**Shell Scripting**

**Shell:**

* A **Shell** is a program that acts as an interface between the **user** and the **operating system** (mainly the **kernel**).
* It takes commands from the user and gives them to the OS to execute.
* Common shell types:
  + **Bash** (Bourne Again SHell) – most widely used
  + **sh**, **zsh**, **ksh**, **csh**
* Example: When you open a terminal and type ls — the **shell** processes it and shows you the list of files.

**Shell Script:**

* A **Shell Script** is a **text file** containing a sequence of **commands** written for the shell to execute.
* It automates tasks that you would normally type one by one.
* The file usually has a .sh extension.

#!/bin/bash 🡪 Path to the interpreter of shell

echo "Hello, World!" 🡪 Print text

**How to run:**

1. Save it as aswini.sh
2. Open terminal
3. Run:

chmod +x aswini.sh

./aswini.sh

**Variables:**

* A **variable** in a shell script is used to **store data** (like a number, string, or result of a command) so you can use it later in your script.
* You don't need to declare a type like in other languages (e.g., int, string). Everything is treated as a **string** by default.

**Define a Variable**

name="Aswini"

**Note:**

* No spaces around =
* To access the variable, use **$**:

echo "My name is $name"

**Types of Variables in Shell Script**

There are mainly 3 types:

**1. Local Variables**

* Defined and used inside a script or function
* Not accessible outside that scope

**2. Environment Variables**

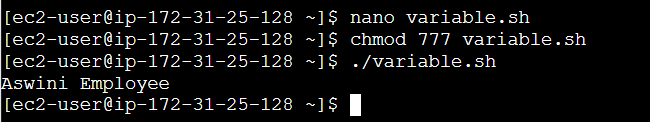
* Available system-wide and inherited by child processes
* Often written in uppercase

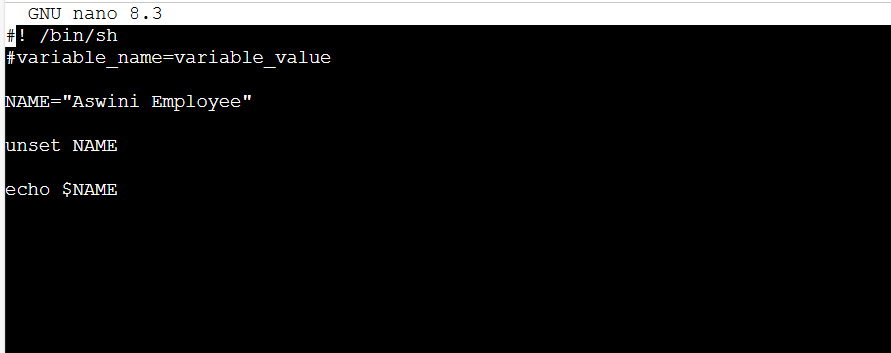
Practice:

Script 1:

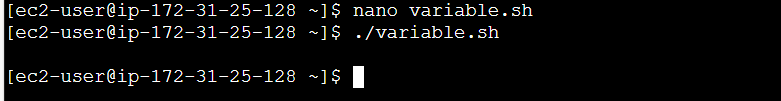


Output:



Script 2: 

Output:



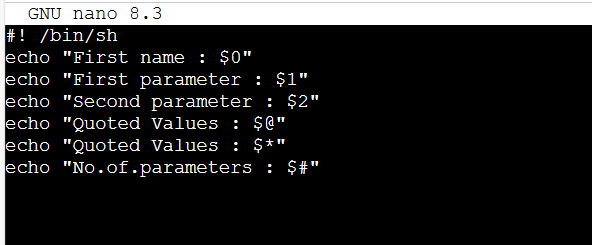
**3. Special Variables**

* Predefined by the shell to handle input, output, and script metadata

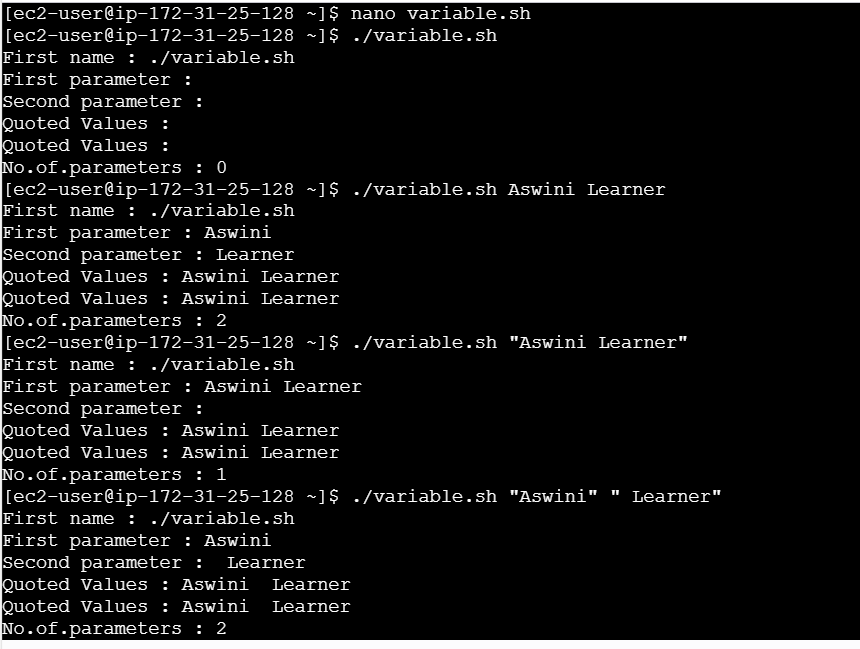
| **Variable** | **Description** |
| --- | --- |
| $0 | Name of the script |
| $1 to $9 | Arguments passed to the script |
| $# | Number of arguments |
| $@ | All arguments |
| $$ | PID of the script |
| $? | Exit status of last command |

**Practice :**

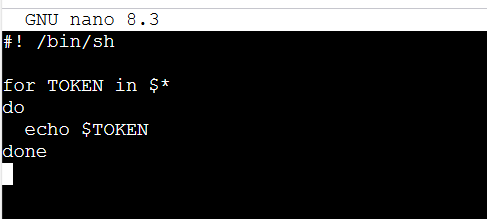
Script 3:



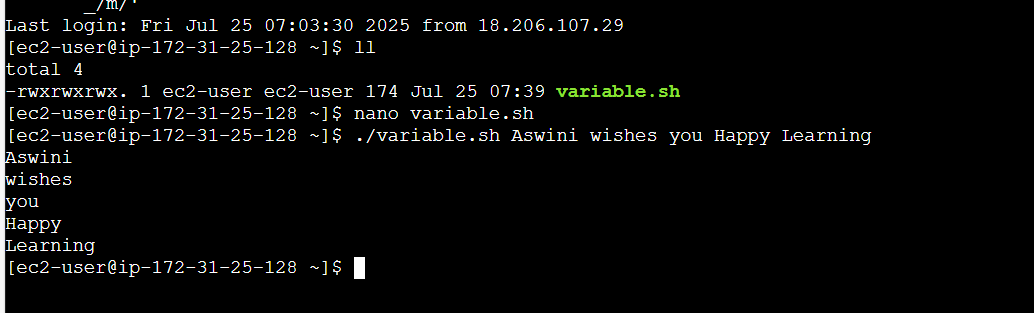
Output:



Script 4:



Output:



**Operators**

* **Operators** are used to **perform operations** on variables and values (just like in other programming languages).
* They are used in **arithmetic, comparisons, conditions, logic**, etc.

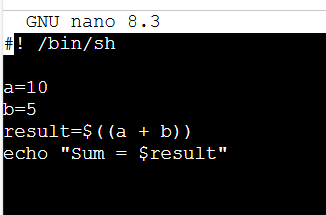
**Categories of Basic Operators**

**1. Arithmetic Operators**

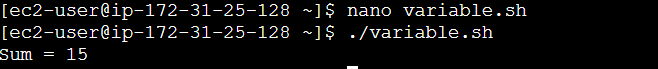
* Used to perform math operations.
* Use expr or (( )) or let for calculation.

| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| + | Addition | expr 2 + 3 |
| - | Subtraction | expr 5 - 1 |
| \* | Multiplication | expr 4 \\* 2 |
| / | Division | expr 8 / 2 |
| % | Modulus (remainder) | expr 5 % 2 |

Script:



Output:

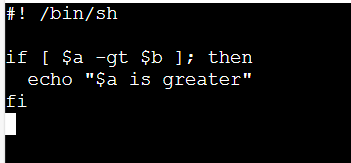


**2. Relational / Comparison Operators**

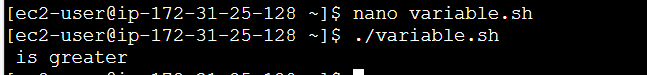
* Used to compare numbers (inside [ ] or [[ ]])

| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| -eq | Equal to | [ $a -eq $b ] |
| -ne | Not equal to | [ $a -ne $b ] |
| -gt | Greater than | [ $a -gt $b ] |
| -lt | Less than | [ $a -lt $b ] |
| -ge | Greater than or equal to | [ $a -ge $b ] |
| -le | Less than or equal to | [ $a -le $b ] |

Script:



Output:



**3. Logical Operators**

* Used to combine conditions.

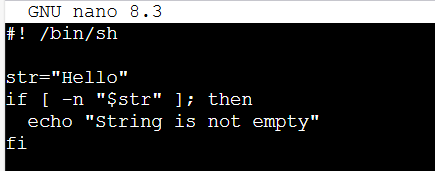
| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| && | AND | [ $a -gt 0 ] && [ $b -gt 0 ] |
|  |  |  |
| ! | NOT | ! [ $a -eq $b ] |

**4. String Operators**

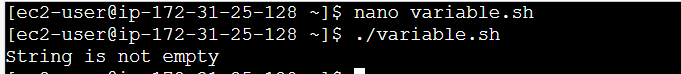
* Used to compare strings.

| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| = | Equal | [ "$a" = "$b" ] |
| != | Not equal | [ "$a" != "$b" ] |
| -z | String is empty | [ -z "$a" ] |
| -n | String is not empty | [ -n "$a" ] |

Script:



Output:

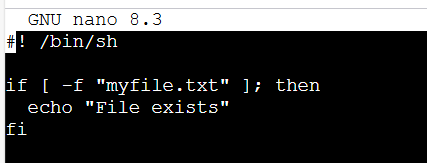


**5. File Test Operators**

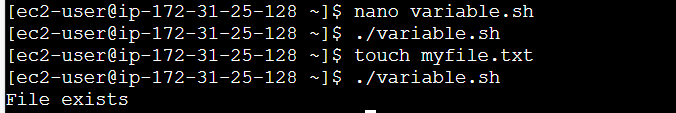
* Used to test properties of files.

| **Operator** | **Meaning** |
| --- | --- |
| -e | File exists |
| -f | Regular file |
| -d | Directory |
| -r | Read permission |
| -w | Write permission |
| -x | Execute permission |
|  |  |

Script:



Output:



**LOOPS**

* Loops are used to repeat a block of commands multiple times.

**Types of Shell Loops**

**1. for Loop**

* Used to iterate over a list of items.

**Syntax:**

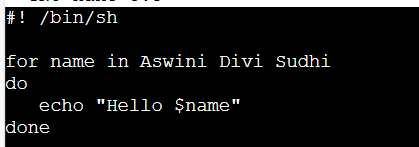
for variable in item1 item2 item3

do

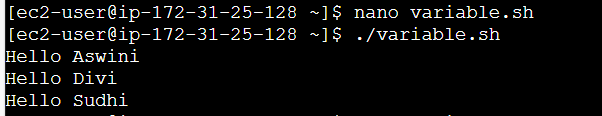
command(s)

done

**Example:**



Output:



**2. while Loop**

* Runs as long as the condition is true.

**Syntax:**

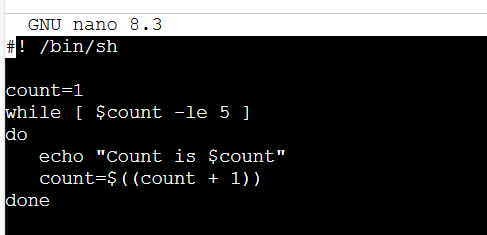
while [ condition ]

do

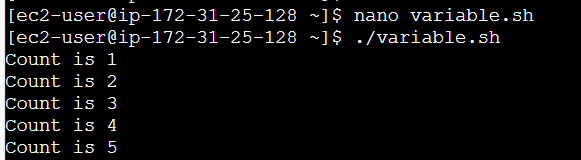
command(s)

done

**Example**



Output:



**3. until Loop**

* Runs as long as the condition is false. (Opposite of while)

**Syntax:**

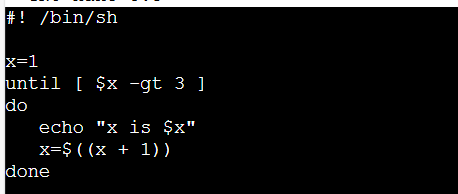
until [ condition ]

do

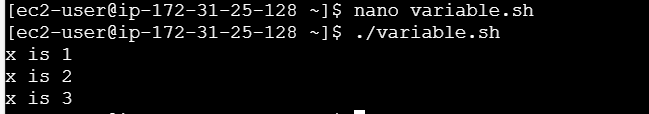
command(s)

done

**Example:**



Output:



**4. select Loop**

* Used to create a menu from a list of options.

**Syntax:**

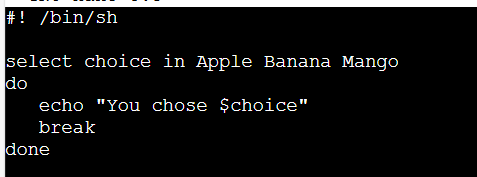
select var in option1 option2 option3

do

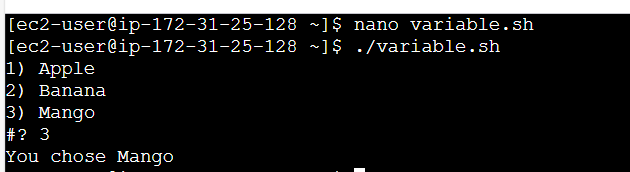
command(s)

done

**Example:**



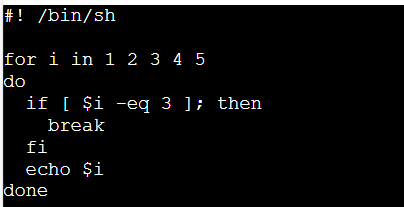
Output:



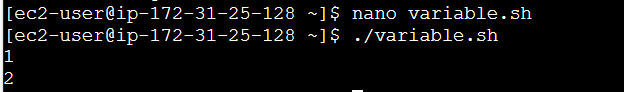
**5. Loop Control Statements**

| **Command** | **Meaning** |
| --- | --- |
| break | Exits the loop immediately |
| continue | Skips the current iteration and continues |

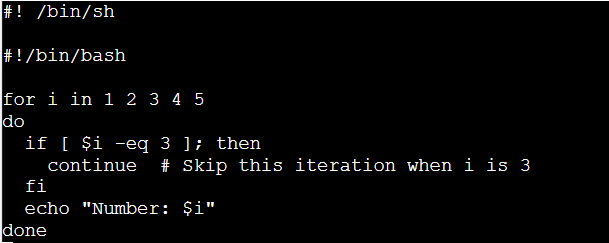
**Example with break:**



Output:



Example with continue :



Output:

