

A black background with white text

AI-generated content may be incorrect.A black screen with numbers and dollar signs

AI-generated content may be incorrect.

What is shell Scripting?

* Shell scripting is basically a scrip that contains series of command which is used to perform a particular task, Shell scripting is basically executed in a sequential manner.
* Shell Scripting is mostly used for automation purposes

Basic Shell Script example  
A black background with white text

AI-generated content may be incorrect.

#!/bin/bash 🡪 is called as the Shebang (it is used to tell the OS which interpreter you will be using) (**Its not mandatory to use the shebang**)

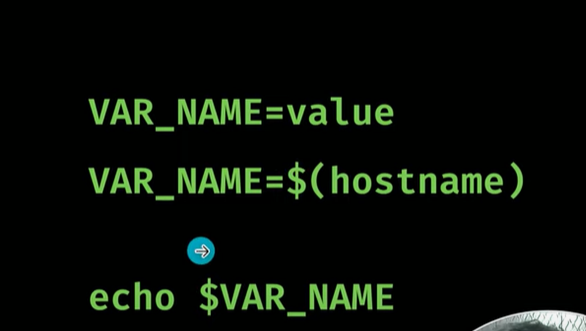
Sending Output to the terminal  


How to execute a shell script  
A screenshot of a computer script

AI-generated content may be incorrect.

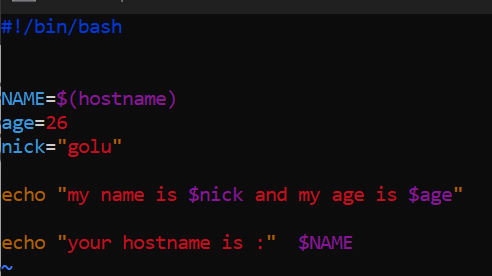
Comments in shell Scripting



Variables in Shell Scripting  
  


A screen shot of a computer

AI-generated content may be incorrect.



What is Constant variable?

One the value for a variable is declared then it cannot be changes, you just need to add the (readonly) term before your variable

Readonly name=”Rohit”

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

**Taking Inputs from the USER**

A black background with green text

AI-generated content may be incorrect.

A black background with red and purple text

AI-generated content may be incorrect.Read -p (p is used to print someting)

A screenshot of a computer screen

AI-generated content may be incorrect.

CONDITIONAL STATEMENT IN SHELL

1. IF – ELSE

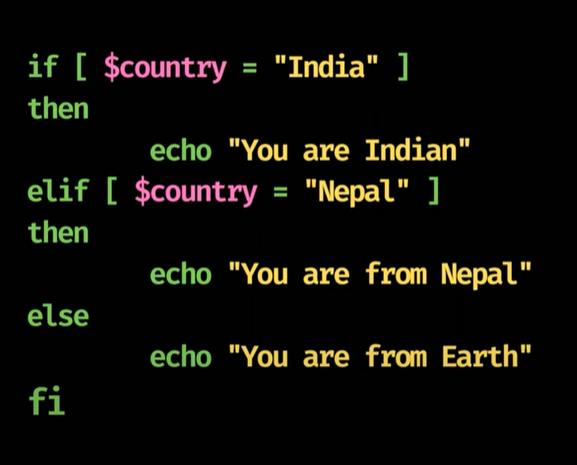
A screenshot of a computer screen

AI-generated content may be incorrect.

A black screen with red and yellow text

AI-generated content may be incorrect.

1. ELIF



A screen shot of a computer

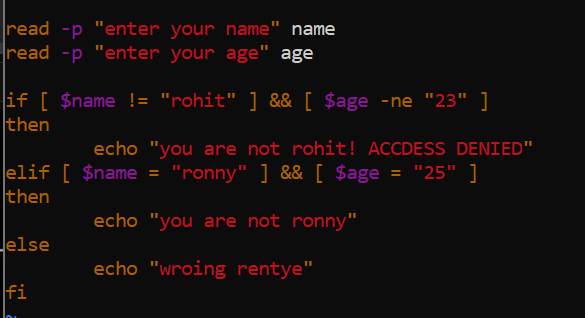
AI-generated content may be incorrect.

NOTE: When you are dealing with number you can use  **-eq**  and when you are dealing with string you can use the **=** symbol  
  
**=**: This is used for string comparison. It checks if two strings are equal. For example:

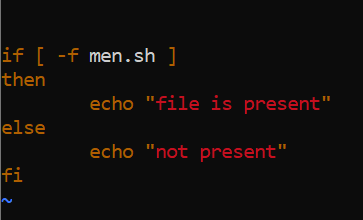
**-eq**: This is used for numerical comparison. It checks if two numbers are equal. For example:



How to add two conditions

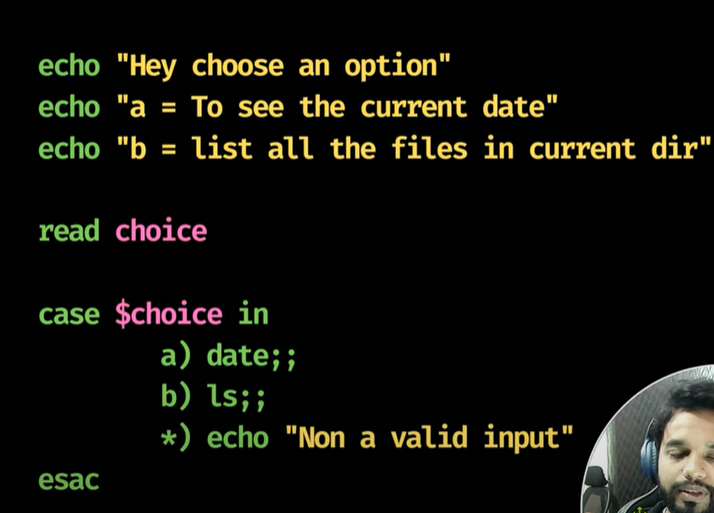


Check if file or directory exist

A screenshot of a computer program

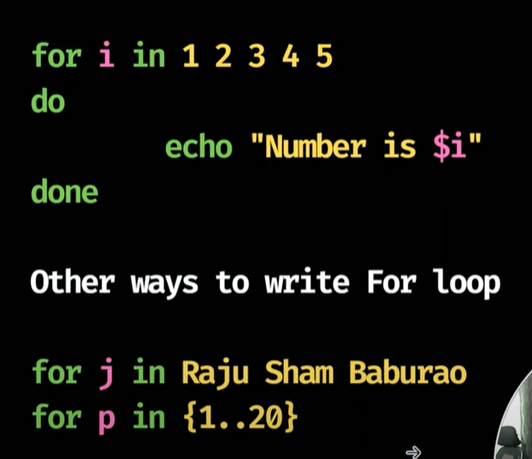
AI-generated content may be incorrect.

Swithc case

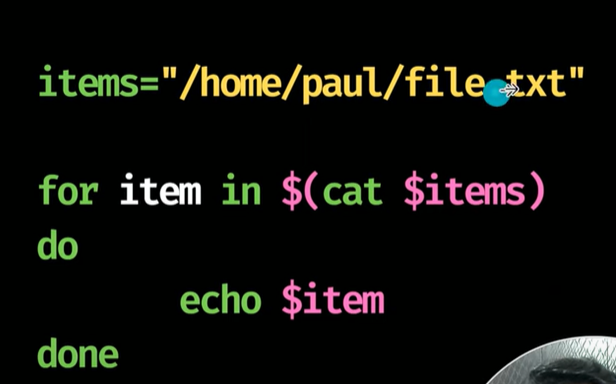


* Is the default case (\*), if no condition is met

For LOOP



How to take valued from a file

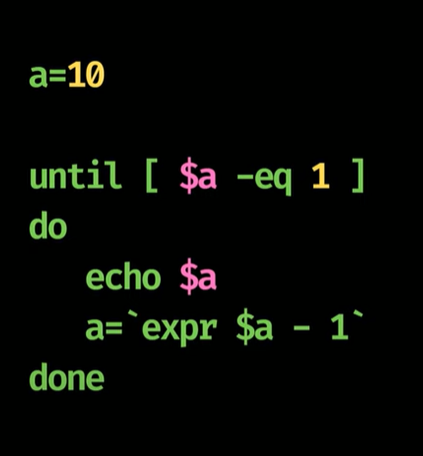


While loop  
A black background with colorful text

AI-generated content may be incorrect.

Until loop

The until loop will run till that time when the condition become false(opposite of while loop)

A black background with white text

AI-generated content may be incorrect.

HOW TO USE EXPRESSION

A black screen with yellow text

AI-generated content may be incorrect.

HOW TO PRINT THE NAME OF THE SCRIPT  
Yellow text on a black background

AI-generated content may be incorrect.

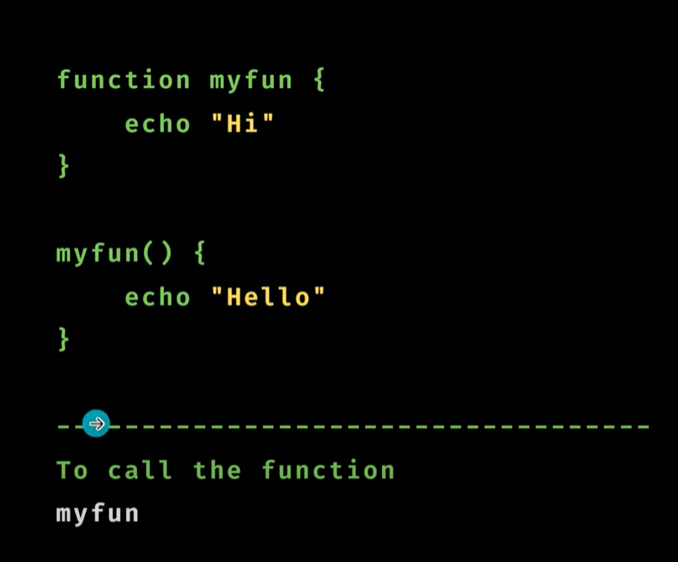
Functions in Shell Scripting  
A screenshot of a computer

AI-generated content may be incorrect.

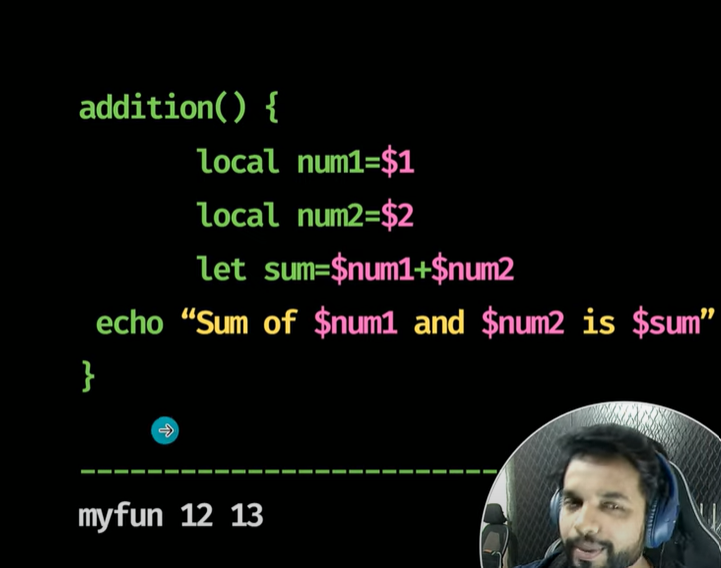
There are two ways to create a function in the shell scripting

1 function func() {}

2 myfun() {}



We can pass argument in functions as well



PARAMETERS

* **positional Parameters ($0, $1, $2, ...):**
  + $0 holds the name of the script itself.
  + $1, $2, $3, and so on, represent the first, second, third, and subsequent command-line arguments, respectively.
  + For arguments beyond $9, use curly braces, such as ${10} for the tenth argument.
* **Special Parameters:**
  + $#: Contains the total number of command-line arguments passed to the script (excluding the script name).
  + $@: Represents all command-line arguments as separate strings, preserving individual arguments even with spaces (e.g., for arg in "$@"; do ...).
  + $\*: Represents all command-line arguments as a single string, where arguments are joined by the first character of IFS (Internal Field Separator), typically a space. When enclosed in double quotes ("$\*"), it treats all arguments as a single argument.
  + $? is a special parameter that holds the exit status of the most recently executed foreground command or pipeline.

Example Usage:

Consider a script named greet.sh:

Code

*#!/bin/bash*  
echo "Hello, $1!"  
echo "You passed $# arguments."  
echo "All arguments: $@"

Executing greet.sh John Doe:

* $0 would be greet.sh
* $1 would be John
* $2 would be Doe
* $# would be 2
* $@ would expand to "John" "Doe"
* $\* would expand to "John Doe"

BASH VARIABLE

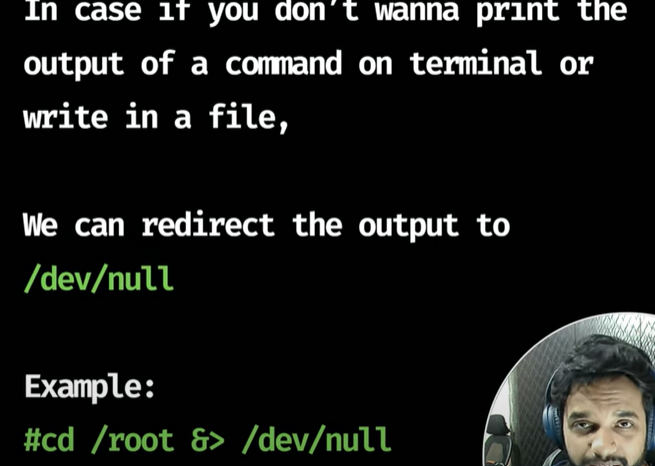
RANDON- prints random values between 0 – 32,607

UID- prints the user id

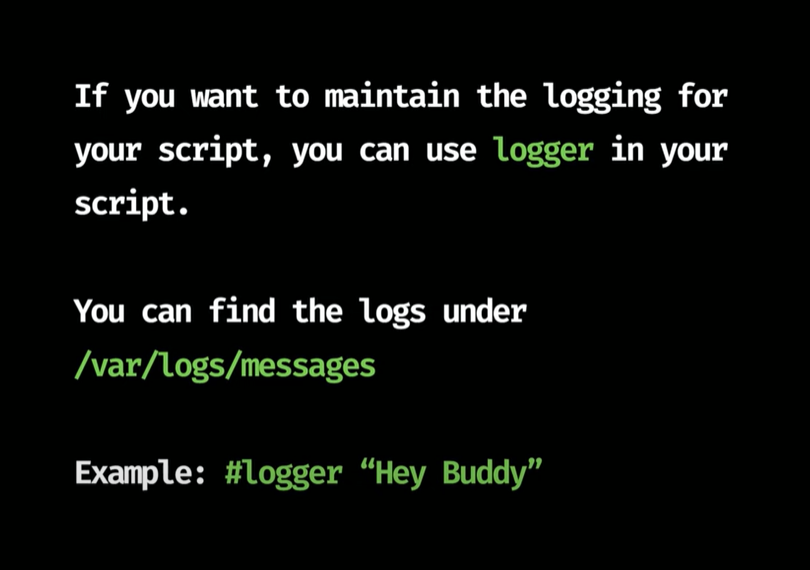
A close up of a text

AI-generated content may be incorrect.

If you don’t want to print any output on terminal, you can use /dev/null



Logger



DEBUGGING SCRIPT

A black screen with white text

AI-generated content may be incorrect.

A screen shot of a computer screen

AI-generated content may be incorrect.A screen shot of a computer

AI-generated content may be incorrect.