Hands-on Lab: Advanced Bash Scripting



Welcome to this hands-on lab, where you will take your Bash scripting chops to the next level. The skills you practice here will serve as logical building blocks for an endless variety of scripting applications. These concepts will also be essential for demonstrating your new skills in the Final Project for this course.

As you develop your Bash scripts, it's always recommended that you test your results at each stage to ensure your logic is behaving as expected. Think of each stage as a building block of your final script that accomplishes an easily digestible sub-task.

Learning Objectives

After completing this lab, you will be able to:

- · Run sets of commands using conditional statements
- Create true / false comparisons with logical operators
- Use arithmetic operators to perform basic mathematical calculations
- Use list-like arrays to store and access data
- Execute repetitive tasks with for loops

Exercise 1 - Using conditional statements and logical operators

In this exercise, you will create a simple Bash script containing a conditional statement to handle the following tasks:

- Prompt the user for a Yes or No response to a question
- · Print a response based on the user's answer

1.1. Create a new Bash script

Create a Bash script file and make it executable.

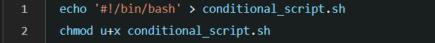
▼ Click here for Hint

Use the <u>echo</u> command to redirect a **shebang** to a new Bash script.

Alternatively, open a new text file using your favourite text editor and add a **shebang** to it. Remember to make your new script executable.

▼ Click here for Solution

Here's a solution using only the command line:



₾ = >_

1.2. Query the user and store their response Now get your script to: 1. Ask the user a binary "yes or no" question of your choosing 2. Store the user's answer in a variable. ▼ Click here for Hint Use the echo and read commands. ▼ Click here for Solution Your Bash script should now look something like this: 1 #I/bin/bash 2 3 echo 'Are you enjoying this course so far?' 4 echo -n "Enter \"y\" for yes, \"n\" for no." 5 read response

1.3. Use a conditional block to select a response for the user

Finally, use a conditional block to print a message to the user based on their response to your query.

Tip: It's best practice to also handle the case where the response doesn't match any of the allowable responses.

▼ Click here for Hint

Use a conditional <u>if elif else fi</u> block that uses a logical operator to compare the user's response to the available response options and prints an appropriate message in each case.

▼ Click here for Solution

Now your Bash script should be similar to the following:

```
4 ■
      echo 'Are you enjoying this course so far?'
      echo -n "Enter \"y\" for yes, \"n\" for no"
4
      read response
      if [ "$response" = "y" ]
          echo "I'm pleased to hear you are enjoying the course!"
          echo "Your feedback regarding what you have been enjoying would be most welcome
      elif [ "$response" = "n" ]
10
12
         echo "I'm sorry to hear you are not enjoying the course."
         echo "Your feedback regarding what we can do to improve the learning experience"
         echo "for this course would be greatly appreciated!"
14
         echo "Your response must be either 'y' or 'n'."
         echo "Please re-run the script to try again."
18
```

Exercise 2 - Performing basic mathematical calculations and numerical logical comparisons

In this exercise, you will create a Bash script that performs basic arithmetic calculations on two integers entered by the user. You will also use logical comparisons to determine which calculation leads to the greatest result.

2.1. Create a Bash script

Create an executable Bash script that prompts the user for two integers, then stores and prints both the sum and product of the two integers.

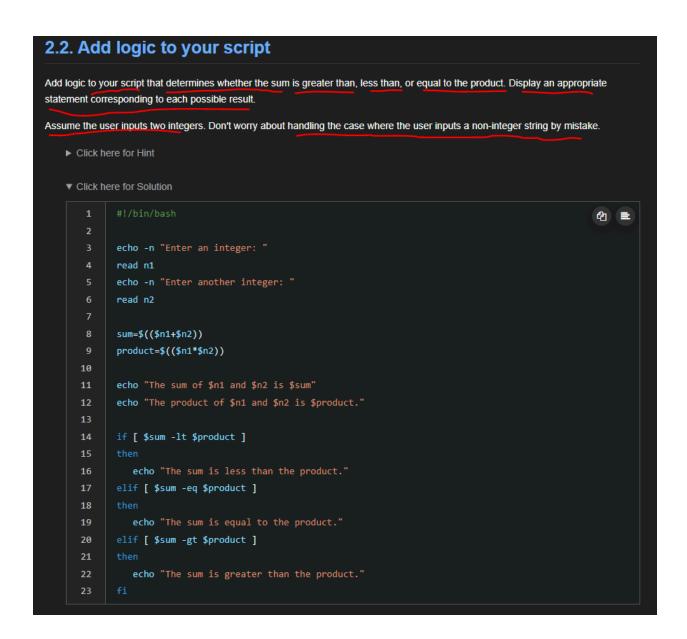
▼ Click here for Hint

Use the echo and read commands as in the previous exercise.

Recall the notation for arithmetic calculations.

▼ Click here for Solution

```
1 #!/bin/bash
2
3 echo -n "Enter an integer: "
4 read n1
5 echo -n "Enter another integer: "
6 read n2
7
8 sum=$(($n1+$n2))
9 product=$(($n1*$n2))
10
11 echo "The sum of $n1 and $n2 is $sum"
12 echo "The product of $n1 and $n2 is $product."
```



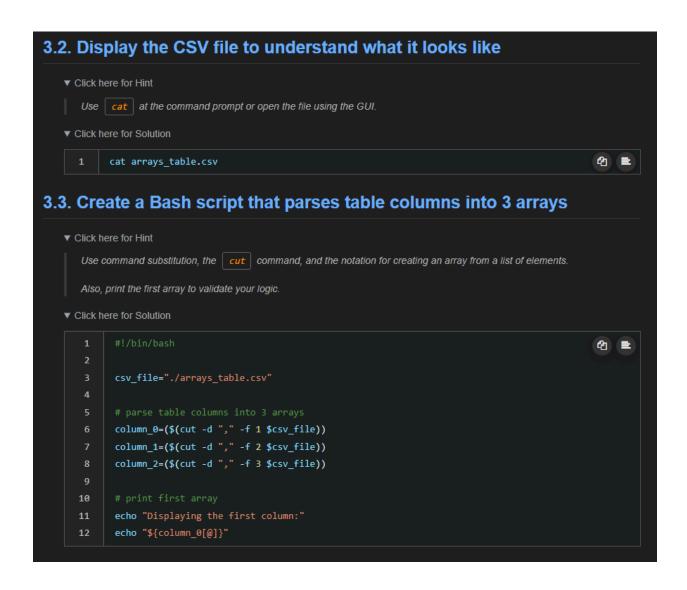
Exercise 3 - Using arrays for storing and accessing data within *for* loops

In this exercise, you will create a report based on a supplied dataset using the CSV format. You will extract the columns of the dataset into separate arrays and create a new column using arithmetic and array logic. Finally, you will combine the dataset with the new column and save the resulting report as a CSV file.

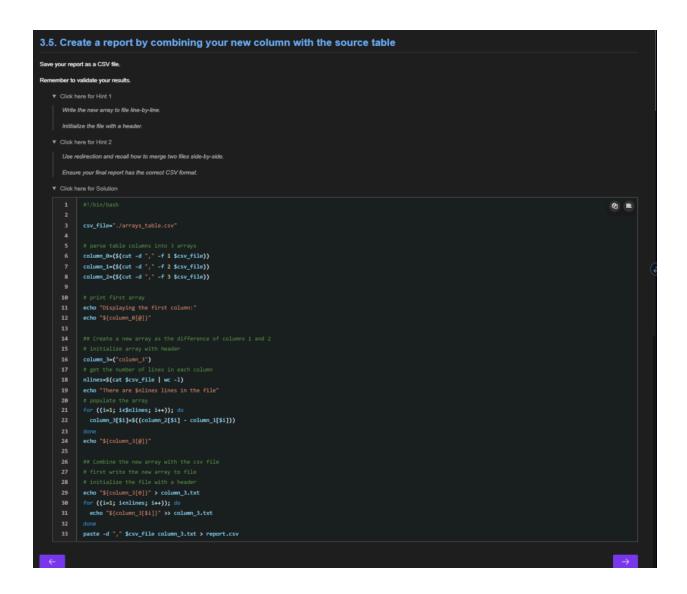
3.1. Download a CSV file to your current working directory

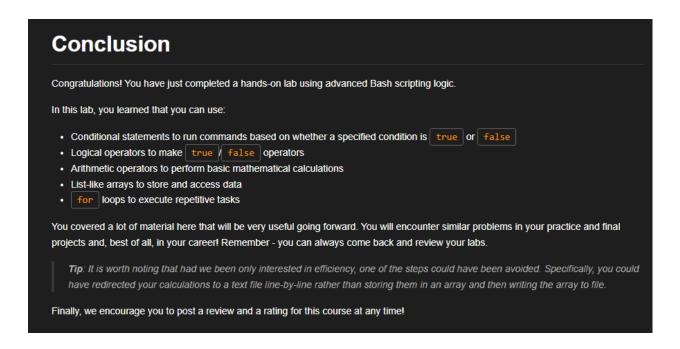
The file, arrays_table.csv, is located at the following url:

- 1. https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-LX0117EN-SkillsNetwork/labs/M3/L
- ▼ Click here for Hint
 - Use the wget command.
- ▼ Click here for Solution
 - csv_file="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-LX0117EN-SkillsNetwork/labs/M3/L2/arrays_table.csv"
 - 2. wget \$csv_file



3.4. Create a new array as the difference of the third and second columns. Initialize your new array with a header (a column name), and remember to validate your results. ▼ Click here for Hint 1 Use a loop to populate your array. Determine the number of elements you need and incorporate within your loop statement. Print both the number of elements and the contents of your new array to validate your logic. ▼ Click here for Hint 2 Recall the for loop notation when you know the number of iterations needed, and that array indexing starts at 0. To get the number of lines, use command subsitution on a pipeline that uses the cat and wc commands as filters and store the result in a variable. ▼ Click here for Solution **2**1 **≡** csv_file="./arrays_table.csv" column_0=(\$(cut -d "," -f 1 \$csv_file)) column_1=(\$(cut -d "," -f 2 \$csv_file)) column_2=(\$(cut -d "," -f 3 \$csv_file)) 11 echo "Displaying the first column:" 16 column_3=("column_3") nlines=\$(cat \$csv_file | wc -1) echo "There are \$nlines lines in the file" for ((i=1; i<\$nlines; i++)); do column_3[\$i]=\$((column_2[\$i] - column_1[\$i])) echo "\${column_3[@]}"





Conclusion

Congratulations! You have just completed a hands-on lab using advanced Bash scripting logic.

In this lab, you learned that you can use:

- Conditional statements to run commands based on whether a specified condition is true or false
- Logical operators to make true / false operators
- Arithmetic operators to perform basic mathematical calculations
- List-like arrays to store and access data
- for loops to execute repetitive tasks

You covered a lot of material here that will be very useful going forward. You will encounter similar problems in your practice and final projects and, best of all, in your career! Remember - you can always come back and review your labs.

Tip: It is worth noting that had we been only interested in efficiency, one of the steps could have been avoided. Specifically, you could have redirected your calculations to a

text file line-by-line rather than storing them in an array and then writing the array to file.

Finally, we encourage you to post a review and a rating for this course at any time!