BATCH	BATCH 85
LESSON	SHELL SCRIPTING
DATE	07.06.2022
SUBJECT	SHELL SCRIPTING

techproeducation

techproeducation

techproeducation

techproeducation

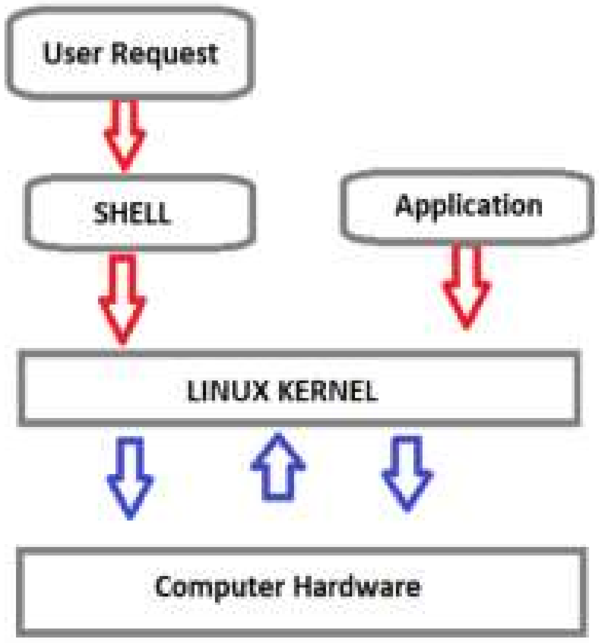
techproedu

Shell Scripting

WHAT IS SHELL?

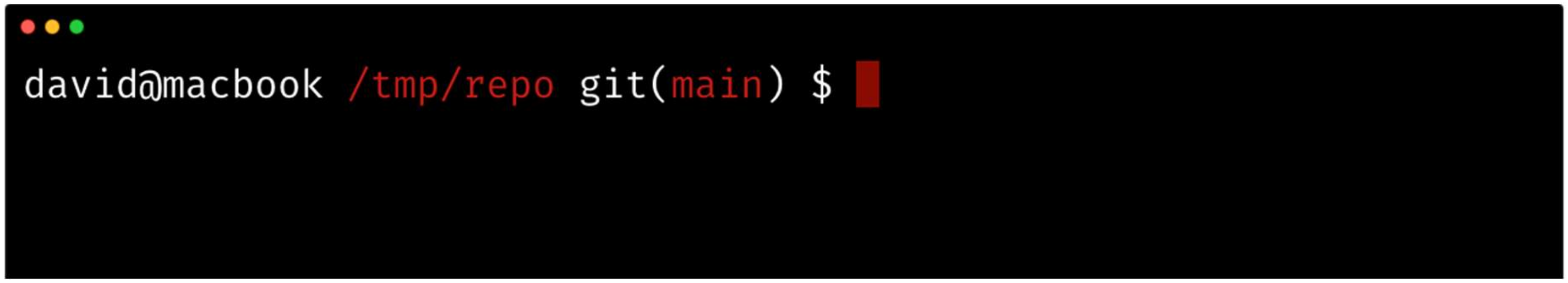
Shell is a program that receives the user’s commands and gives them to the operating system to process and displays the output.

The standard Linux Shell is both a command-line interpreter and a programming language.



The BASH Prompt

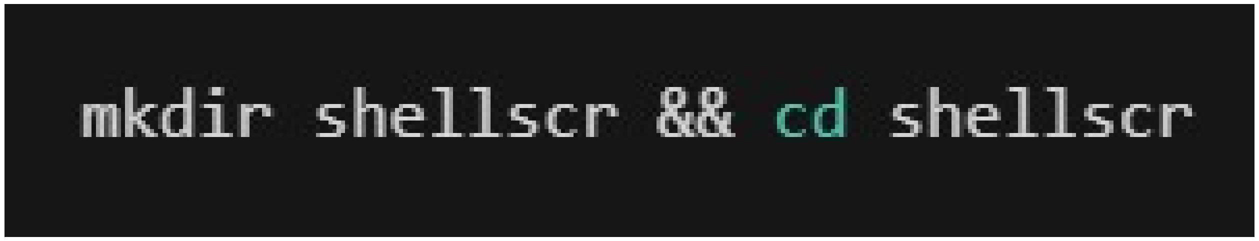
A prompt is text or symbols used to represent the system's readiness to perform the next command. A prompt may also be a text representation of where the user is currently.



```
david@macbook /tmp/repo git(main) $
```

SHELL SCRIPTING BASICS

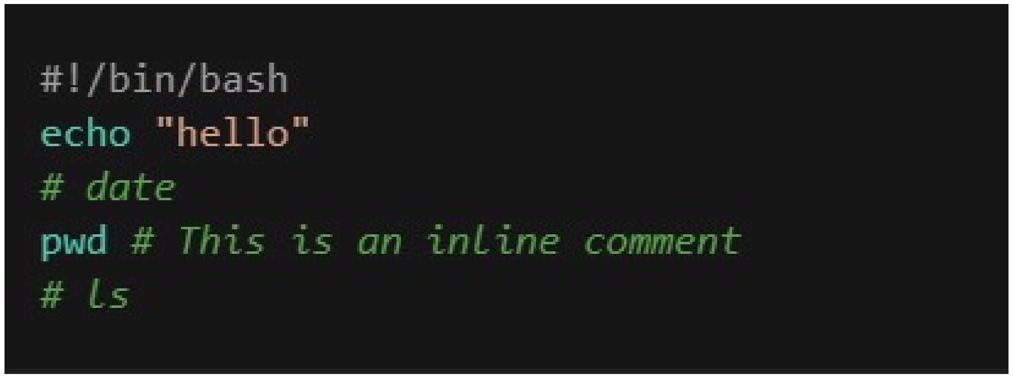
- To create a folder, we use **mkdir**
- To change directory, we use **cd**



```
mkdir shellscr && cd shellscr
```

Shell Comments

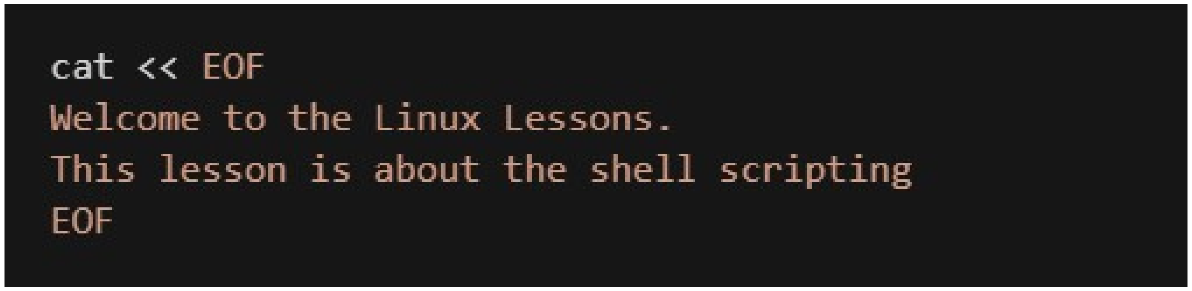
- - Bash ignores everything written on the line after the hash mark (#). The only exception to this rule is the first line of the script that starts with the #! characters.
  - Comments can be added at the beginning on the line or inline with other code.



```
#!/bin/bash
echo "hello"
# date
pwd # This is an inline comment
# ls
```

Heredoc Syntax

- - A heredoc consists of the << (redirection operator), followed by a delimiter token. After the delimiter token, lines of string can be defined to form the content. Finally, the delimiter token is placed at the end to serve as the termination. The delimiter token can be any value as long as it is unique enough that it won’t appear within the content



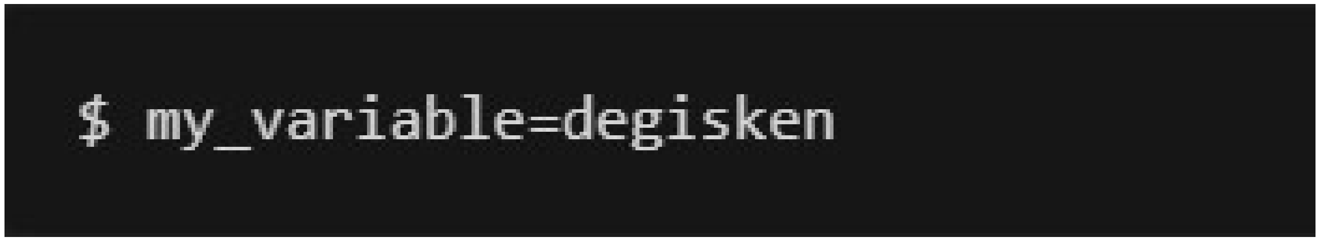
```
cat << EOF
Welcome to the Linux Lessons.
This lesson is about the shell scripting
EOF
```

VARIABLES

- A variable is a value that can vary or change.
- A variable always has a **$** sign before it's name.
- A variable name may only contain alphanumeric or underscores
- A variable is case sensitive.

VARIABLES

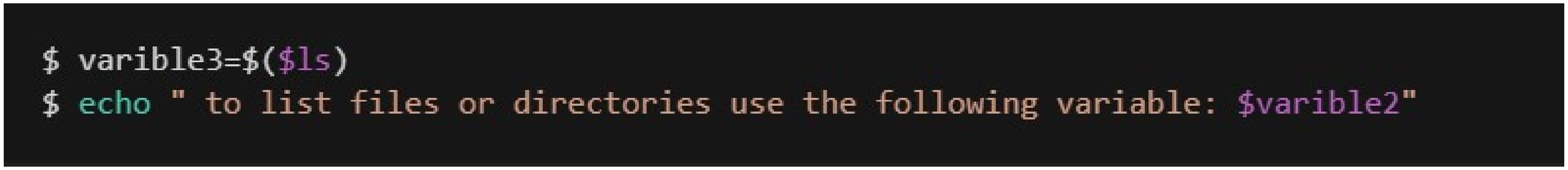
- To set the value for a variable



```
$ my_variable=degisken
```

VARIABLES

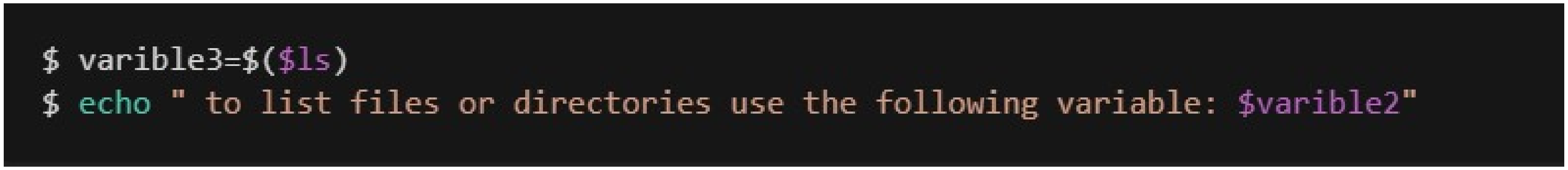
- We can also use variables to store the result of another command and print it.



```
$ varible3=$($ls)
$ echo " to list files or directories use the following variable: $varible2"
```

VARIABLES

- We can also use variables to store the result of another command and print it.



```
$ varible3=$($ls)
$ echo " to list files or directories use the following variable: $varible2"
```

* What we did above is we set a variable for the output of variable3

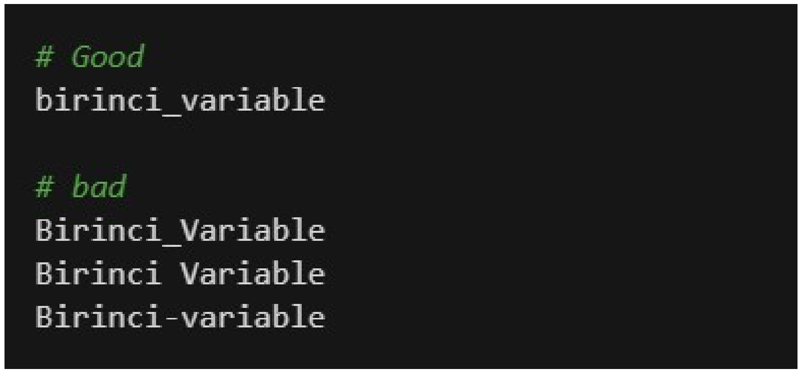
-

* Then, we print the its output with echo

VARIABLES

- **Best Practices**
-

  * Variable names must be in lower-case with underscores to separate words.



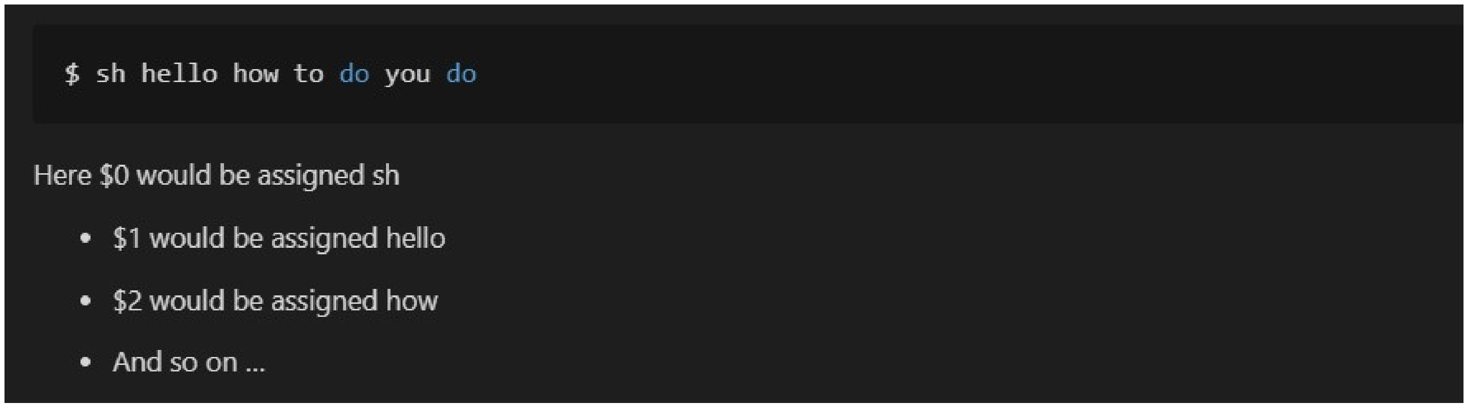
```
# Good
birinci_variable

# bad
Birinci_Variable
Birinci Variable
Birinci-variable
```

COMMAND LINE ARGUMENTS

- We can replace the hardcoded names within our script with the built-in variable.
- The command run by user is spitted to the built-in variables.
- - $0, $1, $2, $3 ... etc. are built in variables. These are called as **command line arguments**.

COMMAND LINE ARGUMENTS

- In our example:



```
$ sh hello how to do you do
```

Here $0 would be assigned sh

- $1 would be assigned hello
- $2 would be assigned how
- And so on ...

COMMAND LINE ARGUMENTS

- **## Best Practices**
  - It is a best practice to
- assign a variable to a meaningful variable name
- Design your script to be re-usable
- Script should not require to be edited before running
- Use command line arguments to pass inputs.

COMMAND LINE ARGUMENTS

- We use best practice since:
- We want to design our script to be re-usable
- Our script should not require to be edited before running
- We also use command line arguments to pass inputs.

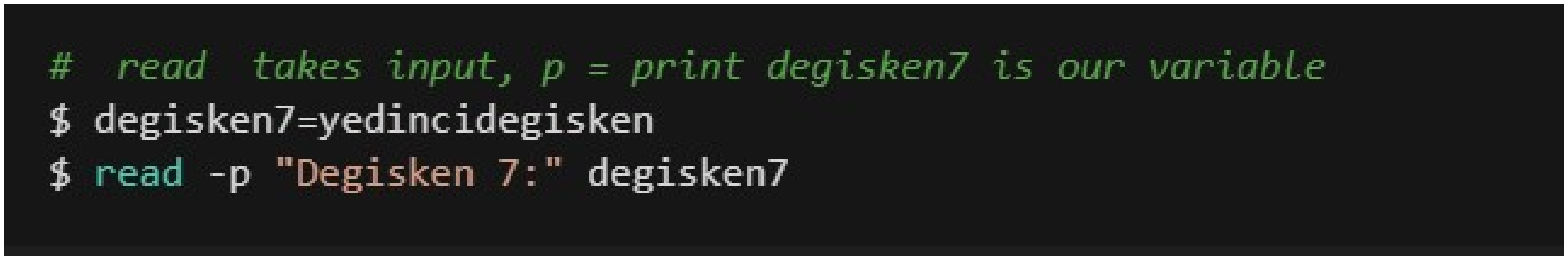
READ INPUTS

- Lets take a look at prompting for inputs in a shell script

- To prompt the user for input, use the **read`** statement followed by adding a **prompt** with **-p** option.

READ INPUTS

- Read takes an input from the user.
- If we only use read , nothing is displayed on the console before we enter the input. To avoid this, we use read with -p statement.



```
#  read  takes input, p = print degisken7 is our variable
$ degisken7=yedincidegisken
$ read -p "Degisken 7:" degisken7
```

READ INPUTS

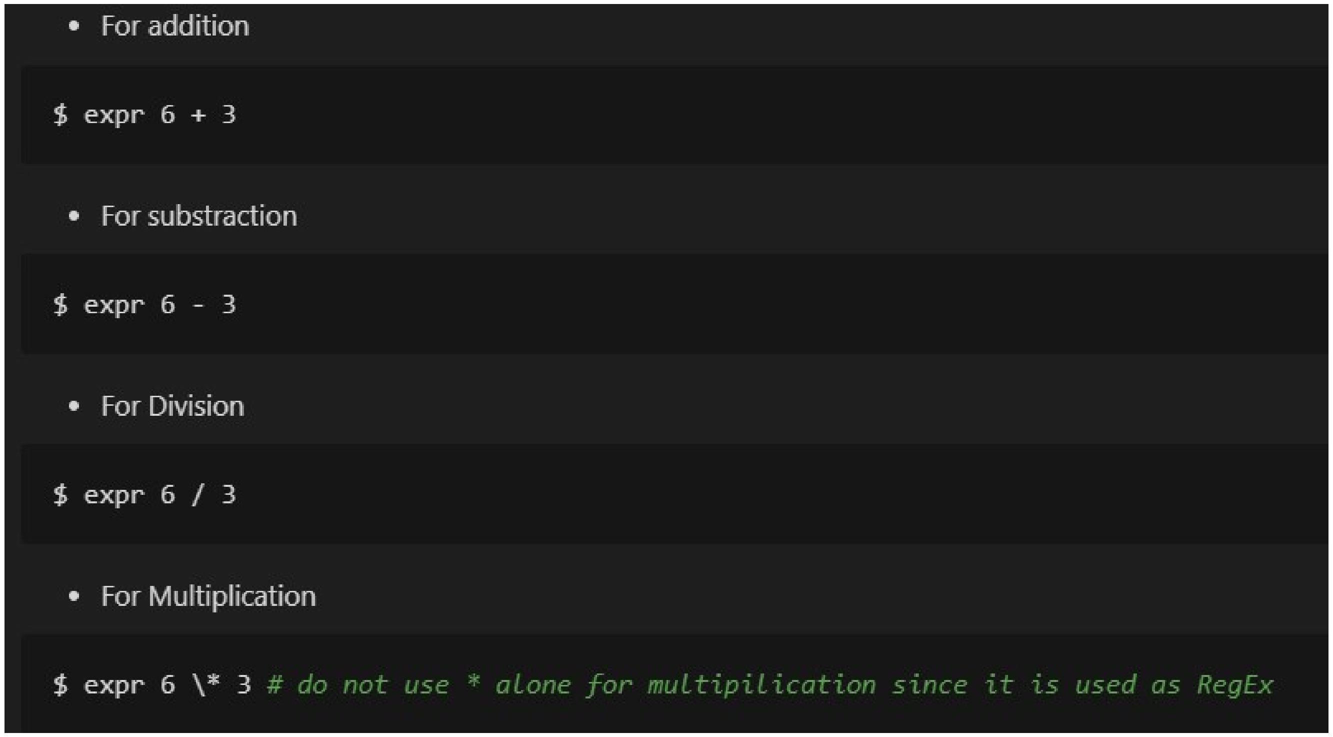
- When to use Command Line Arguments and Read Inputs ?
- When you want manual intervention on your program, use read.
- When you do not want manual intervention on your program, don not use read. It might be the case for most of the time since your program will communicate with other programs without manual intervention.
- You can also use read and CLA at the same time.

ARITHMETHIC OPERATIONS

- Let’s take a look at "Arithmetic Operations"
- There are different methods with which we can perform arithmetic operations. These are:
- expr
- Double Parantheses(())
- bc

ARITHMETHIC OPERATIONS

- Expr



- For addition

```
$ expr 6 + 3
```

- For substraction

```
$ expr 6 - 3
```

- For Division

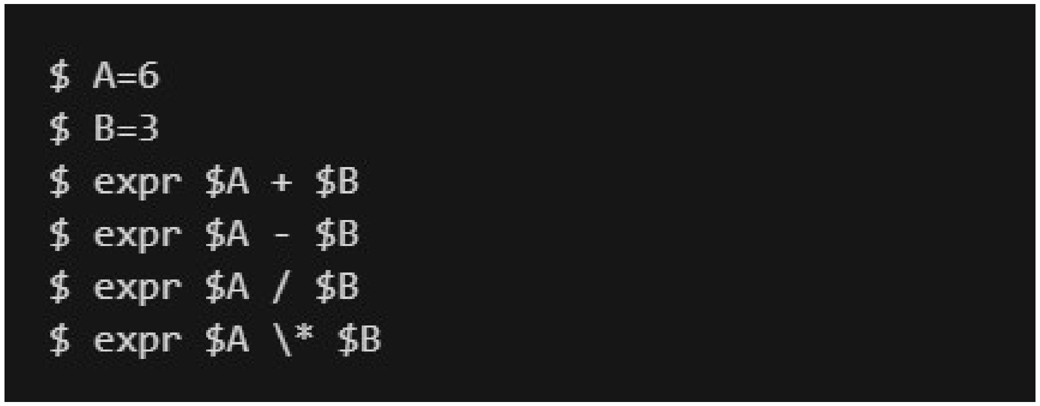
```
$ expr 6 / 3
```

- For Multiplication

```
$ expr 6 \* 3 # do not use * alone for multipilication since it is used as RegEx
```

ARITHMETHIC OPERATIONS

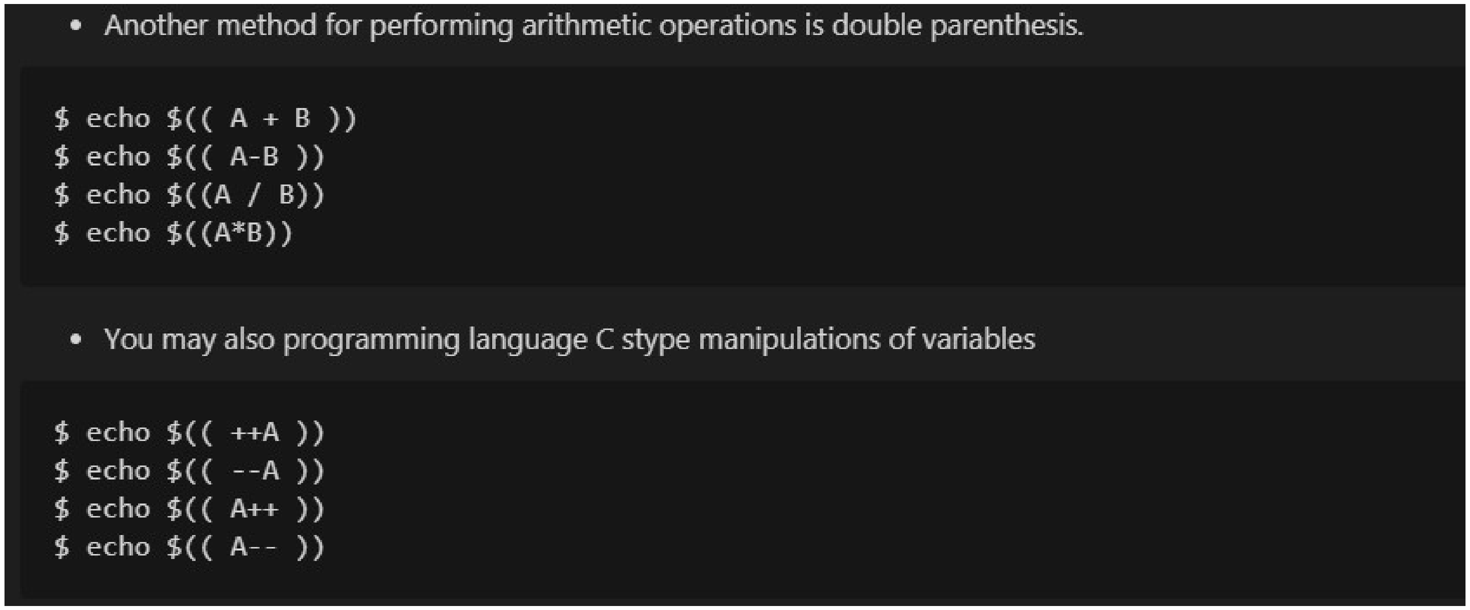
- Expr can be used variable substition as well.



```
$ A=6
$ B=3
$ expr $A + $B
$ expr $A - $B
$ expr $A / $B
$ expr $A \* $B
```

ARITHMETHIC OPERATIONS

- Double Paranthesis



- Another method for performing arithmetic operations is double parenthesis.

```
$ echo $(( A + B ))
$ echo $(( A-B ))
$ echo $((A / B))
$ echo $((A*B))
```

- You may also programming language C stype manipulations of variables

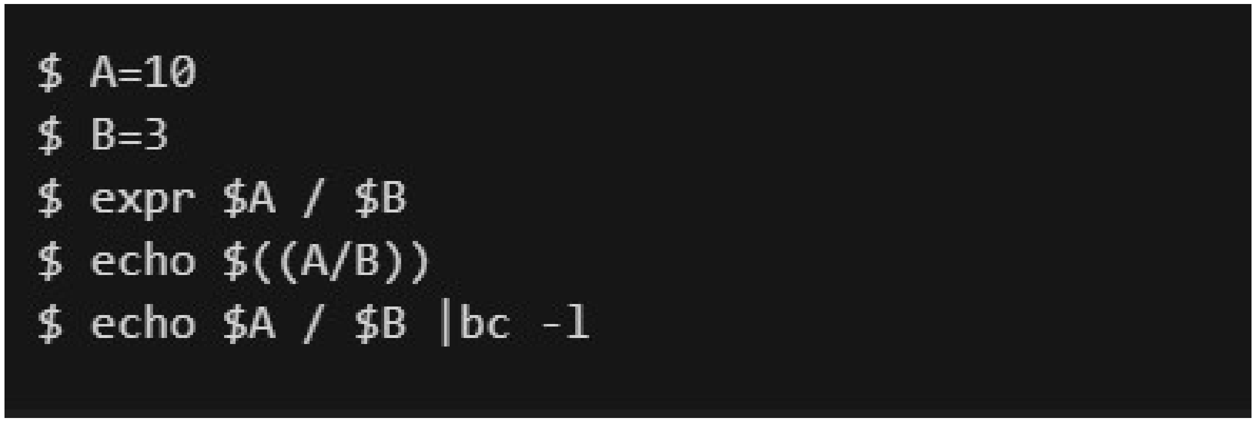
```
$ echo $(( ++A ))
$ echo $(( --A ))
$ echo $(( A++ ))
$ echo $(( A-- ))
```

ARITHMETHIC OPERATIONS

- **Bc**
- expr and "double parentheses" only return decimal output, they do not support floating values.
- For this we use another utility called bc, it referred to as Basic Calculator.
- It works in an interactive mode and also you can input to another command as well

ARITHMETHIC OPERATIONS

- **Bc**



```
$ A=10
$ B=3
$ expr $A / $B
$ echo $((A/B))
$ echo $A / $B |bc -l
```

ARITHMETHIC OPERATIONS

Please complete the exercises from your lecture notes accordingly…