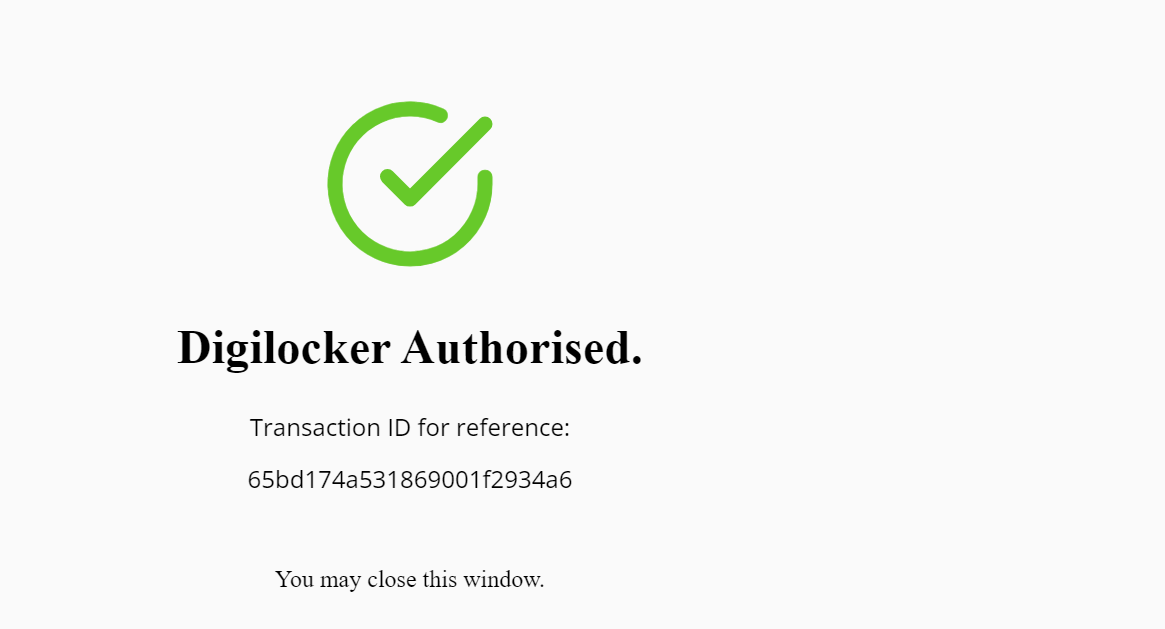
****

Shell Scriptitng

A shell is a special user program that provides an interface for the user to use operating system services. Shell accepts human-readable commands from users and converts them into kernel understandable language. It is a command language interpreter that executes commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or starts the terminal.

### **Why do we need shell scripts?**

There are many reasons to write shell scripts:

* To avoid repetitive work and automation
* System admins use shell scripting for routine backups.
* System monitoring
* Adding new functionality to the shell etc.

### **Some Advantages of shell scripts**

* The command and syntax are exactly the same as those directly entered in the command line, so programmers do not need to switch to entirely different syntax
* Writing shell scripts are much quicker
* Quick start
* Interactive debugging etc.

A unique shell program inerprets user commands wjich are directly by user or can be call by shell scripts

they are interpreted by your system not compilerd to use scripts

Different type of shell

1)learn bash scripting(most commonly used)

2)sh

3)ksh (korn)

4)tsh

5)fish

6)zsh (extended version of bash)

check which shell you are using

command -> echo $0

Lehar Agrawal@DESKTOP-OQPL5PT MINGW64 ~

$ echo $0

/usr/bin/bash

other shells are present

command -> cat /etc/shells

Shell Scripting consist of set of commands to perform a task.

All the cmds execute sequestially

SOme task like file manipulation, program execution, user interaction, automation of task etc can be done.

first scripts created

Lehar Agrawal@DESKTOP-OQPL5PT MINGW64 ~/Desktop/myscripts

$ vi 01\_basic.sh

Lehar Agrawal@DESKTOP-OQPL5PT MINGW64 ~/Desktop/myscripts

$ cat 01\_basic.sh

#!/bin/bash //meaning it;s called shebang

when we use this to excecute the script tell that which shell you want to use bash shell dedault hi bash hai

echo "hii buddy"

// to print the message

//to check permission

Lehar Agrawal@DESKTOP-OQPL5PT MINGW64 ~/Desktop/myscripts

$ ls -ltr

total 1

-rwxr-xr-x 1 Lehar Agrawal 197121 30 Feb 14 14:42 01\_basic.sh\*

how to run an script

Lehar Agrawal@DESKTOP-OQPL5PT MINGW64 ~/Desktop/myscripts

$ ./01\_basic.sh

hii buddy

Lehar Agrawal@DESKTOP-OQPL5PT MINGW64 ~/Desktop/myscripts

$ bash 01\_basic.sh

hii buddy

/\*

./script.sh

/path/script.sh //if you r not in the same path

bash script.sh

ctrl+c //to terminate

ctrl+z //to stop

\*/

Lehar Agrawal@DESKTOP-OQPL5PT MINGW64 ~/Desktop/myscripts

$ cat 01\_comments.sh

echo "Checking comments"

#this is a single line comments

<<<comments

this

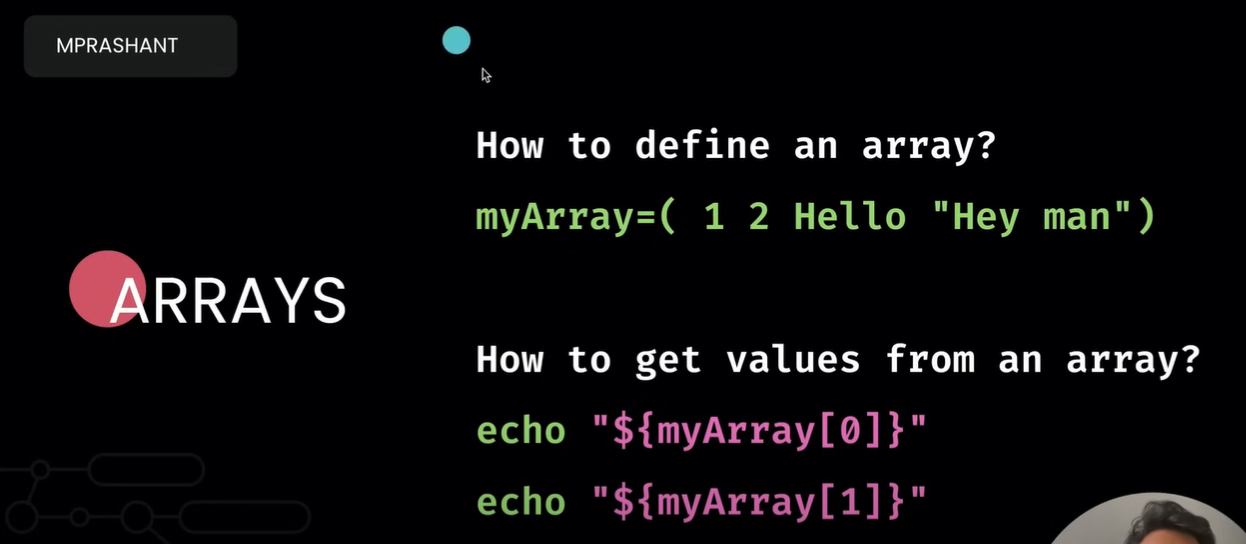
is

multi

line

comments

comment





Output

Lehar Agrawal@DESKTOP-OQPL5PT MINGW64 ~/Desktop/myscripts

$ ./variables.sh

Value in 3rd index Hello

//to display all the values in array are

#! /bin/bash

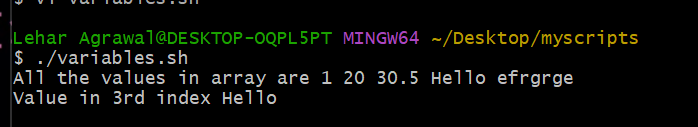
#Array

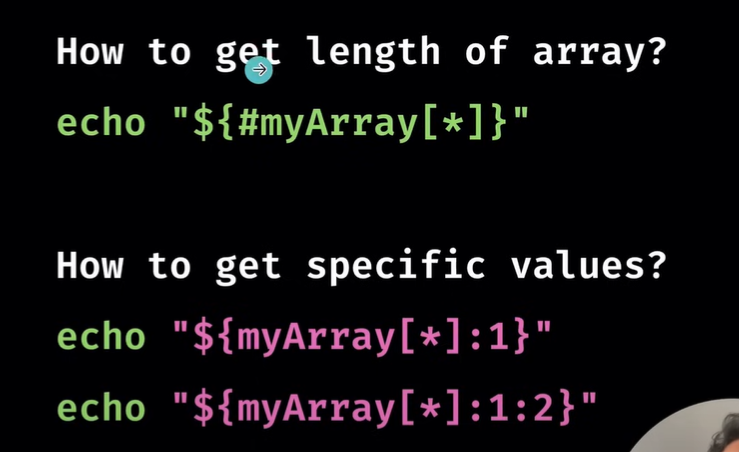
myArray=( 1 20 30.5 Hello "efrgrge" )

echo "All the values in array are ${myArray[\*]}"

echo "Value in 3rd index ${myArray[3]}"

output





#How to find no. of values in an array

echo "No. of values, length of an array is ${#myArray[\*]}"

-

-

Output

No. of values, length of an array is 5

Theory

**Basic shell scripting terms**

Before we delve further into shell scripting, knowing a few definitions may come in handy when exploring this topic:

* **Terminal:** A terminal is a program that establishes a connection with the server.
* **Shell:** This program interprets shell scripting commands from the terminal and runs on the server. This is the [command-line user interface](https://www.coursera.org/articles/command-line-interface) you choose and includes shells such as the Bourne shell, Korn shell, Bourne-Again shell, and C shell.
* **Script:** A [script](https://www.coursera.org/articles/scripting-language) is a short program that performs a specific task.
* **Command-line shell:** A command-line shell (also known as a command prompt) allows you to instruct your computer through textual commands.
* **Shell script:** A shell script is a script run through a command-line shell

When you create a shell script, you might use text editors like Nano or Vim and save the file with a “.sh” extension. A shell interpreter interprets and executes these scripts; you can run them from the command line whenever necessary.

**Read more:**[Information Technology (IT) Terms: A to Z Glossary](https://www.coursera.org/articles/it-terms)

**5 types of shells**

Different types of shells have different capabilities, so choose the type of shell that works best for your applications. Your shell choice will affect how you can execute programs and which system resources you can access. Some common types of shells you can choose from include:

| **Type of shell** | **Description** |
| --- | --- |
| Bourne shell | The Bourne shell, also known as ‘sh,’ was the original UNIX shell or command-line interpreter developed by Stephen Bourne at AT&T Bell Labs. While known for its high operation speed, this type of shell cannot reference previous commands and has limited built-in functionality. |
| C shell | The C shell, ‘csh,’ is a shell developed by Bill Joy at the University of California Berkeley. This shell uses a similar syntax as the C programming language and incorporates aliases into the available features. |
| Bash shell | Bash, short for “Bourne Again Shell,” is a shell created by Brian Fox as a combination of sh, csh, and ksh capabilities. Bash is the default shell on Linux and Mac OSX. This type of shell can recall previous commands and edit them. |
| Korn shell | The Korn shell, or ‘ksh,’ is a shell developed by David Korn in the 1980s. It combines features of both the Bourne and C shells, along with its own improvements, such as string and array manipulation. |
| Z shell | The Z shell, or ‘zsh,’ is a modern extension of the sh shell and offers extensive customization. Some notable features include plugins, function indexes, and theming support. |

**Capabilities of Shell Script**

Shell scripts are incredibly versatile, capable of handling file manipulation, database monitoring, text printing, and more. While the language of shells has conditional operations, looping functions, and command-line arguments, much of the strength of shell programs comes from the ability to call any program through shell scripting. This allows you to design complex programs suited to your needs.

//OPERaTORs

Expressions An expression can be: String comparison, Numeric comparison, File operators and Logical operators and it is represented by [expression]: String Comparisons:

---------------------------------

= compare if two strings are equal

!= compare if two strings are not equal

-n evaluate if string length is greater than zero

-z evaluate if string length is equal to zero

Examples:

[ s1 = s2 ] (true if s1 same as s2, else false)

[ s1 != s2 ] (true if s1 not same as s2, else false)

[ s1 ] (true if s1 is not empty, else false)

[ -n s1 ] (true if s1 has a length greater then 0, else false)

[ -z s2 ] (true if s2 has a length of 0, otherwise false)

Number Comparisons:

------------------------------------

-eq compare if two numbers are equal

-ge compare if one number is greater than or equal to a number

-le compare if one number is less than or equal to a number

-ne compare if two numbers are not equal

-gt compare if one number is greater than another number

-lt compare if one number is less than another number

Examples:

[ n1 -eq n2 ] (true if n1 same as n2, else false)

[ n1 -ge n2 ] (true if n1greater then or equal to n2, else false)

[ n1 -le n2 ] (true if n1 less then or equal to n2, else false)

[ n1 -ne n2 ] (true if n1 is not same as n2, else false)

[ n1 -gt n2 ] (true if n1 greater then n2, else false)

[ n1 -lt n2 ] (true if n1 less then n2, else false)

Note: When using square brackets, spaces around the operators (**==**, **-eq**, etc.) are important.

In summary, parentheses are used for command substitution, grouping commands, and arithmetic operations, while square brackets are commonly used for conditional tests, pattern matching, file tests, and string comparisons in shell scripting.

Besides **\n** for a newline character, here are some commonly used backslash escapes:

1. **\n**: Newline character.

echo -e "Line 1\nLine 2"

1. **\t**: Tab character.

echo -e "Column 1\tColumn 2"

1. **\b**: Backspace character.

echo -e "123\b456" # Outputs: 12456

**\r**: Carriage return character.

echo -e "123\r456" # Outputs: 4563

1. **\a**: Alert (bell) character.

echo -e "This is an alert\a"

1. **\c**: Suppress trailing newline.

echo -e "This line will not have a newline\c" echo "The cursor will be here."

