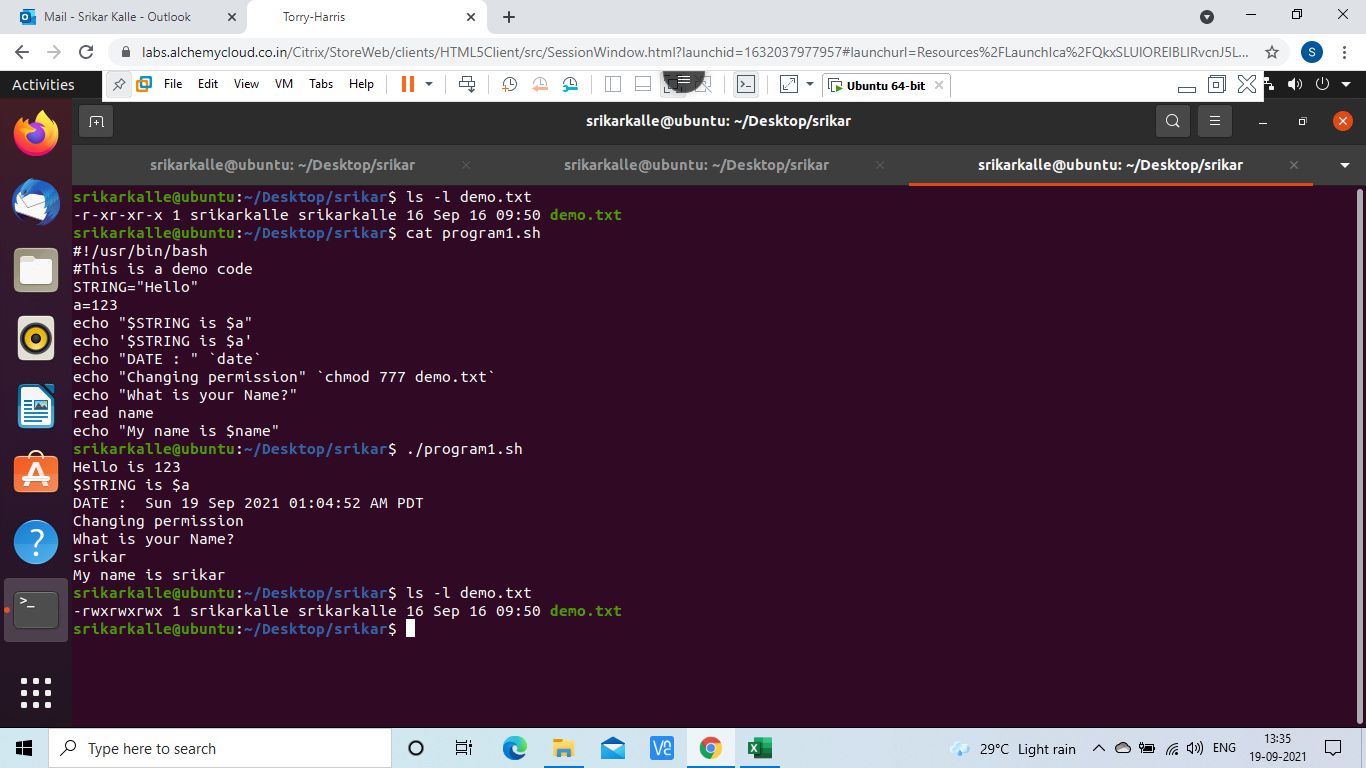
Type of Quotes:

Single quotes:

Enclosing characters in single quotation marks (‘) holds onto the literal value of each character within the quotes. In simpler words, the shell will interpret the enclosed text within single quotes literally and will not interpolate anything including variables, backticks, certain \ escapes, etc. No character in the single quote has special meaning. This is convenient when you do not want to use the escape characters to change the way the bash interprets the input string.

Double quotes:

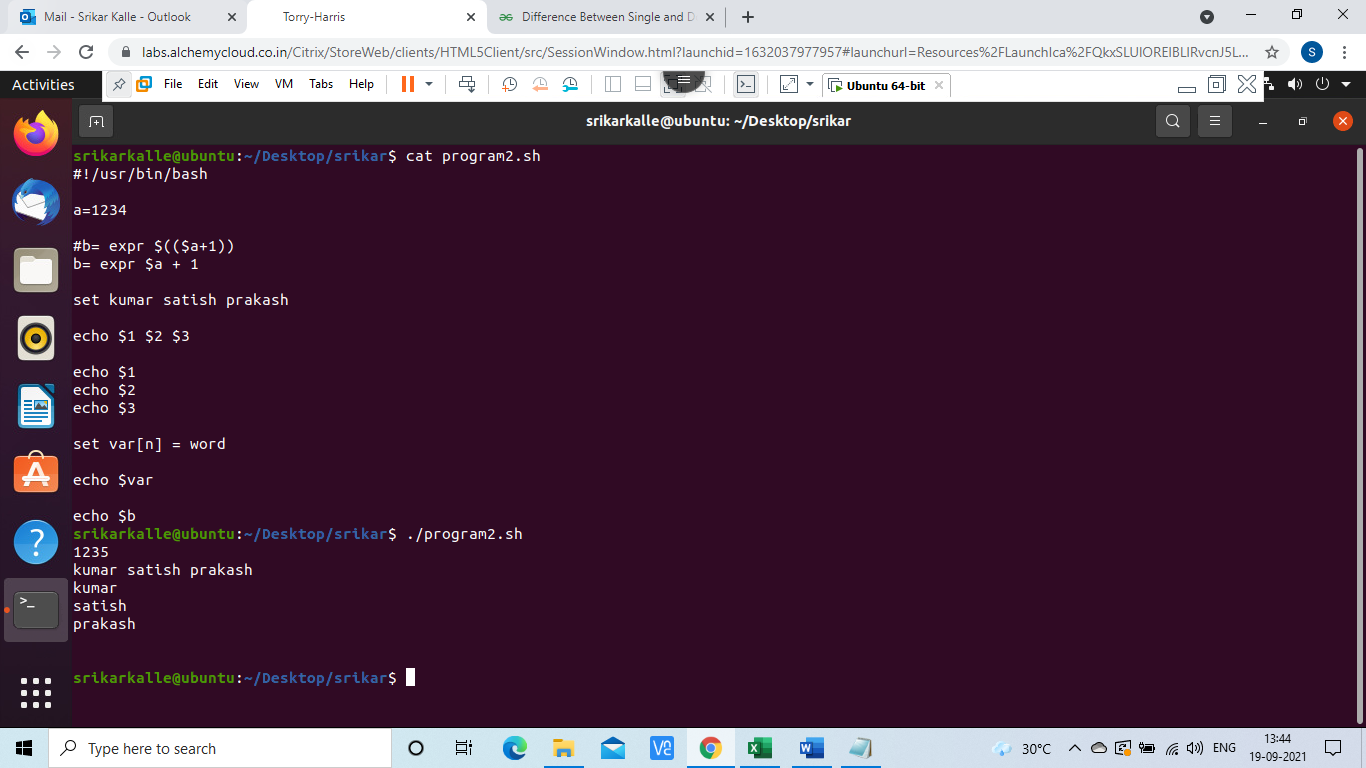
Double quotes are similar to single quotes except that it allows the shell to interpret dollar sign ($), backtick(`), backslash(\) and exclamation mark(!). The characters have special meaning when used with double quotes, and before display, they are evaluated. A double quote may be used within double quotes by preceding it with a backslash.



Variables

A variable is nothing more than a pointer to the actual data.

The name of a variable can contain only letters (a to z or A to Z), numbers ( 0 to 9) or the underscore character ( \_).



CONTROL STATEMENTS:

1. If else
2. If elif else
3. While loop
4. Until loop
5. For loop

COMPARISION OPERATORS

-eq = Equality

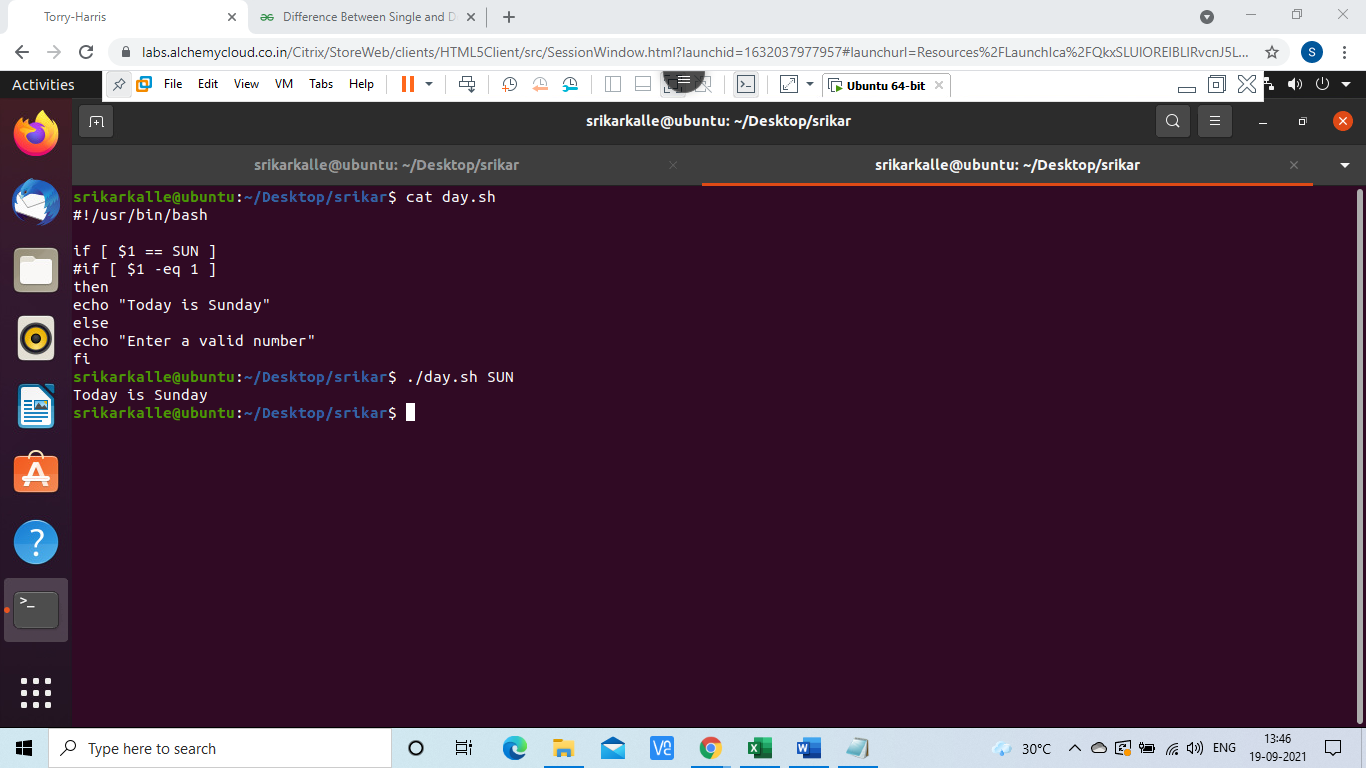
-ne = not equals

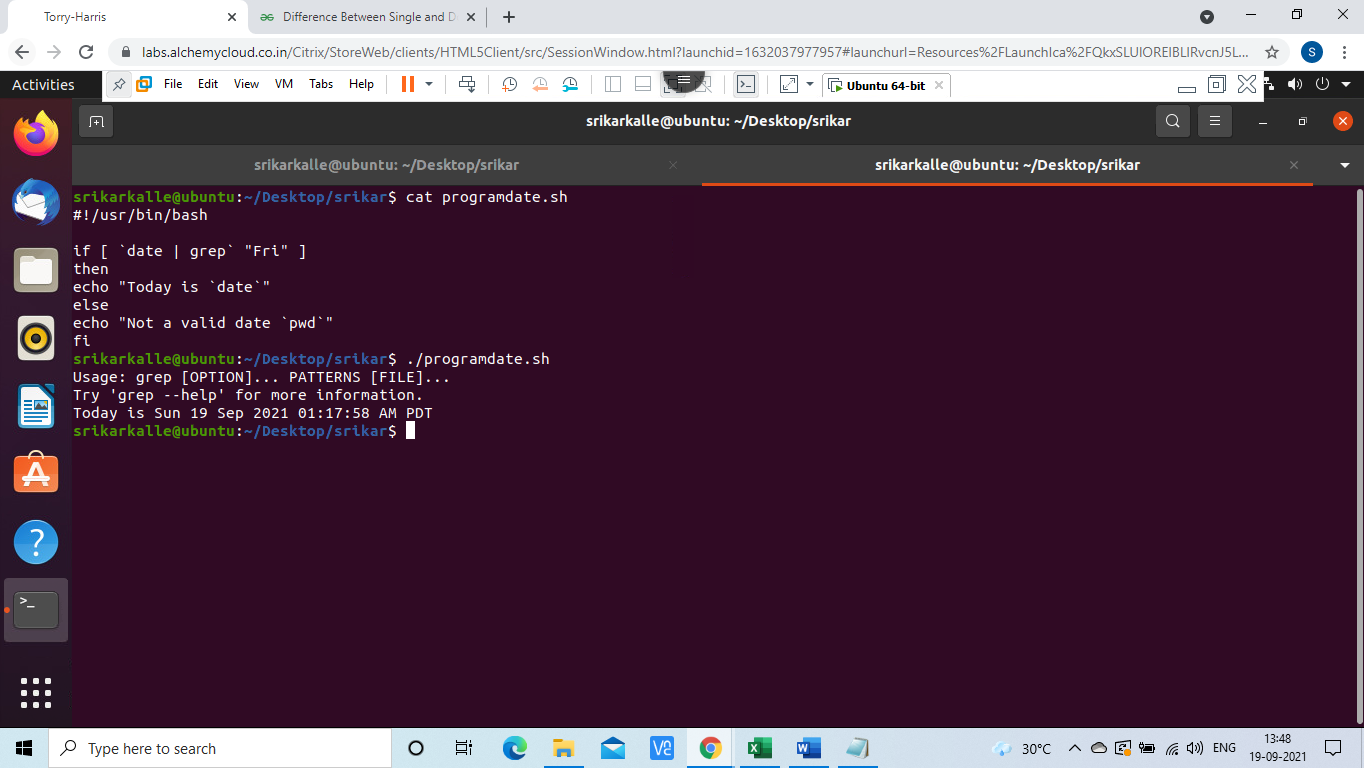
-lt = less than

-gt = greater than

-le = less than or equal

-ge = greater than or equal

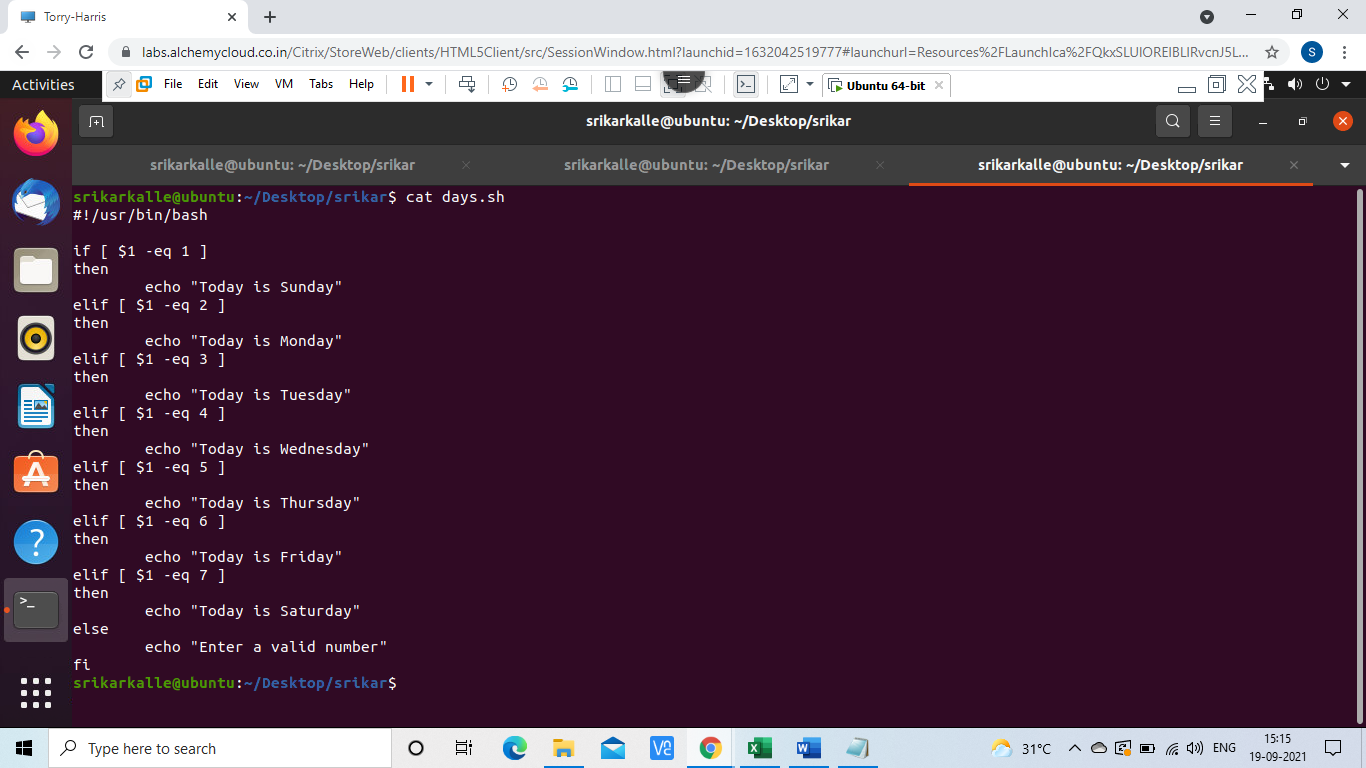




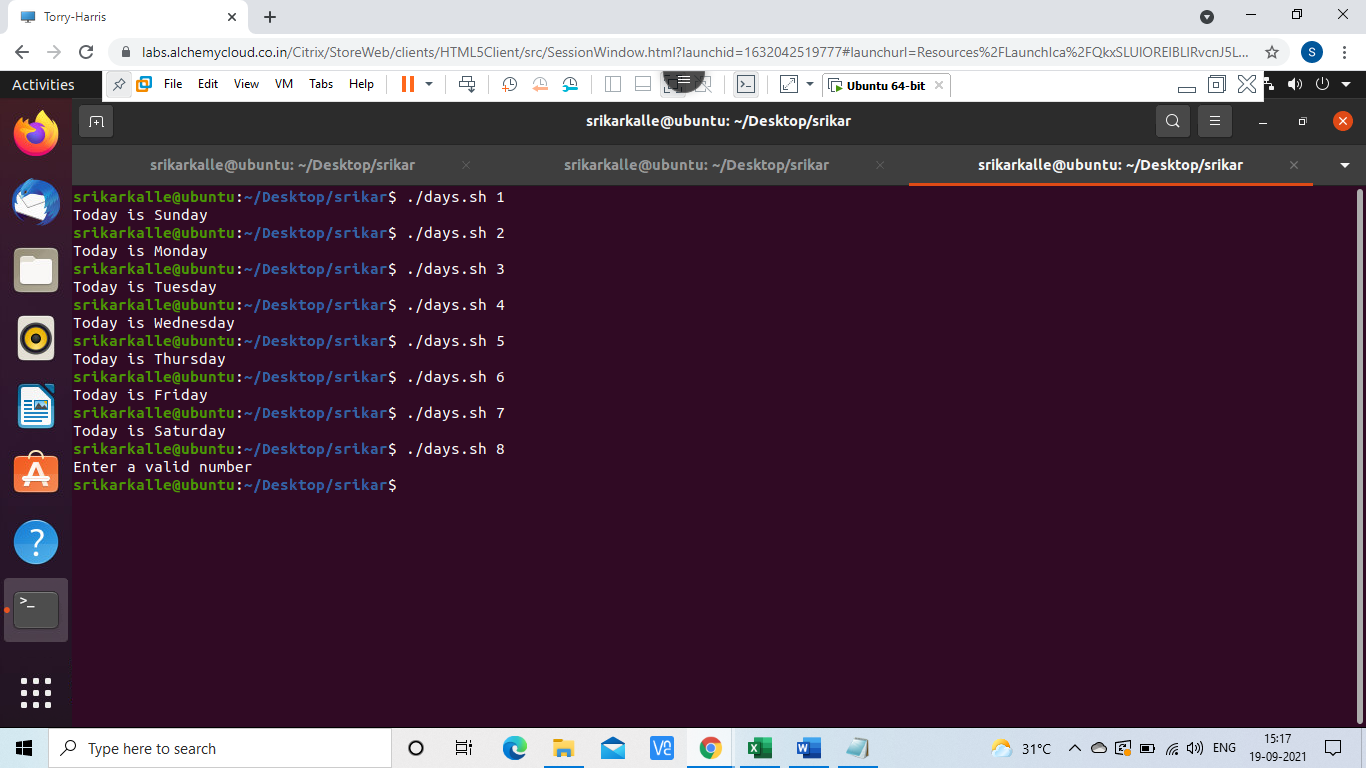
If elif else:

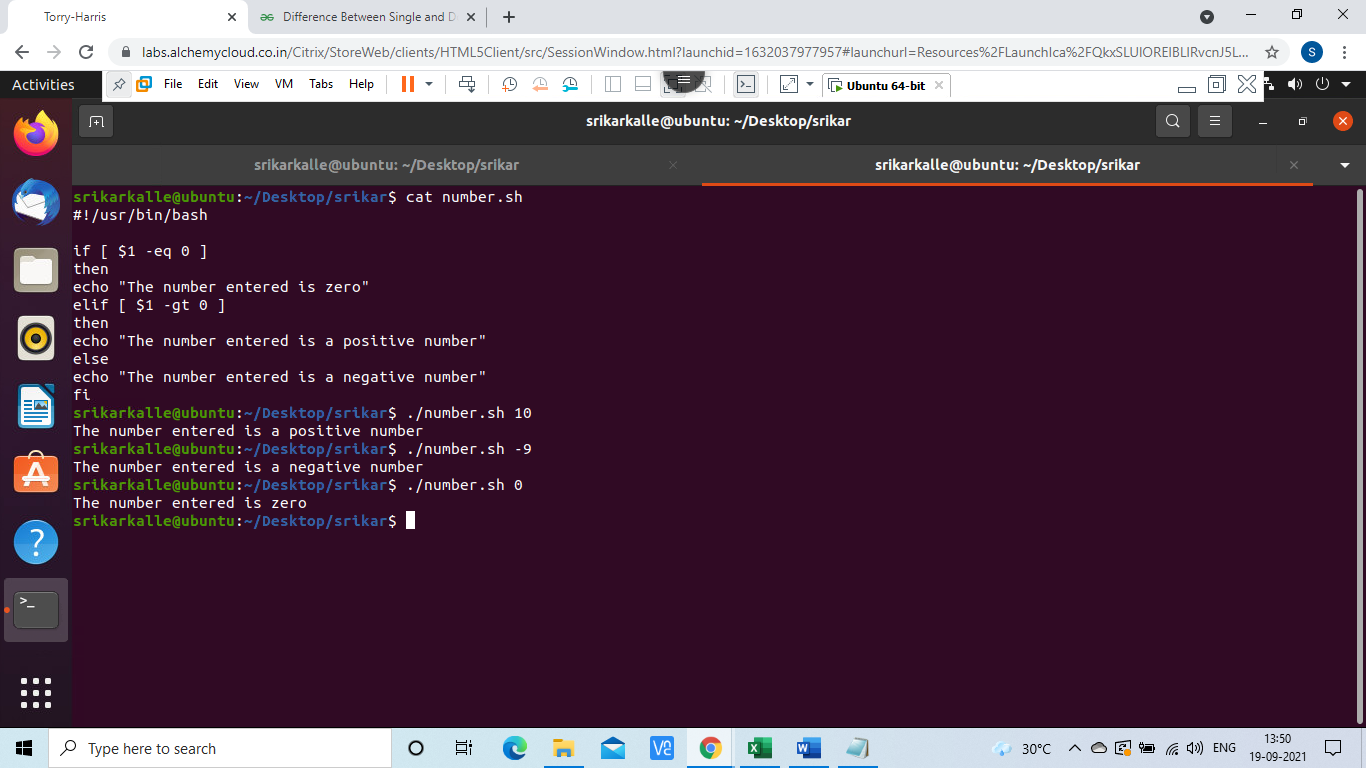
Write a simple program to check days in a week.

Program:

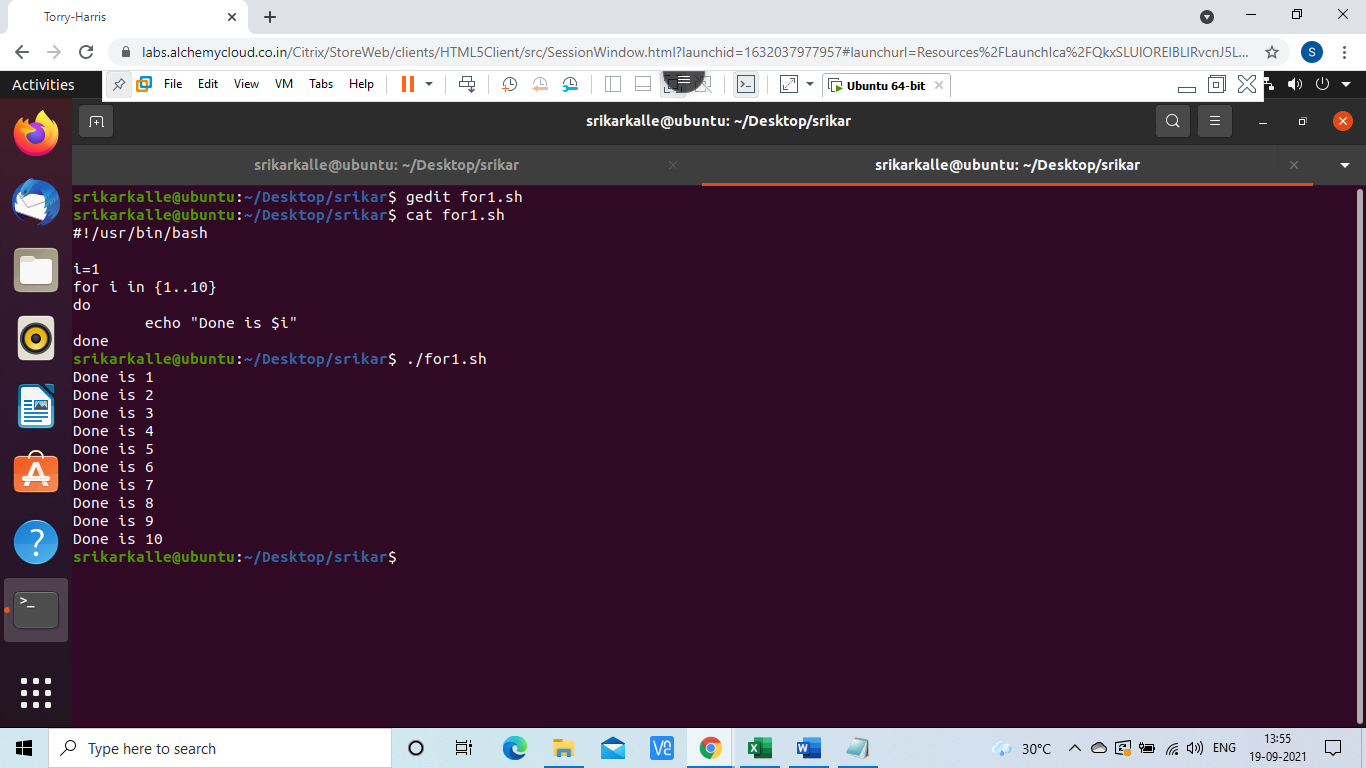


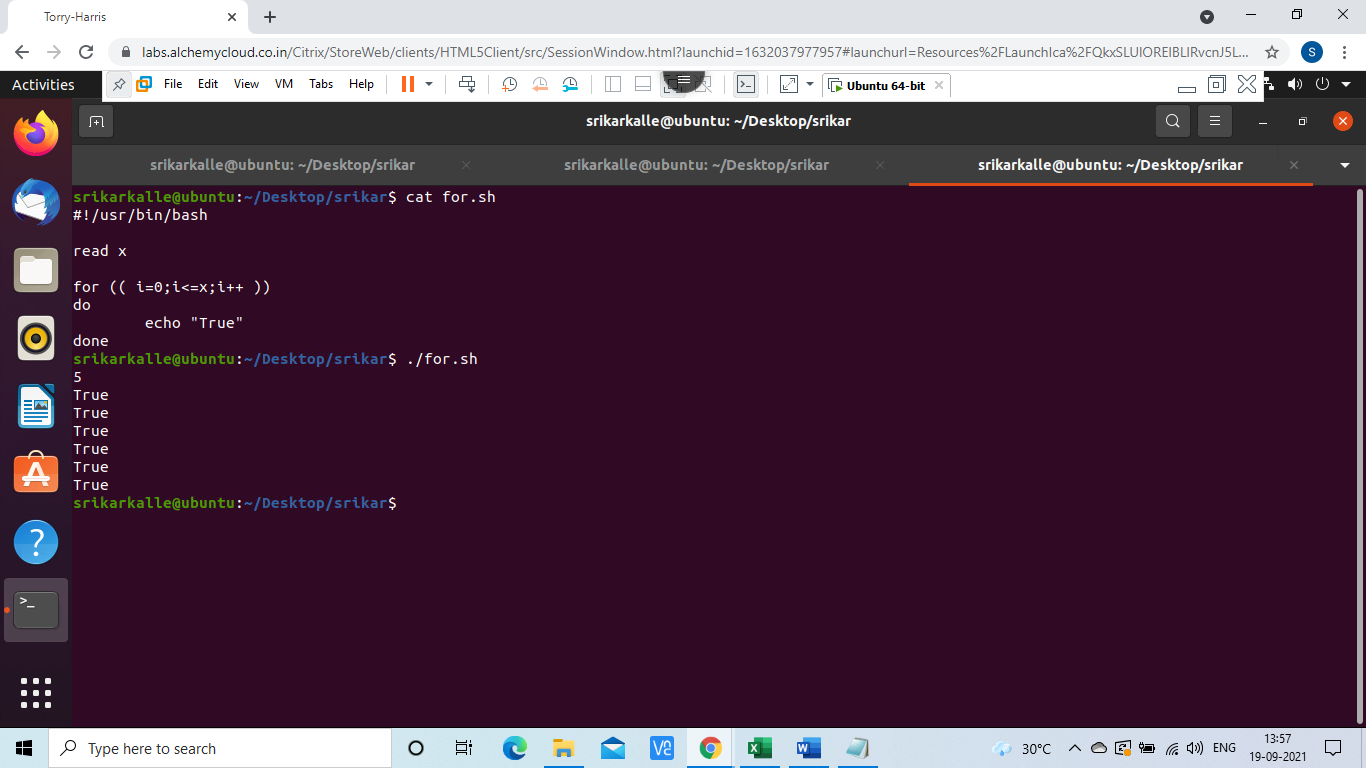
Output:



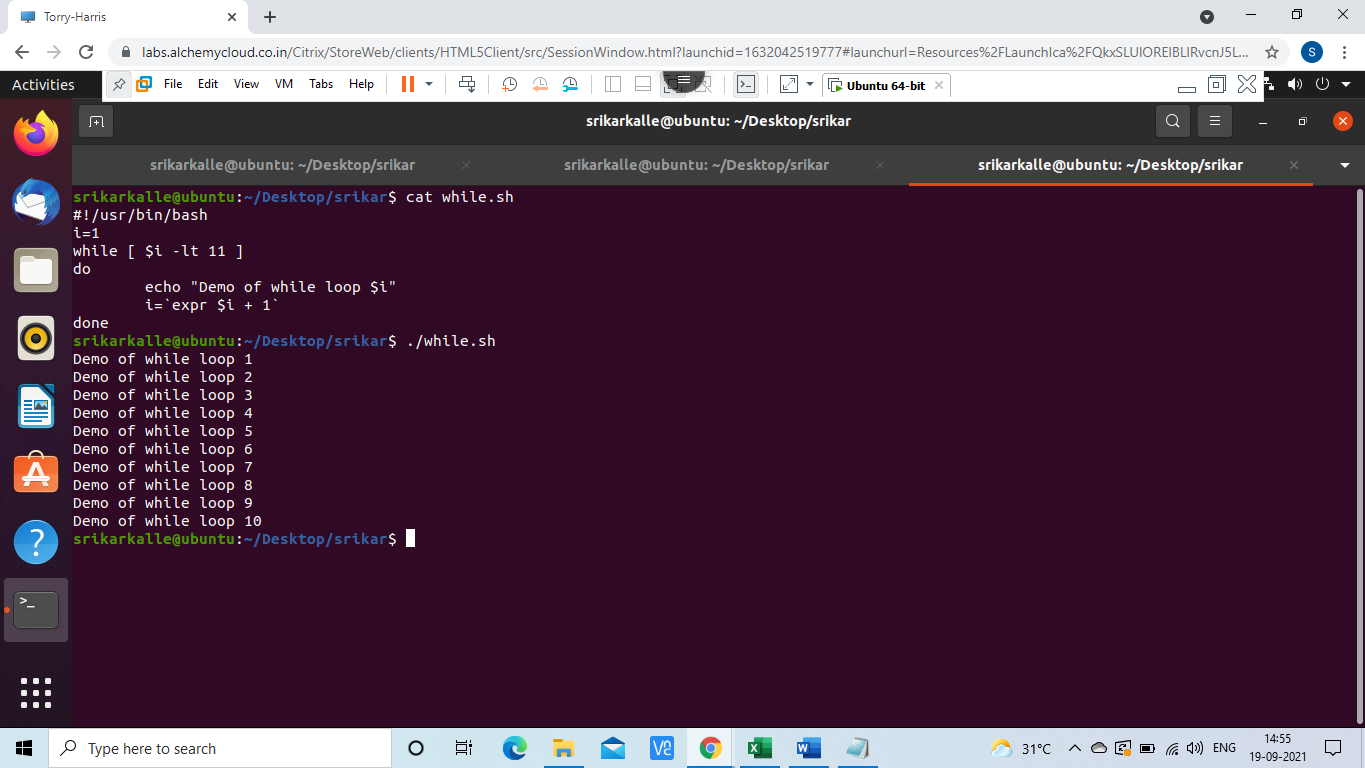


For loop: The for loop operate on lists of items. It repeats a set of commands for every item in a list.

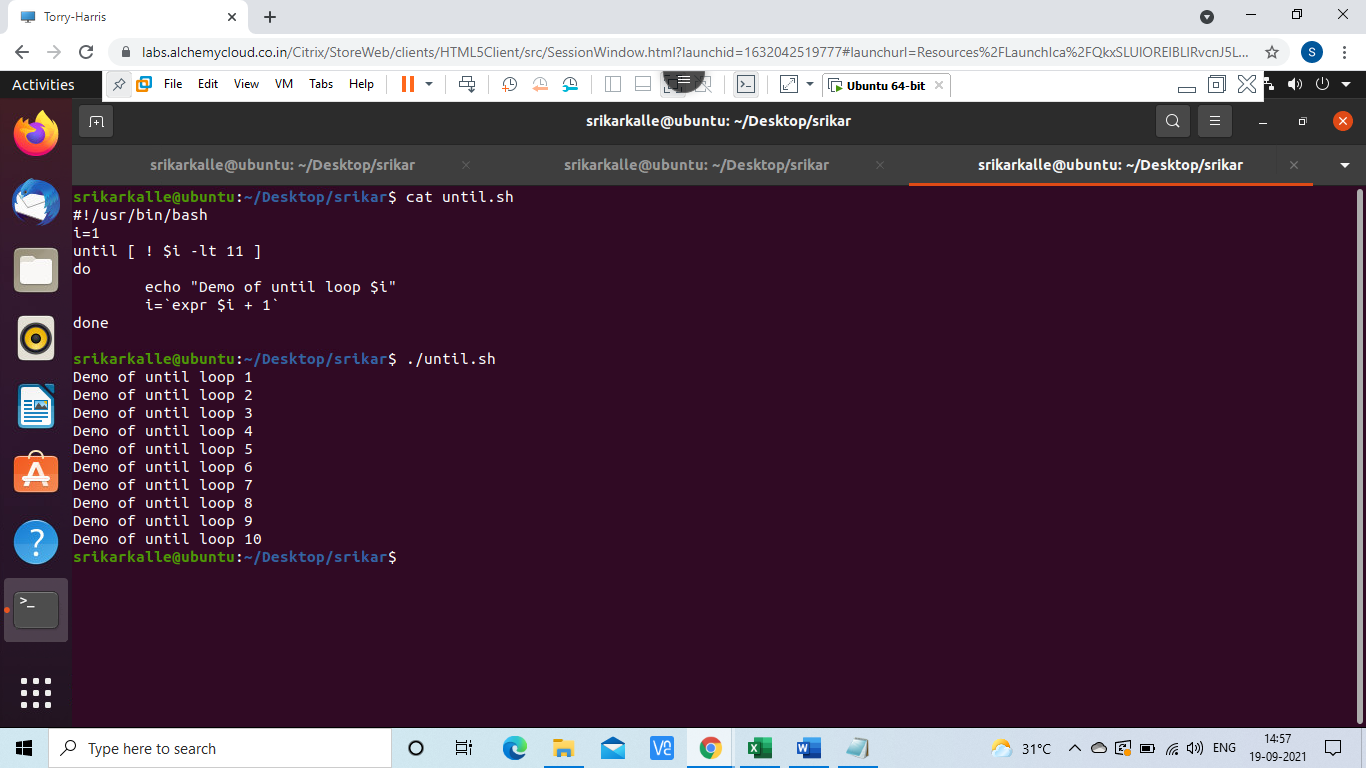




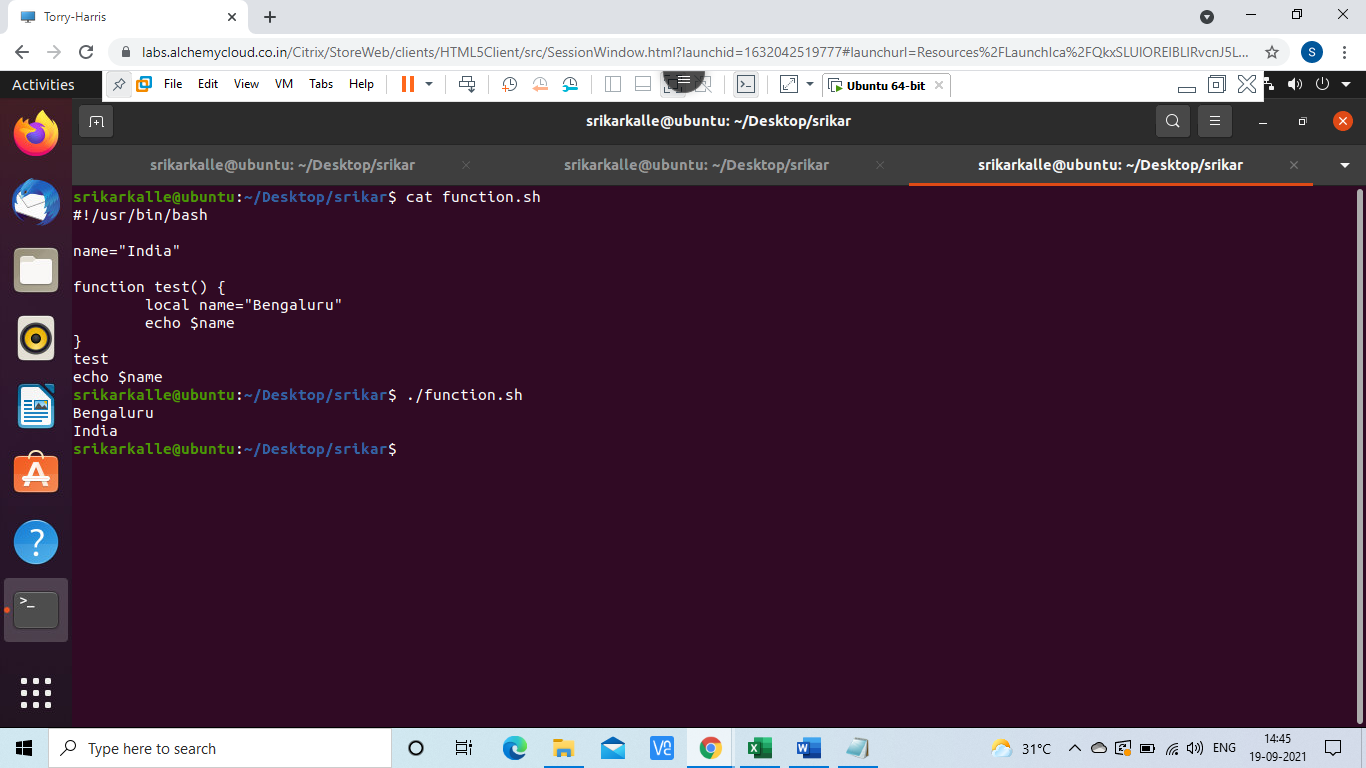
While loop: The while loop enables you to execute a set of commands repeatedly until some condition occurs. It is usually used when you need to manipulate the value of a variable repeatedly.



Until: The while loop is perfect for a situation where you need to execute a set of commands while some condition is true. Sometimes you need to execute a set of commands until a condition is true.



Function: Functions enable you to break down the overall functionality of a script into smaller, logical subsections, which can then be called upon to perform their individual tasks when needed.

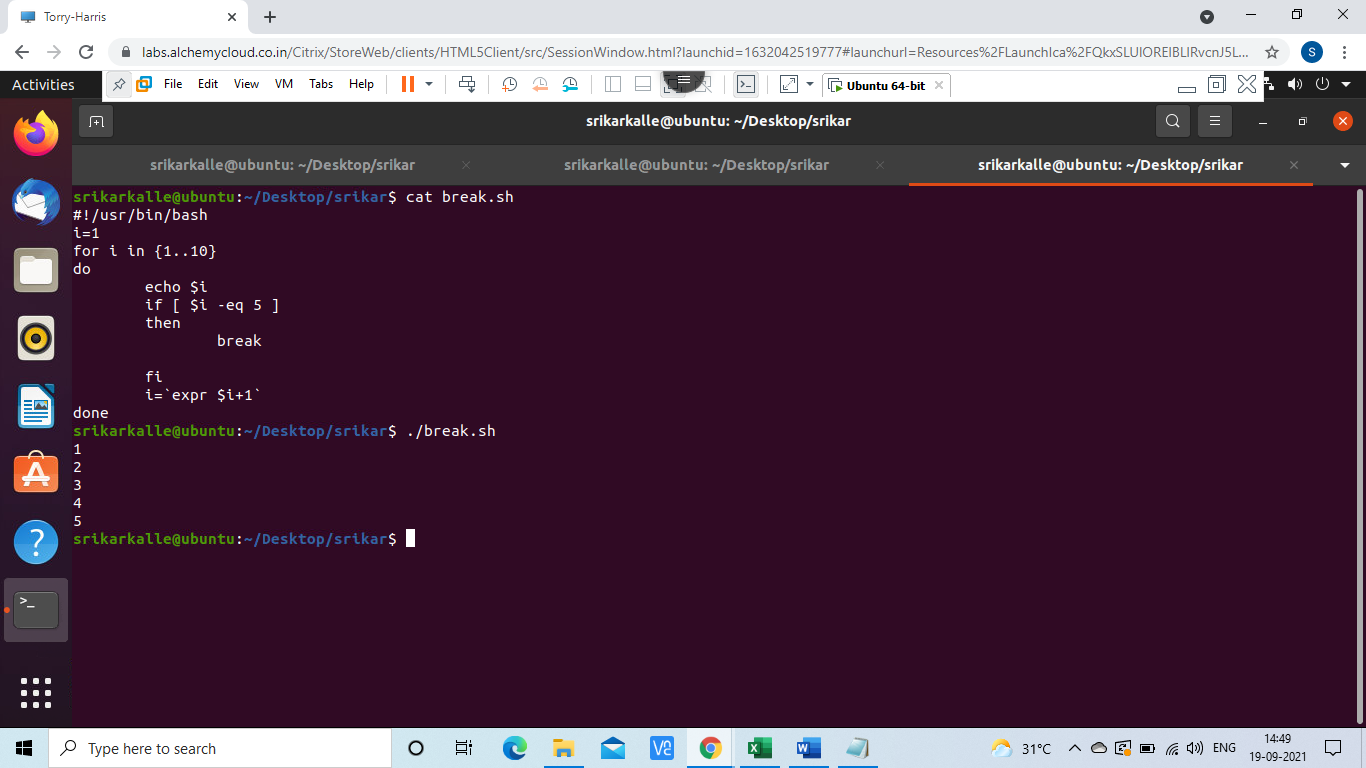


Break & continue:

The continue statement is similar to the break command, except that it causes the current iteration of the loop to exit, rather than the entire loop.

This statement is useful when an error has occurred but you want to try to execute the next iteration of the loop.

Ex: Break statement



Ex: Continue statement

