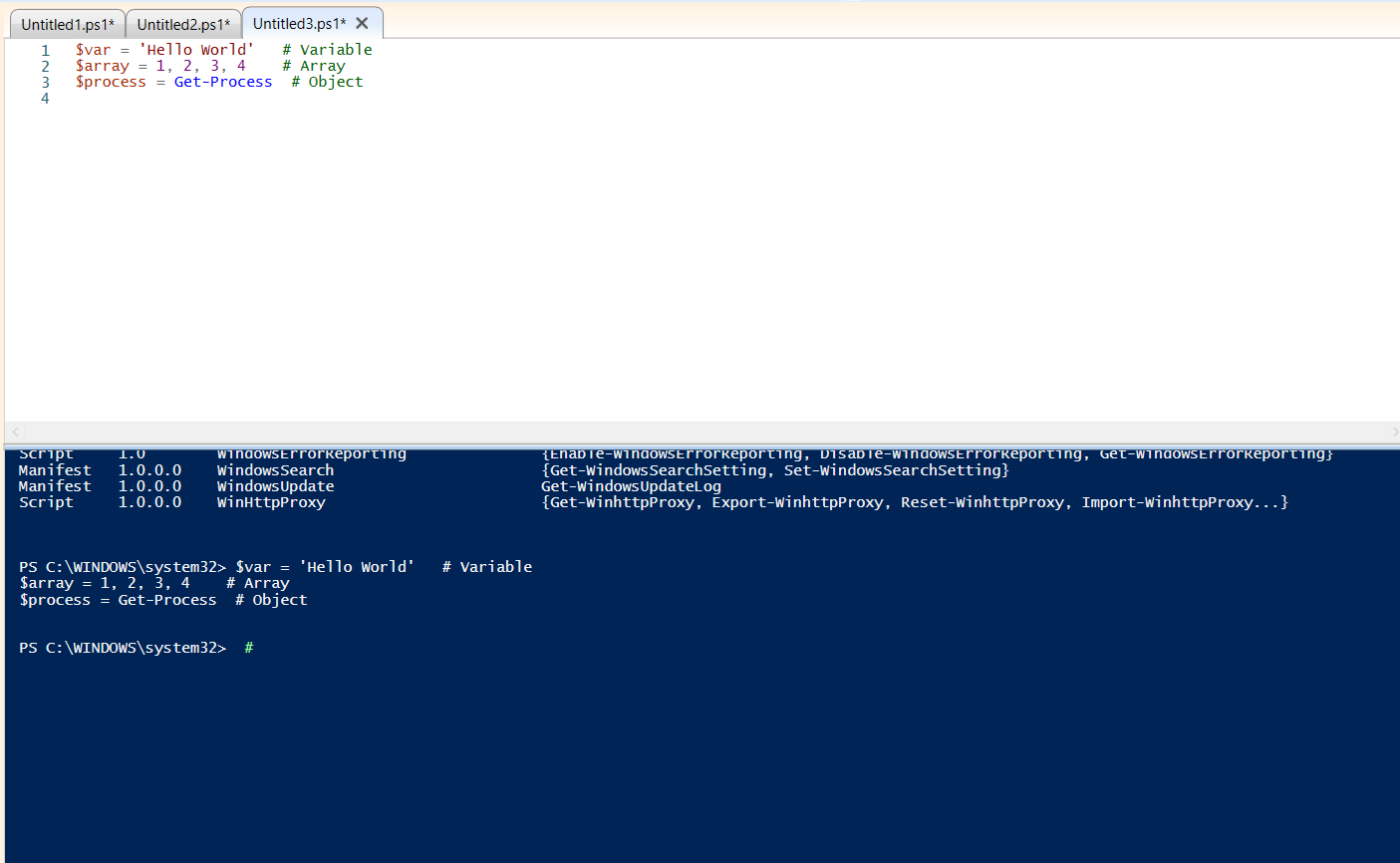
# PowerShell Concepts

## 1. Objects, Arrays, Variables

In PowerShell, everything is treated as an object. Objects have properties (data) and methods (actions). Variables store values, and arrays can hold multiple values.

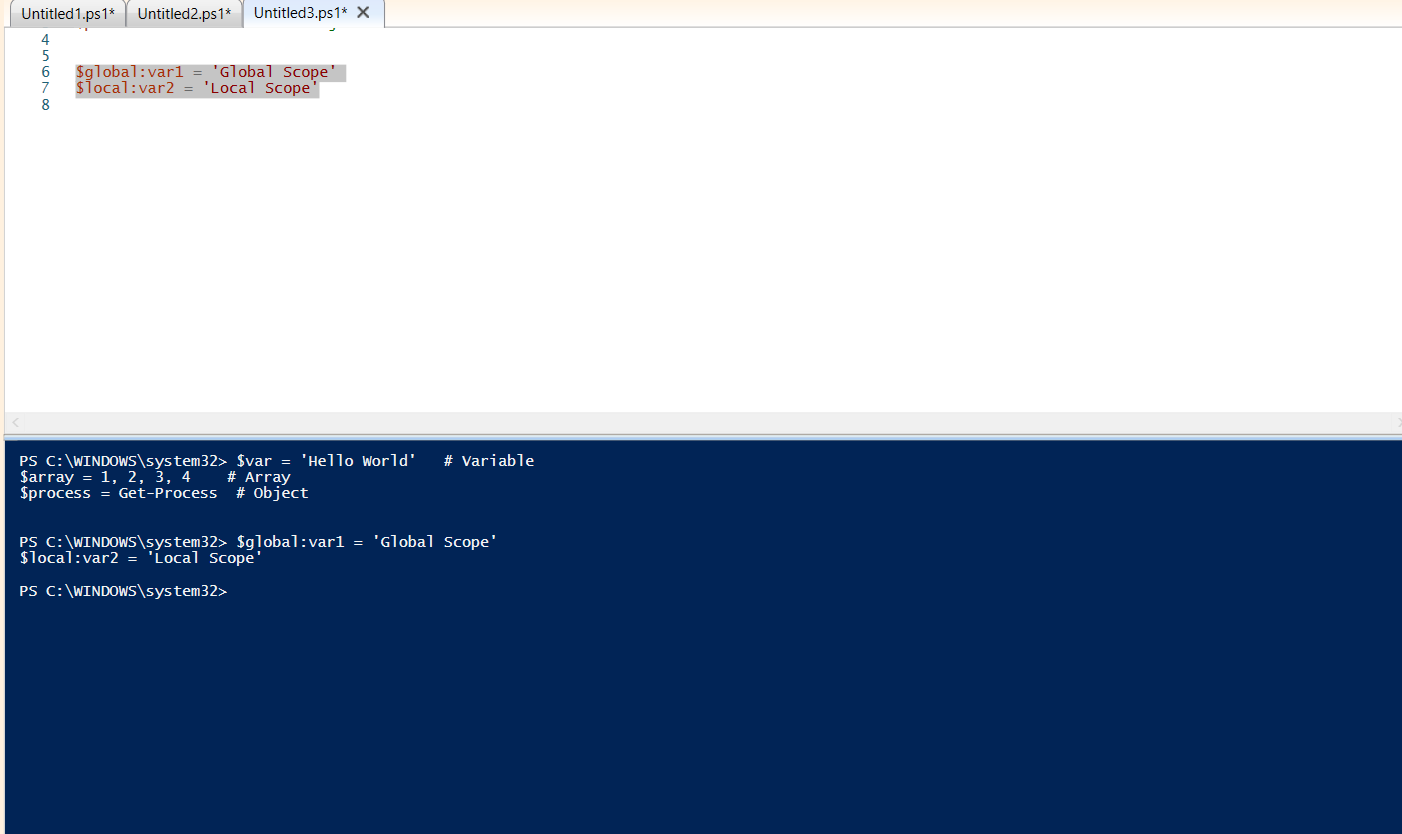
Example:  
$var = 'Hello World' # Variable  
$array = 1, 2, 3, 4 # Array  
$process = Get-Process # Object

****

## 2. Scope

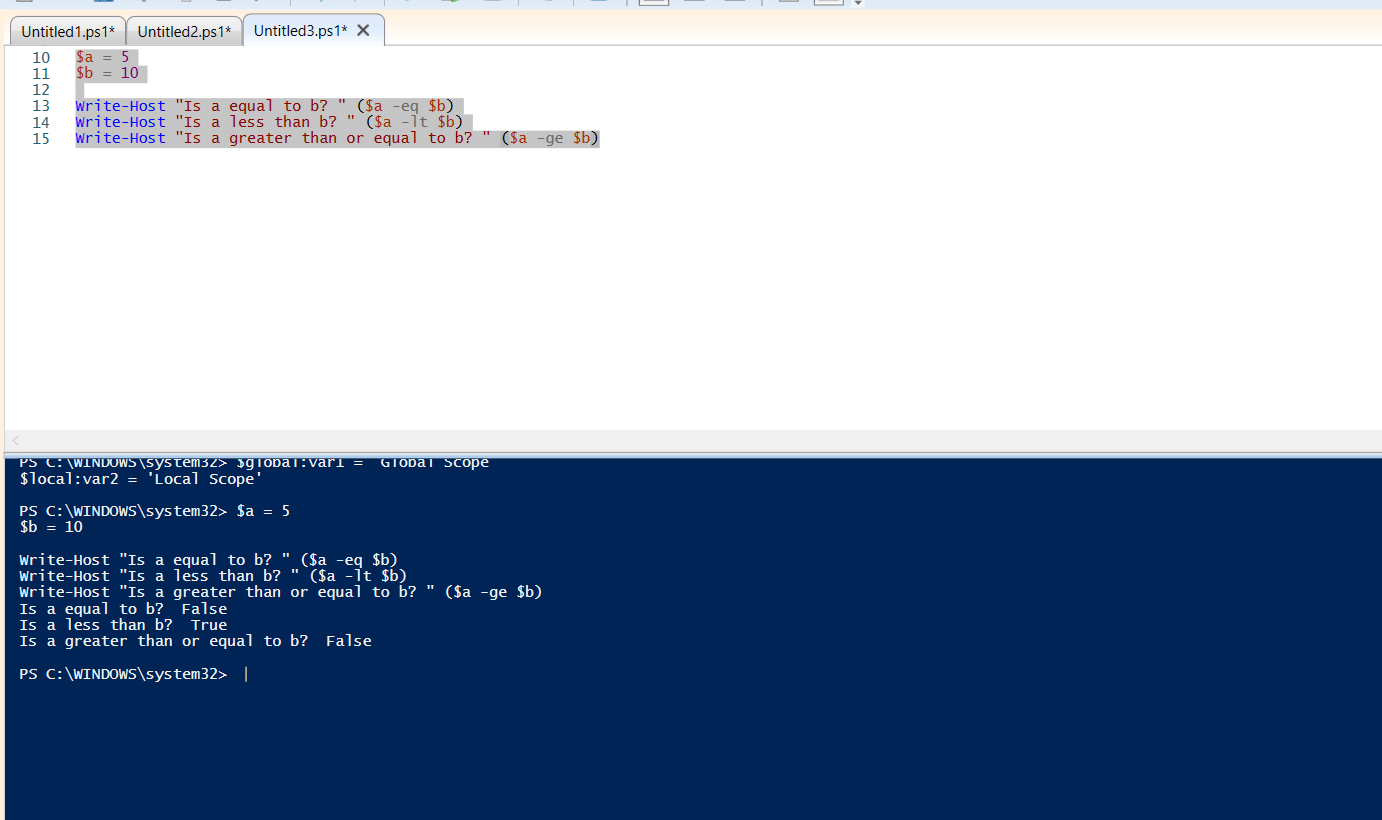
Scope determines where a variable or function is accessible. Common scopes include Global, Local, Script, and Private.

Example:  
$global:var1 = 'Global Scope'  
$local:var2 = 'Local Scope'



## 3. More Operators

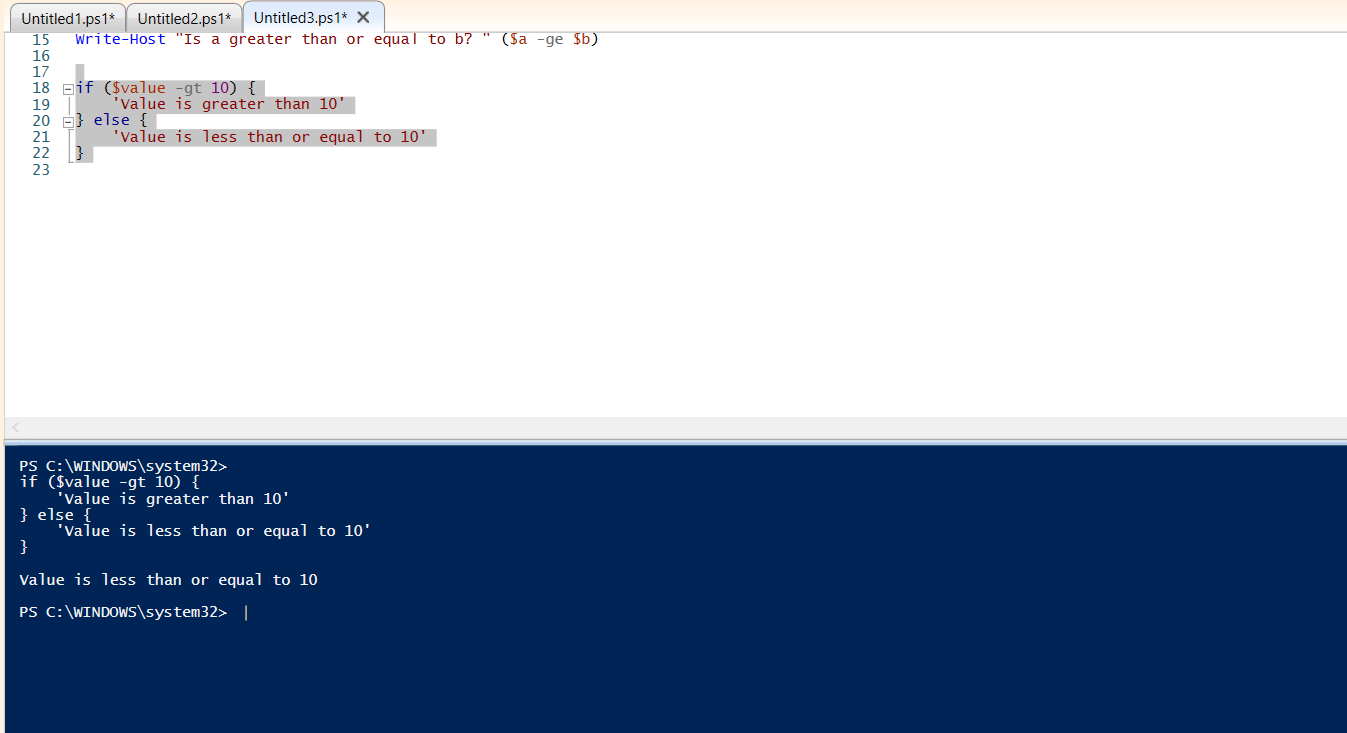
* **Arithmetic:** +, -, \*, /
* **Comparison:** -eq, -ne, -gt, -lt
* **Logical:** -and, -or, -not
* **Pipeline:** | (passes objects from one cmdlet to the next)
* **Format operator:** -f (string formatting)
* **Splatting:** @params (clean way to pass many parameters)



4. Scripting Constructs

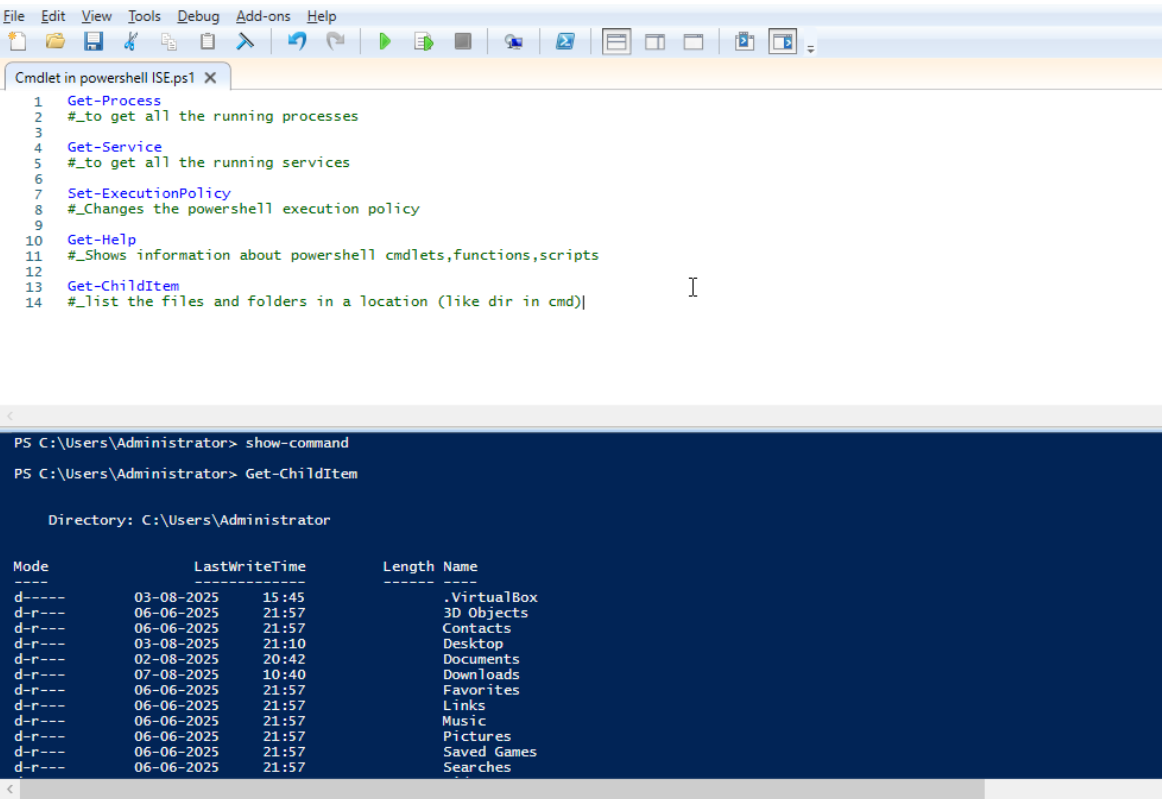
PowerShell supports scripting constructs like loops and conditionals:  
- if, elseif, else  
- switch  
- for, foreach, while, do-while

Example:  
if ($value -gt 10) {  
 'Value is greater than 10'  
} else {  
 'Value is less than or equal to 10'  
}



## 5. Modularization

Modularization means breaking code into reusable units (modules). Modules can contain functions, variables, and other resources.



## 6. Basic Platform Values, Concepts, and Usage

PowerShell is a cross-platform automation tool built on .NET. It is designed to work with objects rather than text, making it powerful for system administration, automation, and configuration management.

Key concepts:  
- Object-based output  
- Pipeline for chaining commands  
- Rich scripting language with full access to .NET classes  
- Cross-platform compatibility