# Lab Exercises

This section of the PowerShell training is supposed to be executed in a PowerShell console or PowerShell ISE. Results may vary depending if the commands are executed in the console, the ISE, 32&64-bit environment or running with administrative privileges. Feel free to experiment with the alternatives to discover the differences.

If there are any questions or if there is anything you would like to have explained in greater detail, feel free to contact me directly. As always there are a hundred different ways of achieving the same result, so trying out different approaches for the same problem can be a learning experience.

## Exercise 1 – Working with different methods for creating loops

Using the foreach key word it is possible to create a loop, experiment with the following code:

foreach ($Number in 1..10) {

"The number is: $Number"

}

It is also to create a for-loop, this is similar to how loops were created in a batch file:

$Files = Get-ChildItem -Path $Env:USERPROFILE\Documents -File

for ($i = 0; $i -lt $Files.Count; $i++) {

"File($i): {0}" -f $Files[$i].Name

}

The preferred method for creating loops is by utilizing the PowerShell pipeline in combination with the ForEach-Object cmdlet:

Get-Content -Path $env:windir\win.ini | ForEach-Object {

"The win.ini contains: '$PSItem'"

}

## Exercise 2 – Working with different methods for creating loops

Using do-while-until it is possible to create loops that run until a condition is reached:

$Count = 1

do {

"The counter is $Count"

$Count++

} while ($Count -le 10)

Alternatively, this block can also be changed to this by omitting the do keyword:

$Count = 1

while ($Count -le 10) {

"The counter is $Count"

$Count++

}

To see this in a real world scenario, attempt to run the following code:

Stop-Service -Name wuauserv

do {

Start-Sleep -Seconds 5

Start-Service wuauserv

} until ((Get-Service wuauserv).Status -eq 'Running')

'Windows Update service is {0}' -f (Get-Service wuauserv).Status

To learn more about the operators that are used in these cmdlets read through the following help document:

Get-Help about\_Operators -ShowWindow

## Exercise 3 – Using if-else and switch statements to control flow

Using the Get-Random cmdlet we can select a random object from an array of objects, using this

$RandomValue = 1..5 | Get-Random

if ($RandomValue -eq 1) {

'The value is one'

} elseif ($RandomValue -eq 2) {

'The value is two'

} elseif ($RandomValue -eq 3) {

'The value is three'

} elseif ($RandomValue -eq 4) {

'The value is four'

} else {

'The value is not 1 through 4'

}

It is also possible to use the switch statement to generate

$RandomValue = 1..5 | Get-Random

switch ($RandomValue) {

1 {'The value is one'}

{$\_ -eq 2} {'The value is two'}

{$\_ -gt 2} {'The value is greater than two'}

Default {'The value is not as expected'}

}

## Exercise 4 – Create a loop that renames any file starting with an a

In this exercise the goal is to create a loop that looks at a list of files and renames all files that start with the letter a to ***“<filename>.<extension>.ThisFileIsRenamed”***. Ensure you execute this in a folder that contains a number of files that can be renamed, otherwise include the –WhatIf parameter. When writing this code it try to use the pipeline where possible, but using variables to store the data is also a correct solution. Don’t forget about using the following cmdlets to discover the available commands:

* Get-Help
* Get-Command