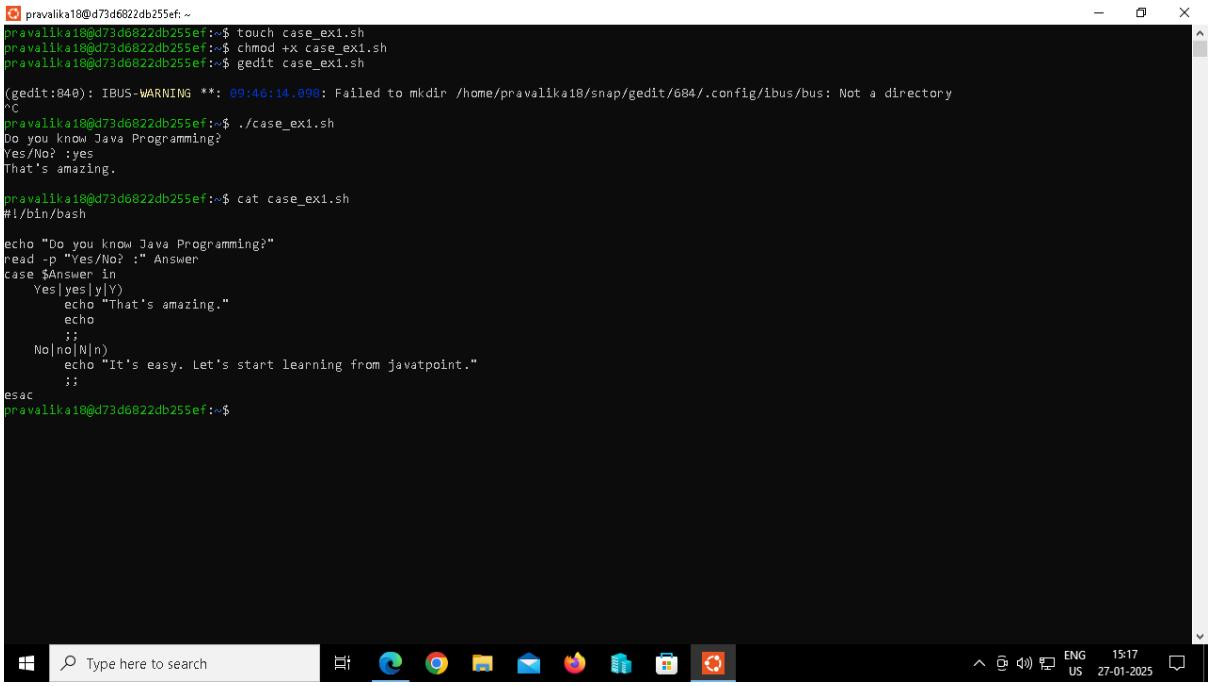
CASE

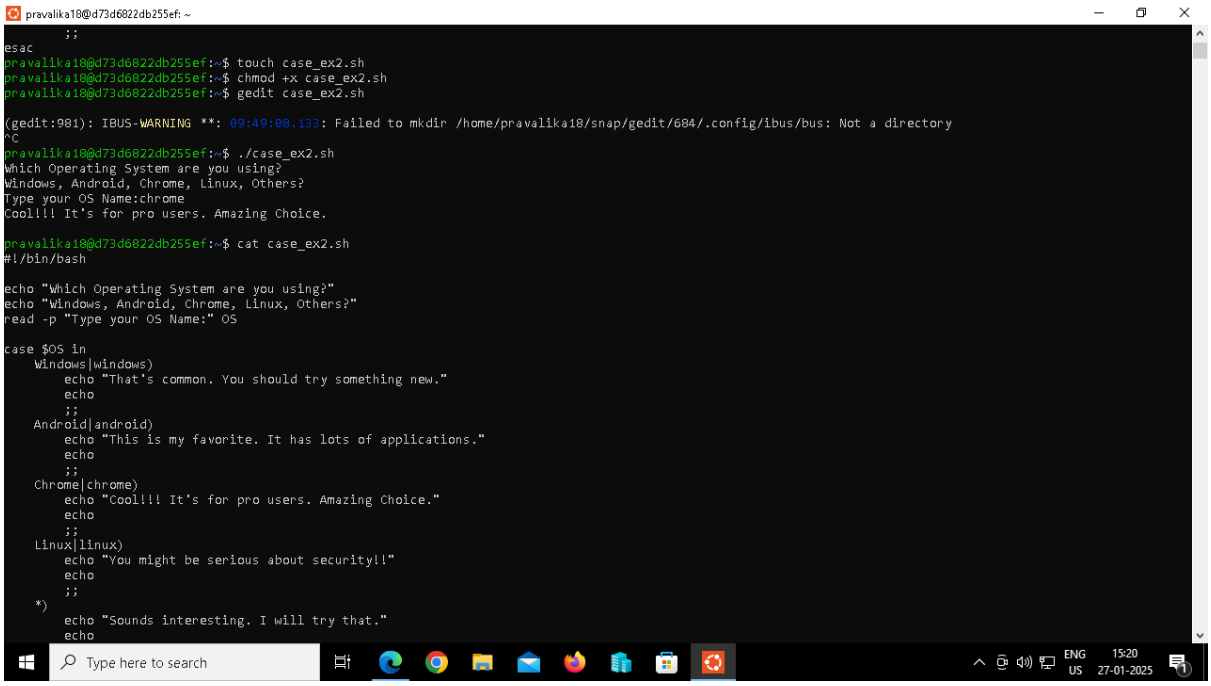
Example 1

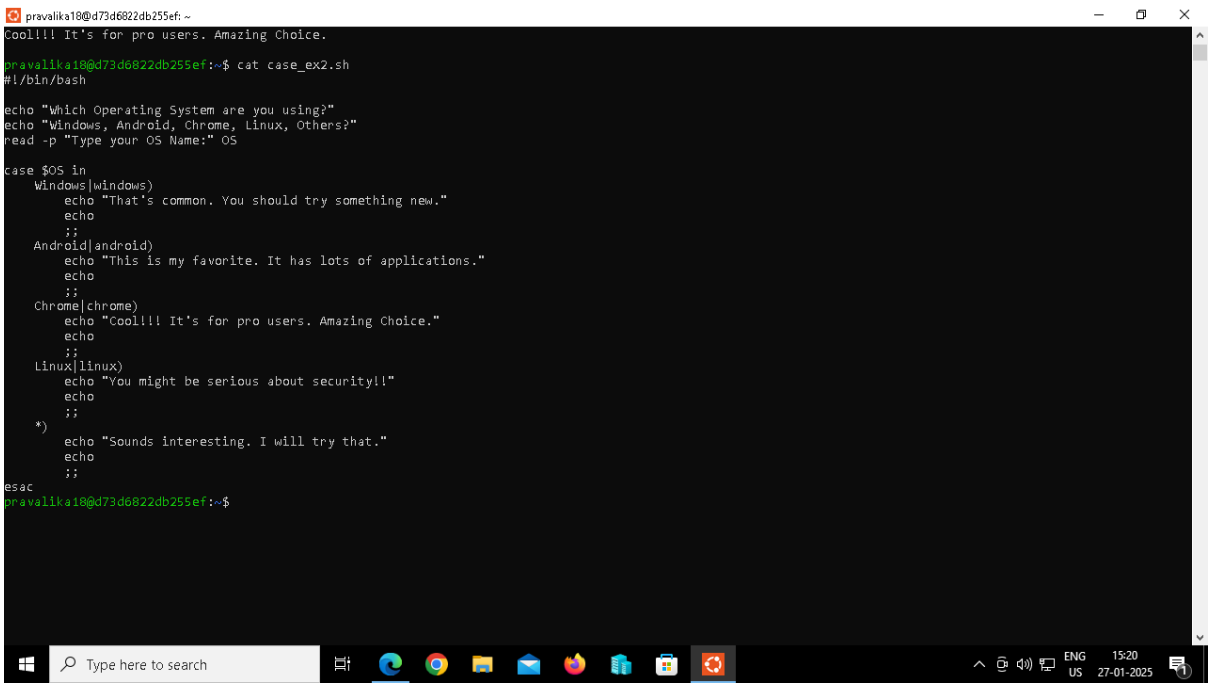
In this example, we have defined a simple scenario to demonstrate the use of the case statement.



Example 2

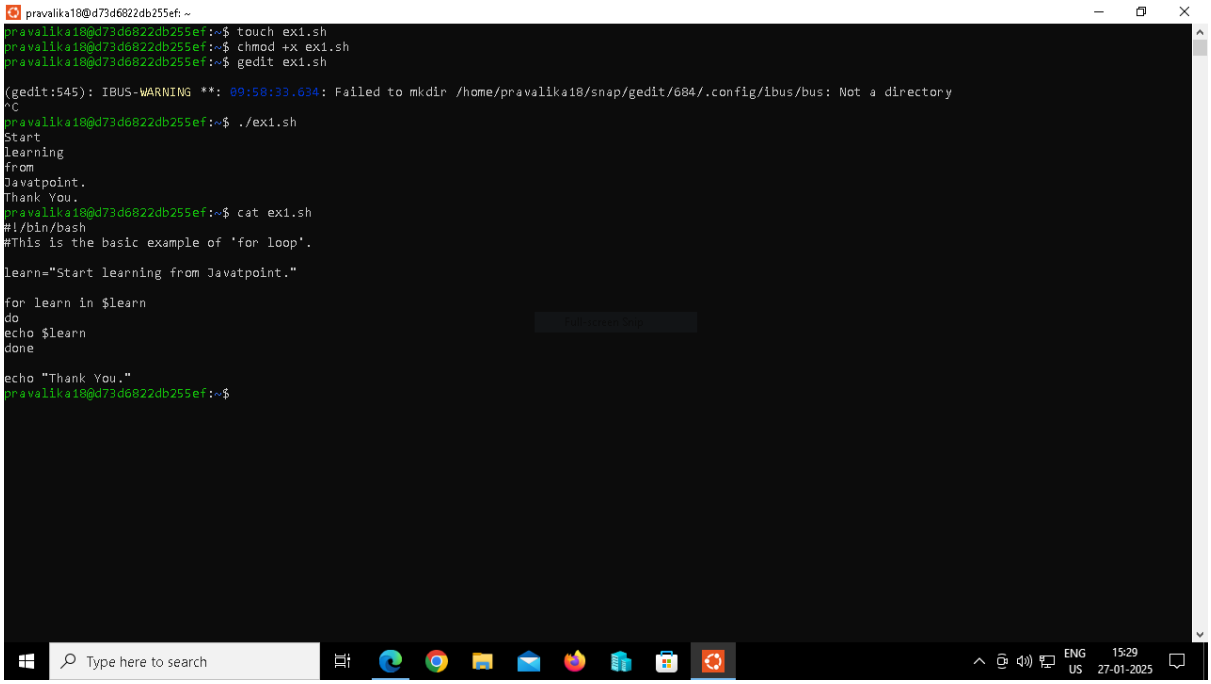
In this example, we have defined a combined scenario where there is also a default case when no previous matched case is found.



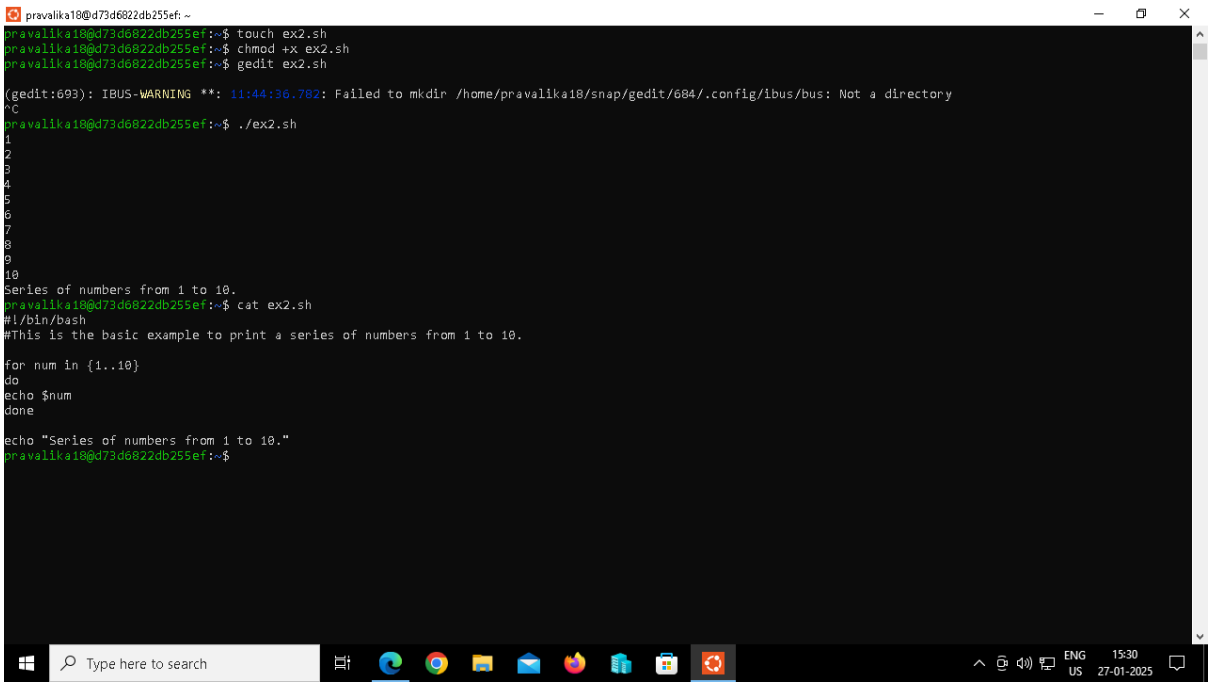


FOR LOOP

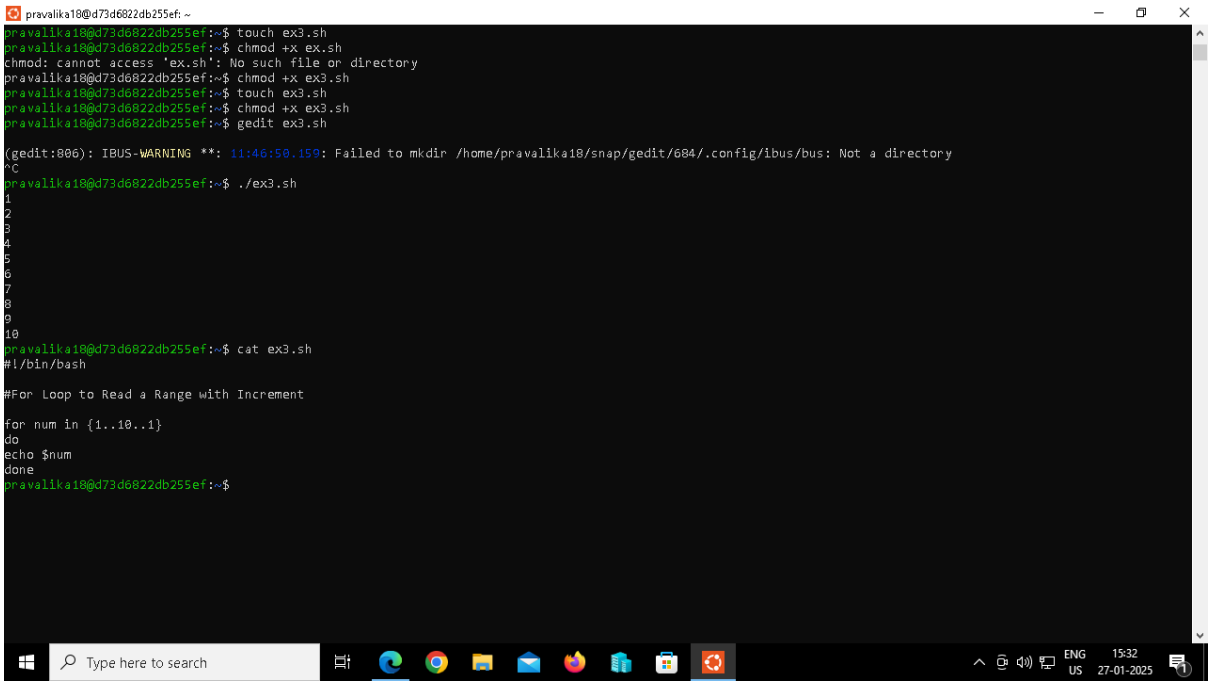
1.Basic 'For Loop' Example



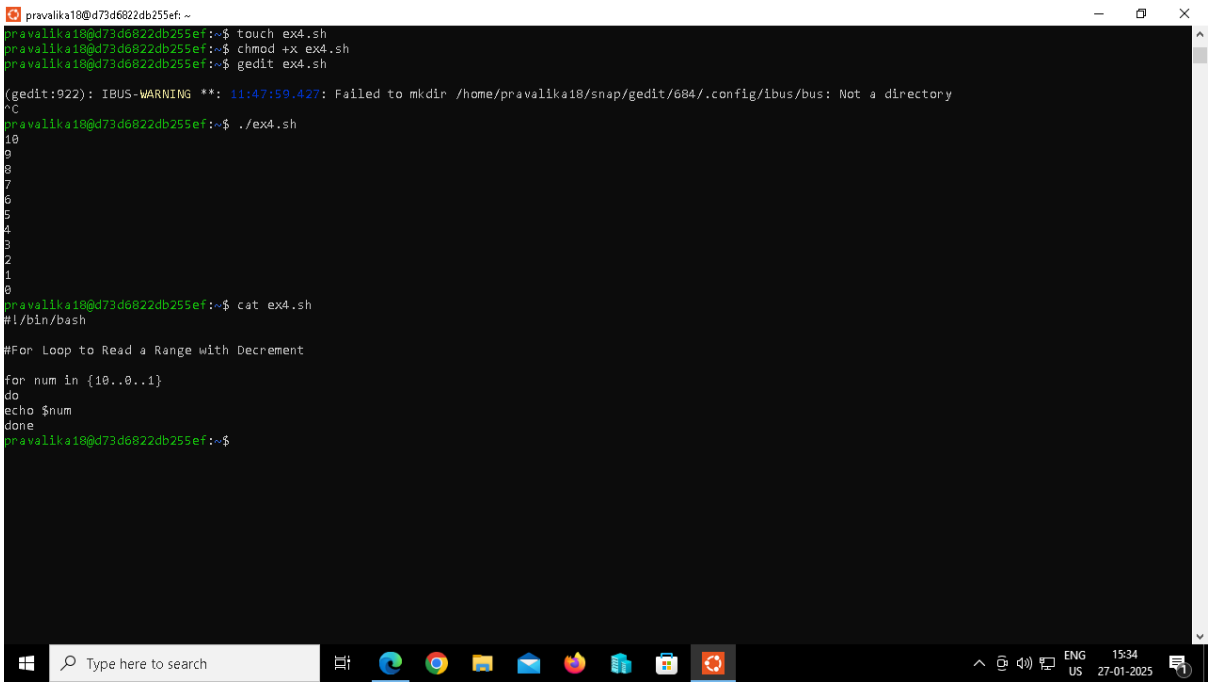
2. For Loop to Read a Range



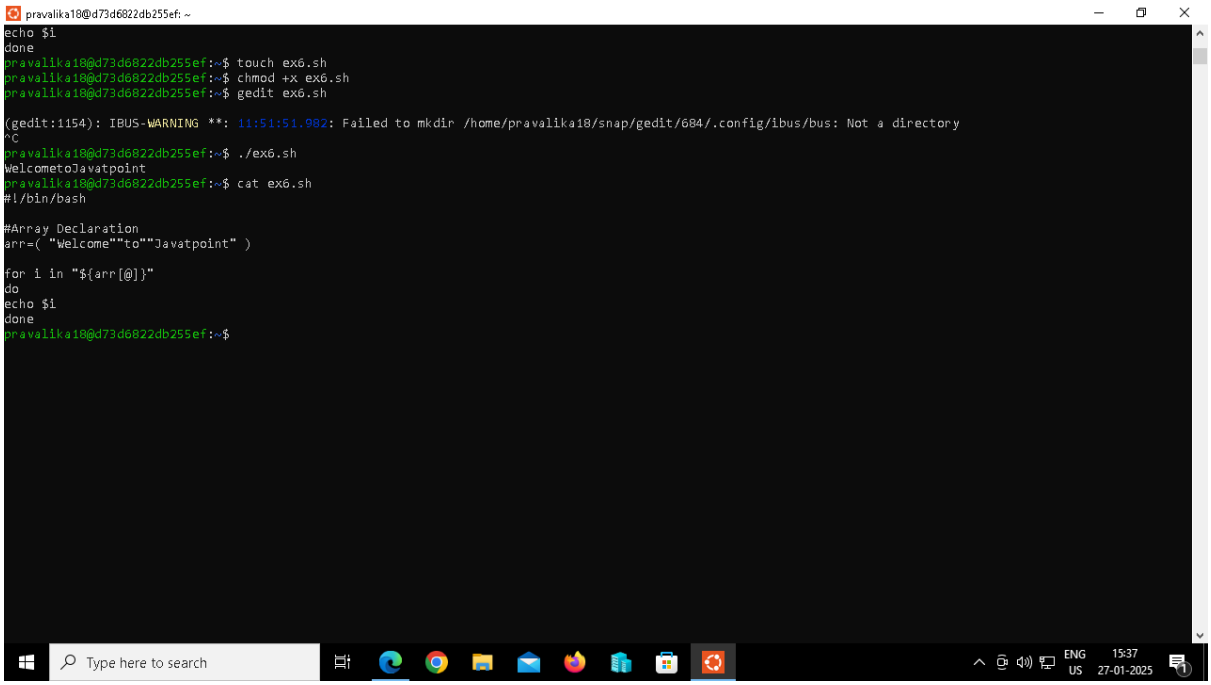
3. For Loop to Read a Range with Increment.



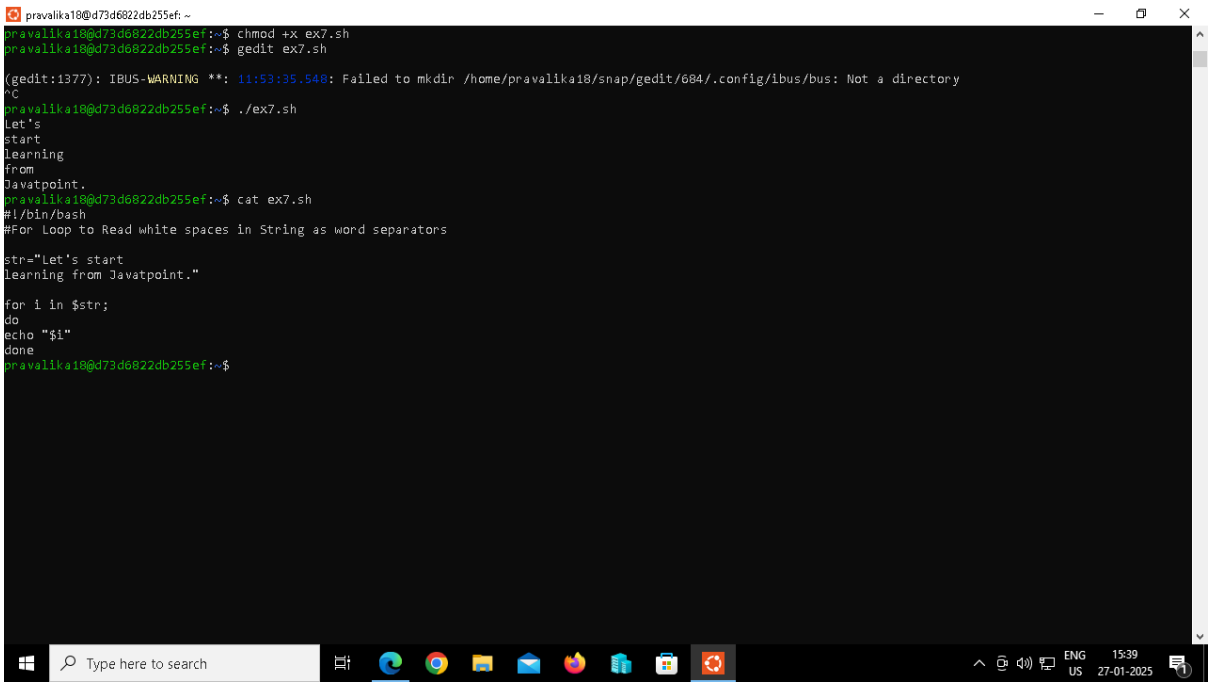
4. For Loop to Read a Range with Decrement.



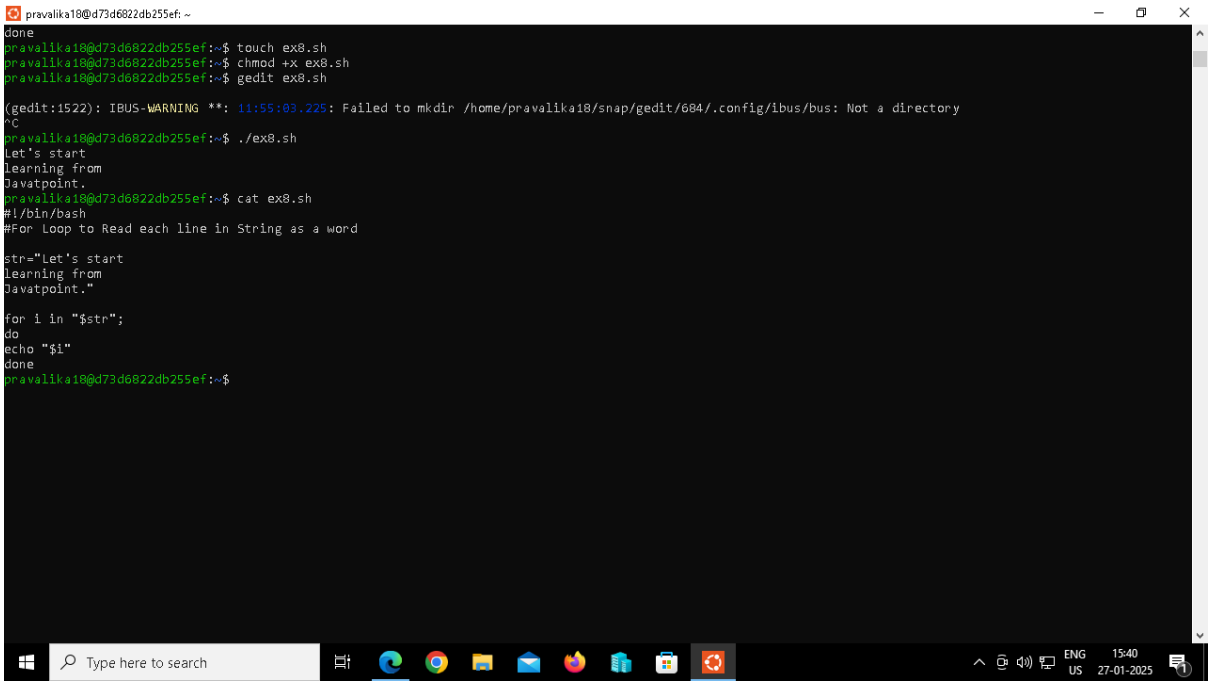
5. For Loop to Read Array Variables.



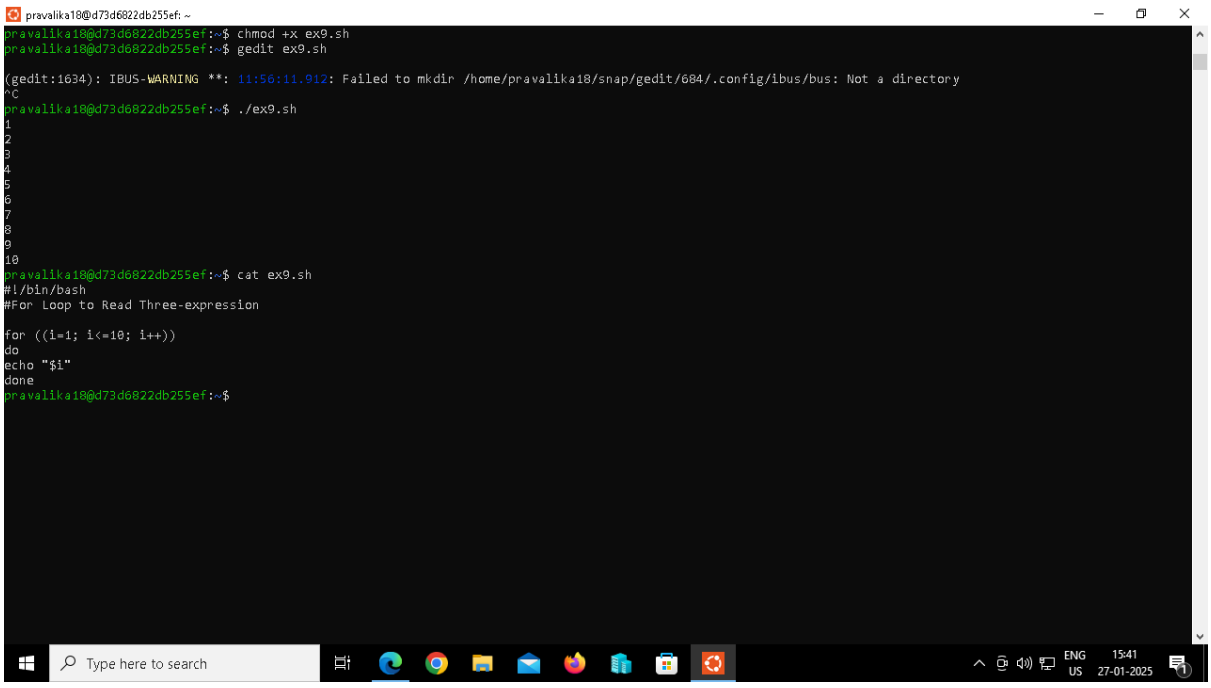
6. For Loop to Read white spaces in String as word separators.



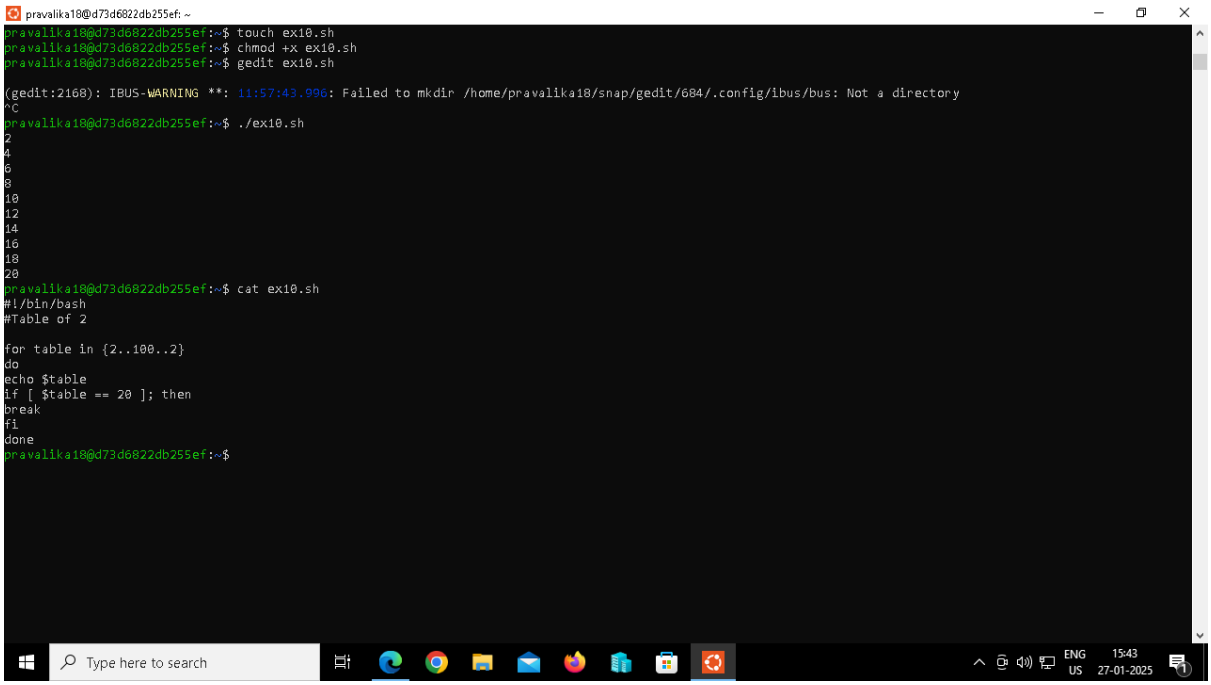
7. For Loop to Read each line in String as a word



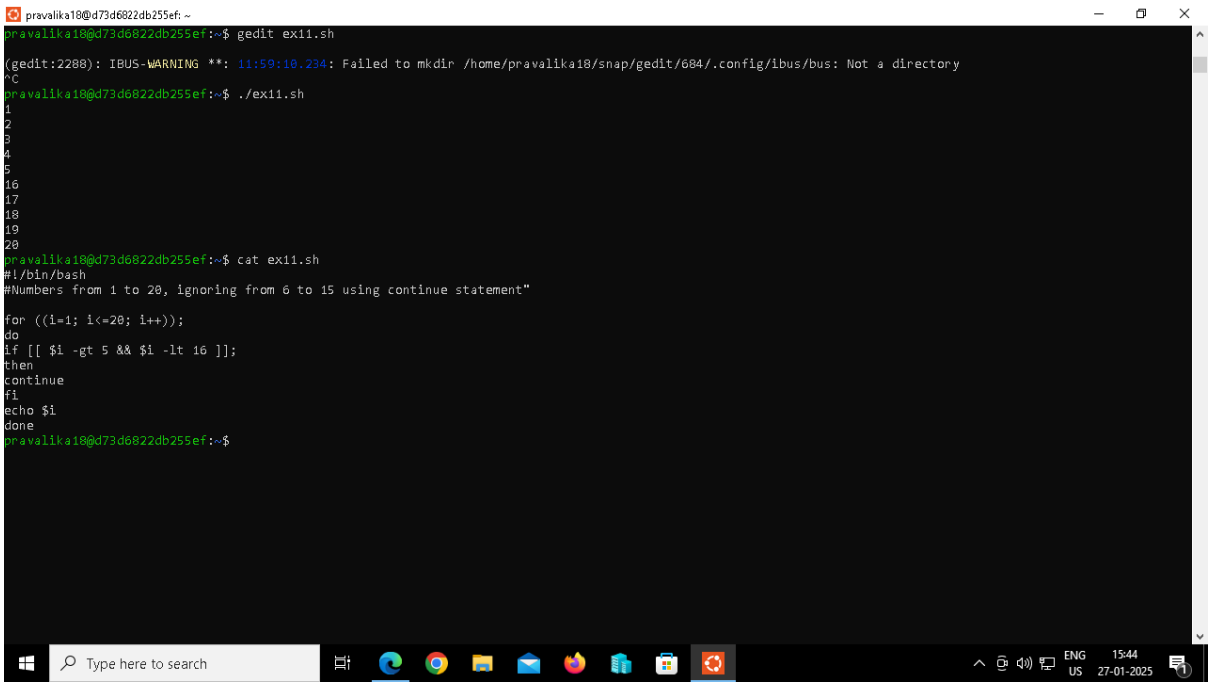
8. For Loop to Read Three-expression



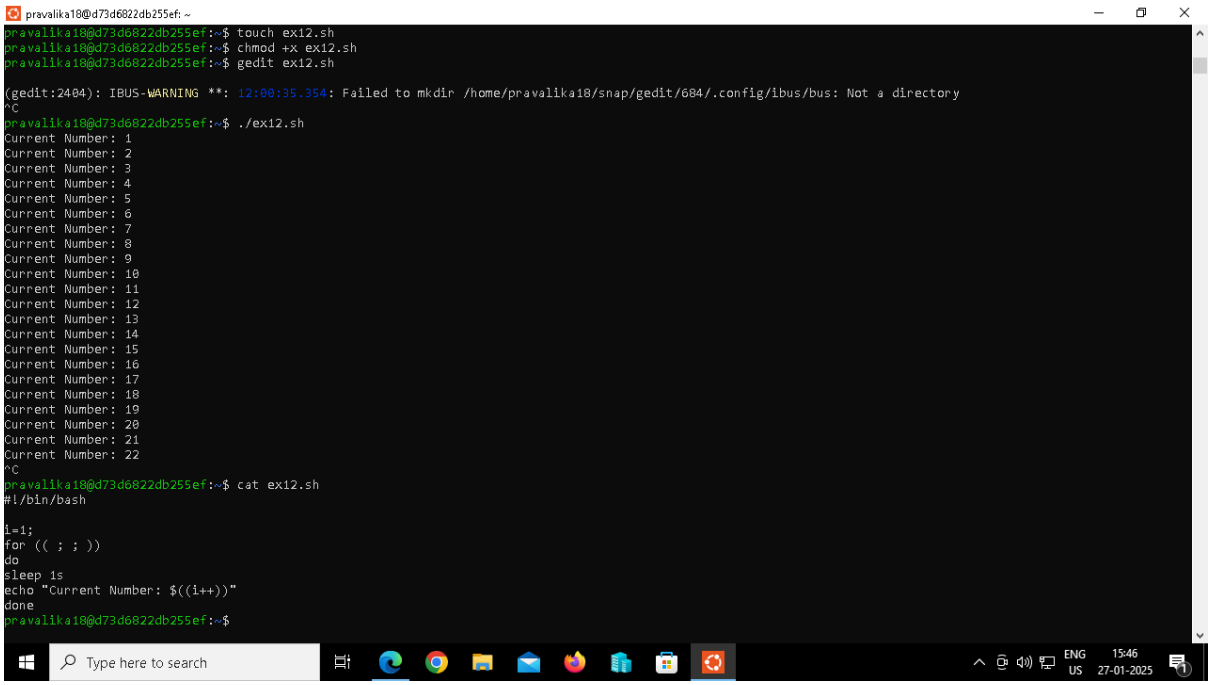
9. For Loop with a Break Statement



10. For Loop with a Continue Statement



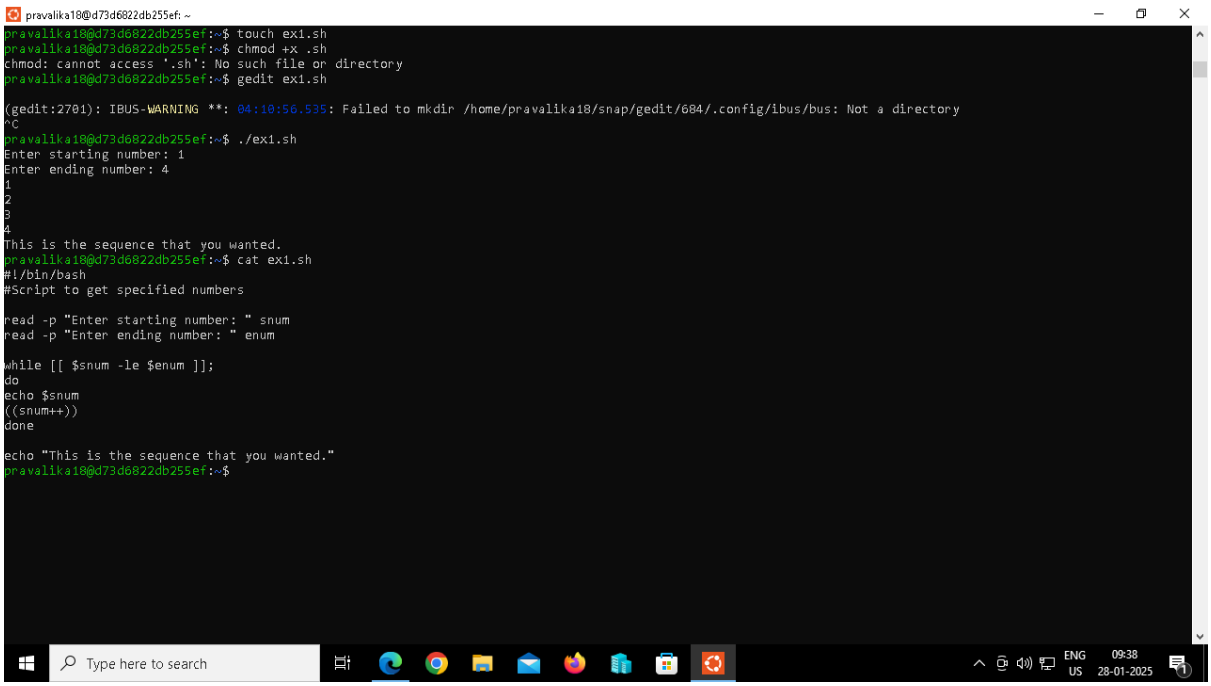
11. Infinite Bash For Loop



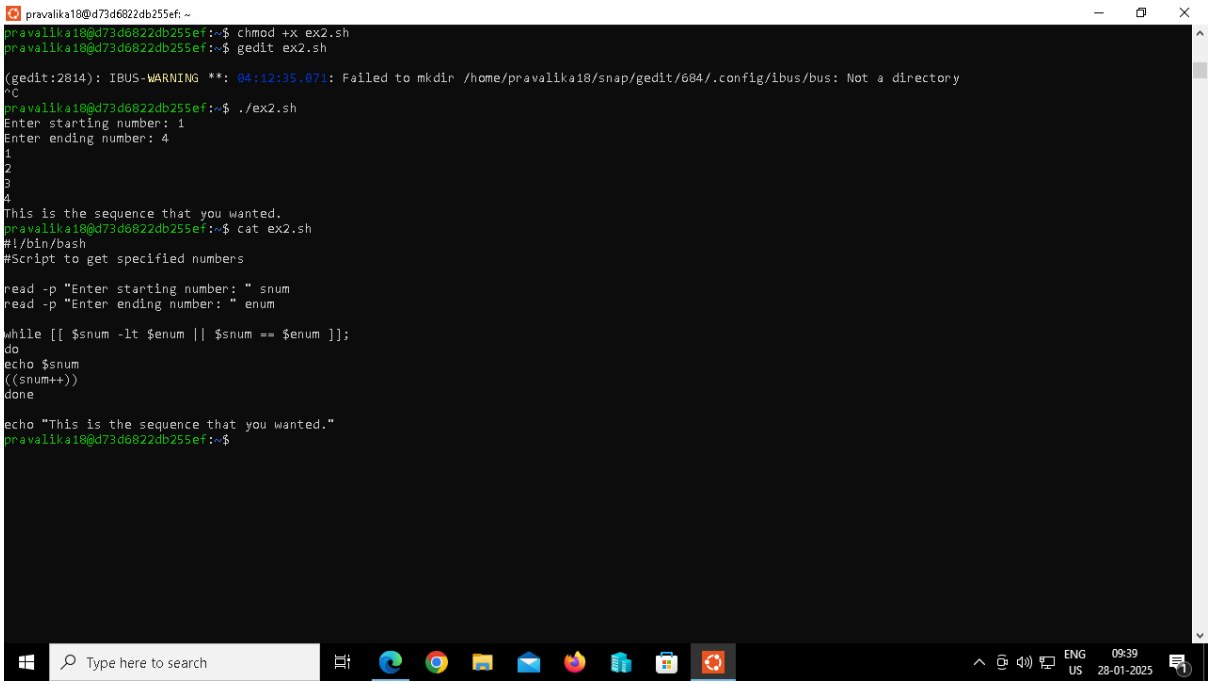
BASH WHILE

1.While Loop with Single Condition:

In this example, the while loop is used with a single condition in expression.It is the basic example of while loop which will print series of numbers as per user input:

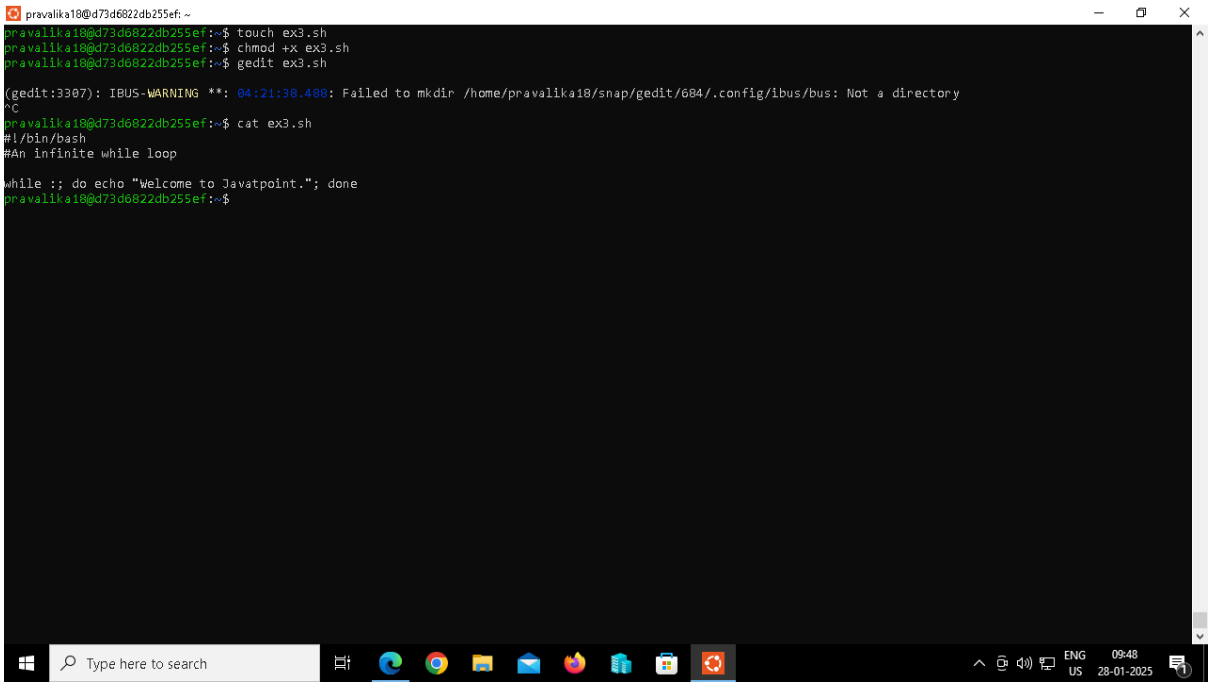


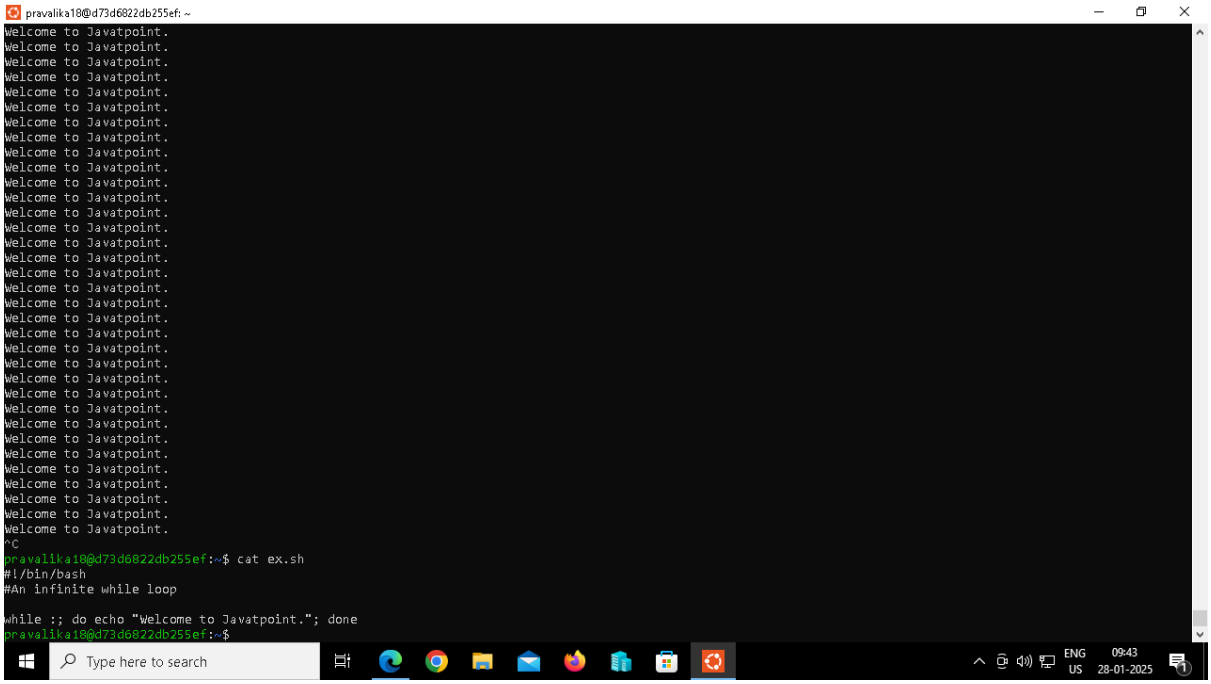
2. While Loop with Multiple Conditions



3. Infinite While Loop

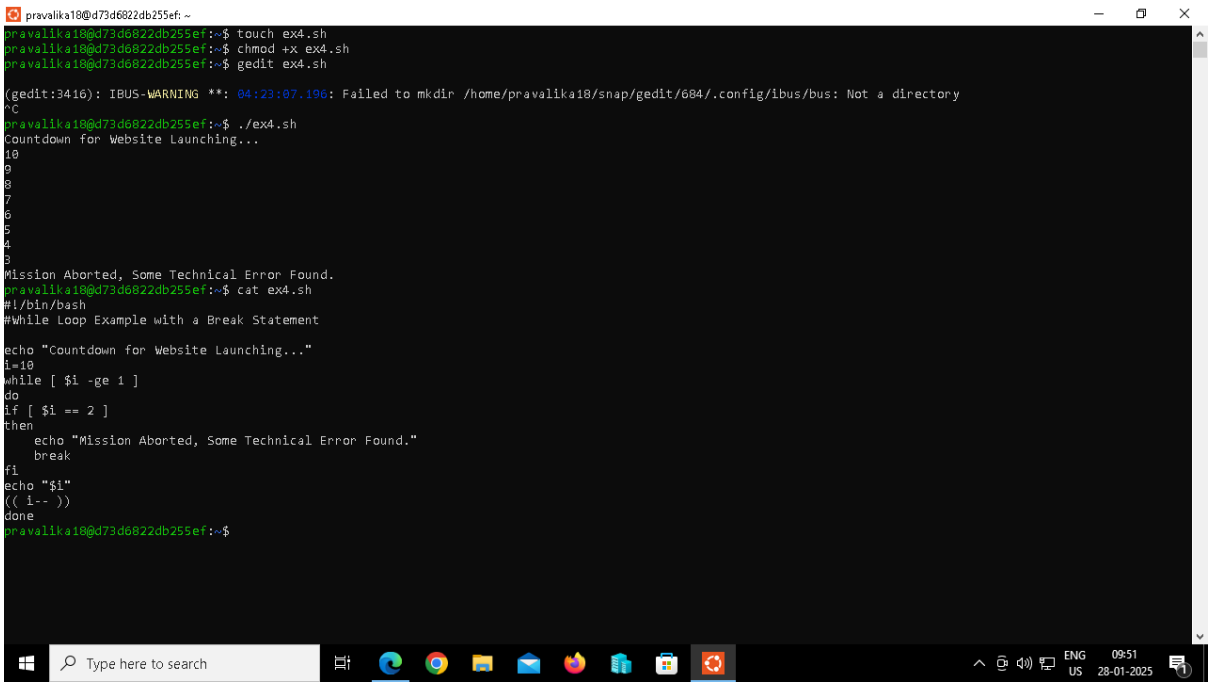
An infinite loop is a loop that has no ending or termination. If the condition always evaluates to true, it creates an infinite loop. The loop will execute continuously until it is forcefully stopped using CTRL+C :





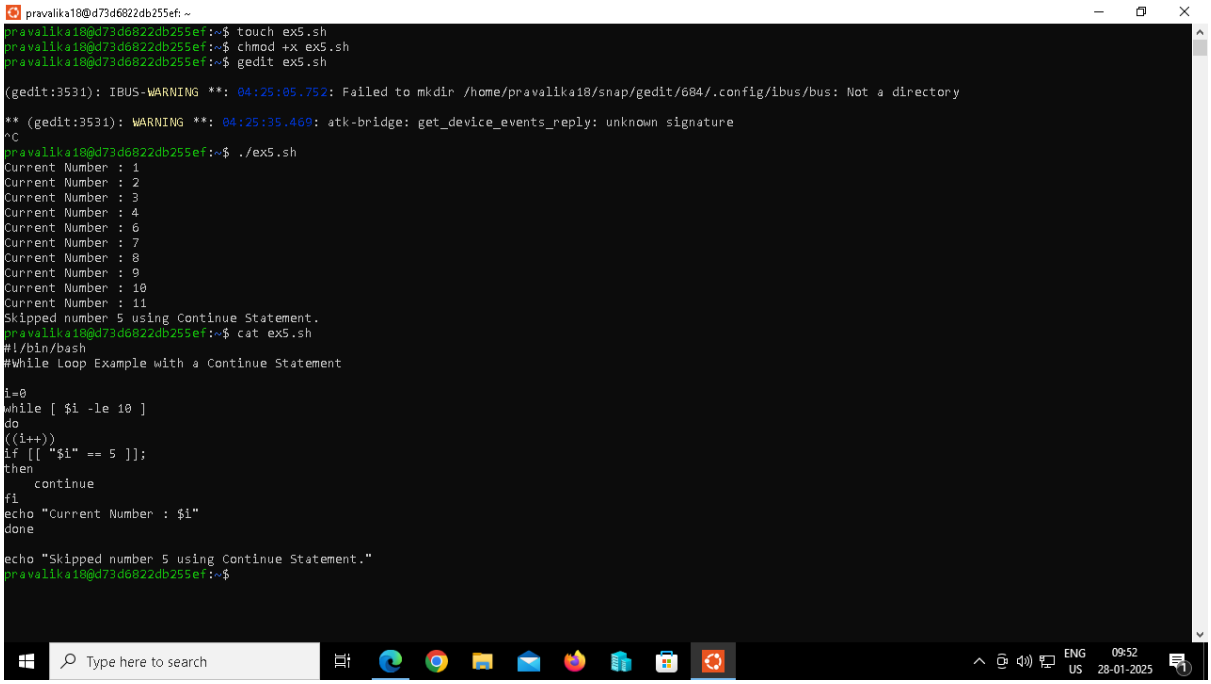
4. While Loop with a Break Statement

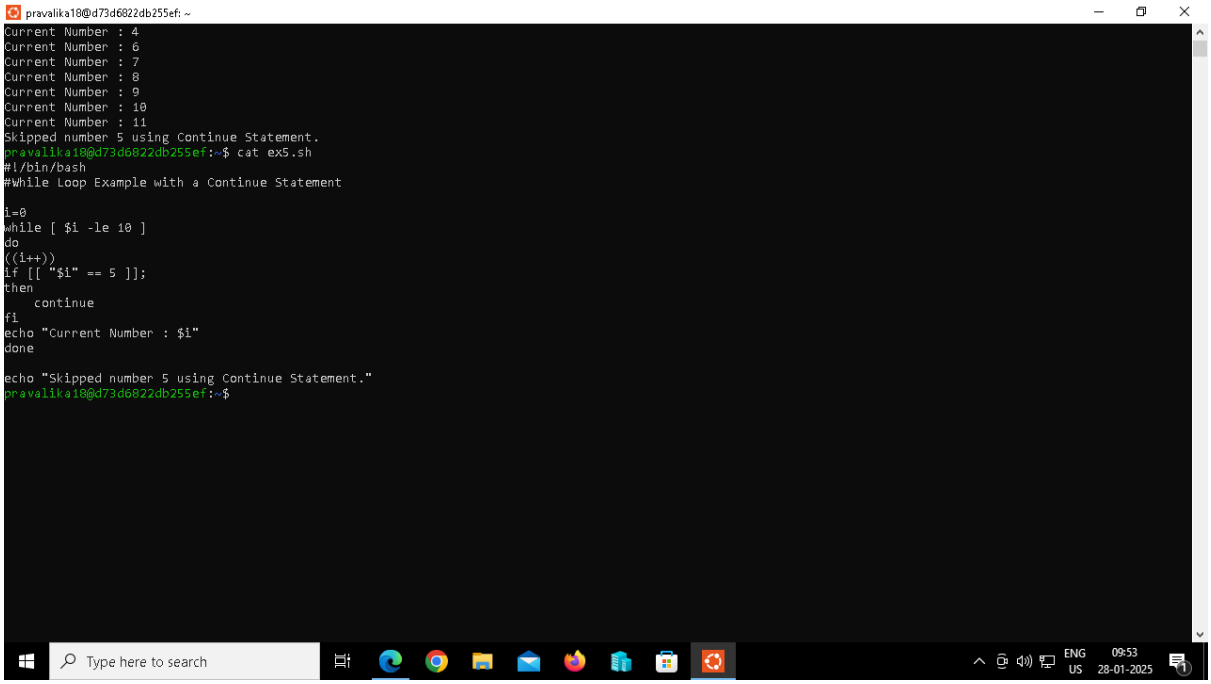
A break statement can be used to stop the loop as per the applied condition. For example:



5. While Loop with a Continue Statement

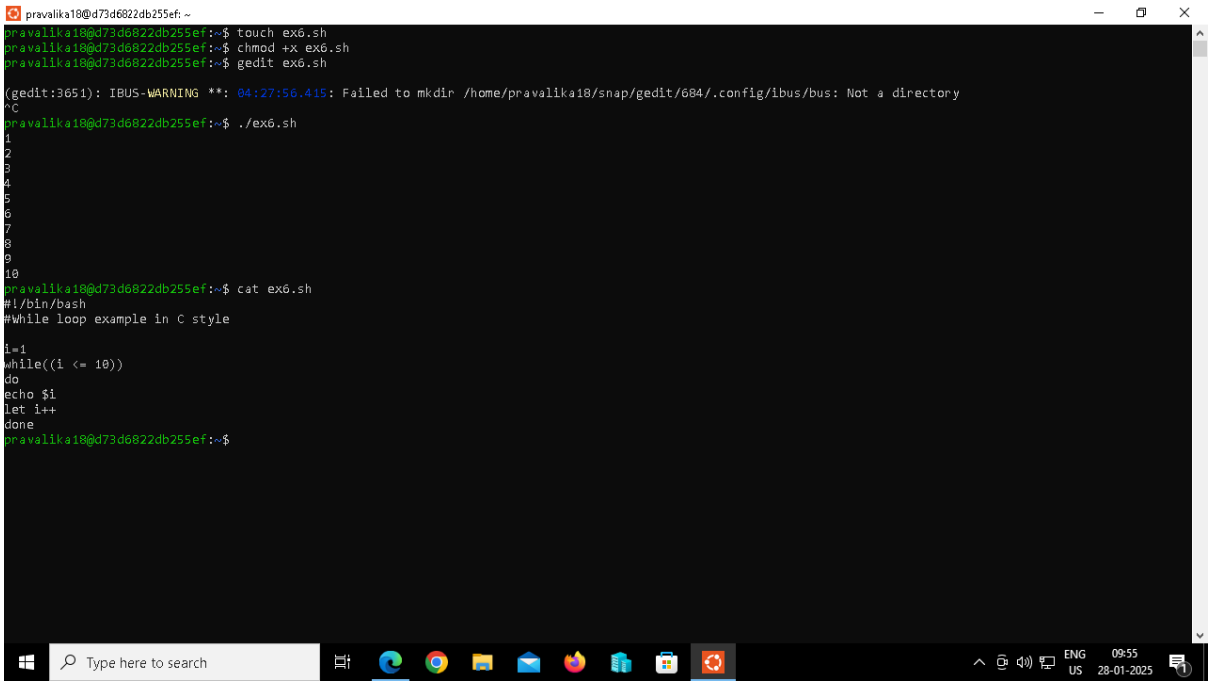
A continue statement can be used to skip the iteration for a specific condition inside the while loop.





6. While Loop with C-Style

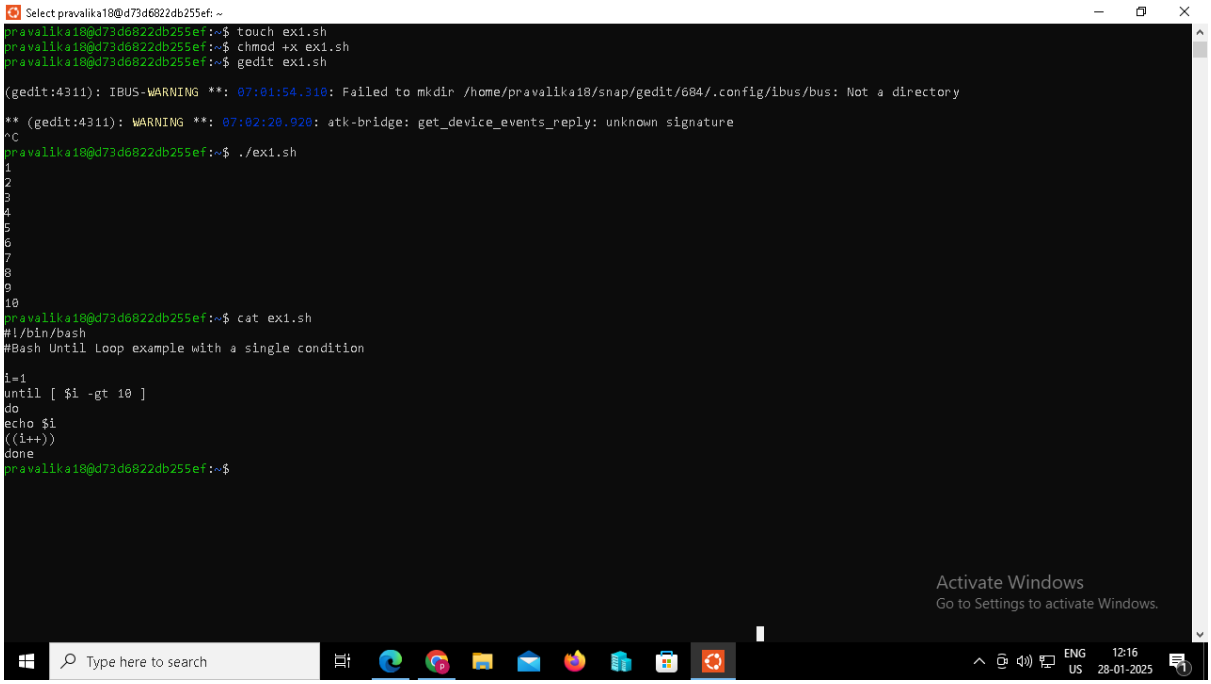
We can also write while loop in bash script as similar as a while loop in C programming language.



BASH UNTIL

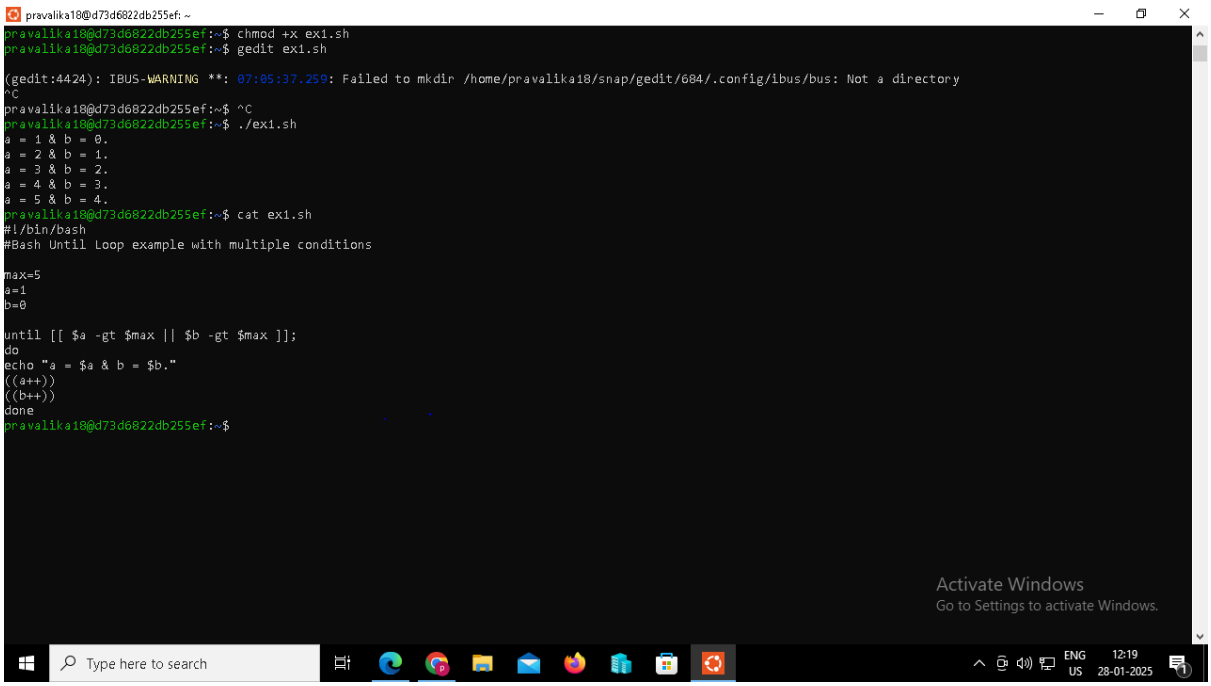
Until Loop with Single Condition

In this example, the until loop contains a single condition in expression. It is the basic example of until loop which will print series of numbers from 1 to 10:



2. Until Loop with Multiple Conditions

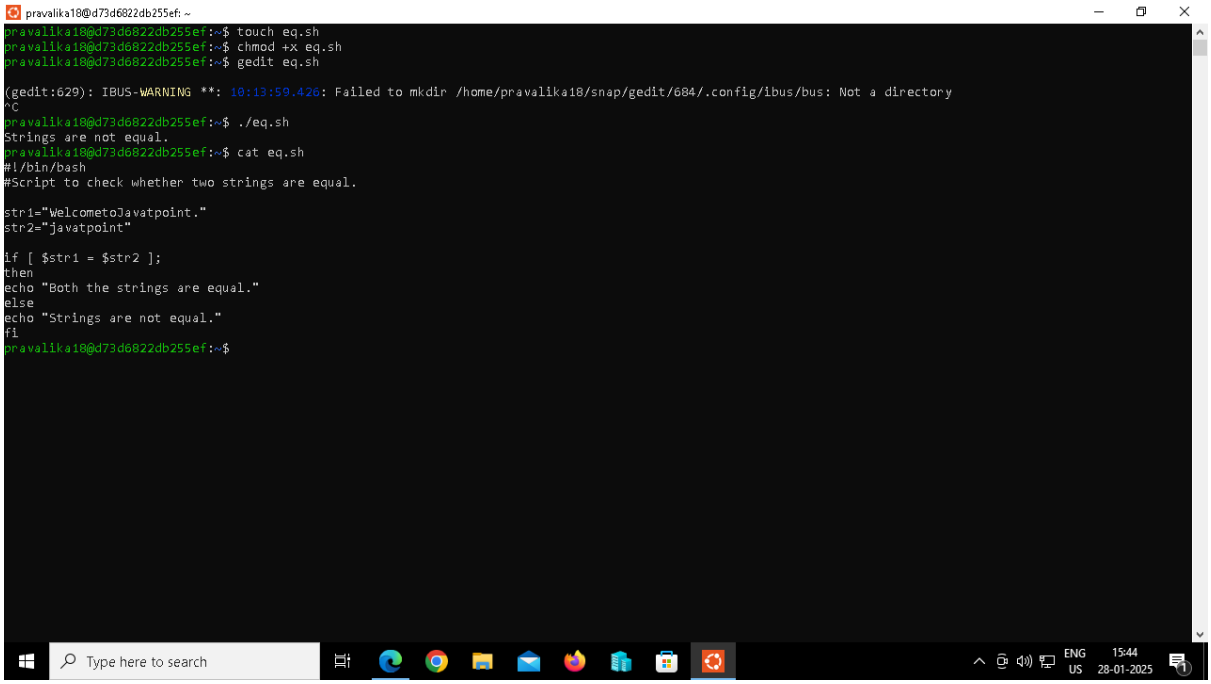
Following is an example with multiple conditions in an expression:



Bash String

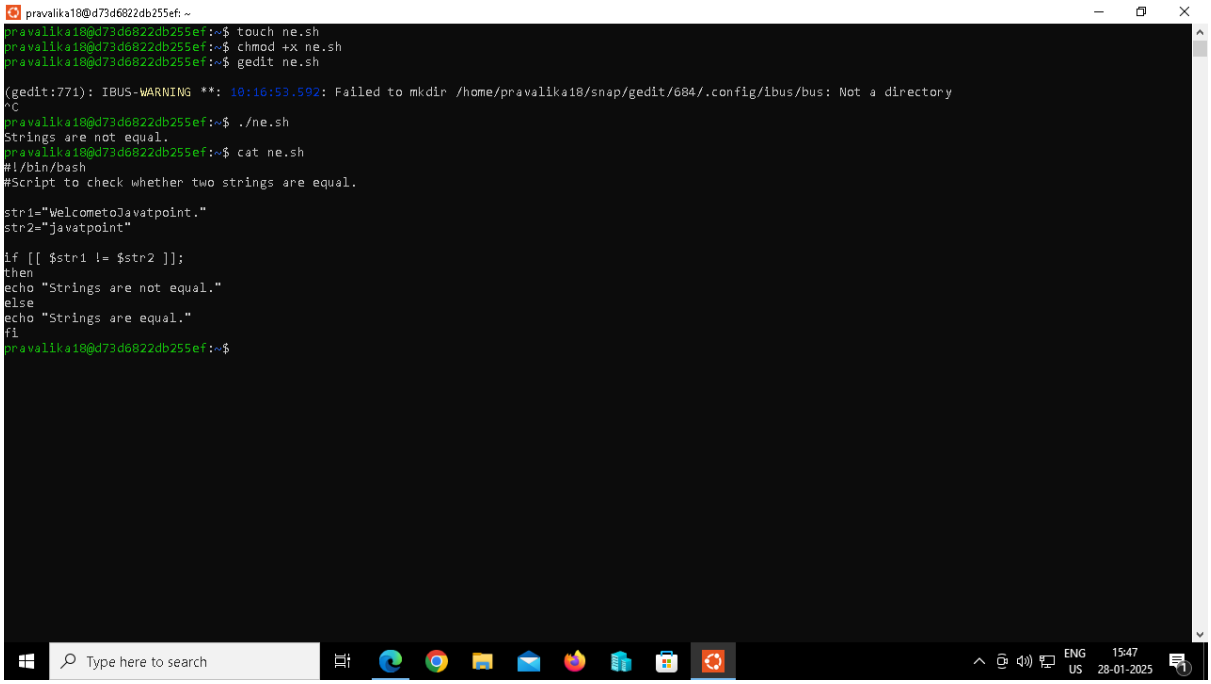
Equal Operator

An equal operator (=) is used to check whether two strings are equal.



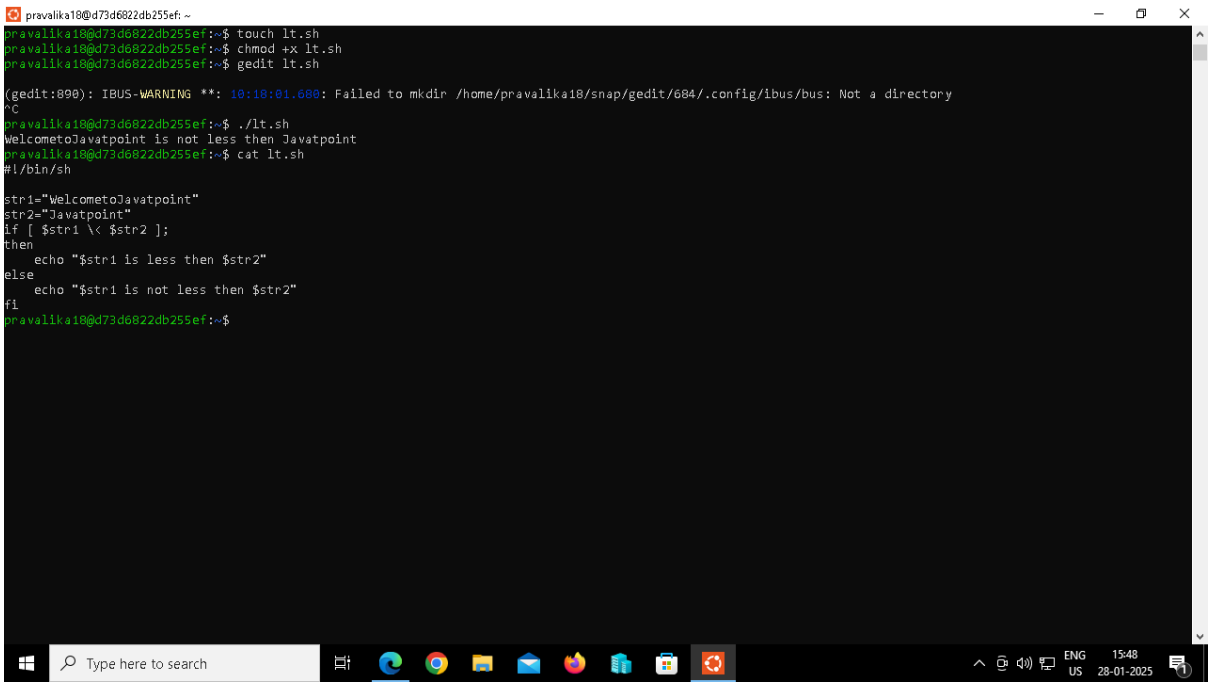
Not Equal Operator

Not equal operator (!=) is used to define that strings are not equal.



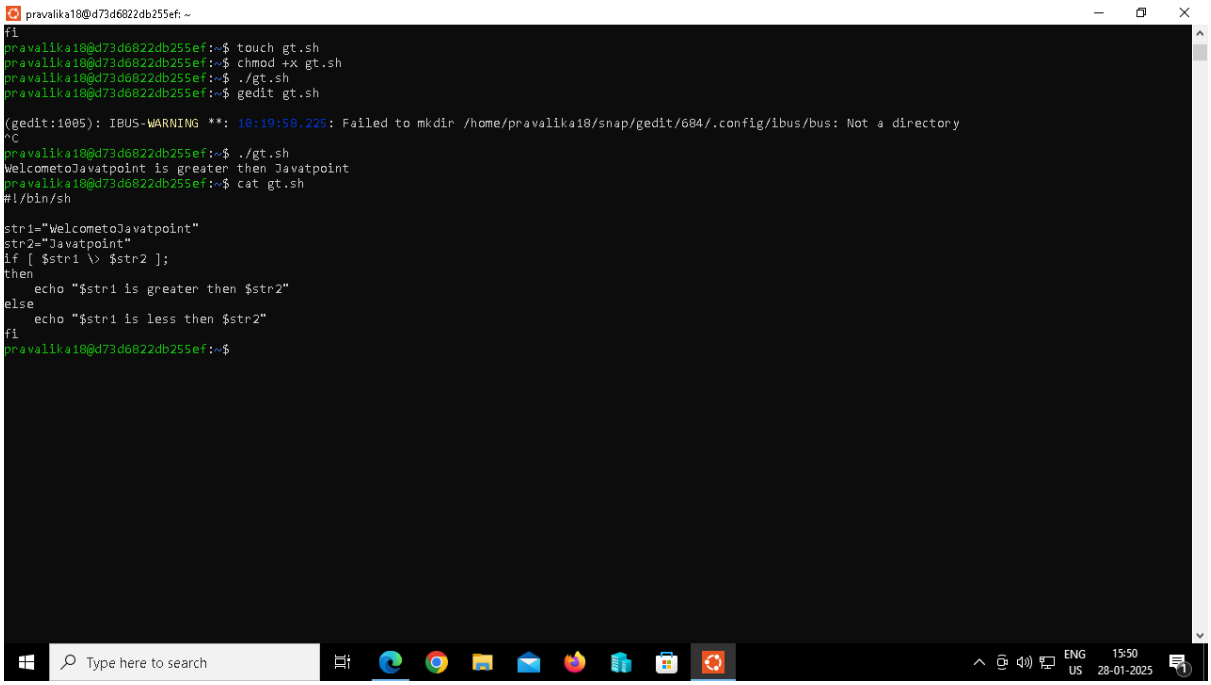
Less than Operator

The 'less than operator (\<)' is a conditional operator which is used to check if string1 is less than string2.



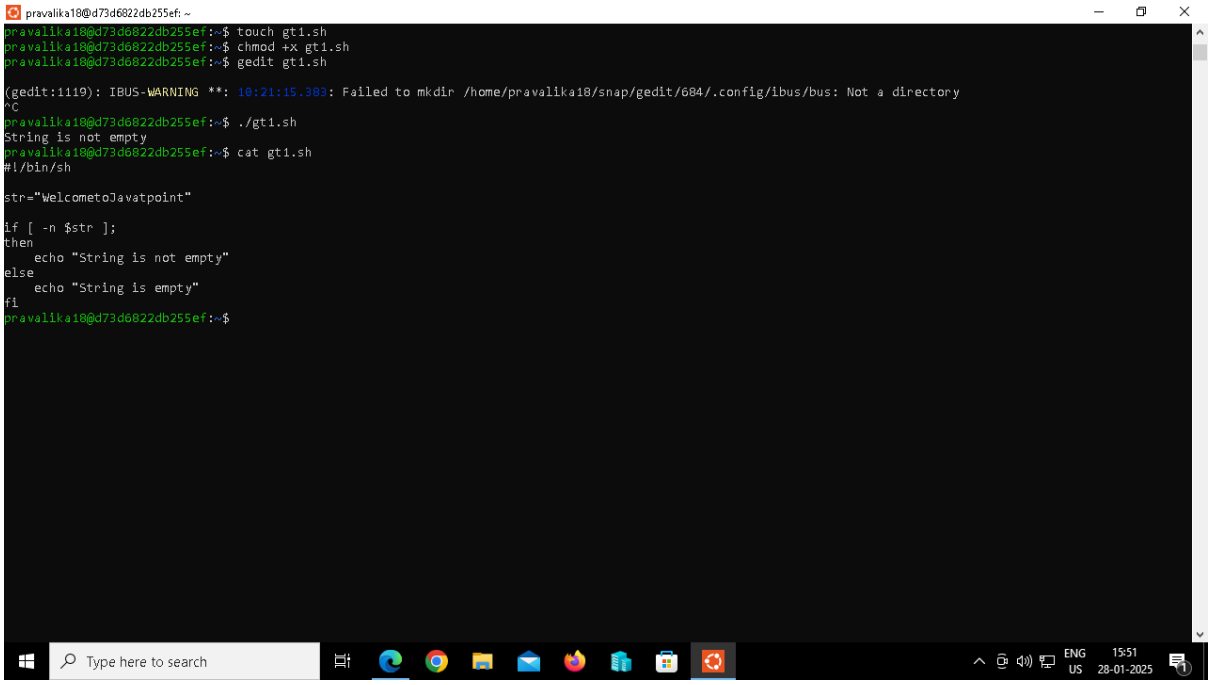
Greater than Operator

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



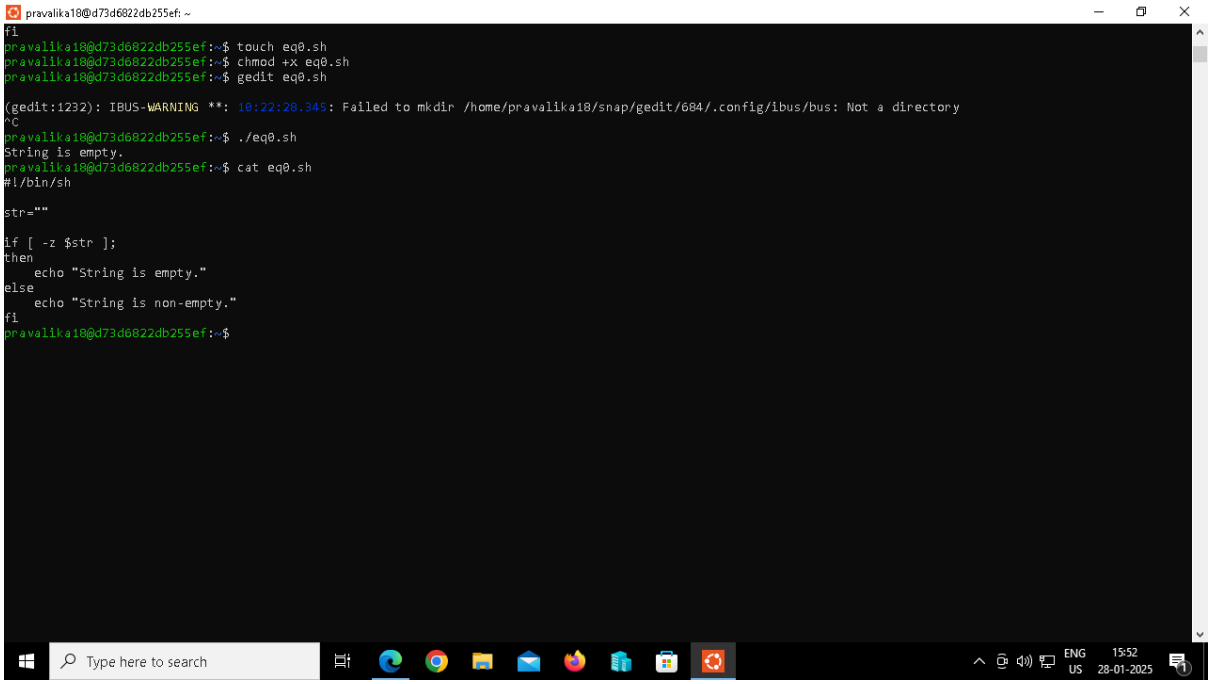
To check if the string length is greater than Zero:

This operator is used to check if the string is zero or greater than zero.



To check if the string length is equal to Zero

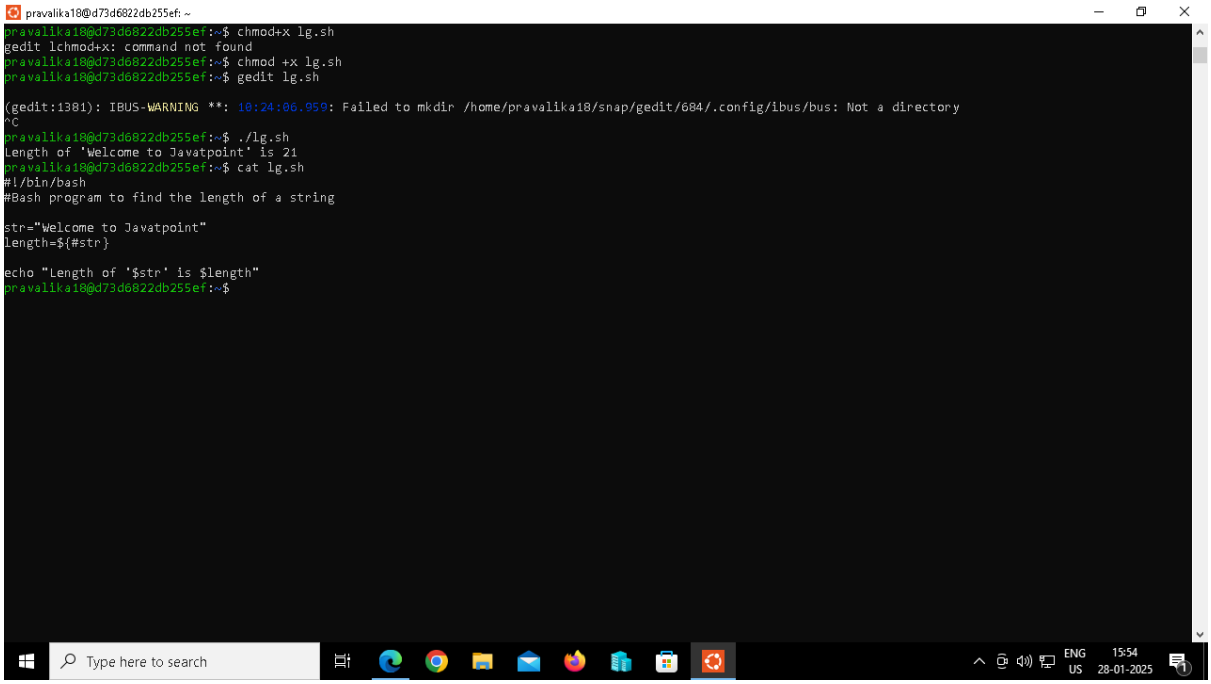
This operator is used to check if the string is empty or equal to zero.



BASH FIND STRING

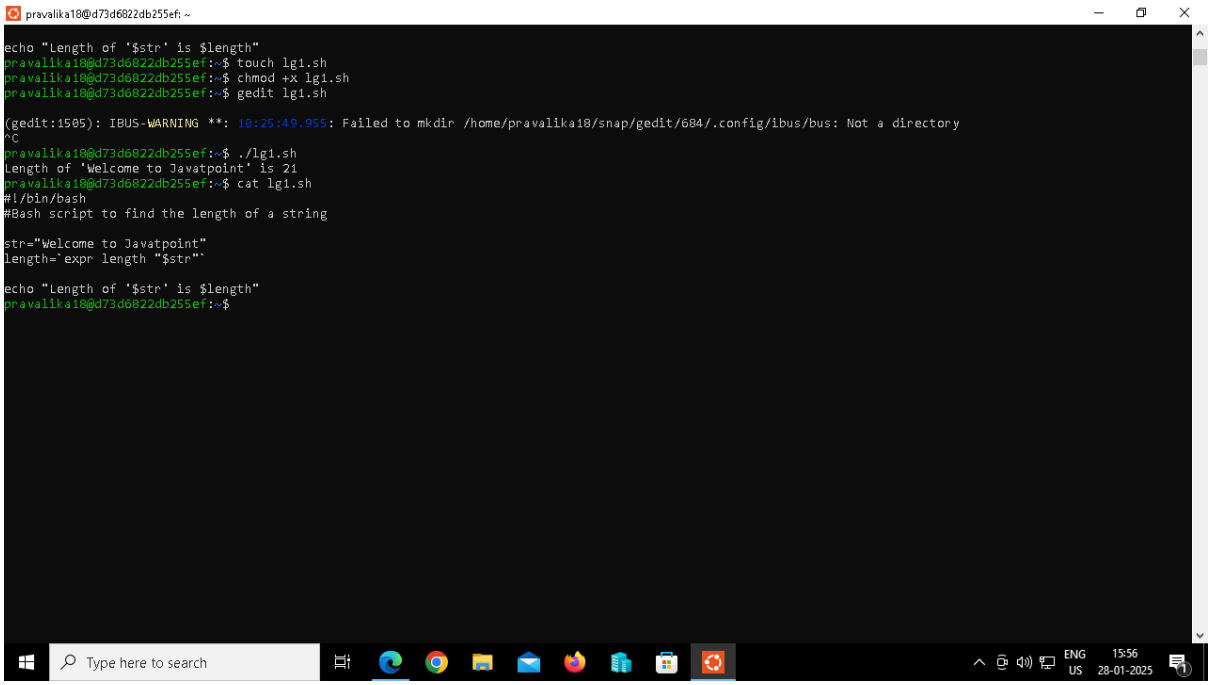
Example 1

The simplest way to calculate the length of a string is to use '#' symbol. In this example, we have used $[#string_variable_name} to find the length of a string.



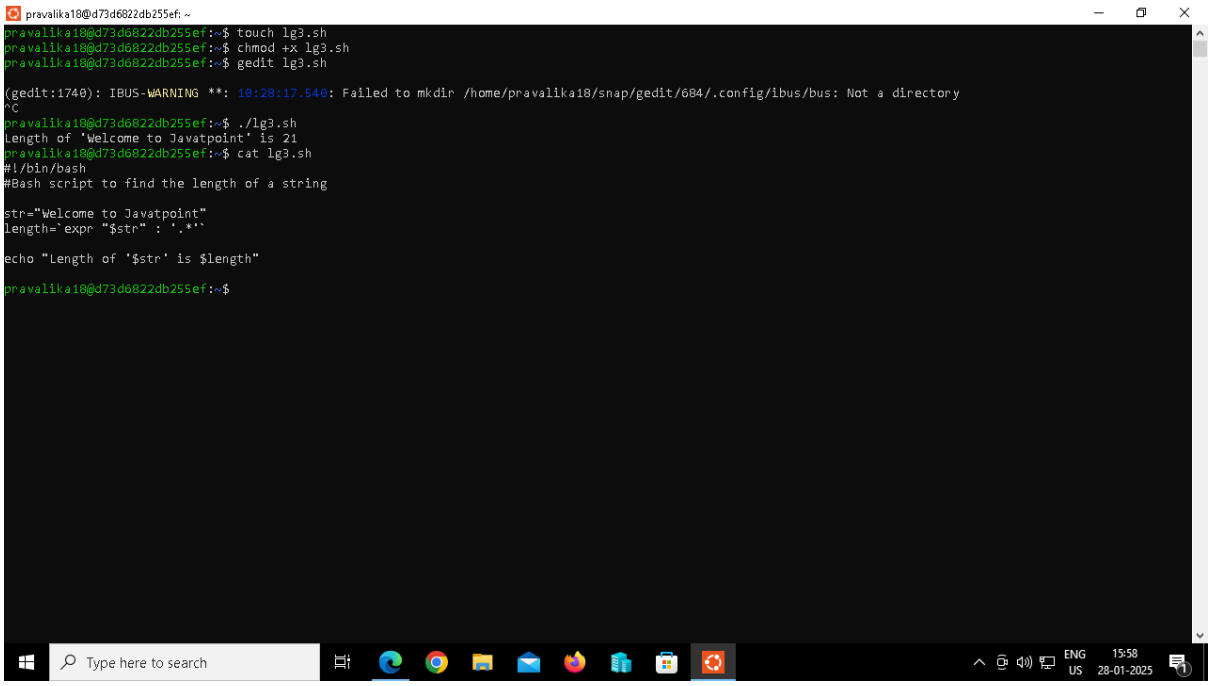
Example 2

In this example, we have used `**expr length "$str"**` to find the length of a string.



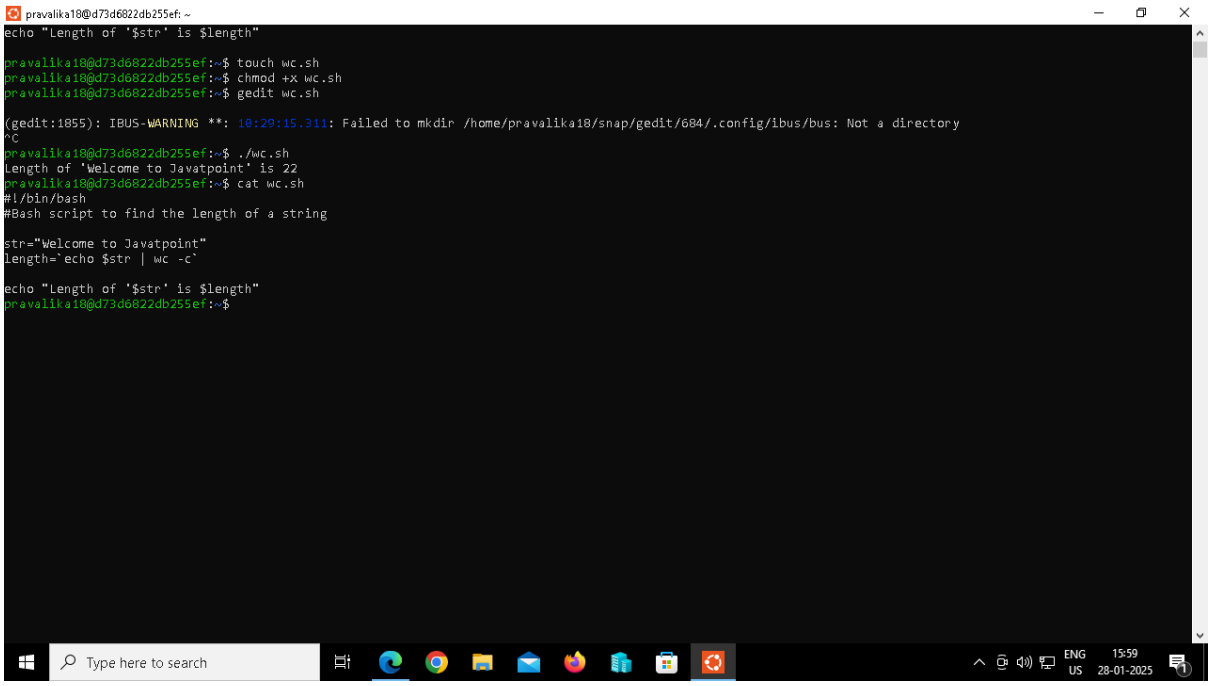
Example 3

In this example, we have used `**expr "$str":** ' .*'`to find the length of a string. Here, str is a string variable.



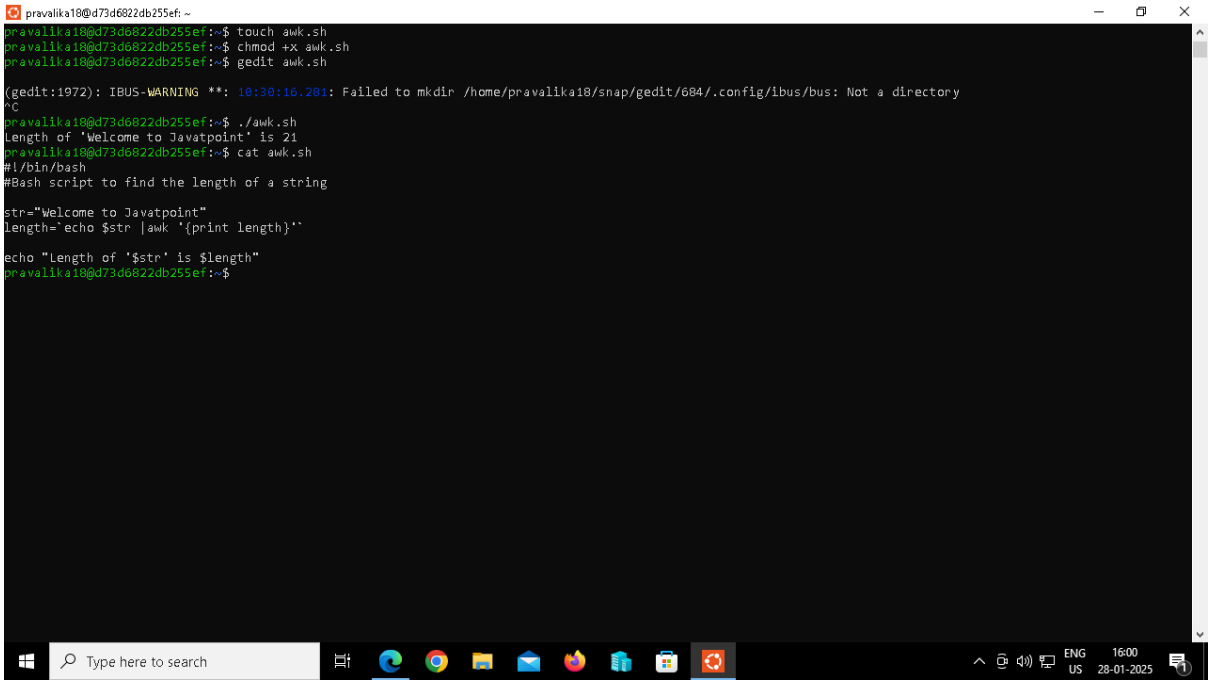
Example 4

In this example, we have used `wc` command to find the length of a string.



Example 5

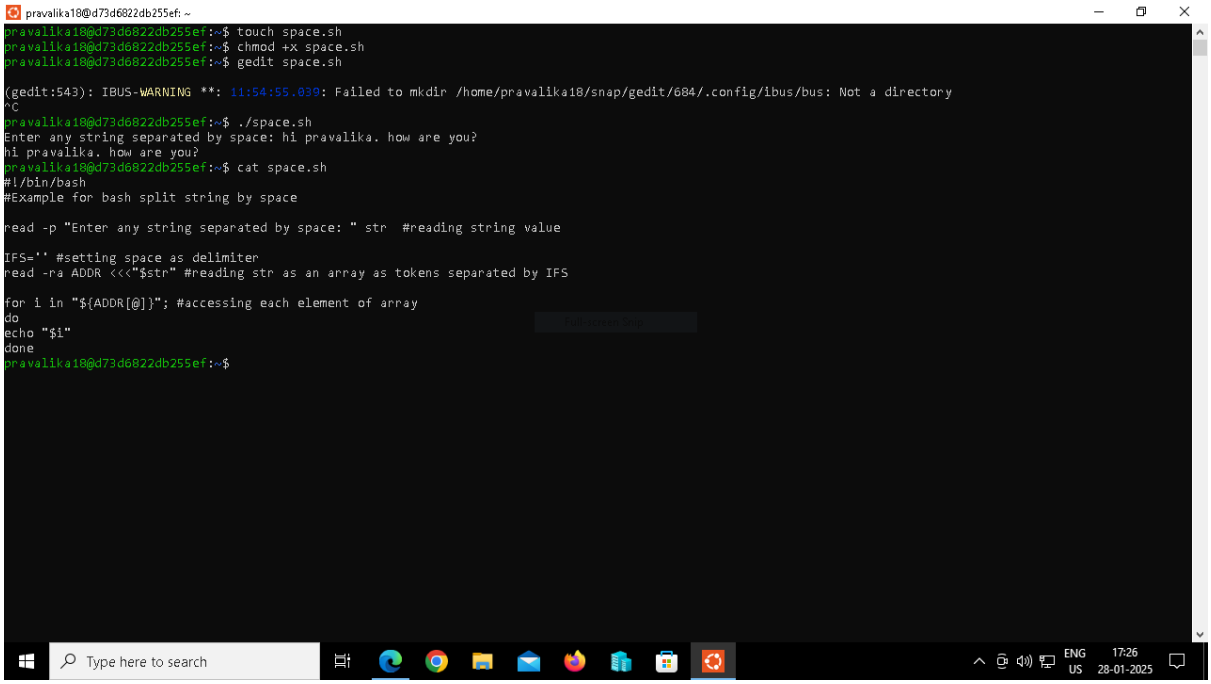
In this example, we have used `**awk**` command to find the length of a string.



Bash Split String (Split using $IFS variable)

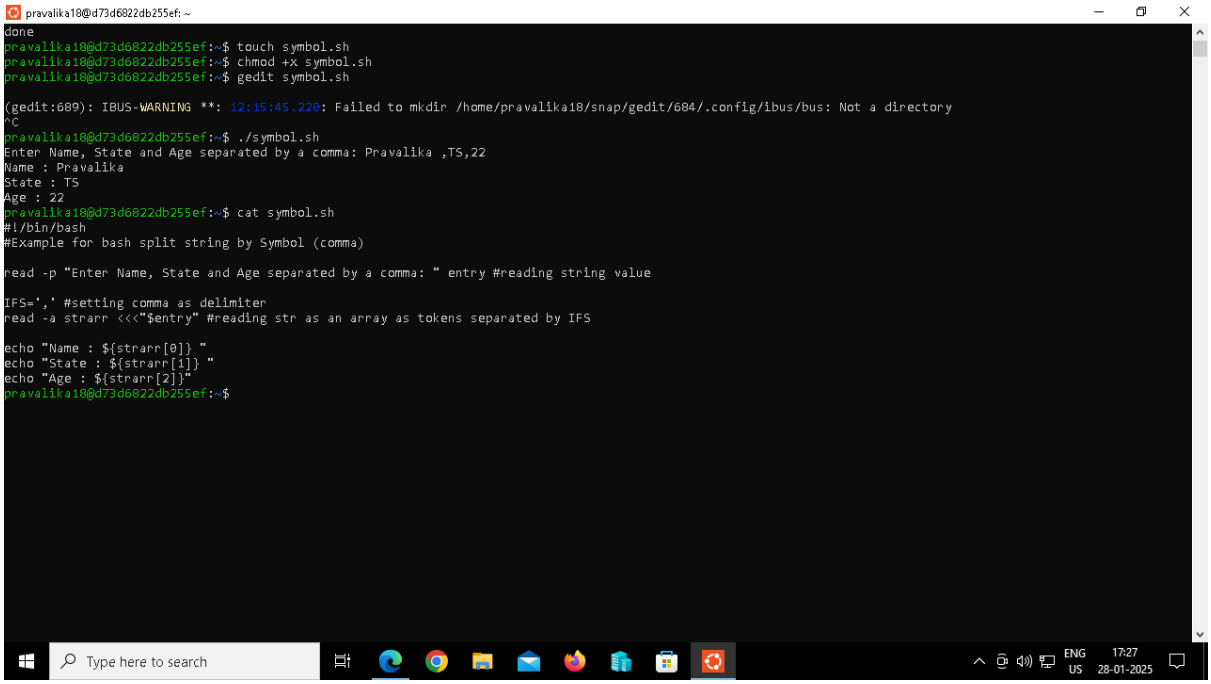
Example 1: Bash Split String by Space

In this example, a string is split using a space character delimiter.



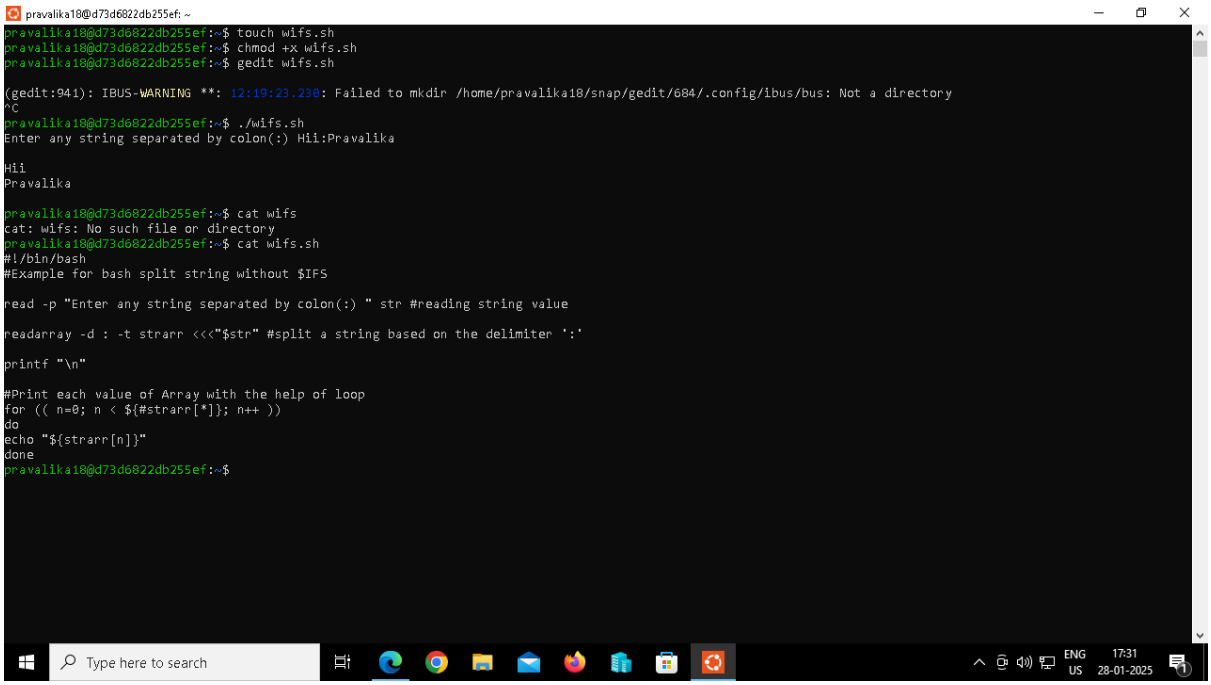
Example 2: Bash Split String by Symbol

In this example, a string is split using a comma (,) symbol character as a delimiter.



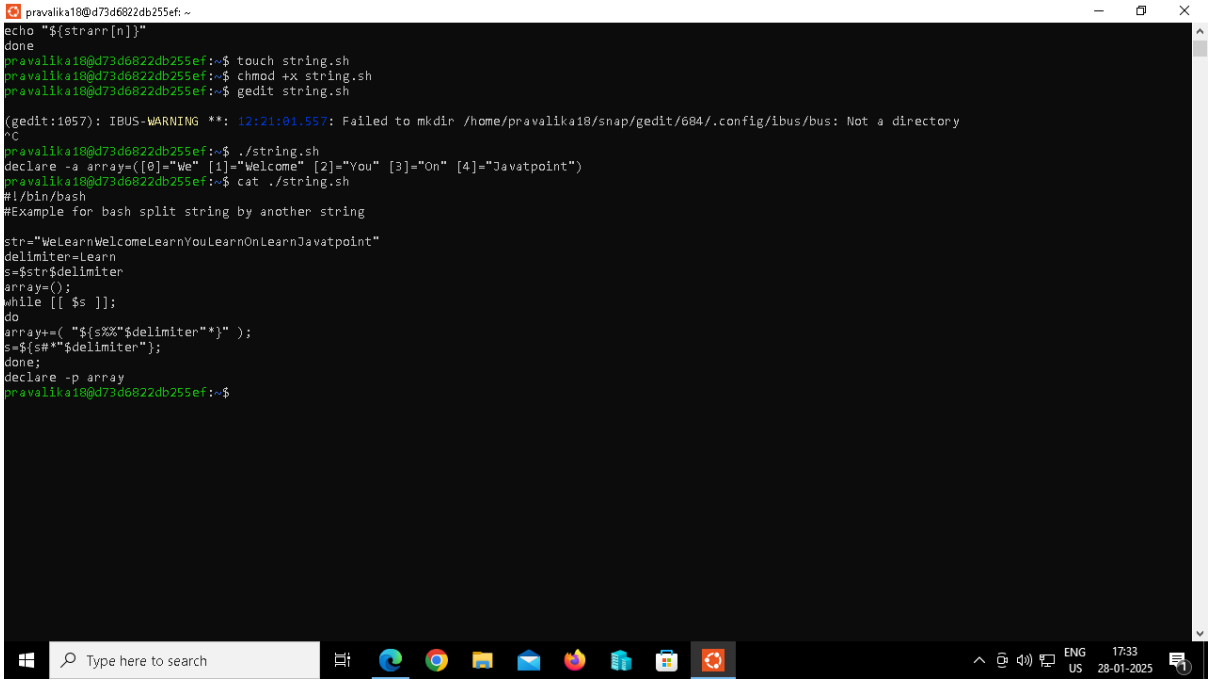
Split using $IFS variable

Example 1: Bash Split String by Symbol



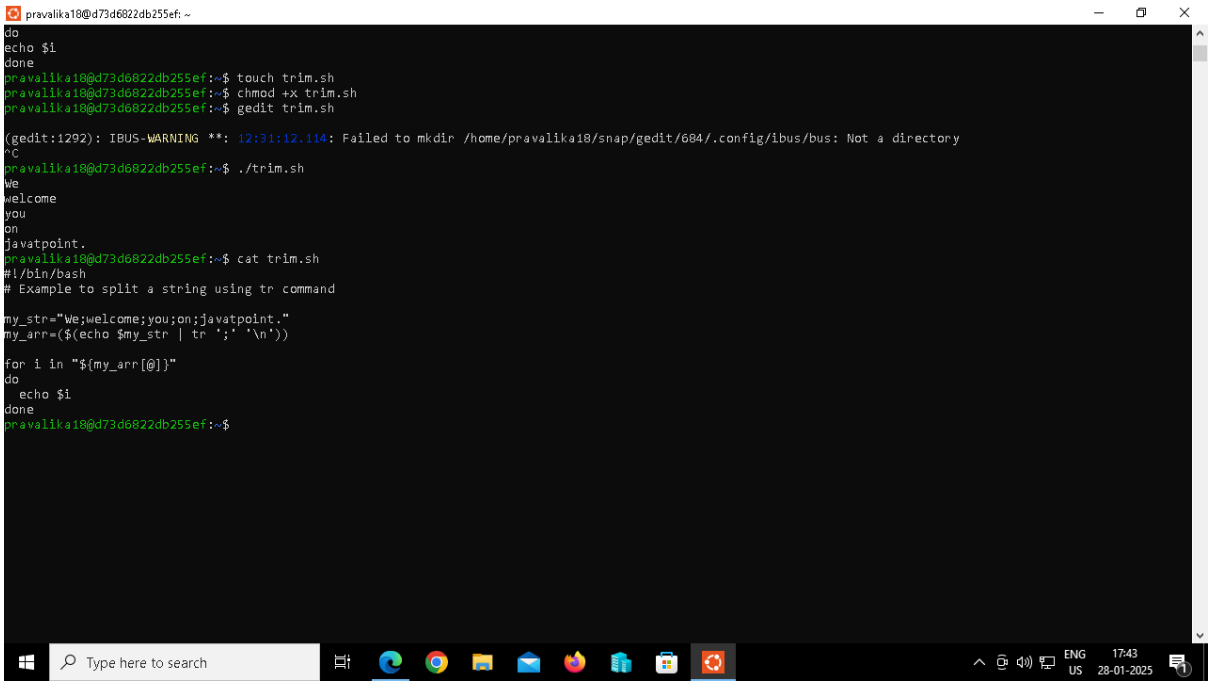
Example 2: Bash Split String by another string

In this example, we have used idiomatic expressions where parameter expansion hascompleted.



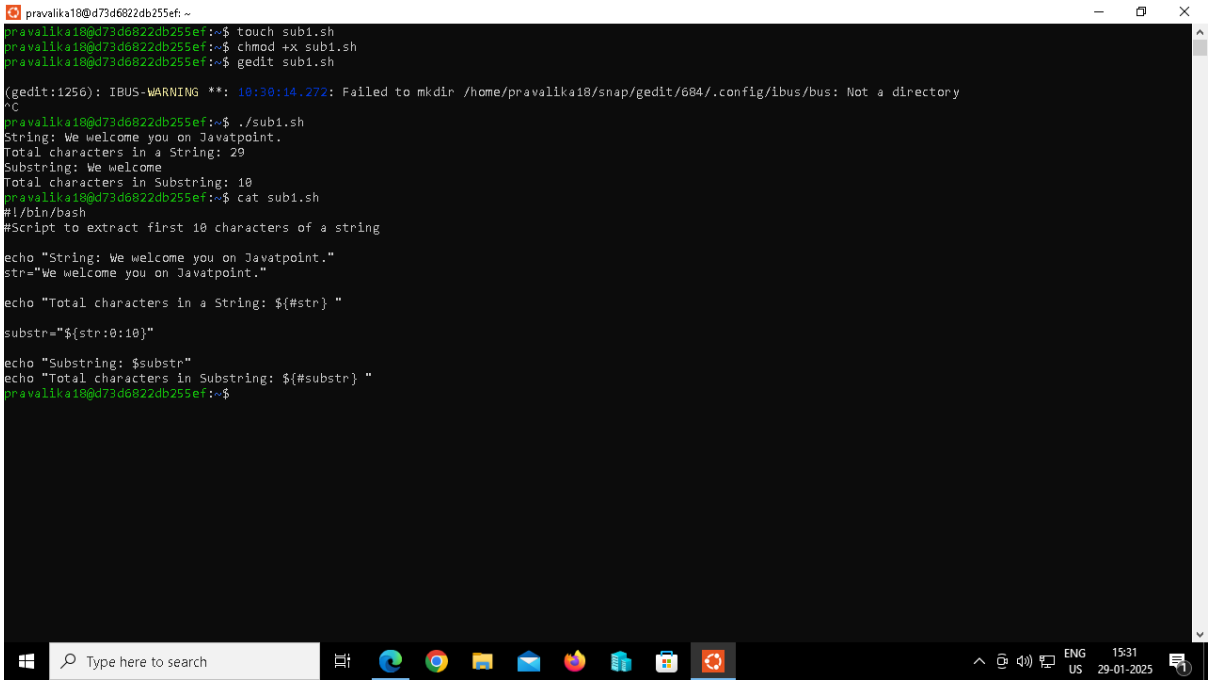
Example 3: Bash Split String using Trim Command

In this example, we have used trim (tr) command to split a string. Instead of using the read command, the trim command is used to split a string on the delimiter.

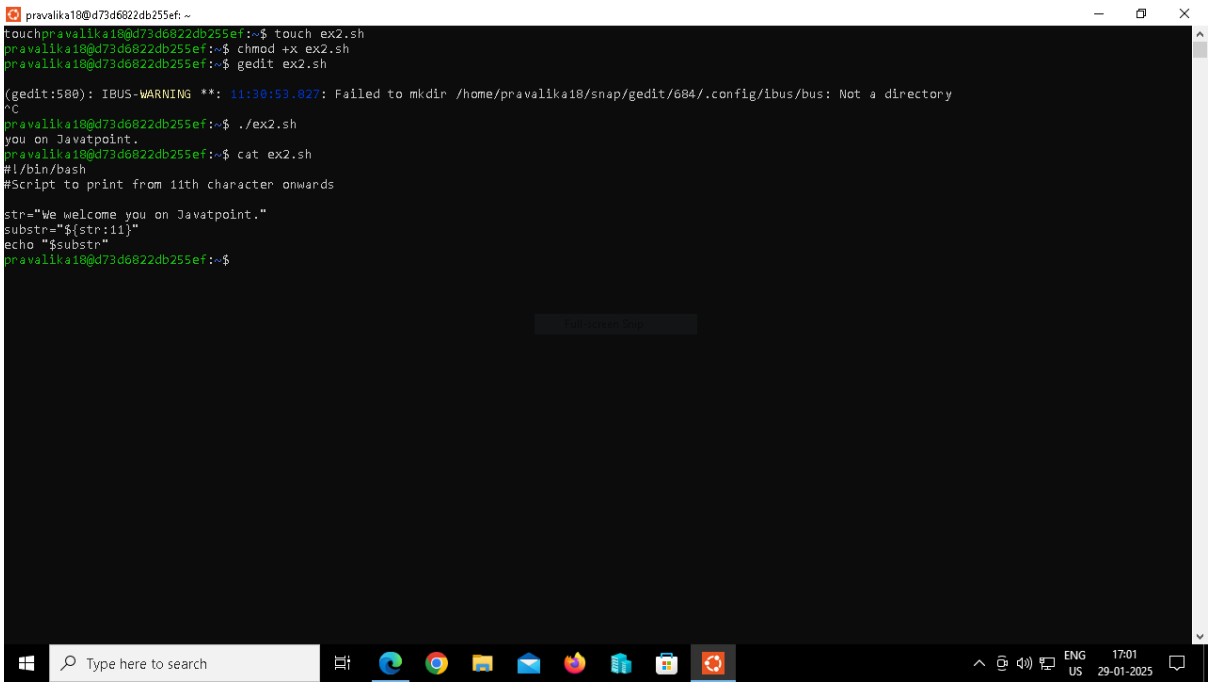


Bash Substring

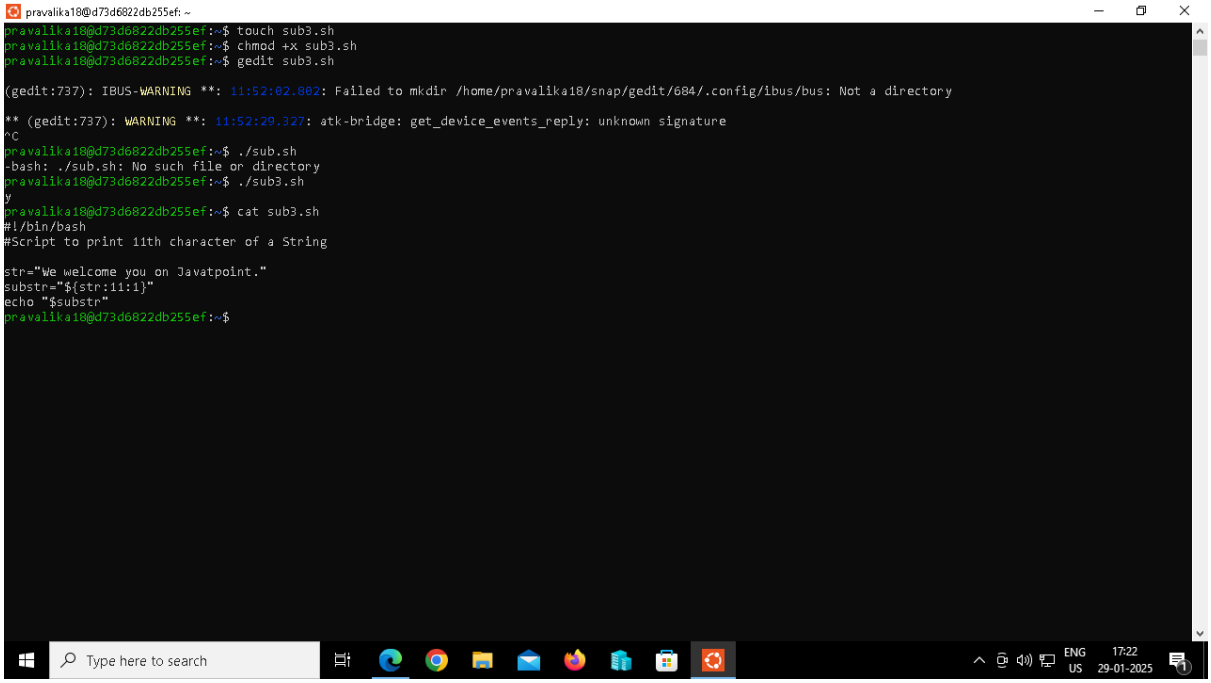
Example 1: To Extract till Specific Characters from Starting



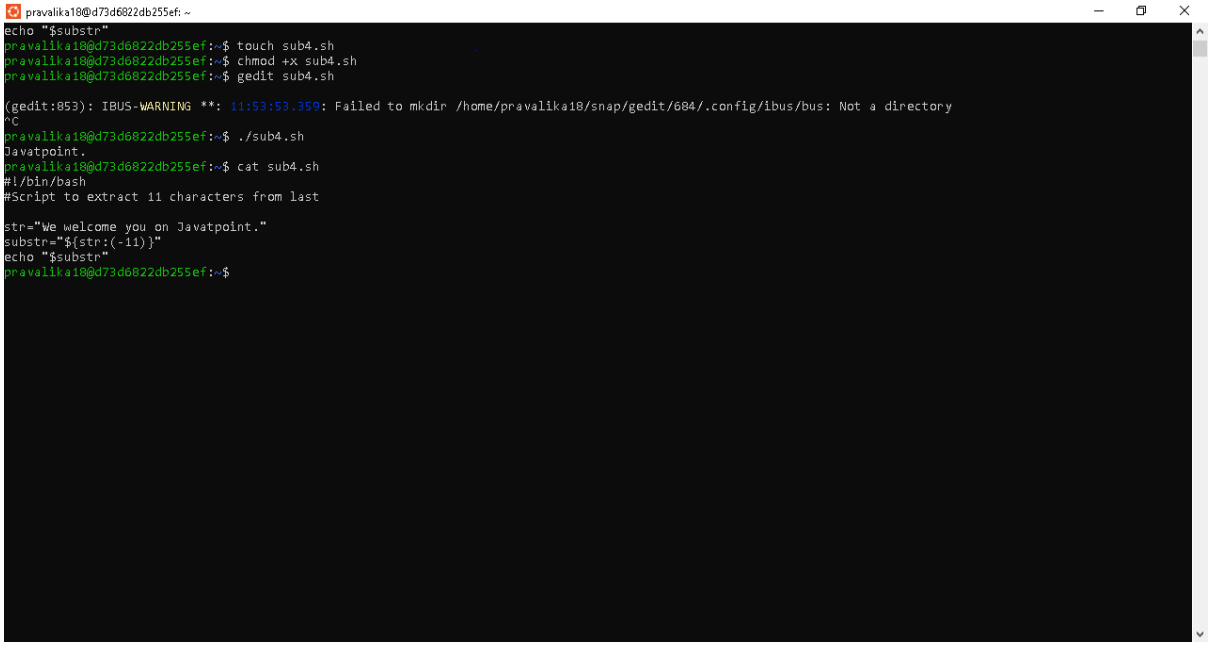
Example 2: To Extract from Specific Character onwards



Example 3: To Extract a Single Character



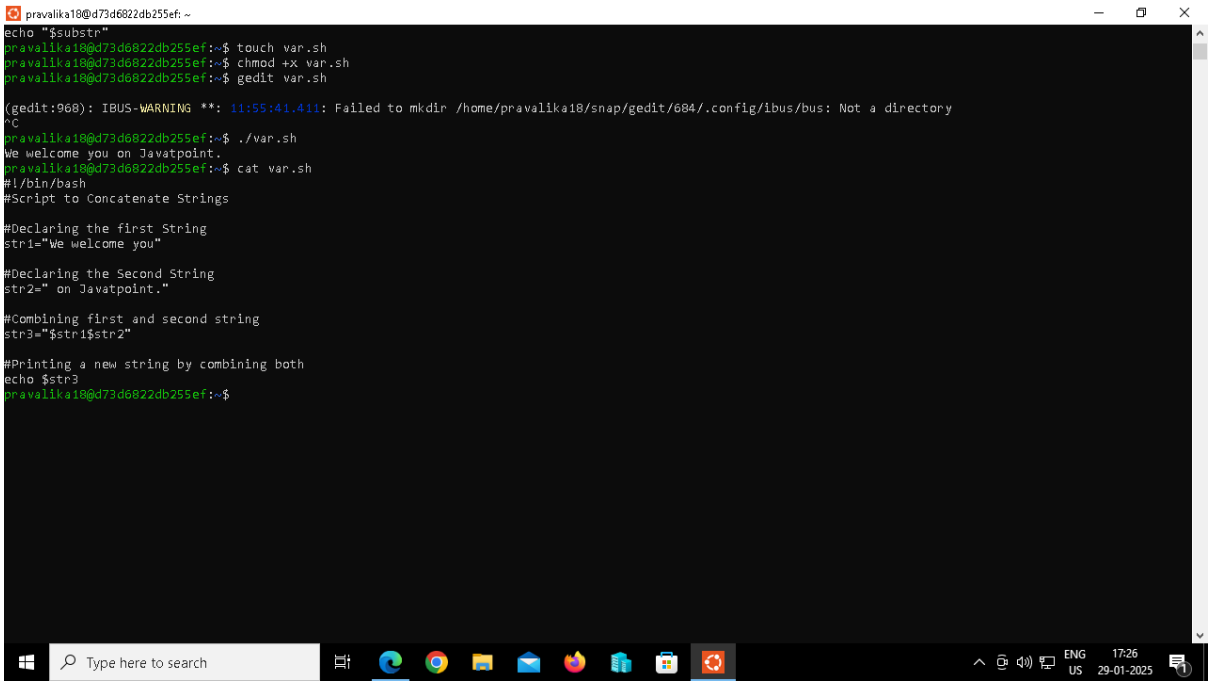
Example 4: To Extract the specific characters from last



Bash Concatenate String

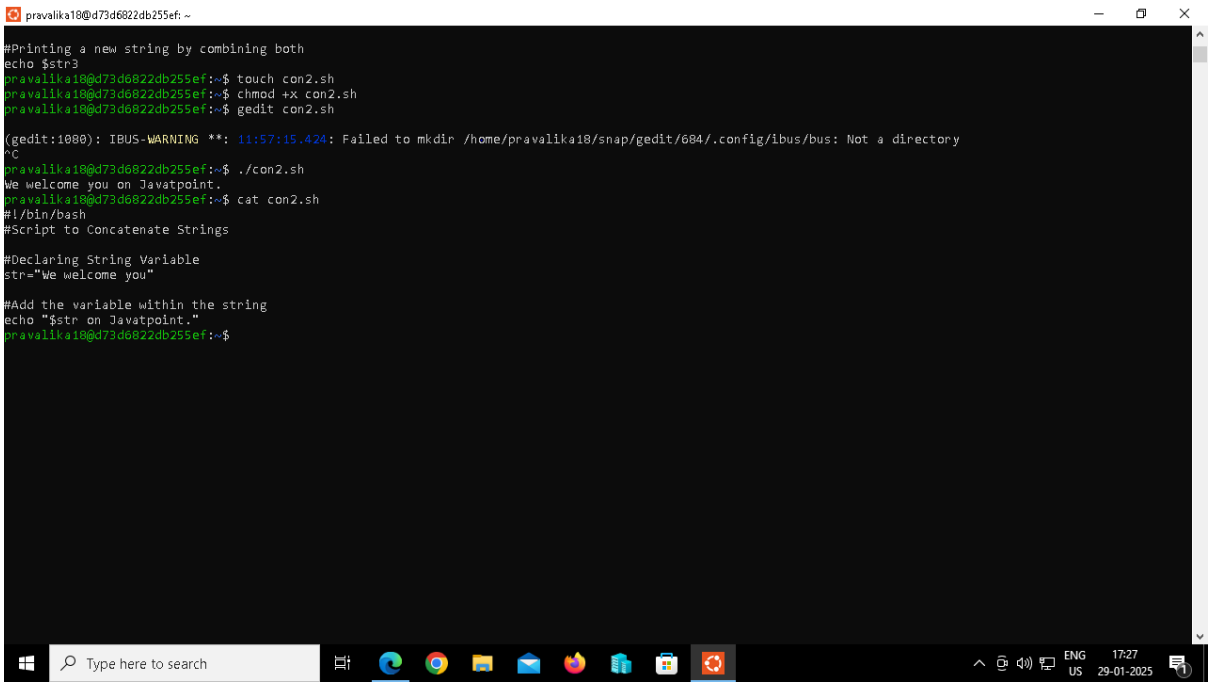
Example 1: Write Variables Side by Side

This is the basic example of String Concatenation, and we do not need any extra operator or function in this method.



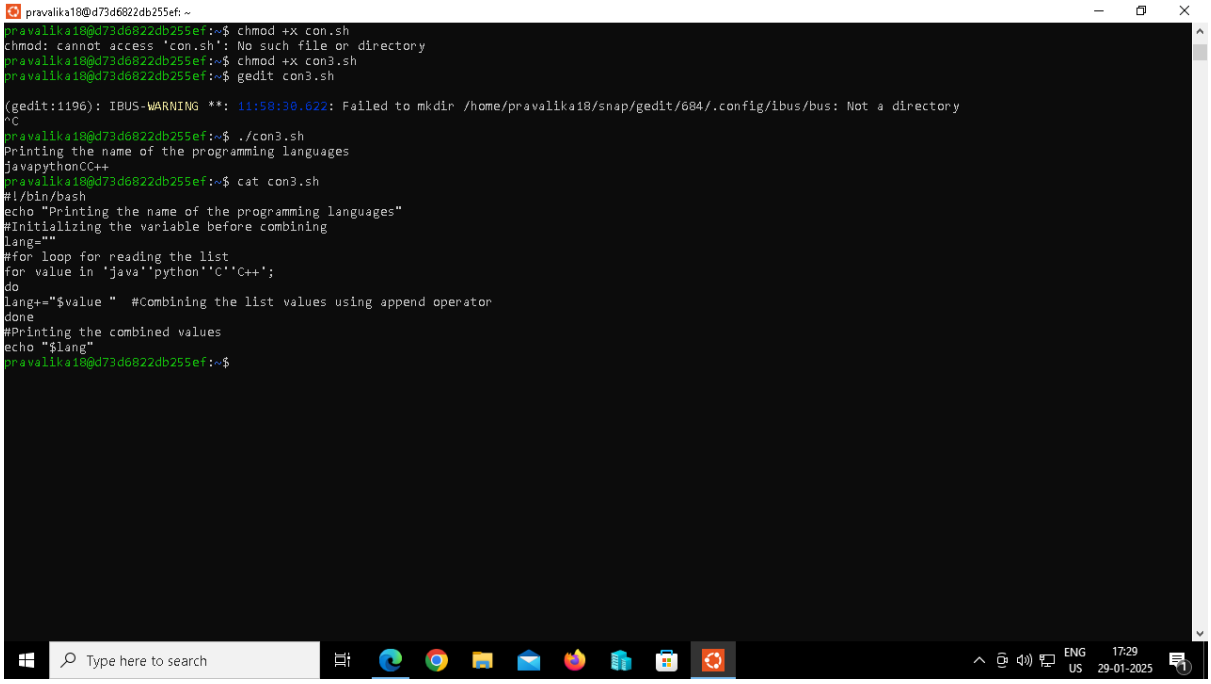
Example 2: Using Double Quotes

We can use variables inside the string, which is defined with double-quotes. The string variable can be applied in any position of the string data.



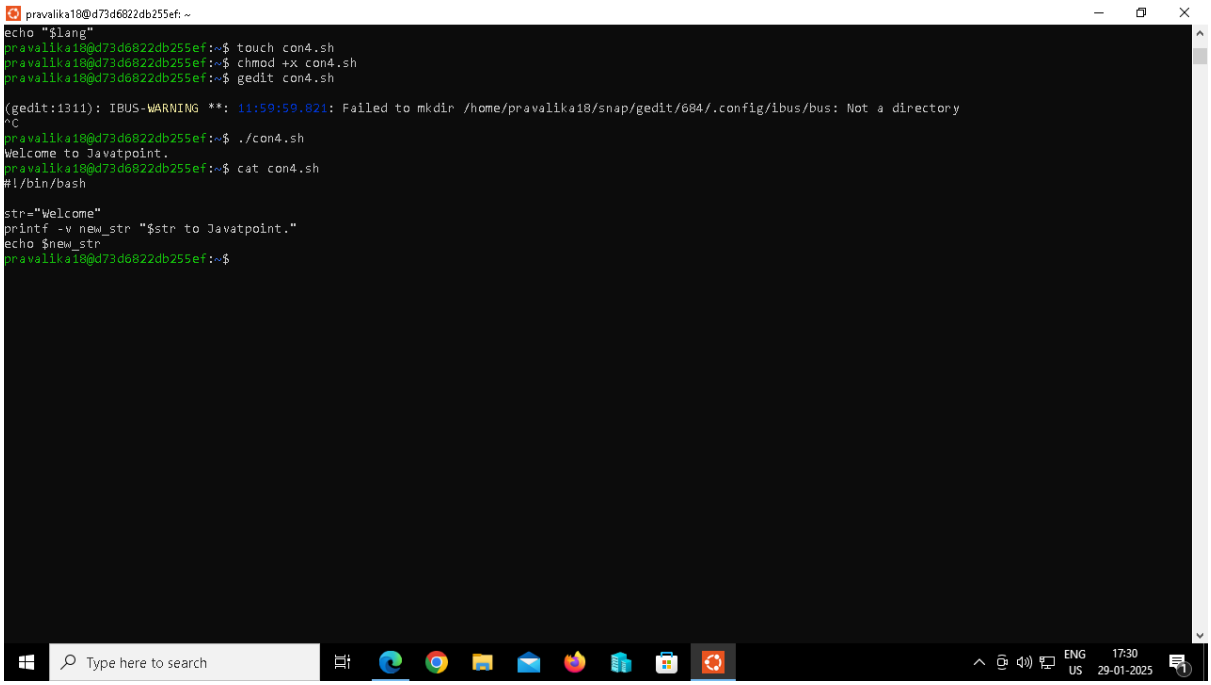
Example 3: Using Append Operator with Loop

Most of the popular programming languages provide support for append operator (+=) which is the combination of the plus and equal sign. It will add new strings to the end of the string variable.



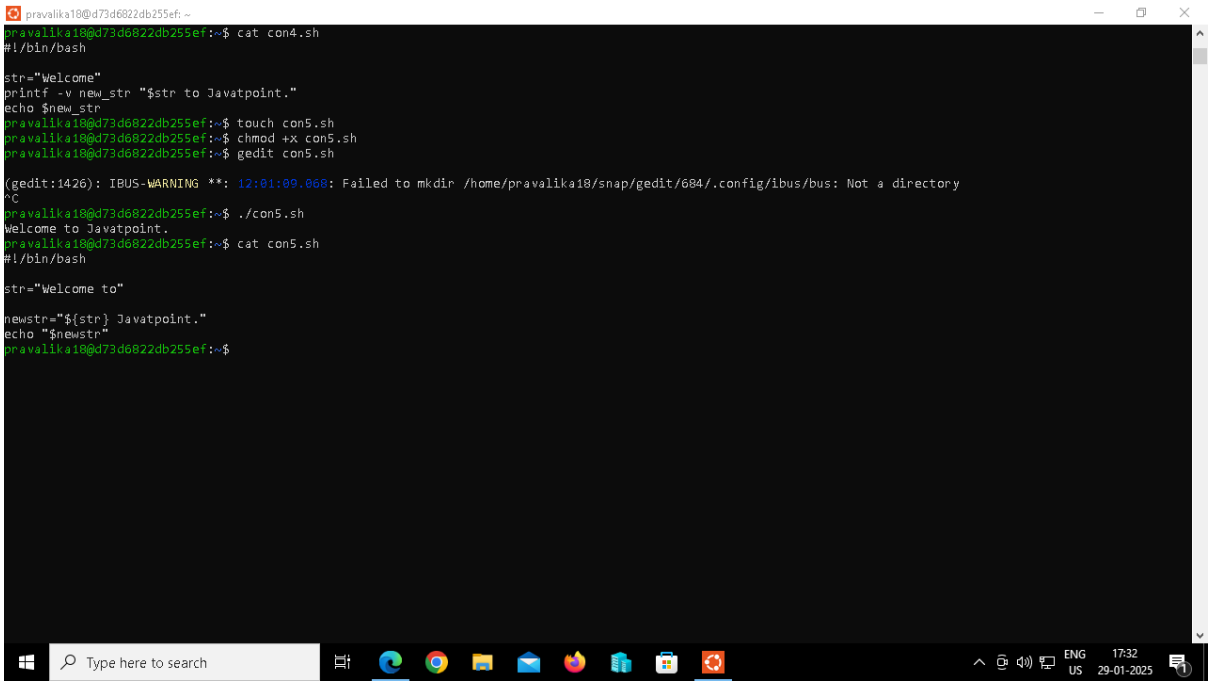
Example 4: Using the Printf Function

In bash, printf is a function which is used to print and concatenate the strings.



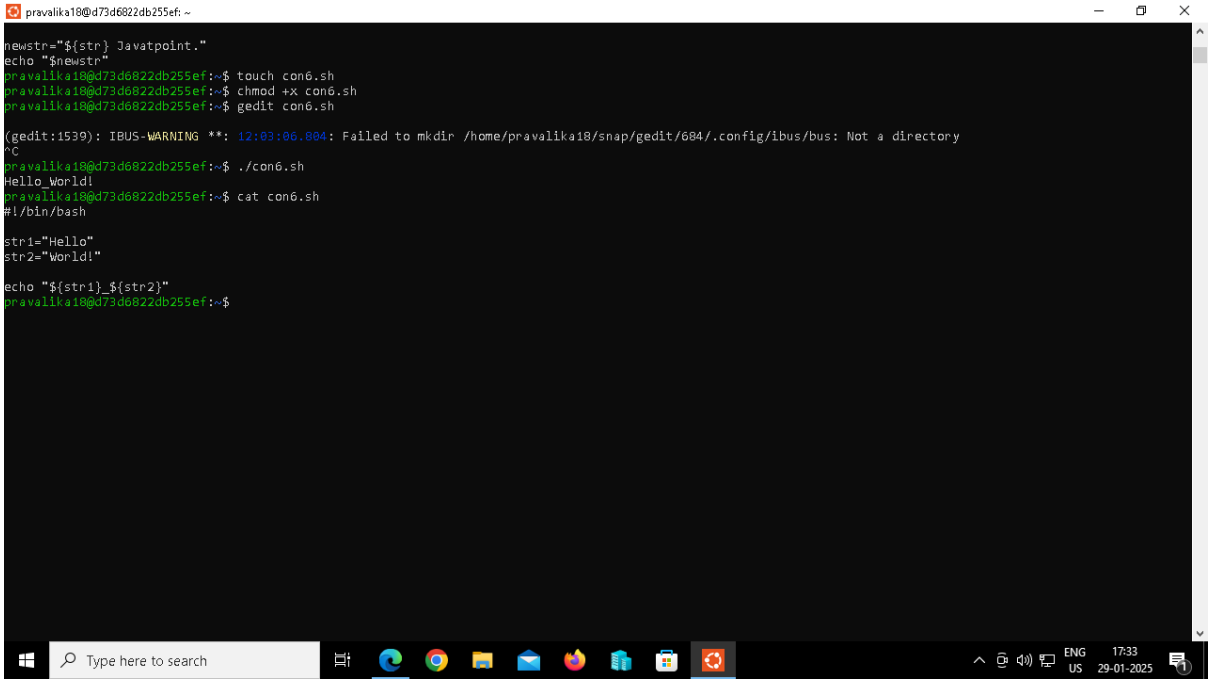
Example 5: Using Literal Strings

String concatenation can also be performed with a literal string by using curly braces{}. They should be used in such a way that the variable does not mix up with the literal string.

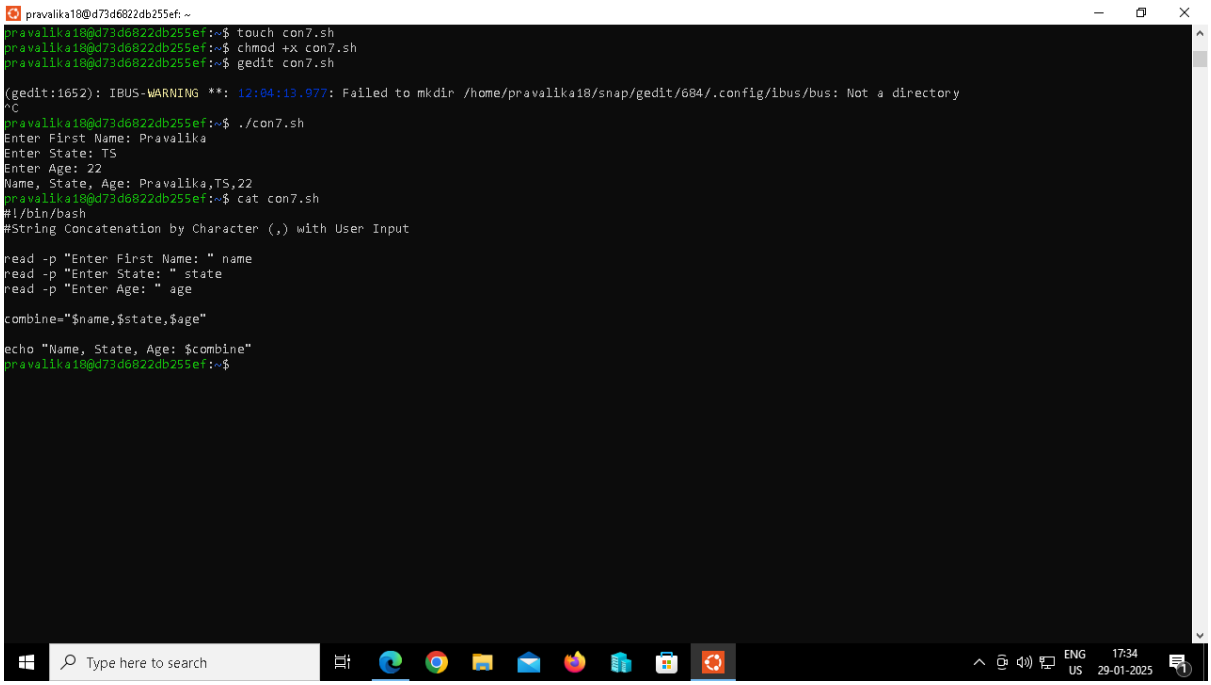


Example 6: Using Underscore

Using underscore for concatenating the string in bash shell is one of the common tasks. It is mostly used for assigning a name to the files.



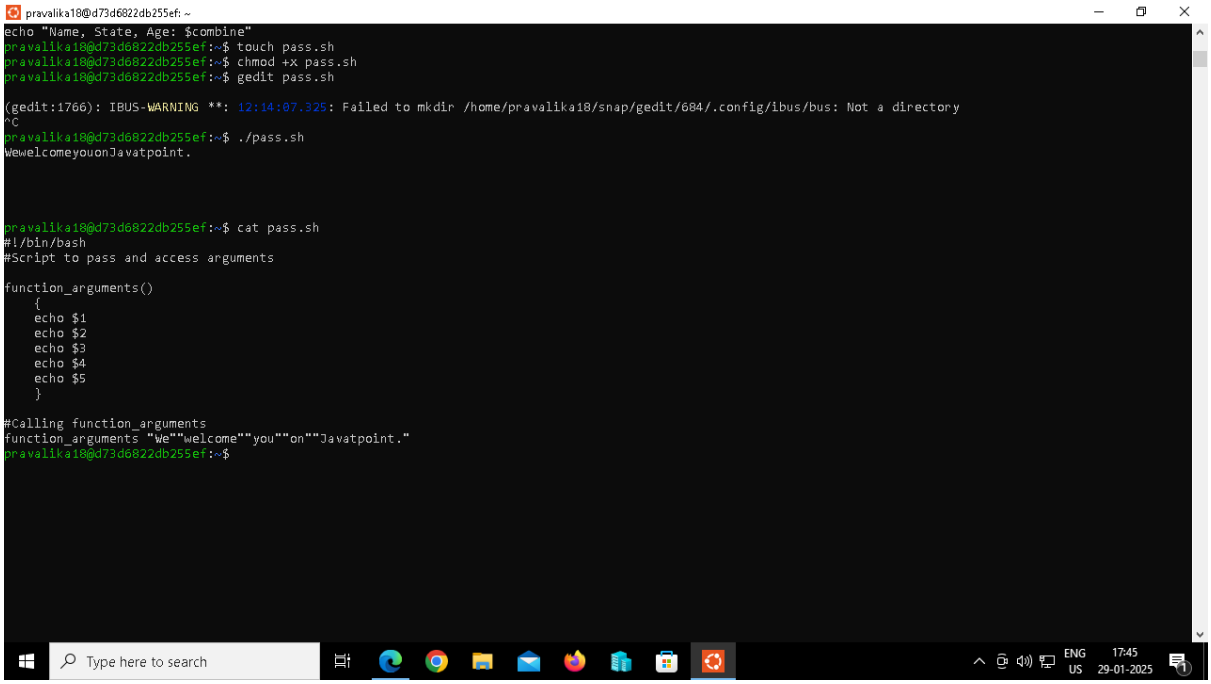
Example 7: Using any Character



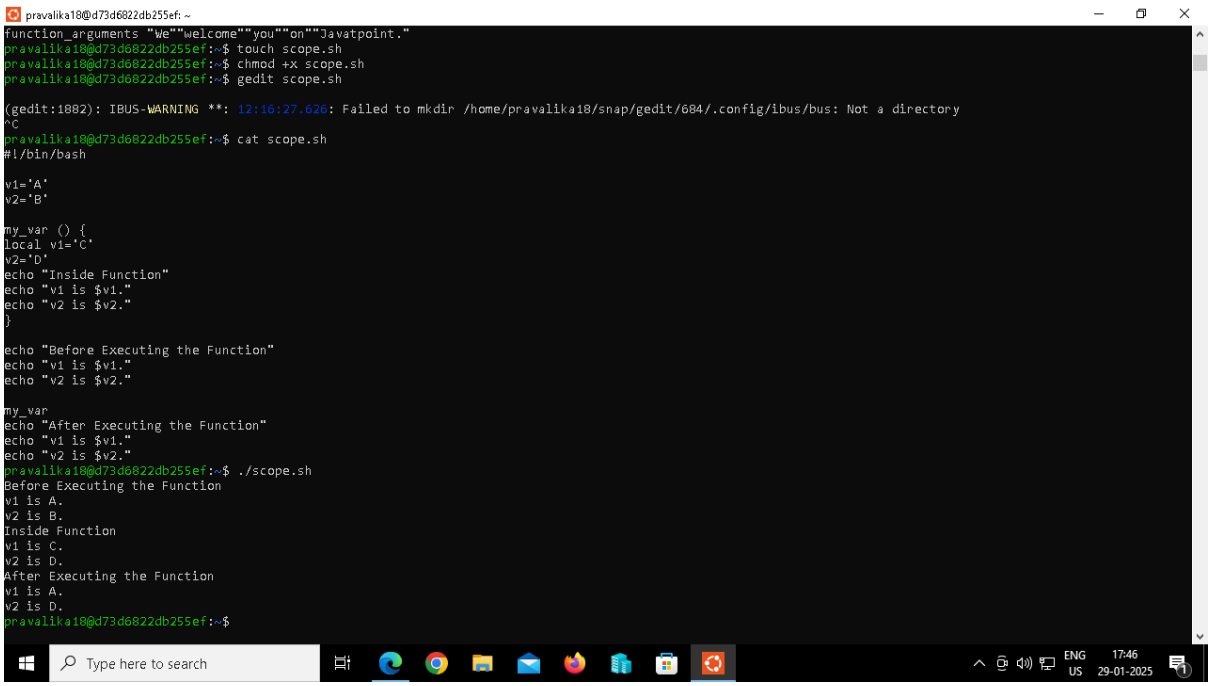
Bash Functions

Passing Arguments

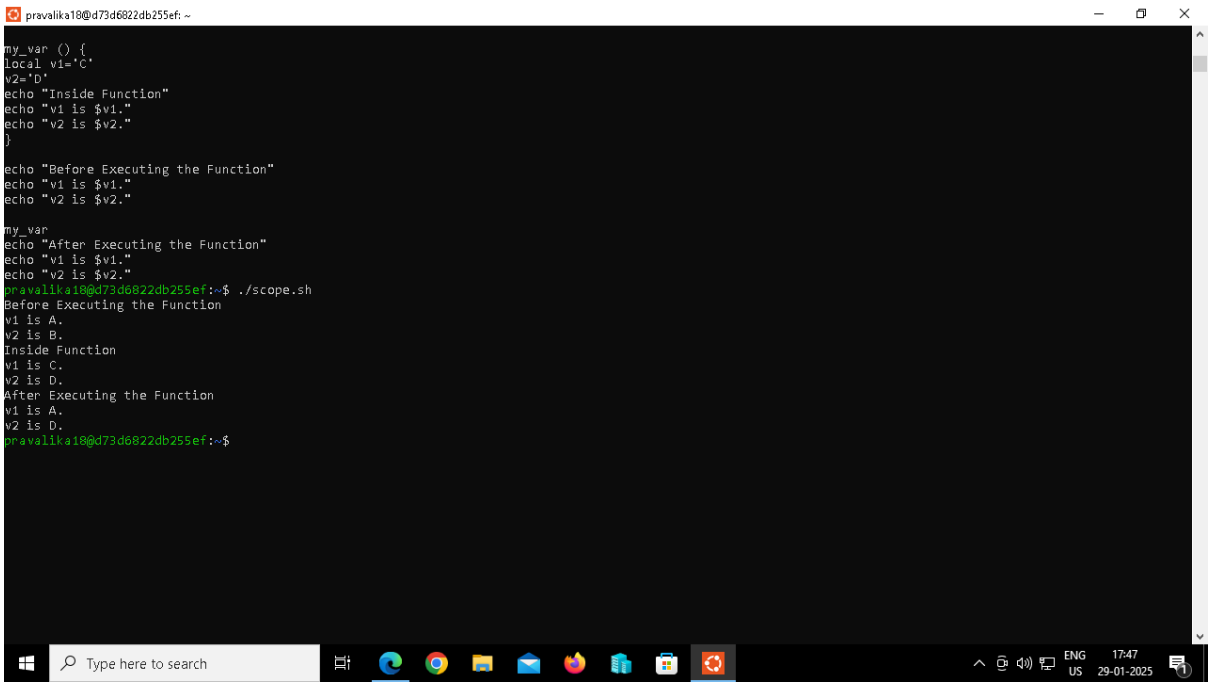
Following isthe code that illustrates the procedure on how to pass arguments to functions, and access the arguments inside the function.



Variable Scope

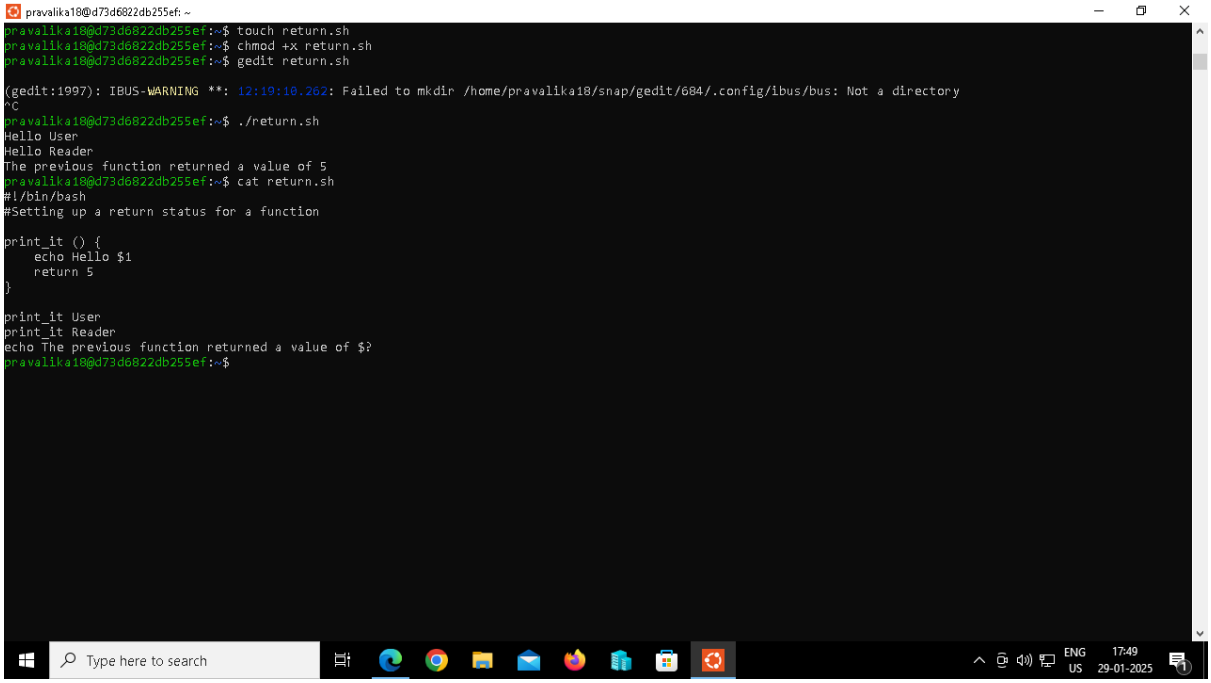


```
pravalika18@d73d6822db255ef: ~
function_arguments "We""welcome""you""on""Javatpoint."
pravalika18@d73d6822db255ef:~$ touch scope.sh
pravalika18@d73d6822db255ef:~$ chmod +x scope.sh
pravalika18@d73d6822db255ef:~$ gedit scope.sh

(gedit:1882): IBUS-WARNING **: 12:16:27.626: Failed to mkdir /home/pravalika18/snap/gedit/684/.config/ibus/bus: Not a directory
^C
pravalika18@d73d6822db255ef:~$ cat scope.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 $v2."
pravalika18@d73d6822db255ef:~$ ./scope.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.
pravalika18@d73d6822db255ef:~$
```



```
pravalika18@d73d6822db255ef: ~
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 $v2."
pravalika18@d73d6822db255ef:~$ ./scope.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.
pravalika18@d73d6822db255ef:~$
```

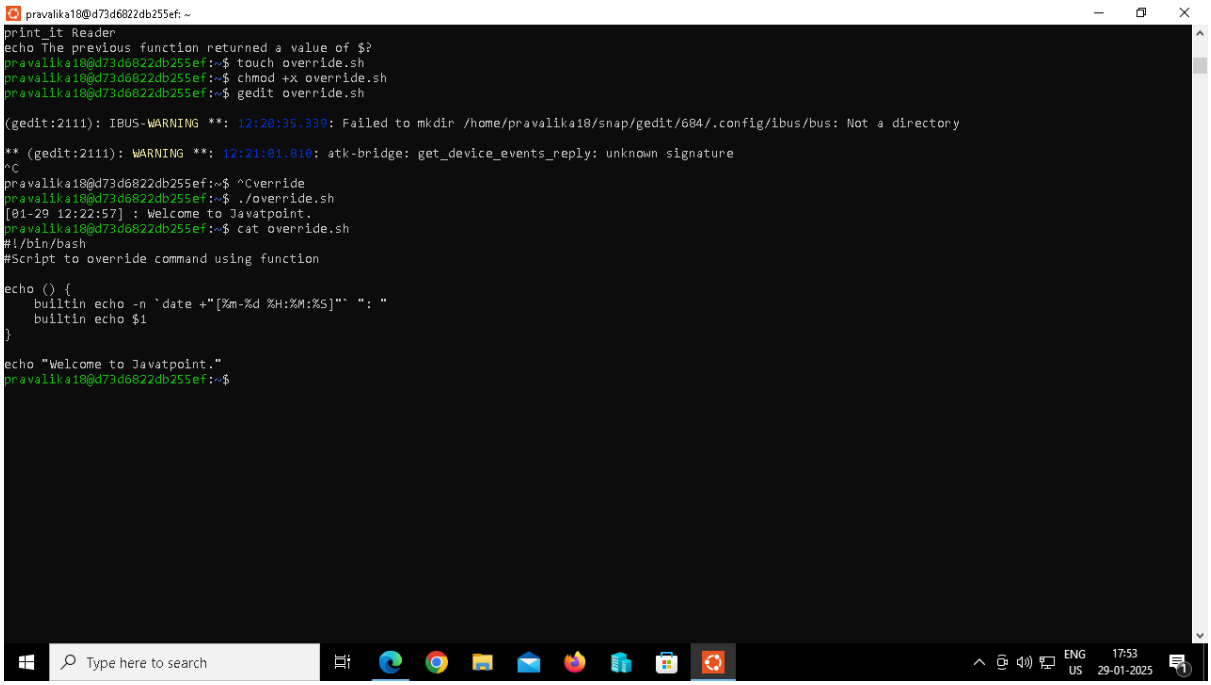
Return Values



Overriding Commands

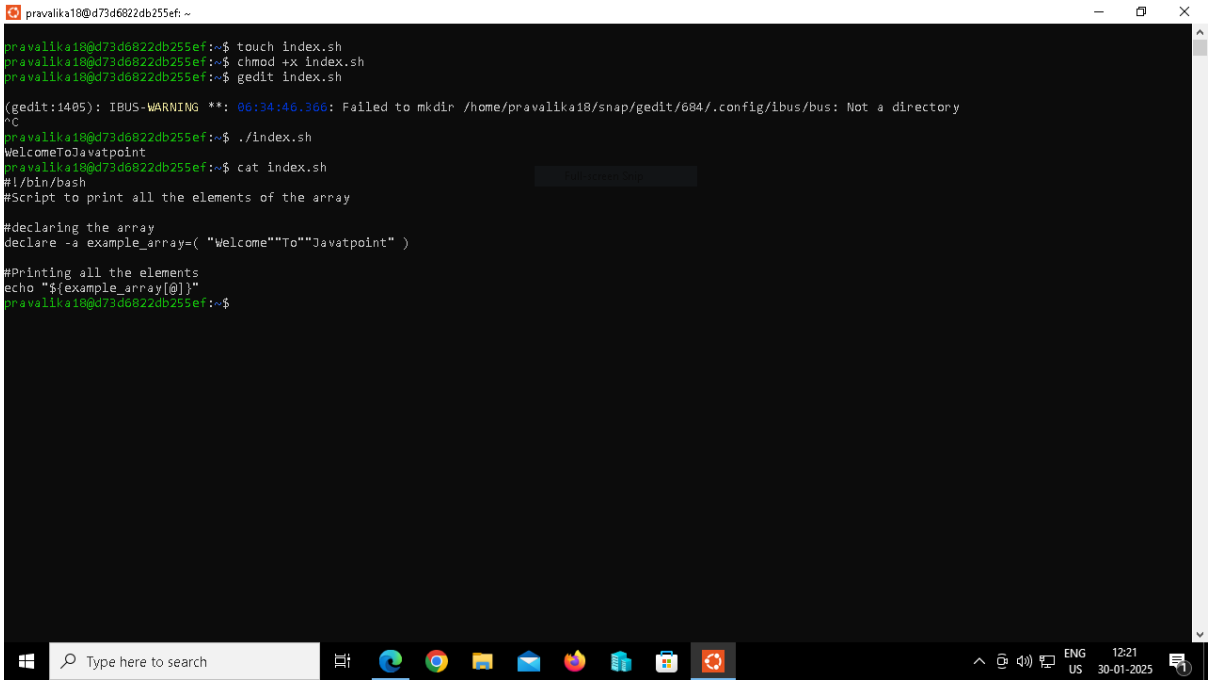
Example

In this example, we have overridden the 'echo' command and added the time stamp in the form of the argument to the 'echo' command.

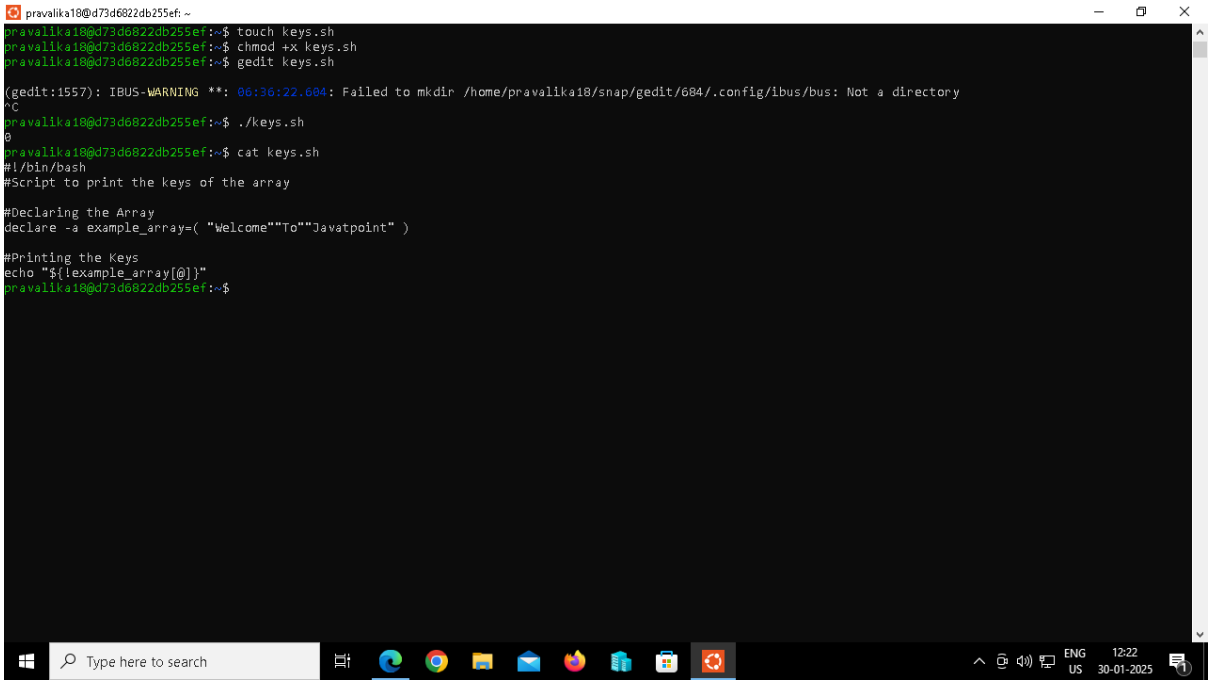


BASH ARRAY

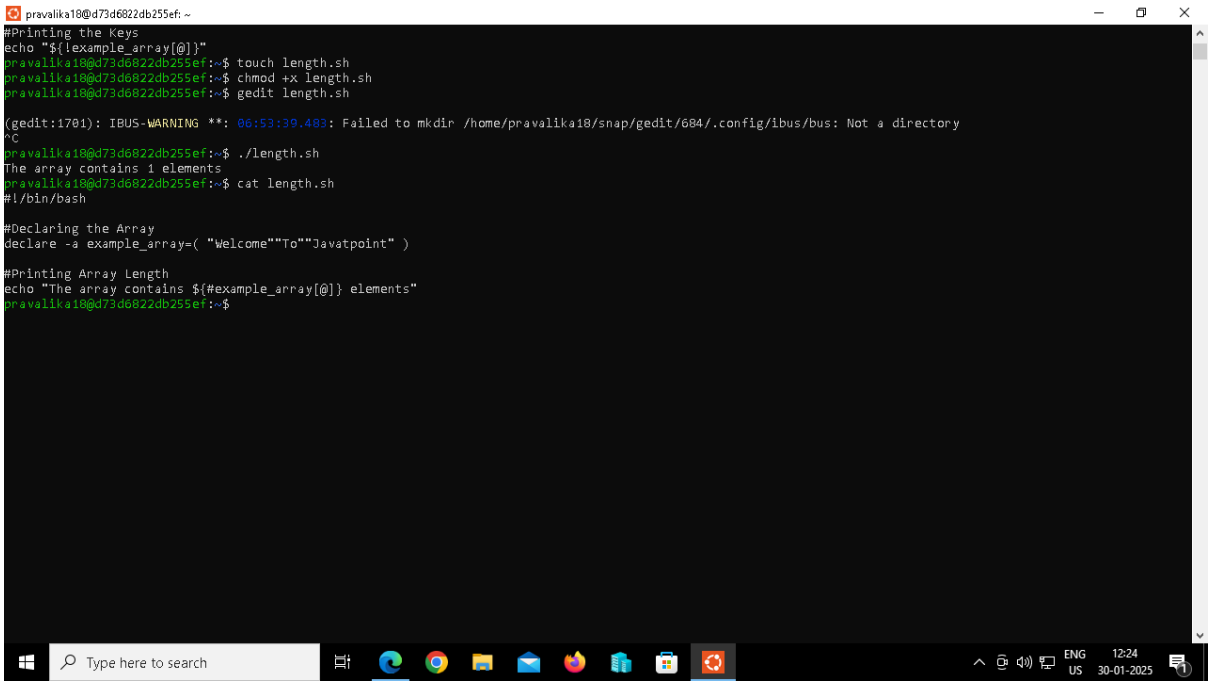
1.let's print an element of an array with an index of 2:



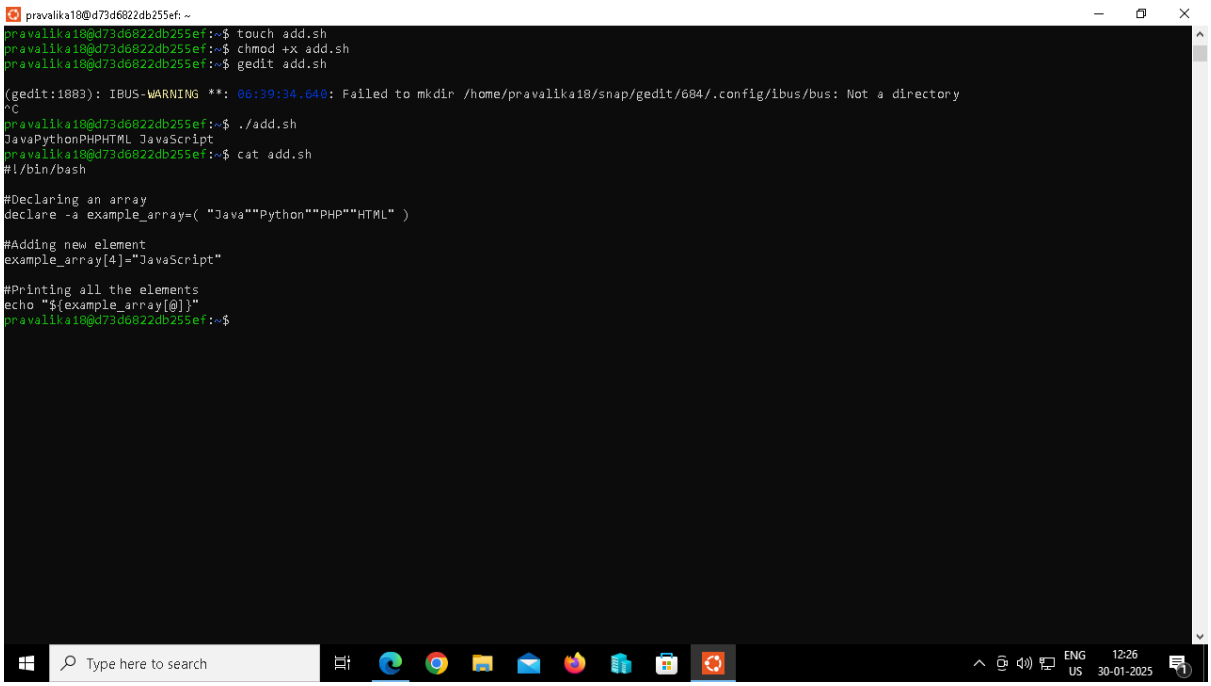
2. Printing the Keys of an Array



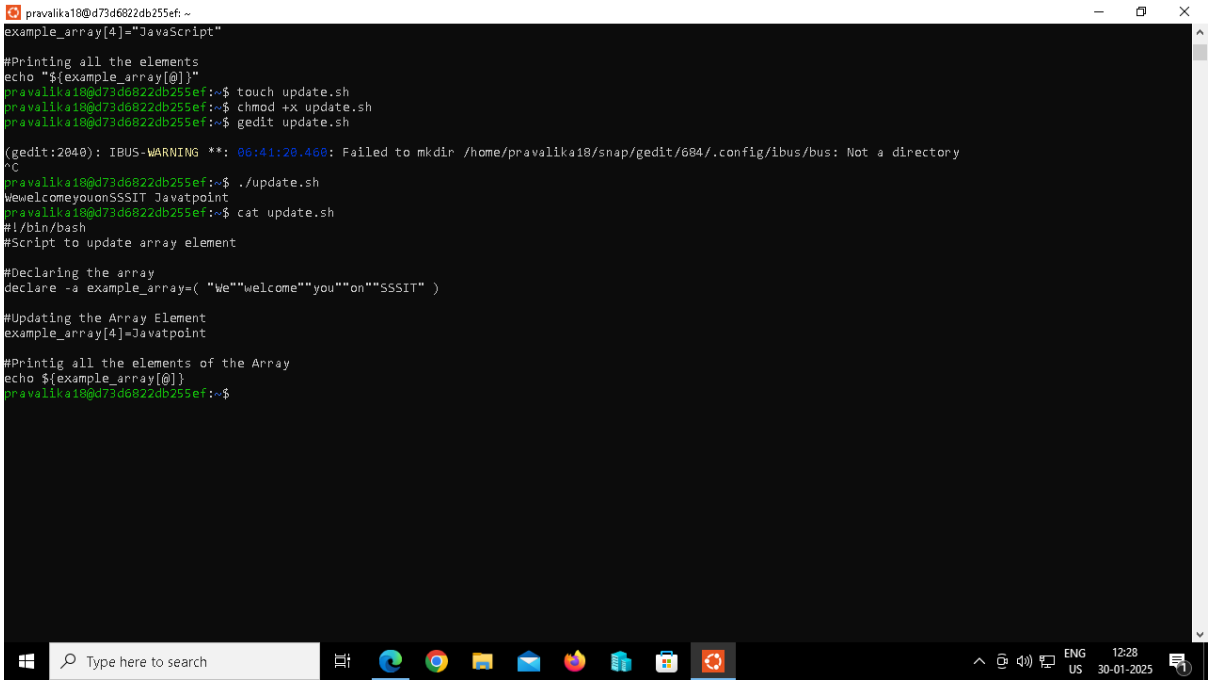
3. Finding Array Length



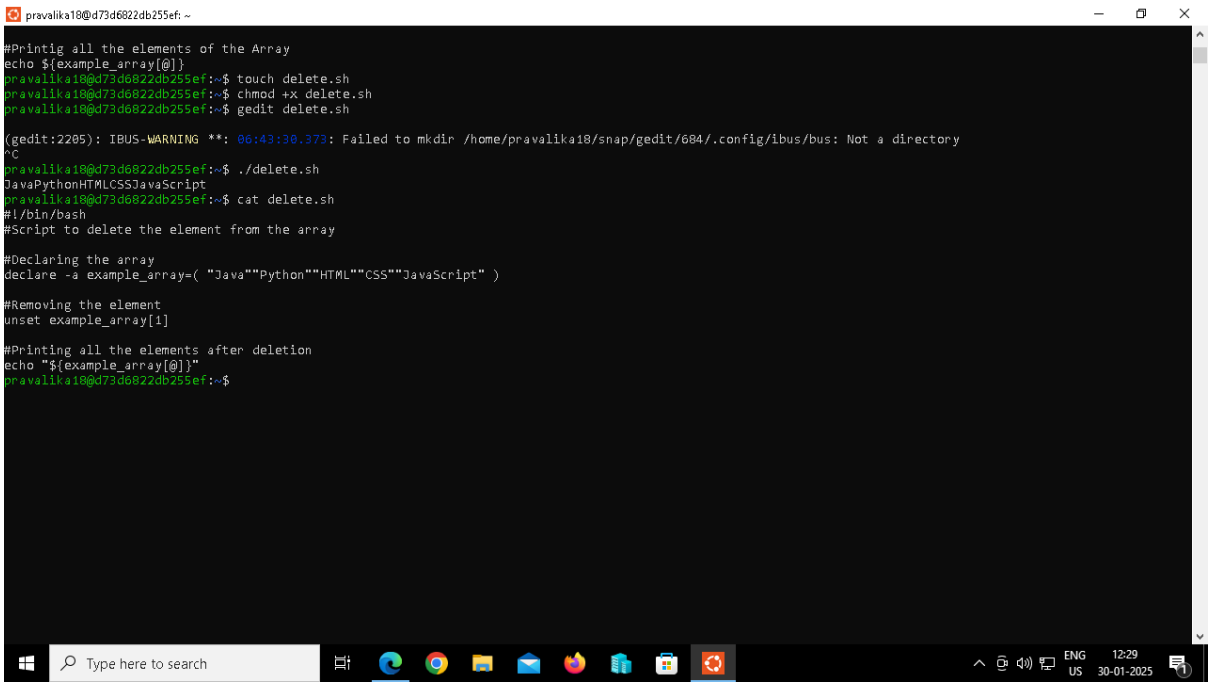
4. Adding Elements to an Array



5. Updating Array Element



6. Deleting an Element from an Array



7. Slice Array Elements