OPERATING SYSTEMS CCGC-5000

Module - 10

Agenda

Authentic information is available from the given resources in course outline and URL's mentioned from this slides, and this presentation is only supportive document to be read with the given resources and corrected accordingly if required..

- PowerShell
- PowerShell Cmdlets
- PowerShell Scripting

https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.1
https://docs.microsoft.com/en-us/learn/modules/introduction-to-powershell/2-what-is-powershell

Power Shell

- What is PowerShell ?
  - Windows PowerShell is a command-line interface that offers a shell, a customized environment for executing commands and scripts.
  - PowerShell is a cross-platform task automation solution made up of a command-line shell, a scripting language, and a configuration management framework. PowerShell runs on Windows, Linux, and macOS.
  - PowerShell is modern command shell that includes the best features of other popular shells. Unlike most shells that only accept and return text, PowerShell accepts and returns .NET objects
  - As a scripting language, PowerShell is commonly used for automating the management of systems. It is also used to build, test, and deploy solutions, often in CI/CD (Continuous Integration/Continuous Deployment)environments. PowerShell is built on the .NET Common Language Runtime (CLR). All inputs and outputs are .NET objects.

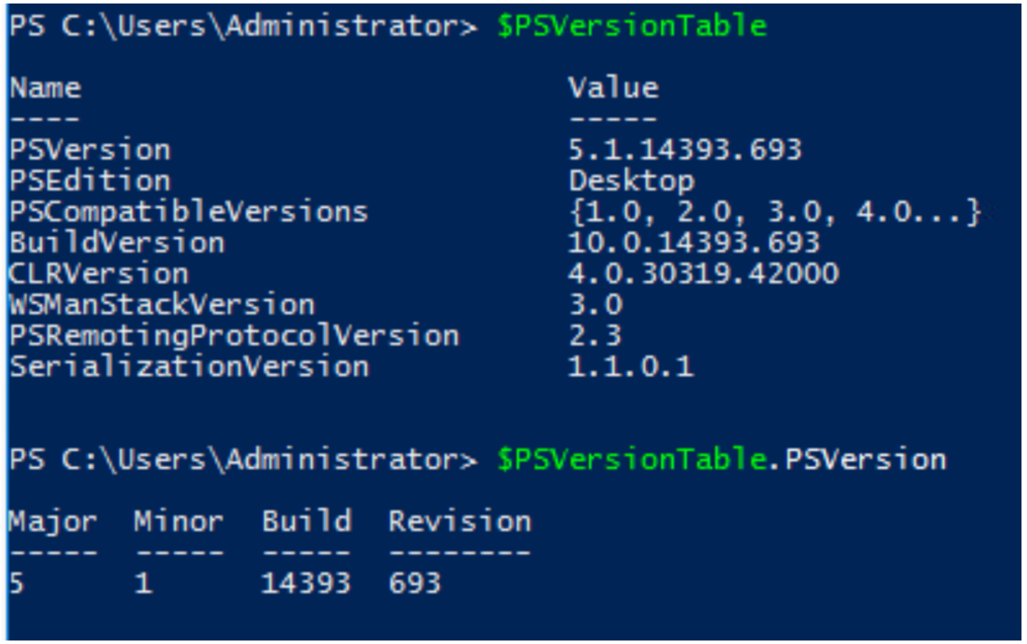
https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.1

https://docs.microsoft.com/en-us/learn/modules/introduction-to-powershell/2-what-is-powershell

HUMBER
Faculty of Applied Sciences & Technology

Powershell

- Powershell is 64 bit
- Powershell(x86) is 32 bit
- Powershell ISE is Integrated Scripting Environment
- Latest version is 7.2 (stable release)
- To check version in PowerShell : **$PSVersionTable**
- Version 7.2.0, where 7 is Major version, 2 is Minor version and 0 is Patch which can be checked with **$PSVersionTable.PSVersion**



```
PS C:\Users\Administrator> $PSVersionTable

Name                           Value
----                           -----
PSVersion                      5.1.14393.693
PSEdition                      Desktop
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
BuildVersion                   10.0.14393.693
CLRVersion                     4.0.30319.42000
WSManStackVersion              3.0
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1


PS C:\Users\Administrator> $PSVersionTable.PSVersion

Major  Minor  Build  Revision
-----  -----  -----  --------
5      1      14393  693
```

WE ARE HUMBER

HUMBER
Faculty of Applied Sciences & Technology

PowerShell - Cmdlets

- Cmdlets are developed in .NET or .NET Core
- Cmdlets are compiled commands
- Cmdlets names are standard in verb-noun format
- **Get- Verb** lists approved verbs
- **Get-Command** lists all available cmdlets
- **Get-Help** invokes built-in help
- **Get-Help** *cmdlet* displays help for the cmdlet.

https://docs.microsoft.com/en-us/powershell/scripting/developer/cmdlet/cmdlet-overview?view=powershell-7.1

WE ARE HUMBER

cmdlets

Anatomy of cmdlet
Set -> Verb
Location -> Noun
Path -> Parameter
C:\Windows -> Parameter Value

- To get Computer Info: Get-ComputerInfo
- To get location: Get-Location
- To change location: Set-Location -Path C:\Windows -PassThru
  (PassThru to get information about the result)
- To get AD domain name in the DC : Get-ADDomain
- Filtering of specific info can be done with |, For example to filter only the Name and DNSRoot from Get-ADDomain output:
  Get-ADDomain |select Name, DNSRoot
- Similarly, try Get-ADForest, Get-ADGroup in the DC.
- cmdlets can get information and also add or modify as required.
- Output of one command can be given as input to second command by using pipelining between first and second command
- Example: **Get-ADDomain |select DistinguishedName**

PowerShell Scripts

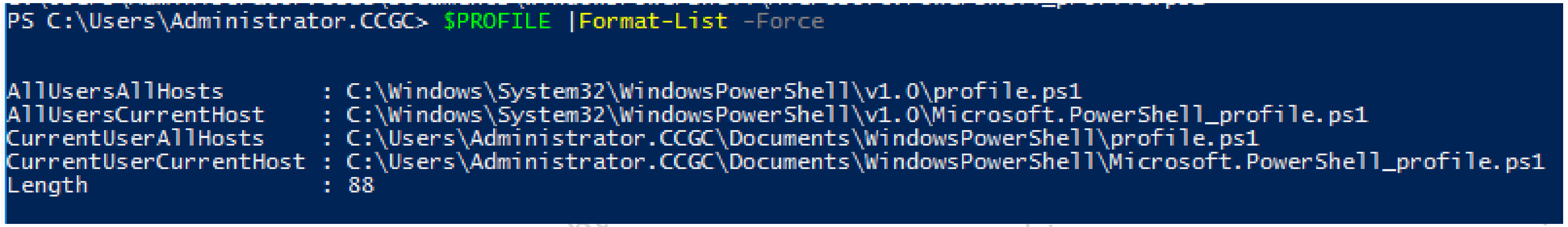
- Powershell scripts can be developed in PowerShell ISE
- Powershell extension Visual Studio Code provides rich language support
- Powershell script the file extension must be **ps1**
- Powershell scripts require ExecutionPolicy permission to execute scripts
- There are four permissions
  - Restricted - No scripts can be run,
  - RemoteSigned - Allows scripts created on the device, but does not run scripts from other computer unless it includes publisher's signature.
  - AllSigned - All scripts will run only if trusted publisher has signed
  - Unrestricted - Runs any scripts without any restriction
- Use **Get-ExecutionPolicy** to know the current permission
- Use **Set-ExecutionPolicy** *permission* to set the required permission.
- It can also be set for current user only : **Set-ExecutionPolicy remotesigned -Scope currentuser**

PS scripting

HUMBER
Faculty of Applied Sciences & Technology

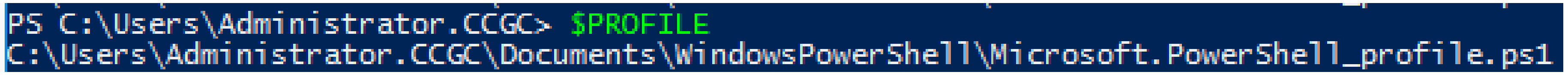
- A PowerShell profile is a script that runs when PowerShell starts.
- Profile can be used as logon script to customize the environment.
- Profiles Description and Name is given in the table, and profiles place holders are given below

DESCRIPTION	NAME
Current User, PowerShell ISE	$PROFILE.CurrentUserCurrentHost, or $PROFILE
All Users, PowerShell ISE	$PROFILE.AllUsersCurrentHost
Current User, All Hosts	$PROFILE.CurrentUserAllHosts
All User, All Hosts	$PROFILE.AllUserAllHosts



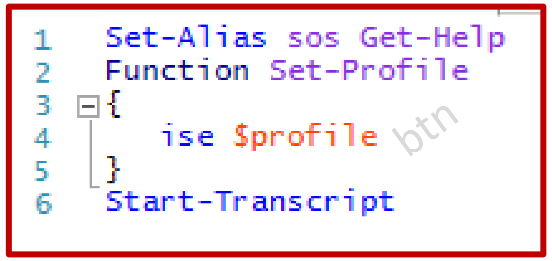
```
PS C:\Users\Administrator.CCGC> $PROFILE |Format-List -Force


AllUsersAllHosts       : C:\Windows\System32\WindowsPowerShell\v1.0\profile.ps1
AllUsersCurrentHost    : C:\Windows\System32\WindowsPowerShell\v1.0\Microsoft.PowerShell_profile.ps1
CurrentUserAllHosts    : C:\Users\Administrator.CCGC\Documents\WindowsPowerShell\profile.ps1
CurrentUserCurrentHost : C:\Users\Administrator.CCGC\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1
Length                 : 88
```

- Cmdlet **Test-path $profile** returns false if no profile exist, which is default.
- Create a new profile: `New-Item -path $profile -itemtype file -force`
- To check your profile :



```
PS C:\Users\Administrator.CCGC> $PROFILE
C:\Users\Administrator.CCGC\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1
```

- A sample profile is given in screenshot
- Line 1, sets **Get-Help** cmdlet to an alias **sos,** Line 2-5, creates a function called Set-Profile which opens Powershell ISE & Line 6, **Start-Transcript** cmdlet creates a record of all or part of a PowerShell Session to a text file.



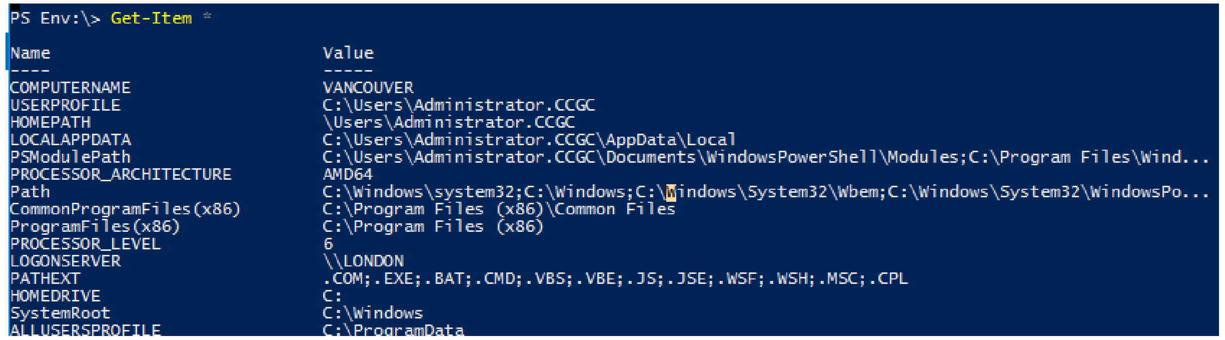
```
1  Set-Alias sos Get-Help
2  Function Set-Profile
3  {
4      ise $profile
5  }
6  Start-Transcript
```

WE ARE HUMBER

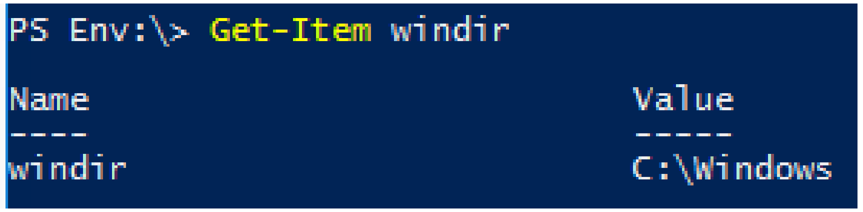
HUMBER
Faculty of Applied Sciences & Technology

PowerShell Scripting

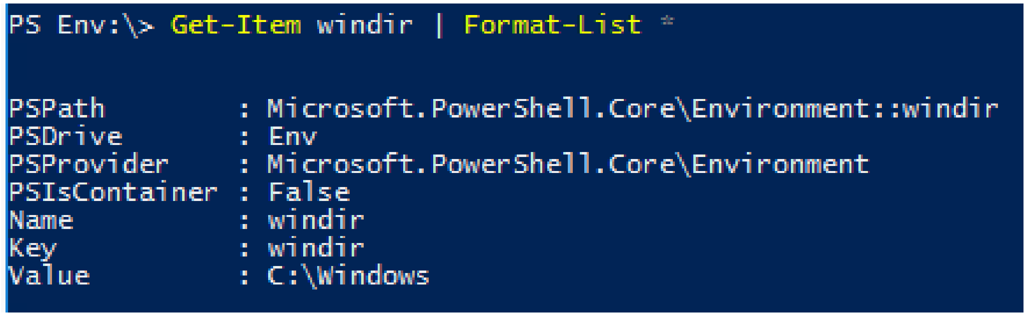
- Environment Variables stores information about the operating system environment
- To display Environmental Variables:
  - View drives in current session: **Get-PSDrive**
  - Set location to ENV:\ : **Set-Location Env:**
  - To display Environmental Variables: **Get-Item ***
- To view the value of the environmental variables: **Get-Item** *variablename*
- Get properties & their associated values: **Get-Item** *variablename* **| Format-List ***



```
PS Env:\> Get-Item *

Name                           Value
----                           -----
COMPUTERNAME                   VANCOUVER
USERPROFILE                    C:\Users\Administrator.CCGC
HOMEPATH                       \Users\Administrator.CCGC
LOCALAPPDATA                   C:\Users\Administrator.CCGC\AppData\Local
PSModulePath                   C:\Users\Administrator.CCGC\Documents\WindowsPowerShell\Modules;C:\Program Files\Wind...
PROCESSOR_ARCHITECTURE         AMD64
Path                           C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPo...
CommonProgramFiles(x86)        C:\Program Files (x86)\Common Files
ProgramFiles(x86)              C:\Program Files (x86)
PROCESSOR_LEVEL                6
LOGONSERVER                    \\LONDON
PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.CPL
HOMEDRIVE                      C:
SystemRoot                     C:\Windows
ALLUSERSPROFILE                C:\ProgramData
```



```
PS Env:\> Get-Item windir

Name                           Value
----                           -----
windir                         C:\Windows
```



```
PS Env:\> Get-Item windir | Format-List *


PSPath        : Microsoft.PowerShell.Core\Environment::windir
PSDrive       : Env
PSProvider    : Microsoft.PowerShell.Core\Environment
PSIsContainer : False
Name          : windir
Key           : windir
Value         : C:\Windows
```

WE ARE HUMBER

https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_environment_variables?view=powershell-7.1

PowerShell Scripting

HUMBER
Faculty of Applied Sciences & Technology

- Variables in powershell scripting are named as $*variablename*
- Example: **$var1** = Get-Date,
- the value assigned in $var1 can be displayed using Write-Host $var1
- To list variable related cmdlets :

  Get-Help *variable | Where-Object category -eq "cmdlet" | Format-List name, category, synopsis
- Cmdlet **Get-Variable** lists both user-defined and system-defined variables
- Cmdlet Get-Variable with PWD lists value of PWD variable
- To store values in array: **$array1** = 8, 12, 19, 22
- To display the 4th array elements: **$array1[3]**
- To find the length of the array: **$array1.length**
- Range of data can also be assigned to array *(assigning 2 to 10)*: **array2 = 2 .. 10**
- To assign data type to the array: [int32[]]$array3 = 100, 200, 300, 400
- To display datatype: $array3.GetType()

https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/get-variable?view=powershell-7.1

HUMBER
Faculty of Applied Sciences & Technology

PowerShell scripts

- Use Powershell ISE, Click on File->New and enter script commands

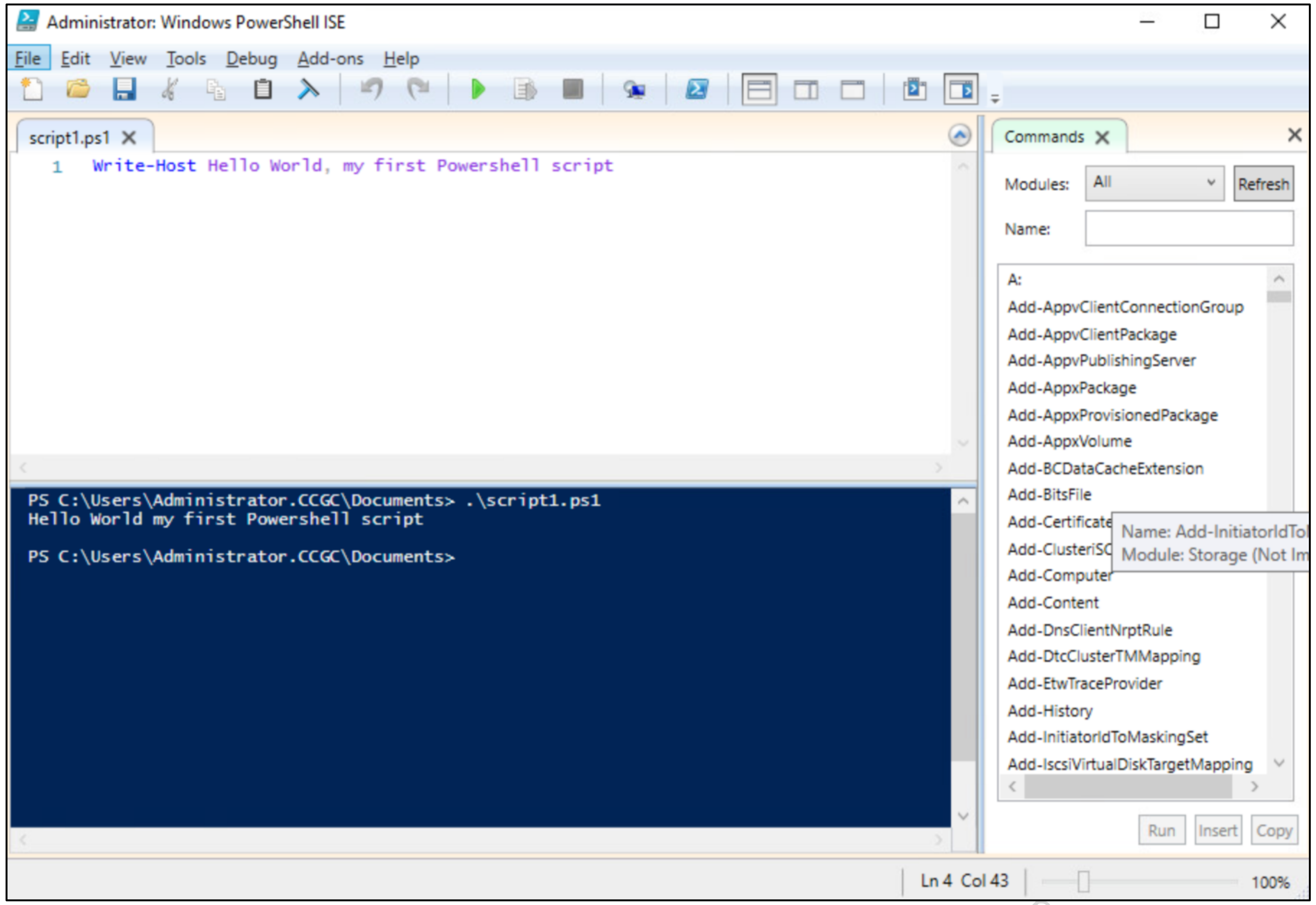
  Write-Host Hello World, my first PowerShell Script

- Save the script and run it in powershell command prompt as **.\*scriptname*.ps1**
- Cmdlet **Read-Host**, reads line of input from console (stdin)
- Getting input in Powershell:

  $course = Read-Host "Enter course "

- Powershell **function**

```
function Welcome ($var1)
{
    Write-Host "Welcome, $var1!"
    Write-Output "Welcome, $var1!"
}
Welcome("User1")
```



```
Write-Host Hello World, my first Powershell script
```

```
PS C:\Users\Administrator.CCGC\Documents> .\script1.ps1
Hello World my first Powershell script

PS C:\Users\Administrator.CCGC\Documents>
```

```
Welcome, User1!
Welcome, User1!
```

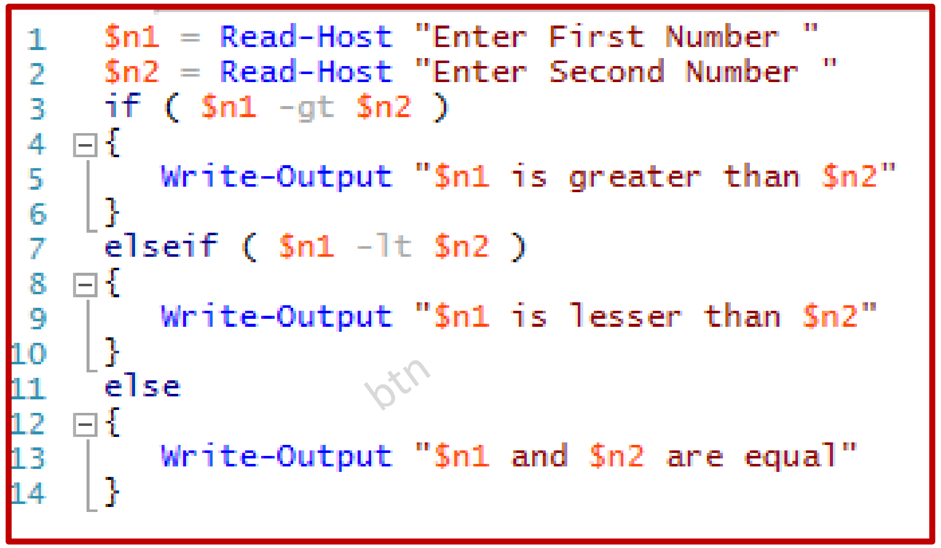
WE ARE HUMBER

https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/write-host?view=powershell-7.1

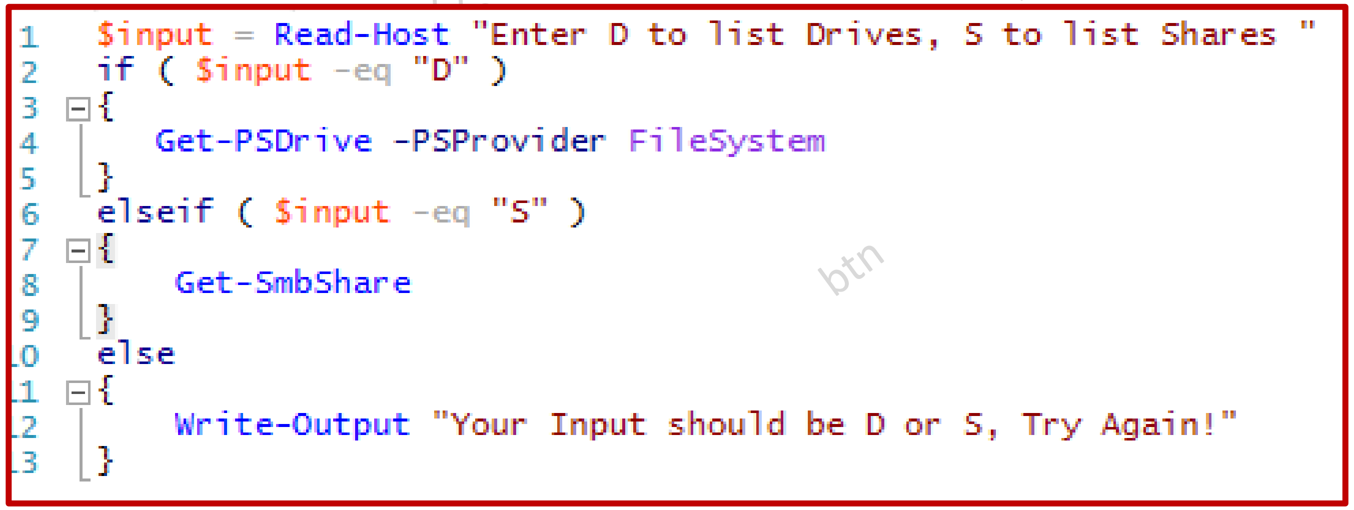
PowerShell Scripting

- **If Statement**

```
if ( condition1 )
{
    Statement1
}
elseif ( condition2 )
{
    Statement2
}
else
{
    Statement3
}
```



```
$n1 = Read-Host "Enter First Number "
$n2 = Read-Host "Enter Second Number "
if ( $n1 -gt $n2 )
{
    Write-Output "$n1 is greater than $n2"
}
elseif ( $n1 -lt $n2 )
{
    Write-Output "$n1 is lesser than $n2"
}
else
{
    Write-Output "$n1 and $n2 are equal"
}
```



```
$input = Read-Host "Enter D to list Drives, S to list Shares "
if ( $input -eq "D" )
{
   Get-PSDrive -PSProvider FileSystem
}
elseif ( $input -eq "S" )
{
    Get-SmbShare
}
else
{
    Write-Output "Your Input should be D or S, Try Again!"
}
```

https://docs.microsoft.com/en-us/powershell/scripting/learn/deep-dives/everything-about-if?view=powershell-7.1

PowerShell Scripting

- **While loop**

  While (Condition)

  {

  Statements

  }

- **Do while loop**

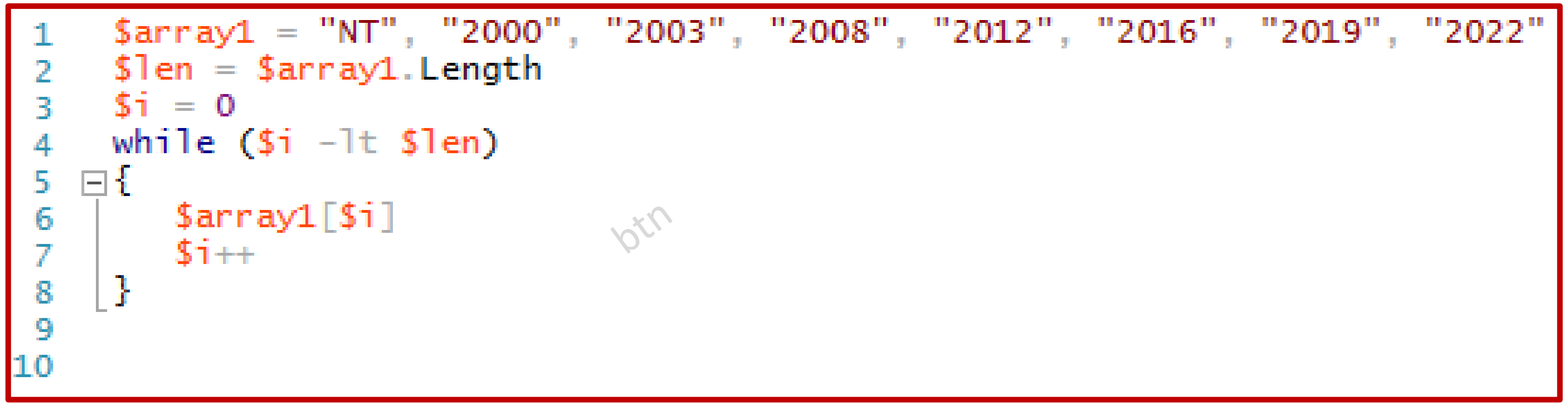
  Do

  {

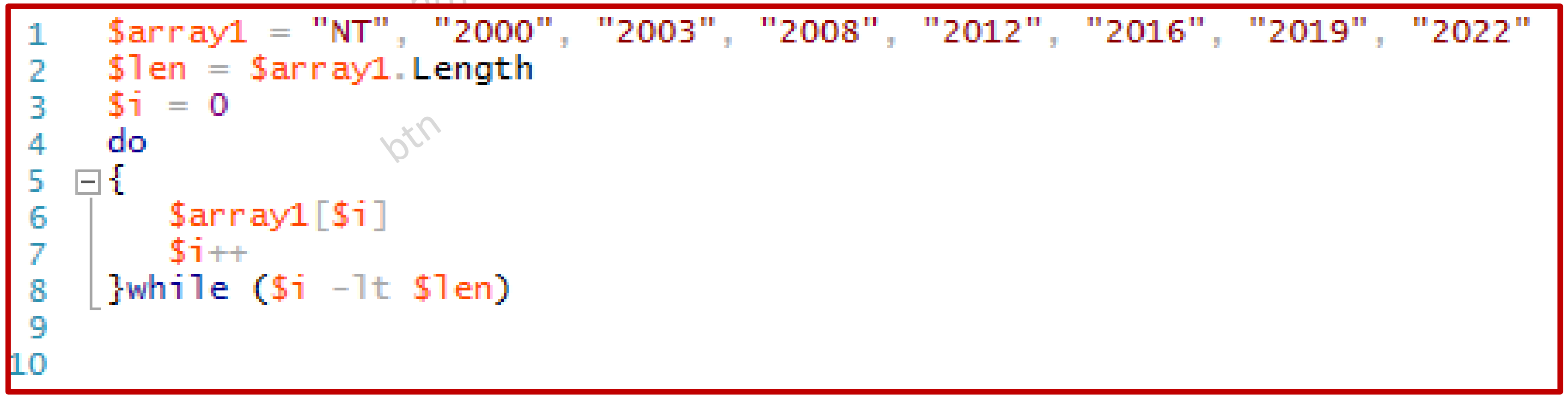
  Statements

  }

  while (condition)



```
$array1 = "NT", "2000", "2003", "2008", "2012", "2016", "2019", "2022"
$len = $array1.Length
$i = 0
while ($i -lt $len)
{
    $array1[$i]
    $i++
}
```



```
$array1 = "NT", "2000", "2003", "2008", "2012", "2016", "2019", "2022"
$len = $array1.Length
$i = 0
do
{
    $array1[$i]
    $i++
}while ($i -lt $len)
```

HUMBER
Faculty of Applied Sciences & Technology

PowerShell Scripting

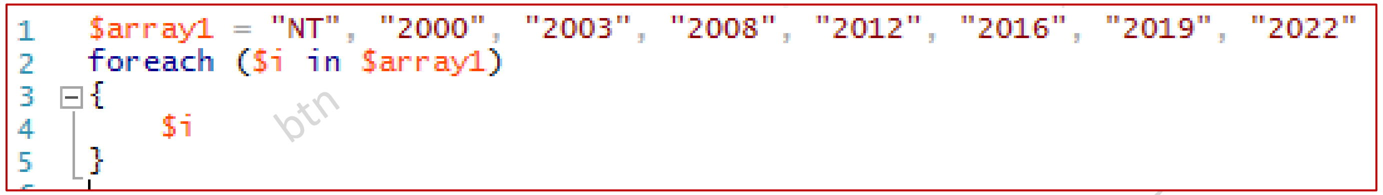
- **<u>foreach loop</u>**

  foreach ($variable **in** collection)

  {

  $variable/Statements

  }



```
$array1 = "NT", "2000", "2003", "2008", "2012", "2016", "2019", "2022"
foreach ($i in $array1)
{
    $i
}
```

https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_foreach?view=powershell-7.1

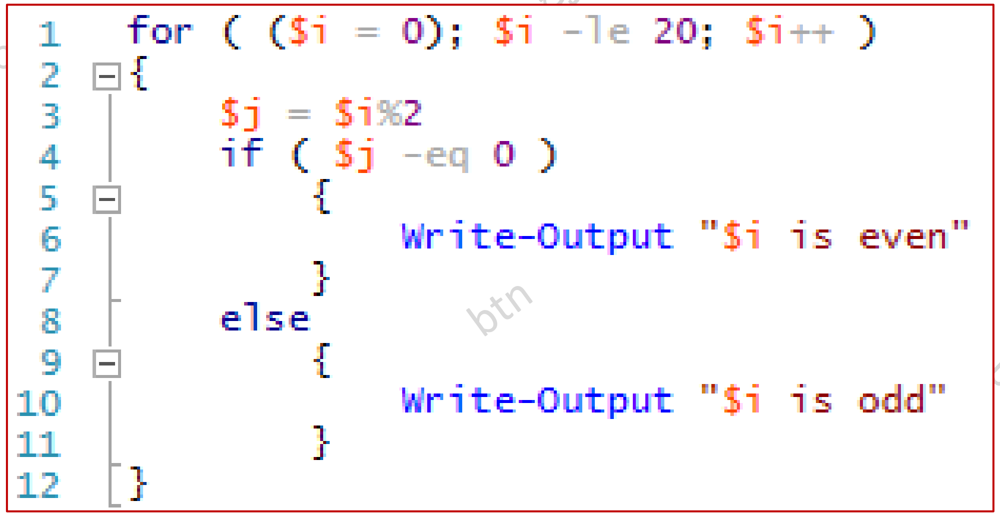
- **<u>for loop</u>**

  for ((init); condition, repeat)

  {

  Statement list

  }



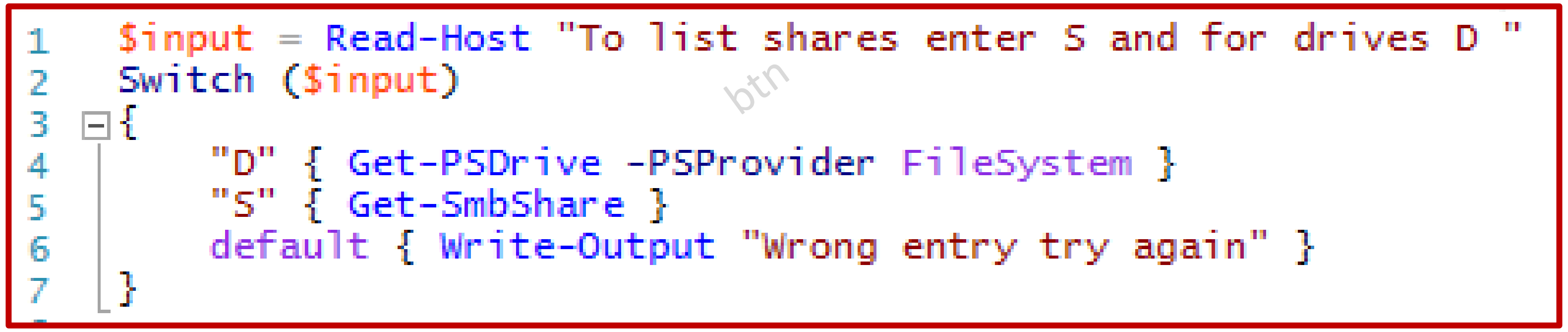
```
for ( ($i = 0); $i -le 20; $i++ )
{
    $j = $i%2
    if ( $j -eq 0 )
        {
            Write-Output "$i is even"
        }
    else
        {
            Write-Output "$i is odd"
        }
}
```

https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_for?view=powershell-7.1

PowerShell Scripting

- **Switch Statement**

```
Switch (<test-value>)
 {
      <condition> {<action>}
      <condition> {<action>}
 }
```

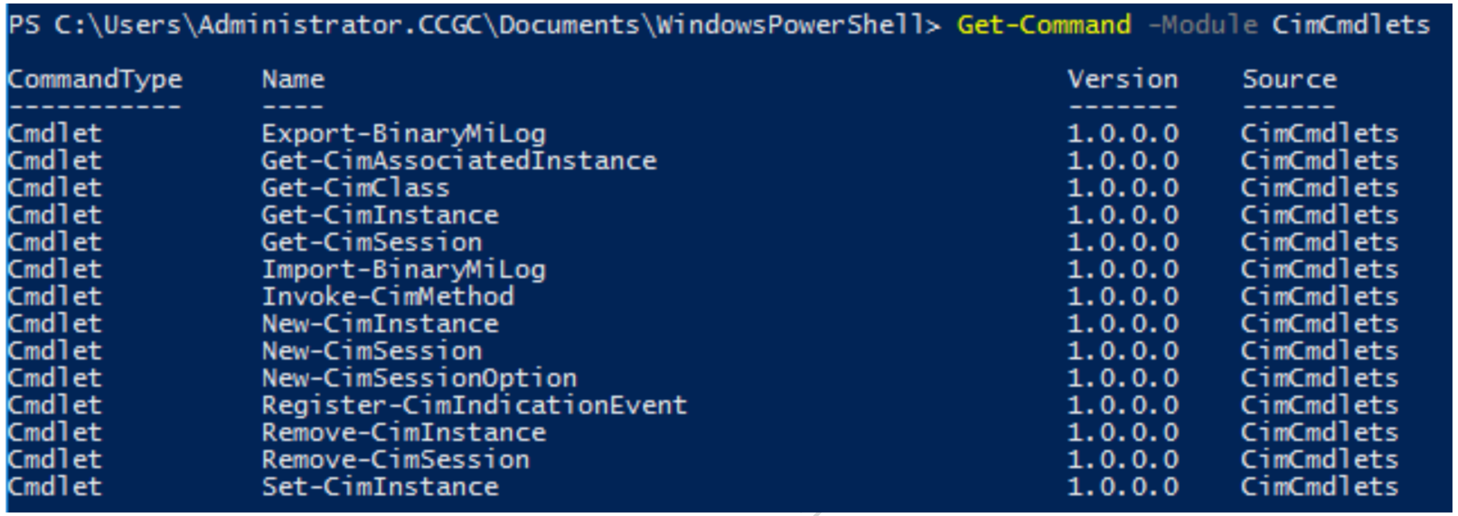


```
$input = Read-Host "To list shares enter S and for drives D "
Switch ($input)
{
    "D" { Get-PSDrive -PSProvider FileSystem }
    "S" { Get-SmbShare }
    default { Write-Output "Wrong entry try again" }
}
```

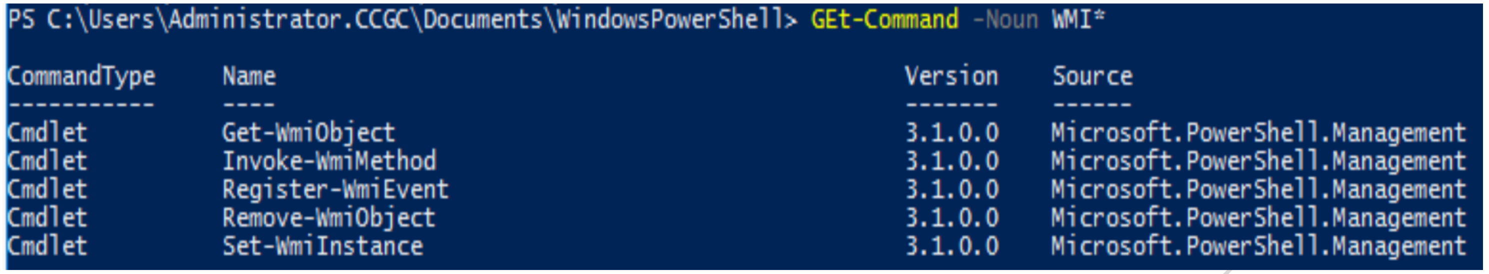
PowerShell Scripting

HUMBER Faculty of Applied Sciences & Technology

- PowerShell provides two similar management interfaces for querying information on computers
- **CIM** (Common Information Model) provides a common definition of management information for systems, networks, applications and services, and allows for vendor extensions. https://www.dmtf.org/standards/cim
- **WMI** (Windows Management Instrumentation) is a core technology for Windows system administration because it exposes a wide range of information in a uniform manner.



```
PS C:\Users\Administrator.CCGC\Documents\WindowsPowerShell> Get-Command -Module CimCmdlets

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Cmdlet          Export-BinaryMiLog                                 1.0.0.0    CimCmdlets
Cmdlet          Get-CimAssociatedInstance                          1.0.0.0    CimCmdlets
Cmdlet          Get-CimClass                                       1.0.0.0    CimCmdlets
Cmdlet          Get-CimInstance                                    1.0.0.0    CimCmdlets
Cmdlet          Get-CimSession                                     1.0.0.0    CimCmdlets
Cmdlet          Import-BinaryMiLog                                 1.0.0.0    CimCmdlets
Cmdlet          Invoke-CimMethod                                   1.0.0.0    CimCmdlets
Cmdlet          New-CimInstance                                    1.0.0.0    CimCmdlets
Cmdlet          New-CimSession                                     1.0.0.0    CimCmdlets
Cmdlet          New-CimSessionOption                               1.0.0.0    CimCmdlets
Cmdlet          Register-CimIndicationEvent                        1.0.0.0    CimCmdlets
Cmdlet          Remove-CimInstance                                 1.0.0.0    CimCmdlets
Cmdlet          Remove-CimSession                                  1.0.0.0    CimCmdlets
Cmdlet          Set-CimInstance                                    1.0.0.0    CimCmdlets
```



```
PS C:\Users\Administrator.CCGC\Documents\WindowsPowerShell> GEt-Command -Noun WMI*

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Cmdlet          Get-WmiObject                                      3.1.0.0    Microsoft.PowerShell.Management
Cmdlet          Invoke-WmiMethod                                   3.1.0.0    Microsoft.PowerShell.Management
Cmdlet          Register-WmiEvent                                  3.1.0.0    Microsoft.PowerShell.Management
Cmdlet          Remove-WmiObject                                   3.1.0.0    Microsoft.PowerShell.Management
Cmdlet          Set-WmiInstance                                    3.1.0.0    Microsoft.PowerShell.Management
```

https://docs.microsoft.com/en-us/powershell/scripting/learn/ps101/07-working-with-wmi?view=powershell-7.1

https://docs.microsoft.com/en-us/powershell/scripting/samples/getting-wmi-objects--get-ciminstance-?view=powershell-7.1

PowerShell Scripting

HUMBER
Faculty of Applied Sciences & Technology

- Get-CimInstance :
  https://docs.microsoft.com/en-us/powershell/module/cimcmdlets/get-ciminstance?view=powershell-7.1

- Get-CimSession:
  https://docs.microsoft.com/en-us/powershell/module/cimcmdlets/get-cimsession?view=powershell-7.1

- Get-WmiObject :
  https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-wmiobject?view=powershell-5.1

- To get a logged in user of a remote system using Get-WmiObject
  Get-WmiObject –ComputerName *ComputerName* –Class Win32_ComputerSystem | Select-Object UserName

- To get a logged in user of a remote system using Get-CmiInstance
  Get-CimInstance –ComputerName *ComputerName* –ClassName Win32_ComputerSystem | Select-Object UserName

https://docs.microsoft.com/en-us/powershell/scripting/learn/ps101/07-working-with-wmi?view=powershell-7.1

WE ARE HUMBER

https://docs.microsoft.com/en-us/powershell/scripting/samples/getting-wmi-objects--get-ciminstance-?view=powershell-7.1

Recommended reading

- Sams Teach Yourself, Windows Shell in 24 hours
- https://docs.microsoft.com/en-us/powershell/module/cimcmdlets/?view=powershell-7.1
- https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_arithmetic_operators?view=powershell-7.1
- https://www.techrepublic.com/blog/10-things/10-fundamental-concepts-for-powershell-scripting/
- https://docs.microsoft.com/en-us/powershell/scripting/samples/sample-scripts-for-administration?view=powershell-5.1