Must have commands

Command	Meaning
`gsv | clip`	Clips output of `gsv` to the clipboard

Basic commands

Command	Meaning
`cls` & `Clear-Host`	clear the screen
`cd`	change directory
`cd\`	go to root directory
`dir` & `ls` & `Get-Childitem`	shows whats in the directory
`cat` & `Get-Content`	output the file
`Copy` & `cp` & `Copy-Item`	copy the file
`Measure-command {Get-Process}`	Shows you the execution time

Help system

Command	Meaning
`Update-Help -Force`	Update the help
`Save-Help`	Save the help to a local file
`man`	Scrollable
`help`	Alias
`Get-Help "*keyword*"`	Get help
`Get-Help Get-*`	Get all commands which start with “Get-”
`Get-Help`	`-ShowWindow`
	`-Detailed`
	`-Examples`
	`-Full`
	`-Online`
	`-ShowCommand`
`Get-Help -Category Provider`	
`Get-verb`	Get the verbs used in PowerShell
`Get-Alias -Definition get-process`	Get all aliases for `Get-Process`
`Get-Help About_*`	Get all the about topics
`Get-Process | Get-Member`	Get all the methods, properties,... of the object
`Get-Command -Module AD*`	Get all commands from modules which start with AD

Export

Command	Meaning
`Export-csv`	
`Export-Clixml`	
`Out-File`	
`Out-Printer`	
`Out-GridView`	
`Out-File`	
`ConvertTo-Csv`	
`ConvertTo-Html`	

Pipeline

Example: you want to check if a service is running on all the computers in the AD

Command	Meaning
`Get-ADComputer -Filter * | gm`	Check what object you are working with (in this case `ADComputer`)
`Get-Help Get-Service -ShowWindow`	Search for `-InputObject` and see what it accepts (in this case `ServiceController` != `ADCom` `-ComputerName` supports `ByPropertyName`
`Get-ADComputer -Filter *`	Check for propertyname `-Name` gives you the name of the computer

```
Get-ADComputer -Filter * -Name | Select -Property @{n(ame)='ComputerName';e(xpression)={$_.name}} |
Get-Service -Name bits
```

Example: command does not accept pipeline input (Get-WmiObject)

```
Get-WmiObject -class win32_bios -ComputerName (Get-ADComputer -Filter *).Name
```

Or

```
Get-ADComputer -Filter * | Get-WmiObject win32_bios -Computername {$_.Name}
```

Create new property

Command	Meaning
`Select -Property @{n(ame)='ComputerName';e(xpression)={$_.name}}`	Creates a new property called `ComputerName` from the
`Select -ExpandProperty name`	Return the property in a array

WmiObject & CimInstance

Remoting

Command	Meaning
`Invoke-Command -ComputerName dc {Restart-Computer}`	Execute the command on specific computers
`icm dc {Restart-Computer}`	Short version

Execution Policy

Command	Meaning
`Restricted`	
`Unrestricted`	Run any script **(not recommended)**
`AllSigned`	
`RemoteSigned`	Only run scripts that are remote signed
`Bypass`	
`Undefined`	

Powershell Remote Session

Command	Meaning
`Get-PSSession`	Get open sessions
`$sessions = New PSSession -ComputerName dc`	Open session
`icm -Session $sessions {...}`	Execute command in script block to all sessions

Parallelism

Parallel: `icm -Computername (Get-Content servers.txt) { ... }`

Serial: `foreach ($s in (Get-Content servers.txt) ) { icm -Computername $s { ... }}`

Scripting

Command	Meaning
`CTRL + J`	Helps you build a script
`$var = 1`	Declares new variabele `var` with value `1`
`param($ComputerName)`	Create new parameter
`[ValidateSet("a","b","c")][string]$x = "a"`	Value of $x **must** be a value of the ValidateSet
`"this is $var"`	" Resolves the value of $var
`'this is $var'`	' Does **not** resolve the value
`"The value of `$var is $var"`	the ‘ (```) prevents that the value is resolved
`@' '@`	Multiline string, usefull for `New-Snippet`
`-ErrorAction`	
`-ErrorVariabele`	