LAZYADMIN

PowerShell Cheat Sheet

Good to know

Ctrl + Space	Display all parameters of the cmdlet.
Get-History -Count x	Get command history
Get-Alias	View all defined aliasses
Notepad $profile	Edit your PowerShell Profile

Operators

$a **+=** 5	$a **-=** 5	Add or Subtract and assign
$a ***=** 2	$a **/=** 2	Multiple or Devide and assign
2 **-eq** $a	2 **-ne** $a	Equal or Not Equal Variable on right side!
$a **-gt** 2	$a **-lt** 2	Greater or Less then
2 **-ge** $a	2 **-le** $a	Greater/less Than or Equal Variable on right side!

The comparison operators below also have a counterpart that start with -not. For example, -notlike or -notin

"file.txt" **-like** "*.txt"	Pattern matching using wildcards
"FileName" **-match** "^File"	Regex matching
1 **-in** @(1,3,5)	Pattern matching using wildcards
@(1,3,5) **-contains** 2	Check if collection contains

Variables & Objects

$var **=** "Hello"	Assign value to variable
$a, $b **=** "Hello","Bye"	Multiple variable assignment
$range **=** 1..10	Create array with sequence
$_	Get current pipeline object
$null	Null value
[type]$var	Declared typed (int, string) variable
$global:var	Assign variable in global scope

Input/Output

$var = **Read-Host**	Read user input
Write-Host "Hello"	Write to console
Write-Host "$($obj.test)"	Write to console with obj property
Get-Content	Read file content
Import-Csv	Import from CSV file
Export-Csv	Export to CSV file
$var **| Out-GridView**	Output to interactive grid view
$var **| Clip**	Copy results to clipboard
Get-Clipboard	Get results from clipboard

Notes

Running Scripts & Processing

. .script.ps1	Run script in current scope
& .script.ps1	Run script in new scope
Start-Process notepad	Start external process or application
Start-ThreadJob -ScriptBlock { Get-Process **}** PS7	Run script or task in the background

Flow Control Statements

if ($condition**) { } else { }**		Simple If - Else statement
($x -gt 10) **?** "High" **:** "Low"	PS7	Ternary operator for if-else
switch ($var**) { }**		Switch statement
for ($i=0; $i -lt 10; $i++**) { }**		Standard for loop
foreach ($item **in** $collection**) {}**		Loop to iterate collections
while ($condition**) { }**		While loop with a condition
$result = $value **??** "Default"	PS7	Assigns default if value is null
$value **??=** DefaultValue	PS7	Assigns if null
$result = ${object}**?.**Property	PS7	Access prop. if object exists
$element = ${array}**?**[index]	PS7	Access element if array exists

Collections & Hashtables

$array = **@(**'item1', 'item2', 'item3'**)**	Create array with value
$array**[**index**]**	Access array element
$array**.Length**	Get array length
$array.**Add(**item**)**	Add item to array
$array.**Remove(**item**)**	Remove item from array
$hash = **@{** key1 = 'value1'; key2 = 'value2';**}**	Create hash table
$hash.key1	Access hash key
$hash.key2 = 'new value'	Assign value to hash key
$hash.**Add(**'key3', 'value'**)**	Add key-value pair
$hash.**Remove(**'key2'**)**	Remove key-value pair
$obj = **[PSCustomObject]@{** Prop1 = 'Val1' Prop2 = 'Val2' **}**	Custom object

Pipeline and Formatting

Get-Process **| Sort-Object** -Propery Name
Get-ChildItem *.txt **| Where-Object** {$_.Length -gt 1KB}
Get-Process **| Select-Object** -Property Name, ID, CPU
Get-Process **| ForEach-Object** {$_.ProcessName}
Get-ChildItem **| Format-List** -Property Name, Length
Get-Process **| Format-Table** -Property ID, CPU **-AutoSize**