operators

<table>
<thead>
<tr>
<th>Useful Commands</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Useful Commands</strong></td>
<td><strong>Operators</strong></td>
</tr>
<tr>
<td>Update-Help</td>
<td>Downloads and installs newest help files</td>
</tr>
<tr>
<td>Get-Help</td>
<td>Displays information about commands and concepts</td>
</tr>
<tr>
<td>Get-Command</td>
<td>Gets all commands</td>
</tr>
<tr>
<td>Get-Member</td>
<td>Gets the properties and methods of objects</td>
</tr>
<tr>
<td>Get-Module</td>
<td>Gets the modules that have been imported or that can be imported into the current session</td>
</tr>
<tr>
<td><strong>Assignment Operators</strong></td>
<td>=, +=, -=, *=, /=, %=, ++, --</td>
</tr>
<tr>
<td><strong>Comparison Operators</strong></td>
<td>-eq, -ne, -gt, -ge, -lt, -le, -replace</td>
</tr>
<tr>
<td>“abcde” -replace “bc”, “TEST”</td>
<td>“abcddefghi” -split “de”</td>
</tr>
<tr>
<td>-match, -notmatch, -like, -notlike, -contains, -notcontains</td>
<td>Regular expression matching, Wildcard matching</td>
</tr>
<tr>
<td>1,2,3,4,5 -contains 3</td>
<td>Returns TRUE if the scalar value on its right is contained in the array on its left</td>
</tr>
<tr>
<td>-in, -notin</td>
<td>Returns TRUE only when test value exactly matches at least one of the reference values</td>
</tr>
<tr>
<td>“Windows”—in “Windows”, “PowerShell”</td>
<td></td>
</tr>
<tr>
<td><strong>Bitwise Operators</strong></td>
<td>-.band, -bor, -bxor, -bnot, -shl, -shr</td>
</tr>
<tr>
<td>Bitwise shift operators. Bit shift left, bit shift right (arithmetic for signed, logical for unsigned values)</td>
<td></td>
</tr>
<tr>
<td><strong>Other Operators</strong></td>
<td>-Split, -join, -replace</td>
</tr>
<tr>
<td>“abc”, “def”, “ghi” -join “,”</td>
<td></td>
</tr>
<tr>
<td>1..10</td>
<td>Range operator</td>
</tr>
<tr>
<td>..</td>
<td></td>
</tr>
<tr>
<td><strong>Logical Operators</strong></td>
<td>-and, -or, -xor, -not, !</td>
</tr>
<tr>
<td><strong>Redirection Operators</strong></td>
<td>&gt;, &gt;&gt;</td>
</tr>
<tr>
<td>Output streams</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>Success output</td>
</tr>
<tr>
<td>2</td>
<td>Errors</td>
</tr>
<tr>
<td>3</td>
<td>Warning messages</td>
</tr>
<tr>
<td>4</td>
<td>Verbose output</td>
</tr>
<tr>
<td>5</td>
<td>Debug messages</td>
</tr>
<tr>
<td># Writes warning output to warning.txt</td>
<td>Do-Something &gt; warning.txt</td>
</tr>
<tr>
<td># Appends verbose.txt with the verbose output</td>
<td>Do-Something &gt;&gt; verbose.txt</td>
</tr>
<tr>
<td># Writes debug output to the output stream</td>
<td>Do-Something &gt;=1</td>
</tr>
<tr>
<td># Redirects all streams to out.txt</td>
<td>Do-Something &lt;=1</td>
</tr>
<tr>
<td>Do-Something &gt; out.txt</td>
<td></td>
</tr>
</tbody>
</table>
### Arrays

- "a", "b", "c"  
  Array of strings
- 1,2,3  
  Array of integers
- @()  
  Empty array
- @(2)  
  Array of one element
- 1,2,3,4  
  Array within array
- "hi"  
  Array of one element
- $arr[5]  
  Sixth element of array*
- $arr[2..20]  
  Returns elements 3 thru 21
- $arr[-1]  
  Returns the last array element
- $arr[-3..-1]  
  Displays the last three elements of the array
- $arr[1,4..6.9]  
  Displays the elements at index positions 1,4, and 6 through 9
- @(Get-Process)  
  Forces the result to an array using the array sub-expression operator
- $arr1=1..10  
  $arr2=[$arr1.length-1].0  
  Reverses an array
- $arr1++ 200  
  Adds to an existing value of the second array item (increases the value of the element)
- $b = $arr[0,1+3..6]  
  Creates a new array based on selected elements of an existing array
- $z = $arr + $b  
  Combines two arrays into a single array, use the plus operator (+)

*Arrays are zero-based

### Associative Arrays (Hash tables)

- $hash = @{}  
  Creates empty hash table
- @{$foo=1; bar='value2'}  
  Creates and initialize a hash table
- [ordered]@{$a=1; $b=2; $c=3}  
  Creates an ordered dictionary
- $hash.key1 = 1  
  Assigns 1 to key key1

### Hash keys

- $hash.key1  
  Returns value of key1
- $hash["key1"]  
  Returns value of key1
- $hash.GetEnumerator | sort Key  
  Sorts a hash table by the Key property
- {[pscustomobject] @(x=1; y=2)}  
  Creates a custom object

### Methods

- $hash.key1  
  Returns value of key1
- $hash["key1"]  
  Returns value of key1
- $hash.GetEnumerator | sort Key  
  Sorts a hash table by the Key property
- {[pscustomobject] @(x=1; y=2)}  
  Creates a custom object

### Comments

- # This is a comment because # is the first character of a token
- $a = "#This is not a comment..."
- $a = "something" # ...but this is.
- Write-Host Hello#world

### Block Comments

- A multi-line comment #>

### Object Properties

- An object’s properties can be referenced directly with the "." operator.
- $a = Get-Date  
  $a | Get-Member -MemberType Property
- $a.Date  
  $a.TimeOfDay.Hours  
  $a | Get-Member -MemberType Property -Static

### Strings

- "This is a string, this $variable is expanded as is $(2+2)"
- "This is a string, this $variable is not expanded"
- @'
  This is a here-string which can contain anything including carriage returns and quotes. Expressions are evaluated: $(2+2*5). Note that the end marker of the here-string must be at the beginning of a line!

### Variables

- Format: $scope:name or $anyname or $any path
- $path = "C:\Windows\System32"
- Get-Childitem $env:ProgramFiles(x86)
- $processes = Get-Process
- $global:a = 1 # visible everywhere
- $local:a = 1 # defined in this scope and visible to children
- $private:a = 1 # same as local but invisible to child scopes
- $script:a = 1 # visible to everything is this script
- # Using scope indicates a local variable in remote commands and with Start-Job
- $localVar = Read-Host "Directory, please"
- Invoke-Command -ComputerName localhost -ScriptBlock {
  dir $using:localVar
}
- Start-Job { dir $using:localVar -Recurse}
- $env:Path += ";D:\Scripts"
Get-Command -Noun Variable # the Variable Cmdlets
Get-ChildItem variable: # listing all variables using the variable drive

# strongly-typed variable (can contain only integers)
[int]$number=8

# attributes can be used on variables
[ValidateRange(1,10)]$number = 1
$number = 11 #returns an error

# flip variables
$a=1;$b=2
$a,$b = $b,$a

# multi assignment
$a,$b,$c = 0
$a,$b,$c = 'a','b','c'
$a,$b,$c = 'a b c'.split()

# create read only variable (can be overwritten with -Force)
Set-Variable -Name ReadOnlyVar -Value 3 -Option ReadOnly

# create Constant variable (cannot be overwritten)
Set-Variable -Name Pi -Value 3.14 -Option Constant

Windows PowerShell Preference Variables

$ConfirmPreference Determines whether Windows PowerShell automatically prompts you for confirmation before running a cmdlet or function
$DebugPreference Determines how Windows PowerShell responds to debugging
$ErrorActionPreference Determines how Windows PowerShell responds to a non-terminating error
$ErrorView Determines the display format of error messages in Windows PowerShell
$FormatEnumerationLimit Determines how many enumerated items are included in a display
$MaximumHistoryCount Determines how many commands are saved in the command history for the current session

$Host Reference to the application hosting the POWERSHELL language
$Input Enumerator of objects piped to a script
$LastExitCode Exit code of last program or script
$Matches Exit code of last program or script
$MyInvocation An object with information about the current command
$PSHome The installation location of Windows PowerShell
$profile The standard profile (may not be present)
$Switch Enumerator in a switch statement
$True Boolean value for TRUE
$False Boolean value for FALSE
$PSCulture Current culture
$PSUCulture Current UI culture
$PsVersionTable Details about the version of Windows PowerShell
$PwD The full path of the current directory

$OFS Output Field Separator. Specifies the character that separates the elements of an array when the array is converted to a string. The default value is: Space.
$OutputEncoding Determines the character encoding method that Windows PowerShell uses when it sends text to other applications
$PSDefaultParameterValues Specifies default values for the parameters of cmdlets and advanced functions
$PSEmailServer Specifies the default e-mail server that is used to send e-mail messages
$PSModuleAutoLoadingPreference Enables and disables automatic importing of modules in the session. "All" is the default.
$PSSessionApplicationName Specifies the default application name for a remote command that uses WS-Management technology
$PSSessionConfigurationName Specifies the default session configuration that is used for PSessions created in the current session
$PSSessionOption Establishes the default values for advanced user options in a remote session
$VerbosePreference Determines how Windows PowerShell responds to verbose messages generated by a script, cmdlet or provider
$WarningPreference Determines how Windows PowerShell responds to warning messages generated by a script, cmdlet or provider
$WhatIfPreference Determines whether WhatIf is automatically enabled for every command that supports it

Windows PowerShell Automatic Variables (not exhaustive)

$$ Last token of the previous command line
$? Boolean status of last command
$^ First token of the previous command line
$_,$PSItem Current pipeline object
$Args Arguments to a script or function
$Error Array of errors from previous commands
$ForEach Reference to the enumerator in a foreach loop
$Home The user’s home directory
# Windows PowerShell Learning Resources

## Microsoft Resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows PowerShell</td>
<td><a href="http://www.microsoft.com/powershell">http://www.microsoft.com/powershell</a></td>
</tr>
<tr>
<td>Windows PowerShell Team Blog</td>
<td><a href="http://blogs.msdn.com/PowerShell">http://blogs.msdn.com/PowerShell</a></td>
</tr>
<tr>
<td>MS TechNet Script Center</td>
<td><a href="http://www.microsoft.com/technet/scriptcenter/hubs/msh.mspx">http://www.microsoft.com/technet/scriptcenter/hubs/msh.mspx</a></td>
</tr>
</tbody>
</table>

## Community Resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>PowerShell Community</td>
<td><a href="http://powershellcommunity.org">http://powershellcommunity.org</a></td>
</tr>
<tr>
<td>PowerShell Code Repository</td>
<td><a href="http://poshcode.org">http://poshcode.org</a></td>
</tr>
<tr>
<td>PowerShell.com Community</td>
<td><a href="http://powershell.com">http://powershell.com</a></td>
</tr>
<tr>
<td>PowerGUI.org Community</td>
<td><a href="http://powergui.org">http://powergui.org</a></td>
</tr>
<tr>
<td>PowerShell Community Groups</td>
<td><a href="http://powershellgroup.org">http://powershellgroup.org</a></td>
</tr>
<tr>
<td>PowerShell Magazine</td>
<td><a href="http://powershellmagazine.com">http://powershellmagazine.com</a></td>
</tr>
</tbody>
</table>

## Free eBooks and Guides

<table>
<thead>
<tr>
<th>Title</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secrets of PowerShell Remoting - Don Jones and Dr. Tobias Weltner</td>
<td><a href="http://powershellbooks.com">http://powershellbooks.com</a></td>
</tr>
<tr>
<td>Layman’s Guide to PowerShell 2.0 Remoting - Ravikanth Chaganti</td>
<td><a href="http://www.ravichaganti.com/blog/?page_id=1301">http://www.ravichaganti.com/blog/?page_id=1301</a></td>
</tr>
<tr>
<td>WMI Query Language via PowerShell - Ravikanth Chaganti</td>
<td><a href="http://www.ravichaganti.com/blog/?page_id=2134">http://www.ravichaganti.com/blog/?page_id=2134</a></td>
</tr>
</tbody>
</table>

## Books

- Don Jones, Learn Windows PowerShell in a Month of Lunches
- Bruce Payette, Windows PowerShell in Action, Second Edition