**This Mini project is to test my skill with (Linux shell scripting – control flow).**

**What is control flow:** This is a flow that direct the order in which commands or instructions are executed in a script. It’s like a decisional road map which direct the path to take based on certain conditions OR the number of times to visit / operate in a place**.**

**Control flow** is shell scripting is a tool for making decisions, they are backbone statement that give decisional texture in programming. Those statements allow your script decide what to do Or how to proceed based on the conditions, loops or user inputs. While Bash and others shell interpreter provide control flow statement like the below.

1. **if / else.**
2. **For loops**
3. **While loops**
4. **Until loops**

**The below is the entire script that focus on the if /else which is the stage 1:**

**#!/bin/bash**

**read -p "Enter a number: " num**

**if [ $num -gt 0 ]; then**

**echo "The number is positive."**

**elif [ $num -lt 0 ]; then**

**echo "The number is negative."**

**else**

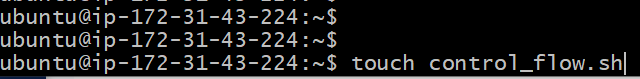
**echo "The number is zero."**

**Fi**

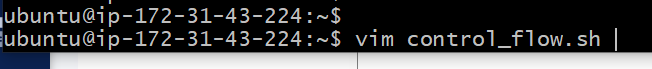
**But I will start with the test of the If and gradually move to the else and stage 2.**

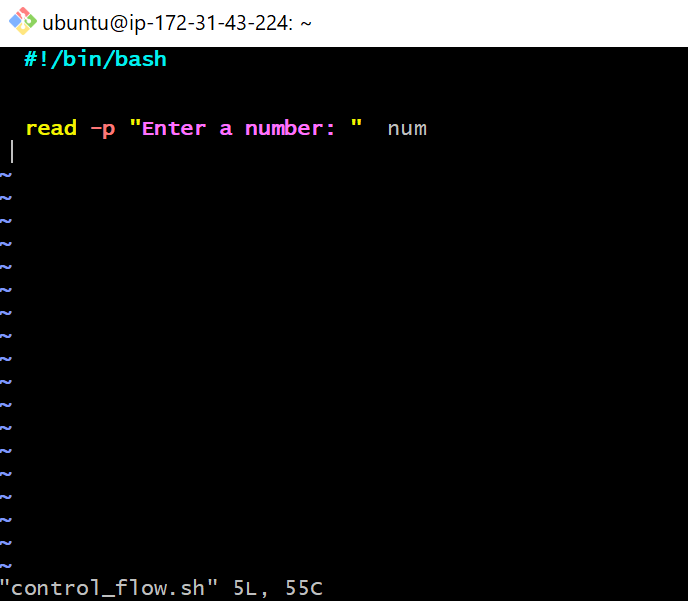
**To start with we need to:**

**1 creat a file and name it control\_flow.sh**

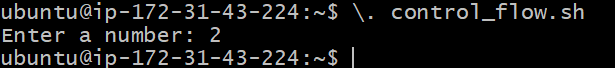


**2. Put this code and execute. read -p "Enter a number: " num**

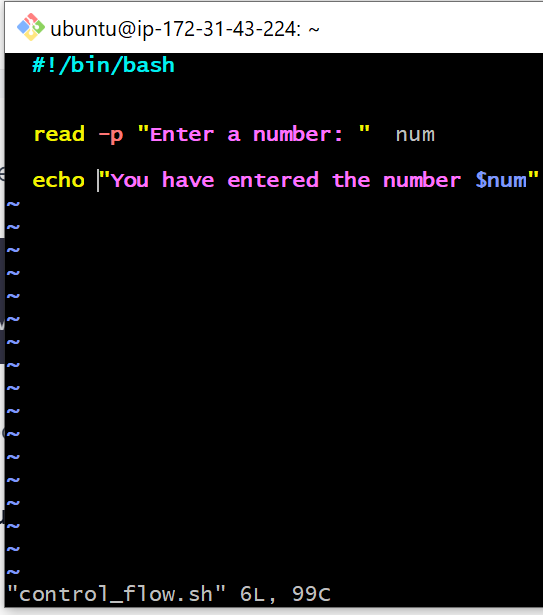




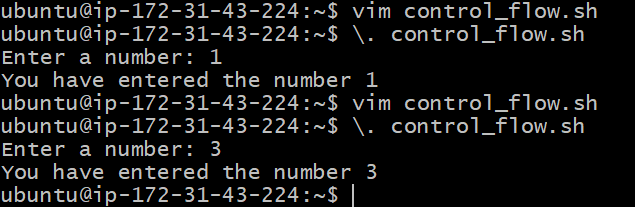
**After Execution: the script give no output because it has no Standard output instruction.**



**Added another layer of script to show Standard output.**



**This is the outcome:**

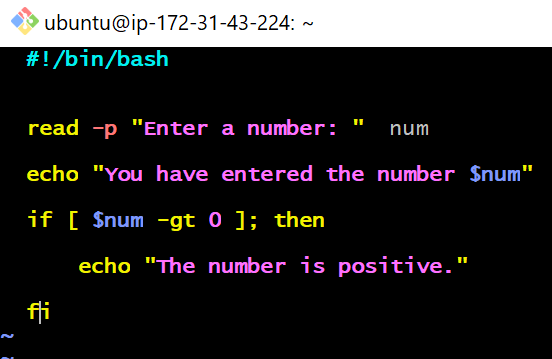


**Moving to the text of IF statement:**

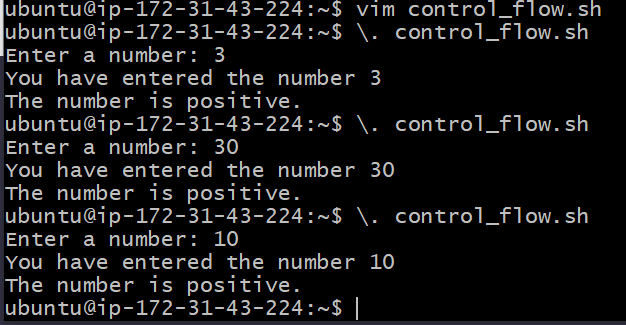
**-gt operator = greater than**

**$num = Number variable**

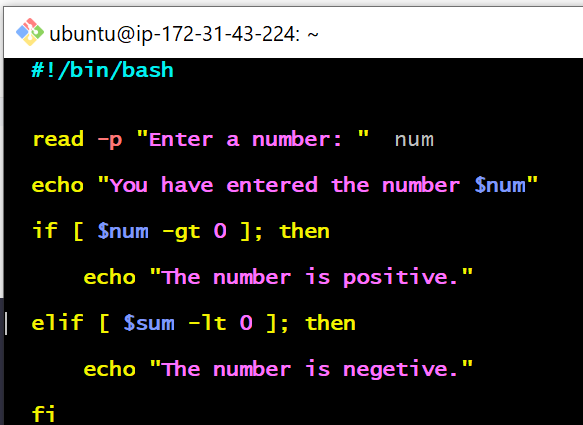
**The ( if ) statement I this script is saying [ if the number is -gt ] reply with the number is positive.**



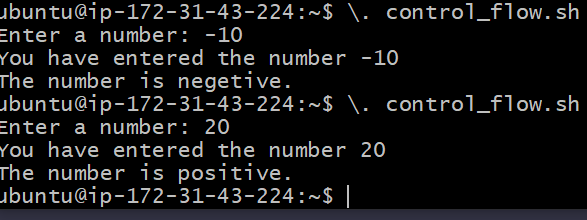
**Outcome.**



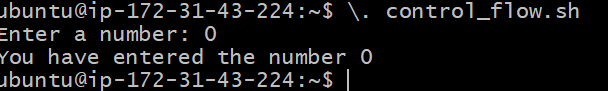
**Elif statement: This allow the test that show when the IF condition is not met. Example if a number is 10 display positive, but if the number is less than 10 display negative… the elif take care of the highlighted part (but if the number is less than 10 display negative).**



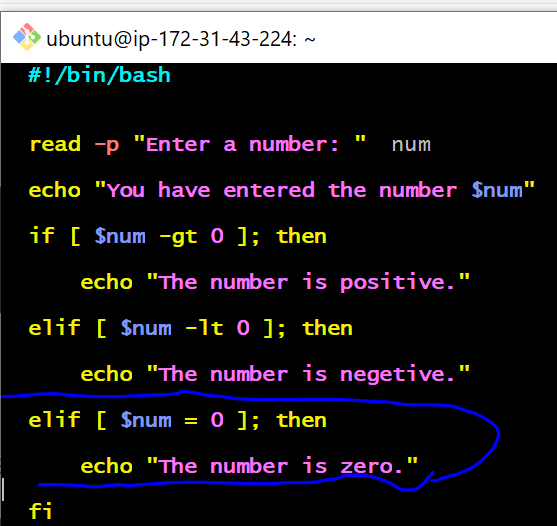
**Outcome:**



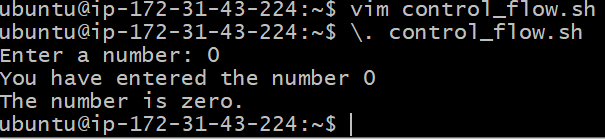
**However, zero was not defined yet, so when tested I got the below message.**



**So let’s capture the Zero script. I decided to test the Zero by myself to understand what will occur.**



**Outcome of the Zero control flow conditional test.**



**STAGE 2 - LOOPS**

Loop are tools used for the auto generation of output result, report. The script command can be used for repetitive task.

Note the below:

**For:** This keyword initiates the loop, signaling the start of the block of code that will be repeated.

**Items:** This is a variables that temporarily holds the value of each items in the list as the loop iterates. For each iteration of the loop, allowing the command inside the loop to act on this value.

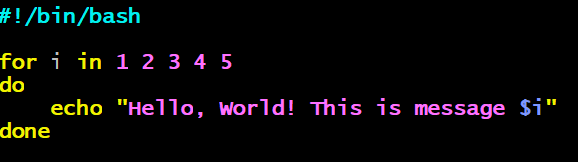
**In:** The in keyword is followed by a list of items that the loop will iterate over. This list can be a series of values, an array. Or the outcome of a command. The loop executes once for each items in this list.

;: A semicolon is used here to separate the list of items from the do keyword that follows. If you place the do keyword on the next line, the semicolon is Optional.

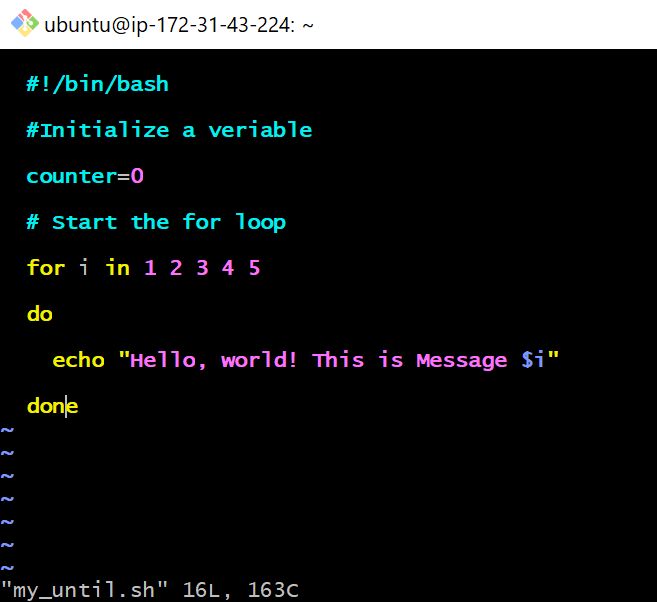
Do: This keyword precedes the block of commands that be executed for each items in the list. The block can contain one or multiple commands, and it perform a wild range of actions, from simple echoes to complex conditional logic.

Done: This keyword mark the end of the loop. It signifies that all commands loop have been executed for each items in the list, and the loop is complete.

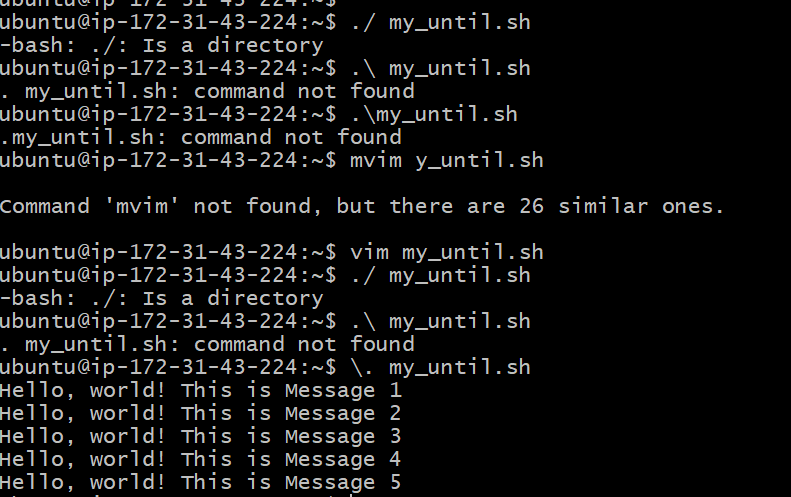
**For: This is for single command for multiple respond at once.**



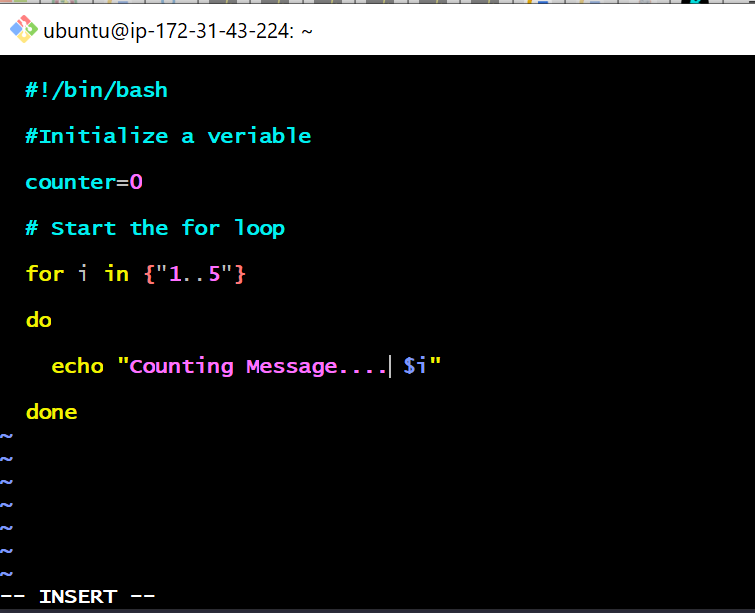
**Configuring the bash script.**



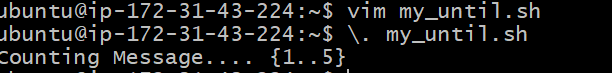
**Output of the command.**



**Counting the number / message.**

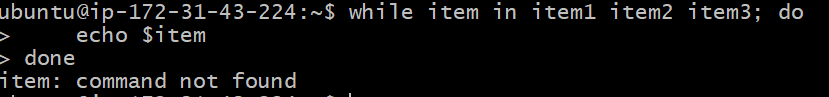


**The outcome of counting message / Number bash scripting.**



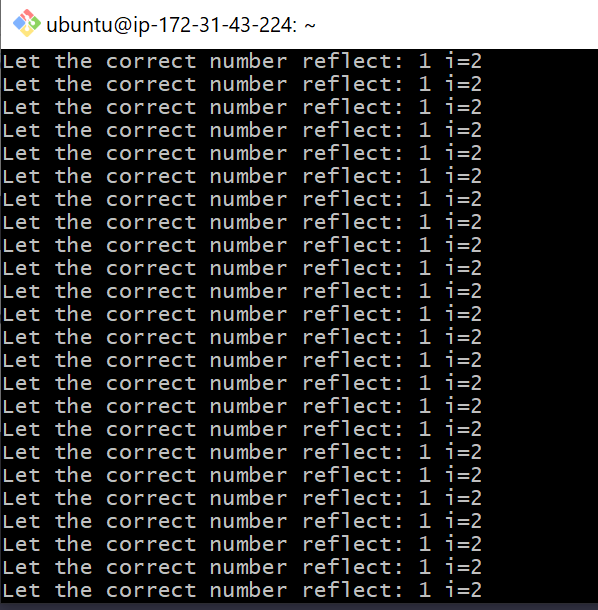
**Using While…**

**Error encountered.**

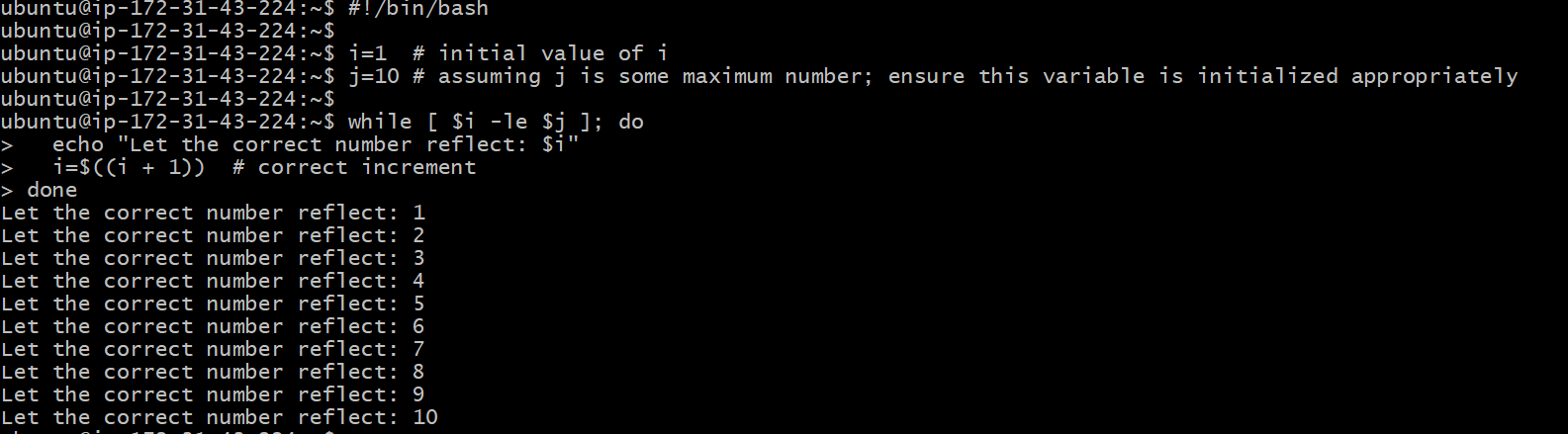


**Using Davina AI to resolve the error message issue.**

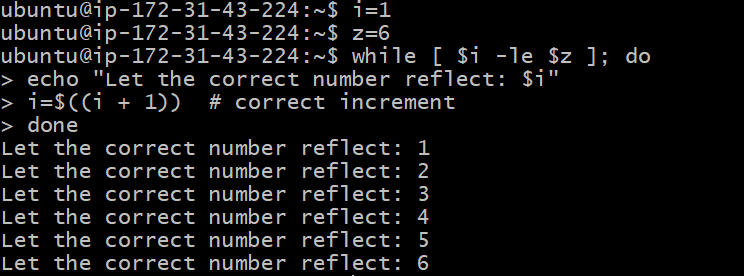
**Another error encountered, not stop running for the script and repeating that same thing.**



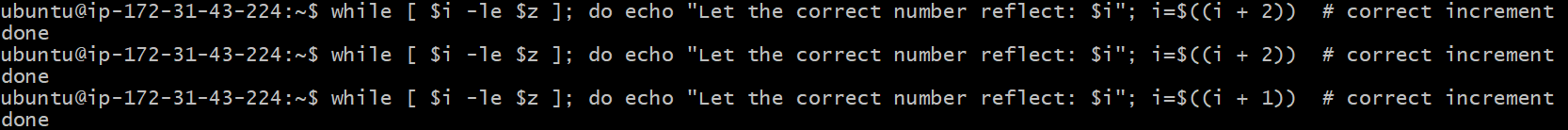
**After Davina Correction. I tried the exact Davina syntax.**



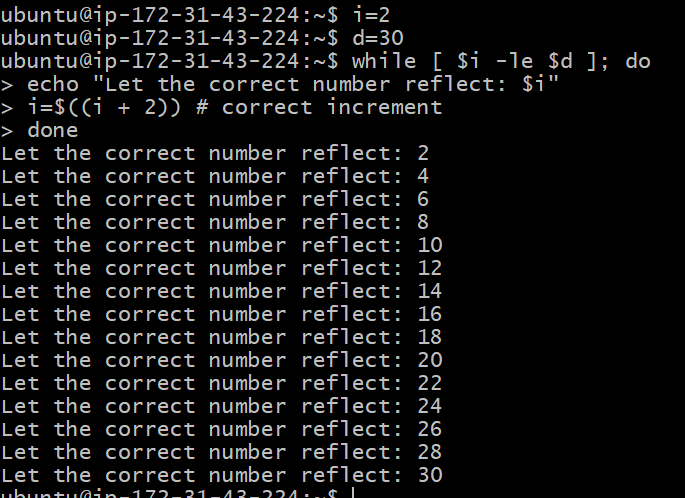
**Then I configured my own syntax.**



**So I tried by changing i to 2 without changing at the initial. So I got this error message.**

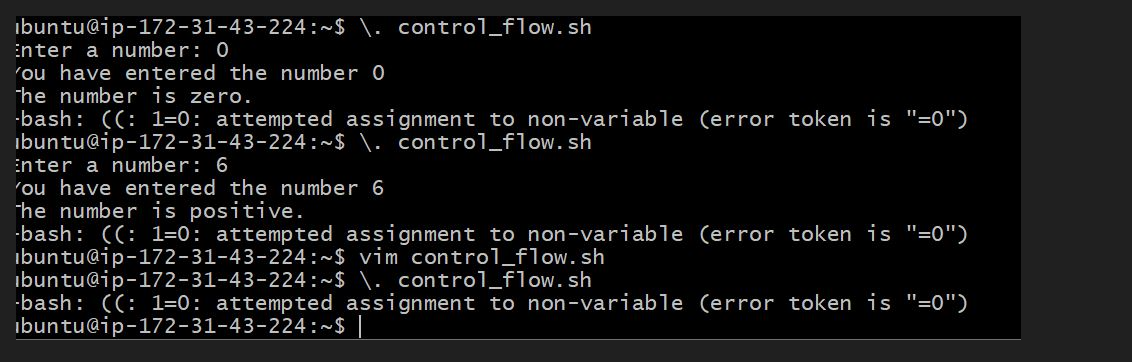


**So I rewrite the script, and this was the outcome.**

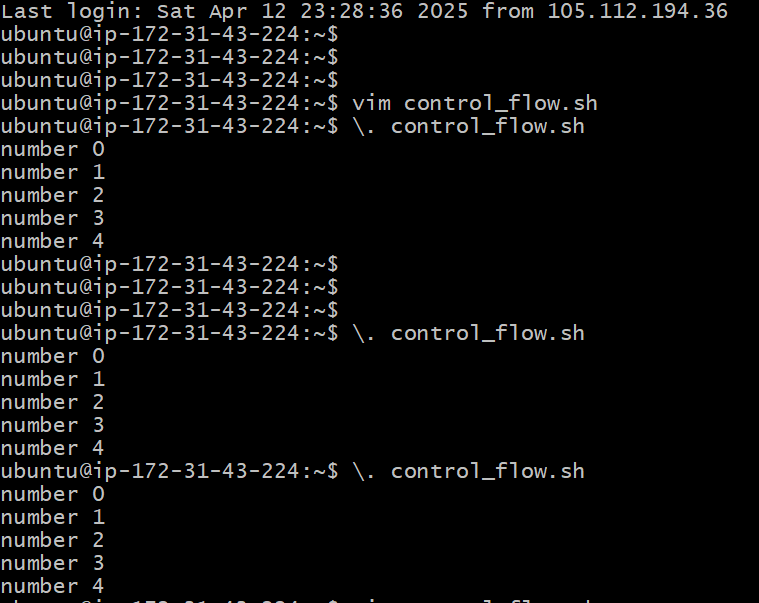


**The next exercise is the use of C- Style form in scripting.**

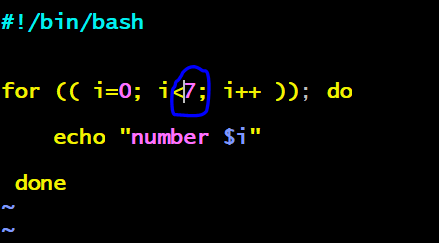
**I typed in the script and erroneously substitute i for 1 and I got the below error message.**

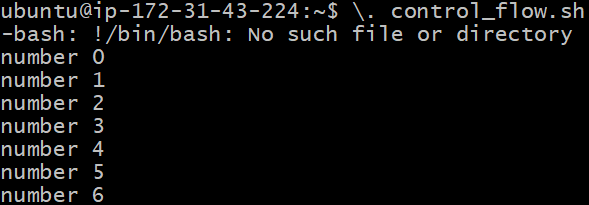


**So, I did not get to see it, but an engineer (Rilwan) in the blocker session assisted and also advise me to always concentrate more on the error message when ever I have any, because most times the error message always have the solution in it. So changed the 1 back to I and execute, then I got this message below.**



**So I modify the script again to my own style and execute.**





**So, that concludes the Mini project on scripting.**