**VARIABLES**

* PowerShell variables are named objects.
* Variable name should start with $ and can contain alphanumeric characters and underscore in their names.
* $location = Get-Location

Get-Member cmdlet can tell the type of variable being used

* $location | Get-Member

**OPRATIORS**

* PowerShell provides a rich set of operators to manipulate variables. We can divide all the PowerShell operators into the following groups −
* Arithmetic Operators
* Assignment Operators
* Comparison Operators
* Logical Operators
* Redirectional Operators
* Spilt and Join Operators
* Type Operators
* Unary Operators

The Arithmetic Operators

* Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. The following table lists the arithmetic operators
* Assume integer variable A holds 10 and variable B holds 20, then −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + (Addition) | Adds values on either side of the operator. | A + B will give 30 |
| - (Subtraction) | Subtracts right-hand operand from left-hand operand. | A - B will give -10 |
| \* (Multiplication) | Multiplies values on either side of the operator. | A \* B will give 200 |
| / (Division) | Divides left-hand operand by right-hand operand. | B / A will give 2 |
| % (Modulus) | Divides left-hand operand by right-hand operand and returns remainder. | B % A will give 0 |

## The Comparison Operators

* Following are the assignment operators supported by PowerShell language −
* Assume integer variable A holds 10 and variable B holds 20, then –

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| eq (equals) | Compares two values to be equal or not. | A -eq B will give false |
| ne (not equals) | Compares two values to be not equal. | A -ne B will give true |
| gt (greater than) | Compares first value to be greater than second one. | B -gt A will give true |
| ge (greater than or equals to) | Compares first value to be greater than or equals to second one. | B -ge A will give true |
| lt (less than) | Compares first value to be less than second one. | B -lt A will give false |
| le (less than or equals to) | Compares first value to be less than or equals to second one. | B -le A will give false |

## The Assignment Operators

* Following are the assignment operators supported by PowerShell language −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Simple assignment operator. Assigns values from right side operands to left side operand. | C = A + B will assign value of A + B into C |
| += | Add AND assignment operator. It adds right operand to the left operand and assign the result to left operand. | C += A is equivalent to C = C + A |
| -= | Subtract AND assignment operator. It subtracts right operand from the left operand and assign the result to left operand. | C -= A is equivalent to C = C - A |

## The Logical Operators

* The following table lists the logical operators −
* Assume Boolean variables A holds true and variable B holds false, then –

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| AND (logical and) | Called Logical AND operator. If both the operands are non-zero, then the condition becomes true. | (A -AND B) is false |
| OR (logical or) | Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true. | (A -OR B) is true |
| NOT (logical not) | Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false. | -NOT(A -AND B) is true |

## Miscellaneous Operators

* Following are various important operators supported by PowerShell language

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| > (Redirectional Opeator) | Redirectional operator. Assigns output to be printed into the redirected file/output device. | dir > test.log will print the directory listing in test.log file |

PowerShell files & folders

1. we'll create a folder in D:\Temp\ with name "Test Folder"

* New-Item -Path 'D:\temp\Test Folder' -ItemType Directory

1. we'll create a file in D:\Temp\Test Folder with name "Test File.txt"

* New-Item -Path 'D:\temp\Test Folder\Test File.txt' -ItemType File

1. we'll copy a folder D:\Temp\Test Folder as D:\Temp\Test Folder1

* Copy-Item 'D:\temp\Test Folder' 'D:\temp\Test Folder1'

1. we'll copy a folder recursively D:\Temp\Test Folder to D:\Temp\Test Folder1

* Copy-Item 'D:\temp\Test Folder' -Destination 'D:\temp\Test Folder1'

1. we'll copy a folder D:\Temp\Test Folder\Test File.txt to D:\Temp\Test Folder1

* Copy-Item 'D:\temp\Test Folder\Test File.txt' 'D:\temp\Test Folder1\Test File1.txt'

1. we'll copy all text file recursively D:\Temp\Test Folder to D:\Temp\Test Folder1

* Copy-Item -Filter \*.txt -Path 'D:\temp\Test Folder' -Recurse -Destination 'D:\temp\Test Folder1'

1. we'll delete a folder D:\Temp\Test Folder1

* Remove-Item 'D:\temp\Test Folder1'

1. we'll move a folder D:\Temp\Test to D:\Temp\Test1

* Move-Item D:\temp\Test D:\temp\Test1

1. we'll move a folder D:\Temp\Test\Test.txt to D:\Temp\Test1

* Move-Item D:\temp\Test\Test.txt D:\temp\Test1

1. we'll rename a folder D:\Temp\Test to D:\Temp\Test1

* Rename-Item "D:\temp\Test Test1"

1. we'll rename a folder D:\Temp\Test\test.txt to test1.txt

* Rename-Item D:\temp\Test\test.txt test1.txt

1. we'll read a file D:\Temp\Test\Test.txt

* Get-Content D:\temp\Test\test.txt

1. we'll read the size of the content of the file read.

* (Get-Content D:\temp\test\test.txt).length

1. we're having a folder test in D:\temp directory

* Test-Path D:\temp\test

1. we're having a file test.txt in D:\temp\test directory

* Test-Path D:\temp\test\test.txt

1. we're using Get-Date to get current date

* Get-Date

1. we're using Set-Date to add one more day to current date.

* set-date -Date (Get-Date).AddDays(1)

1. we're using Get-Date to get current date

* Get-Date -DisplayHint Date

1. we're using Get-Date to get current date

* get-date -displayHint time

**PowerShell** **version**: **$PSVersionTable**

PowerShell cmdlet

* get-content D:\temp\test\test.txt | measure-object -character -line -word
* Lines Words Characters Property
* ----- ----- ---------- --------
* 1 3 29

1. We'll count the no. of files present in current directory.

* Get-ChildItem | Measure-Object

1. Compare the files. Type the following command

* Compare-Object -ReferenceObject $(Get-Content D:\temp\test\TestFolder\star.txt) -DifferenceObject $(Get-Content D:\temp\test\TestFolder\File1.txt

1. Get the list of services.

* Get-Service | Format-List

**PowerShell Script–** getting the content of the website and cheking its up or not.

[string] $\_URL = '[https://git.gtech.com/repos?visibility=public'](https://git.gtech.com/repos?visibility=public%27)  
$webRequest = [net.WebRequest]::Create("[https://git.gtech.com/repos?visibility=public")](https://git.gtech.com/repos?visibility=public%22))  
$webRequest | gm  
$webrequest.GetResponse()  
Invoke-WebRequest "[https://git.gtech.com/repos?visibility=public"](https://git.gtech.com/repos?visibility=public%22) -UseBasicParsing  
function CheckSiteURLStatus($\_URL) {  
try {  
$request= [System.Net.WebRequest]::Create($\_URL)  
$response = $request.getResponse()  
if ($response.StatusCode -eq "200") {  
write-host "`nSite - $\_URL is up (Return code: $($response.StatusCode) -   
$([int] $response.StatusCode)) `n" -ForegroundColor green   
}  
else {  
write-host "`n Site - $\_URL is down `n" ` -ForegroundColor red  
}  
} catch {  
write-host "`n Site is not accessable, May DNS issue. Try again.`n" ` -ForegroundColor red  
}  
}  
CheckSiteURLStatus $\_URL

1. **Read-host** we'll ask the user to pass an input and read the input into a variable.

* $choice = Read-Host "Please put your choice"

1. we'll create objects using Process properties.

* Get-Process | Select-Object -Property ProcessName, Id, WS -Last 5

1. we'll select unique values of an array.

* "a","b","c","a","a","a" | Select-Object -Unique

1. we'll sort objects using Process properties

* Get-Process | Sort-Object -Property WS | Select-Object -Last 5

1. we'll sort an array.

* "d","e","c","a","b","f" | Sort-Object

1. we'll show a warning message.

* Write-Warning "Test Warning"

1. we'll show a customized message.

* Write-Host (2,4,6,8,10,12) -Separator ", -> " -ForegroundColor DarkGreen -BackgroundColor White

1. we'll show a customized message.

* Invoke-Item "D:\temp\Test Folder\test.txt"

1. we'll show how to invoke an expression.

* $Command = 'Get-Process'
* $Command

Get-Process

* Invoke-Expression $Command

1. we'll show how to measure time of Get-EventLog command to log an event in PowerShell event log.

* Measure-Command { Get-EventLog "Windows PowerShell" }

|  |  |
| --- | --- |
| **Operator** | **Description** |
| $$ | Represents the last token in the last line received by the session. |
| $? | Represents the execution status of the last operation. It contains TRUE if the last operation succeeded and FALSE if it failed. |
| $^ | Represents the first token in the last line received by the session. |
| $\_ | Same as $PSItem. Contains the current object in the pipeline object. You can use this variable in commands that perform an action on every object or on selected objects in a pipeline. |
| $ARGS | Represents an array of the undeclared parameters and/or parameter values that are passed to a function, script, or script block. |
| $CONSOLEFILENAME | Represents the path of the console file (.psc1) that was most recently used in the session. |
| $ERROR | Represents an array of error objects that represent the most recent errors. |
| $EVENT | Represents a PSEventArgs object that represents the event that is being processed. |
| $EVENTARGS | Represents an object that represents the first event argument that derives from EventArgs of the event that is being processed. |
| $EVENTSUBSCRIBER | Represents a PSEventSubscriber object that represents the event subscriber of the event that is being processed. |
| $EXECUTIONCONTEXT | Represents an EngineIntrinsics object that represents the execution context of the PowerShell host. |
| $FALSE | Represents FALSE. You can use this variable to represent FALSE in commands and scripts instead of using the string "false". |
| $FOREACH | Represents the enumerator (not the resulting values) of a ForEach loop. You can use the properties and methods of enumerators on the value of the $ForEach variable. |
| $HOME | Represents the full path of the user's home directory. |
| $HOST | Represents an object that represents the current host application for PowerShell. |
| $INPUT | Represents an enumerator that enumerates all input that is passed to a function. |
| $LASTEXITCODE | Represents the exit code of the last Windows-based program that was run. |
| $MATCHES | The $Matches variable works with the -match and -notmatch operators. |
| $MYINVOCATION | $MyInvocation is populated only for scripts, function, and script blocks. PSScriptRoot and PSCommandPath properties of the $MyInvocation automatic variable contain information about the invoker or calling script, not the current script. |
| $NESTEDPROMPTLEVEL | Represents the current prompt level. |
| $NULL | $null is an automatic variable that contains a NULL or empty value. You can use this variable to represent an absent or undefined value in commands and scripts. |
| $PID | Represents the process identifier (PID) of the process that is hosting the current PowerShell session. |
| $PROFILE | Represents the full path of the PowerShell profile for the current user and the current host application. |
| $PSCMDLET | Represents an object that represents the cmdlet or advanced function that is being run. |
| $PSCOMMANDPATH | Represents the full path and file name of the script that is being run. |
| $PSCULTURE | Represents the name of the culture currently in use in the operating system. |
| $PSDEBUGCONTEXT | While debugging, this variable contains information about the debugging environment. Otherwise, it contains a NULL value. |
| $PSHOME | Represents the full path of the installation directory for PowerShell. |
| $PSITEM | Same as $\_. Contains the current object in the pipeline object. |
| $PSSCRIPTROOT | Represents the directory from which a script is being run. |
| $PSSENDERINFO | Represents information about the user who started the PSSession, including the user identity and the time zone of the originating computer. |
| $PSUICULTURE | Represents the name of the user interface (UI) culture that is currently in use in the operating system. |
| $PSVERSIONTABLE | Represents a read-only hash table that displays details about the version of PowerShell that is running in the current session. |
| $SENDER | Represents the object that generated this event. |
| $SHELLID | Represents the identifier of the current shell. |
| $STACKTRACE | Represents a stack trace for the most recent error. |
| $THIS | In a script block that defines a script property or script method, the $This variable refers to the object that is being extended. |
| $TRUE | Represents TRUE. You can use this variable to represent TRUE in commands and scripts. |

**HASHTABLE**

$hash = @{ ID = 1; Shape = "Square"; Color = "Blue"}

or

$hash = @{}

1. The following code snippets are examples of this syntax –

* $hash = [ordered]@{ ID = 1; Shape = "Square"; Color = "Blue"}

Print the hashtable.

* $hash

The hashtable values are accessed through the **keys**.

* $hash["ID"]

Dot notation can be used to access hashtables keys or values.

* $hash.keys
* $hash.values

|  |  |
| --- | --- |
| **ubexpression** | **Matches** |
| ^ | Matches the beginning of the line. |
| $ | Matches the end of the line. |
| . | Matches any single character except newline. Using **m** option allows it to match the newline as well. |
| [...] | Matches any single character in brackets. |
| [^...] | Matches any single character not in brackets. |
| \A | Beginning of the entire string. |
| \z | End of the entire string. |
| \Z | End of the entire string except allowable final line terminator. |
| re\* | Matches 0 or more occurrences of the preceding expression. |
| re+ | Matches 1 or more of the previous thing. |
| re? | Matches 0 or 1 occurrence of the preceding expression. |
| re{ n} | Matches exactly n number of occurrences of the preceding expression. |
| re{ n,} | Matches n or more occurrences of the preceding expression. |
| re{ n, m} | Matches at least n and at most m occurrences of the preceding expression. |
| a| b | Matches either a or b. |
| (re) | Groups regular expressions and remembers the matched text. |
| (?: re) | Groups regular expressions without remembering the matched text. |
| (?> re) | Matches the independent pattern without backtracking. |
| \w | Matches the word characters. |
| \W | Matches the nonword characters. |
| \s | Matches the whitespace. Equivalent to [\t\n\r\f]. |
| \S | Matches the nonwhitespace. |
| \d | Matches the digits. Equivalent to [0-9]. |
| \D | Matches the nondigits. |
| \A | Matches the beginning of the string. |
| \Z | Matches the end of the string. If a newline exists, it matches just before newline. |
| \z | Matches the end of the string. |
| \G | Matches the point where the last match finished. |
| \n | Back-reference to capture group number "n". |
| \b | Matches the word boundaries when outside the brackets. Matches the backspace (0x08) when inside the brackets. |
| \B | Matches the nonword boundaries. |
| \n, \t, etc. | Matches newlines, carriage returns, tabs, etc. |
| \Q | Escape (quote) all characters up to \E. |
| \E | Ends quoting begun with \Q. |

**MATCHCHARACTERS**

#Format value

#Matches exact characters anywhere in the original value.

"book" -match "oo"

#Format .

#Logic Matches any single character.

"copy" -match "c..y"

#Format [value]

#Logic Matches at least one of the characters in the brackets.

"big" -match "b[iou]g"

#Format [range]

#Logic Matches at least one of the characters within the range. The use

# of a hyphen (-) allows you to specify an adjacent character.

"and" -match "[a-e]nd"

#Format [^]

#Logic Matches any characters except those in brackets.

"and" -match "[^brt]nd"

#Format ^

#Logic Matches the beginning characters.

"book" -match "^bo"

#Format $

#Logic Matches the end characters.

"book" -match "ok$"

#Format \*

#Logic Matches any instances of the preceding character.

"baggy" -match "g\*"

#Format ?

#Logic Matches zero or one instance of the preceding character.

"baggy" -match "g?"

|  |  |
| --- | --- |
| **PowerShell** | **Command Prompt (CMD)** |
| 1. It is the automated task-based command-line interface and associated scripting language based on the .NET framework. | 1. It is the default command-line interpreter for the Microsoft Windows operating system. |
| 2. It can interpret both batch and PowerShell commands. | 2. It can interpret only batch commands. |
| 3. It is used to control and automate the applications and Windows operating system on a Windows server. | 3. It is used to execute the given commands on the console, which can be used to debug the problem. |
| 4. The output generated by the PowerShell is not just a stream of characters but a collection of objects. | 4. The output generated by the command prompt is just a stream of characters(text). |
| 5. It is both a shell and scripting environment which supports the creation of large files for managing the windows operating system. | 5. It is just a shell system, which allows a user to do only easy and basic scripts for the execution of the batch file. |

|  |  |  |
| --- | --- | --- |
| **Description** | **Bash** | **PowerShell** |
| **List files and folders** | ls | Ls, Get-childItem |
| **Change directory** | cd | Set-childItem |
| **Show Working directory** | pwd | Get-Location |
| **Clear the Screen** | Clear | cls, clear |
| **To copy a file** | cp | Copy-Item |
| **To remove or delete a file** | rm | Remove-Item, ri, rmdir, rd, del, rm |
| **To print a string** | echo "string" | Write-host "string" OR echo "string" |
| **To create a New text file** | touch <filename.txt> | New-Item <filename.txt> |

**Key features**

Following are the key features of Windows PowerShell ISE:

* **Multiline editing:** In the command pane, to insert a blank line under the current line, press SHIFT + ENTER.
* **Selective Execution**: To execute a part of a script, select the text to run, and click the **Run script** Or, press the **F5** key.
* **Context-sensitive help:** Type the **Invoke-Item**and then press the**f1** A help file opens to the article for an **Invoke-Item** command let**.**

**#Example**

for($i = 1; $i -le 10; $i++){

$x = $i%2

if($x -eq 0){

echo $i

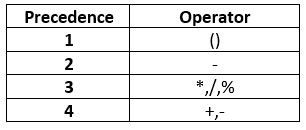
}

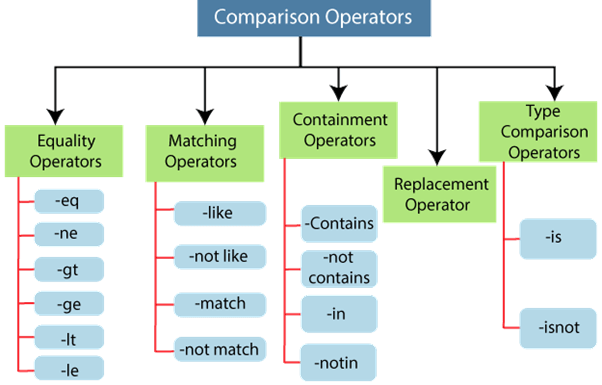
}

# **Basic cmdlets of PowerShell**

|  |  |  |
| --- | --- | --- |
| **Command-Name** | **Description** |  |
| **Add-content** | This cmdlet adds the content to the specified file. |  |
| **Add-Computer** | This cmdlet adds the local computer to a domain or workgroup. |  |
| **Add-History** | This cmdlet is used to add the command in the history. |  |
| **Add-jobTrigger** | This cmdlet adds the job triggers to the scheduled jobs. |  |
| **Add-member** | The cmdlet adds the custom methods or properties to an instance of a PowerShell object. |  |
| **Add-type** | This cmdlet adds a Microsoft .NET framework class to a Windows PowerShell session. |  |
| **Clear-Content** | This cmdlet deletes the content of a file, but not delete that file. |  |
| **Clear-History** | This cmdlet deletes the entries from the command history. |  |
| **Clear-Item** | This cmdlet clears the value of item or variable, but not clear that item or a variable. |  |
| **Clear-ItemProperty** | This cmdlet clears the value of a property but not delete that property. |  |
| **Clear-Variable** | This cmdlet deletes the value of a variable. |  |
| **Clear-Host** | This cmdlet clears the display of the host program. |  |
| **Copy-ItemProperty** | This cmdlet copies a value and property from a specified location to another location. |  |
| **Copy-Item** | This cmdlet copies an item from one location to another. |  |
| **Complete-Transaction** | This cmdlet commits the active transaction. |  |
| **Compare-object** | This cmdlet compares two sets of objects. |  |
| **Disable-PSBreakpoint** | This Cmdlet disables the breakpoint in the current console. |  |
| **Enable-PSBreakpoint** | This Cmdlet enables the breakpoint in the current console. |  |
| **Find-package** | This cmdlet finds the software packages in the available packages sources. |  |
| **Find-script** | This cmdlet is used to find a script |  |
| **ForEach-Object** | This cmdlet performs an operation on each item in a collection of input objects. |  |
| **Get-Alias** | This cmdlet gets the aliases for the current session. |  |
| **Get-childItem** | This cmdlet gets the item and the child items in one or more specified locations. |  |
| **Get-Command** | This cmdlet is used to get all commands. |  |
| **Get-Content** | This cmdlet gets the content of the file at the specified location. |  |
| **Get-Date** | This cmdlet is used to get the current date and time. |  |
| **Get-ExecutionPolicy** | This cmdlet gets the execution policy for the current session. |  |
| **Get-Help** | This cmdlet displays information about PowerShell commands and concepts. |  |
| **Get-History** | This cmdlet displays a list of commands which are entered during the current session. |  |
| **Get-host** | This cmdlet gets an object which represents the current host program. |  |
| **Get-InstalledScript** | This cmdlet gets a script which is installed for the current user. |  |
| **Get-Item** | This cmdlet gets the item or a file at a particular location. |  |
| **Get-ItemProperty** | This cmdlet gets the properties of a particular item. |  |
| **Get-Location** | This cmdlet displays the current working location. |  |
| **Get-PSBreakpoint** | This cmdlet gets the breakpoint which is set in the current session. |  |
| **Get-Package** | This cmdlet displays the list of all installed packages by using the package management. |  |
| **Get-Process** | This cmdlet gets the processes which are running on local or remote computers. |  |
| **Get-Service** | This cmdlet gets the services on local or remote computers. |  |
| **Get-Transaction** | This cmdlet gets the currently active transaction. |  |
| **Get-Variable** | This cmdlet gets the variable in the current console. |  |
| **Install-package** | This cmdlet is used to install one or more software packages. |  |
| **Install-script** | This cmdlet is used to install a script. |  |
| **Invoke-commands** | This cmdlet executes commands on local and remote computers. |  |
| **Move-Item** | This cmdlet is used to move an item from one location to another. |  |
| **Move-ItemProperty** | This cmdlet is used to move the property of an item from one location to another. |  |
| **New-alias** | This cmdlet creates a new alias. |  |
| **New-Item** | This cmdlet creates a new item. |  |
| **New-ItemProperty** | This cmdlet creates a new property for an item and sets its value. |  |
| **New-Service** | This cmdlet creates a new Window service. |  |
| **New-variable** | This cmdlet creates a new variable. |  |
| **Read-Host** | This cmdlet reads a line of inputs from the console. |  |
| **Remove-computer** | This cmdlet removes a local computer from its domain. |  |
| **Remove-Item** | This cmdlet deletes the particular item. |  |
| **Remove-ItemProperty** | This cmdlet deletes the property & its value from an item. |  |
| **Remove-job** | This cmdlet removes the background job of Windows PowerShell. |  |
| **Remove-PSBreakpoint** | This cmdlet deletes the breakpoint from the current console. |  |
| **Remove-variable** | This cmdlet deletes a variable with its value. |  |
| **Rename-computer** | This cmdlet is used to rename a computer. |  |
| **Restart-Service** | This cmdlet stops and starts one or more services. |  |
| **Restart-computer** | This cmdlet is used to restart the Windows operating system on local and remote computers. |  |
| **Resume-job** | This cmdlet is used to restart a suspended job. |  |
| **Save-Help** | It is used to download and save the newest help files to a file system directory. |  |
| **Save-packages** | This cmdlet is used to save the packages to the local computer without installing them. |  |
| **Save-Script** | This cmdlet is used to save a script. |  |
| **Select-string** | This cmdlet is used to find the text in string or files. |  |
| **Send-MailMessage** | This cmdlet is used to send an e-mail message. |  |
| **Set-Alias** | This cmdlet creates or changes the alias for a cmdlet in the current Windows PowerShell. |  |
| **Set-content** | This cmdlet writes the content in a file. |  |
| **Set-Date** | This cmdlet changes the time of the system. |  |
| **Set-Item** | This cmdlet changes the value of an item to the value specified in the command. |  |
| **Set-ItemProperty** | This cmdlet change or creates the value of the property of an item. |  |
| **Set-Location** | This cmdlet is used to set the current working location to a particular location. |  |
| **Set-PSBreakpoint** | This cmdlet sets a breakpoint on a command, line or a variable. |  |
| **Set-ScheduledJob** | This cmdlet changes the scheduled Job. |  |
| **Set-Service** | This cmdlet stops, start and suspend services, and changes its properties. |  |
| **Set-variable** | This cmdlet sets the value of the variable. |  |
| **Show-command** | This cmdlet creates the Windows PowerShell commands in a graphical command Window. |  |
| **Sort-Object** | This cmdlet sorts the object by the property value. |  |
| **Start-Job** | This cmdlet starts a background job of Windows PowerShell. |  |
| **Start-Process** | This cmdlet starts on or more process on a local computer. |  |
| **Start-services** | This cmdlet starts one or more services which are stopped. |  |
| **Start-transaction** | This cmdlet starts a transaction. |  |
| **Stop-Computer** | This cmdlet shut down the local and remote computers. |  |
| **Stop-Job** | This cmdlet stops a background job of Windows PowerShell. |  |
| **Stop-Process** | This cmdlet stops one or more processes. |  |
| **Stop-Services** | This cmdlet stops one or more running services. |  |
| **Suspend-jobs** | This cmdlet temporarily stops the workflow jobs. |  |
| **Suspend-Service** | This cmdlet suspends or pauses one or more running services. |  |
| **Undo-transaction** | This cmdlet rollbacks the active transaction. |  |
| **Uninstall-module** | This cmdlet is used to uninstall a module. |  |
| **Uninstall-Package** | This cmdlet uninstalls one or more packages of software. |  |
| **Unregister-ScheduledJob** | This cmdlet deletes the scheduled jobs on the local computer |  |
| **Update-Help** | This cmdlet is used to download and installs the newest help files on the computer. |  |
| **Write-Output** | This cmdlet sends the particular object down the pipeline to the next command. |  |

## Arithmetic Operators Precedence





Example – using switch case

write-host "enter the value of day between 1 to 7"

$day = read-host

switch($day){

1{echo "The day is Sunday"}

2{echo "The day is Monday"}

3{echo "The day is Tuesday"}

4{echo "The day is Wednesday"}

5{echo "The day is Thursday"}

6{echo "The day is Friday"}

7{echo "The day is Saturday"}

}

Example – calculator using switch

write-host "Enter value of a"

$a = read-host

write-host "Enter value of b"

$b = read-host

write-host "Enter your operator to perform the calculation `nA. Addition `nB.Subtraction `nC. Multiplication `nD. Modulo `nE. Division"

$opt = read-host

switch($opt){

A{echo "The addition of a and b is sum =$(( $a + $b ))"}

B{echo "The subtraction of a and b is sub=$(( $a - $b ))"}

C{echo "The multiplication of a and b is mul=$(( $a \* $b ))"}

D{echo "The modulo of a and b is mod=$(( $a % $b ))"}

E{echo "The division of a and b is div=$(( $a / $b ))"}

}

Example – cheking service

$ServiceName = 'EventLog'

$ServiceInfo = Get-Service -Name $ServiceName

if ($ServiceInfo.Status -ne "Running"){

Write-Host "Service is not Started"

}

else{

Write-Host "The service is already running"

}