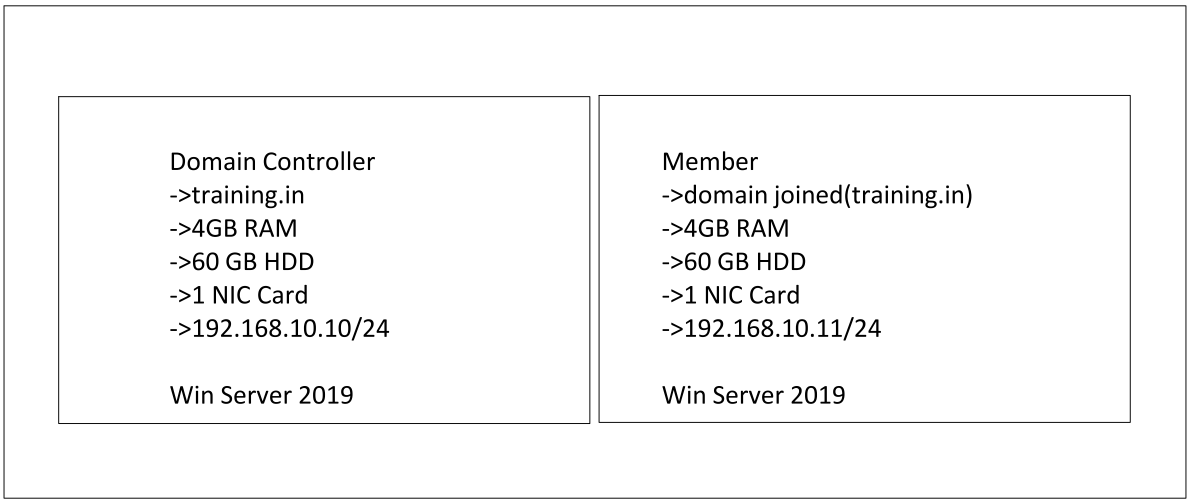
PowerShell Theory~Day1

14 November 2024 09:05

Difference Btw Scripting And Programming(Search)

Reserved Words High In Scripting
Script and command
Suppose if you want to connect your PowerShell to AWS or any cloud based then the commands Will be increased
PowerShell is a automation tool it can automate the job.(10000 user creation example)

HYPER-V (Internal Switch)



Domain Controller
->training.in
->4GB RAM
->60 GB HDD
->1 NIC Card
->192.168.10.10/24

Win Server 2019

Member
->domain joined(training.in)
->4GB RAM
->60 GB HDD
->1 NIC Card
->192.168.10.11/24

Win Server 2019

TASK Automation

Config MGMT

DSC(Desire State Config)->allow you to run a code in any comp (Installing s/w, managing resources Using one machine we can manage all the computers/laptops, scheduling a task)

PowerShell Core is opensource (Win->Win, Win->Linux, Win->MAC)

WSMAN if running it will allow me to connect with the several os.

PowerShell, PowerShell **Integrated Scripting Environment**, Windows Terminal these are the Locations where you run PowerShell cmd.

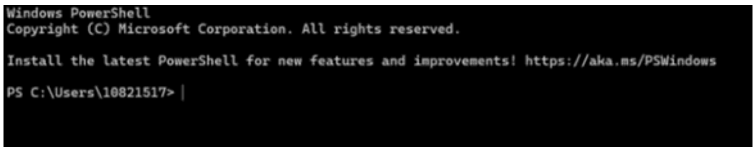
Cmd Prompt VS PowerShell

Cmdlets in PowerShell and Commands in Cmd prompt
Cmdlets syntax -> **verb-noun**
get-date

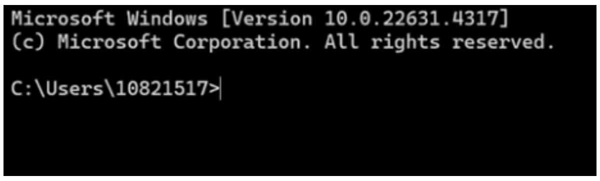
get-process
get-service
new-process
start-process
start-service

These are called Cmdlets
Here capital small no matters.

Manage File system, Registry



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\10821517> |
```



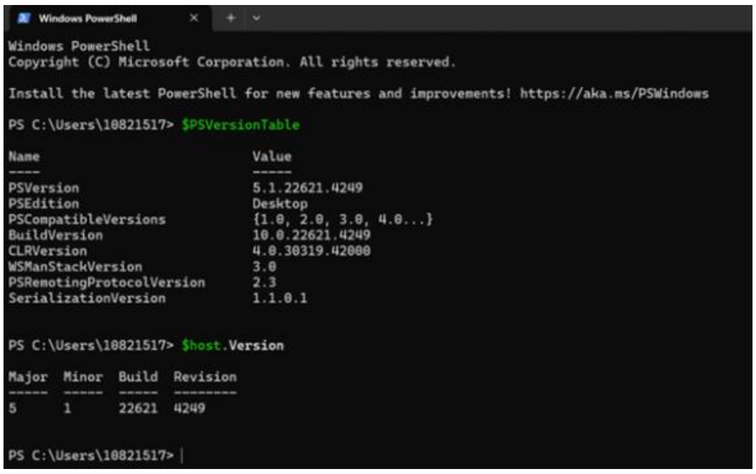
```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\10821517>
```

When you want to automate the entire working the PowerShell is introduced
Connecting All Microsoft products and non MS products PowerShell Does CMD not

PowerShell Versions (See from Jeetu's PPT)

2006 for XP,

Our Current Version



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\10821517> $PSVersionTable

Name                           Value
----                           -----
PSVersion                      5.1.22621.4249
PSEdition                      Desktop
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
BuildVersion                   10.0.22621.4249
CLRVersion                     4.0.30319.42000
WSManStackVersion              3.0
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1


PS C:\Users\10821517> $host.Version

Major  Minor  Build  Revision
-----  -----  -----  --------
5      1      22621  4249


PS C:\Users\10821517> |
```

PowerShell v1~5.1 is a pure MS product and build on dotnet
To open this PowerShell in run type powershell.exe
PowerShell v6.X v7.X is opensource codes are available in GitHub
To open this PowerShell write pwsh.
Ps core is the name for PowerShell V6 and 7.
V1 to 6.X upgradation is not possible it is like you are adding truck in Maruti body.

$->identify a variable

PS VS ISE

White area Scripting PANE In ISE
Allow you to write the script and exe the script

Creating script and writing program
We can use VS code editor also

File Extensions(Jeetu's PPT)

PS1(Default Extension)
PSM1(PowerShell Module files)
These 2 are used to create PowerShell
Module file means in that file we can write all the custom scripts.

Updating PowerShell

To get the latest version
update-help

PowerShell integration with other products(Jeetu's PPT)

NOTE:- PowerShell Core does not support ISE

Day1~Practical PowerShell

14 November 2024 10:06

Create a new dir
New-item -itemtype Directory -Name demo
mkdir demo(it will also work)

History (it will also work for seeing cmd history)

Start-transcript (to start a new command history)
Stop-transcript
Start-transcript-path filename-append

[If system shutdown immediately automatically transcripts are stopped we have to create a new Transcript]
But also we can automate this

get-date
date
(PS supports most of the LINUX areas)

Get-ChildItem
Ls

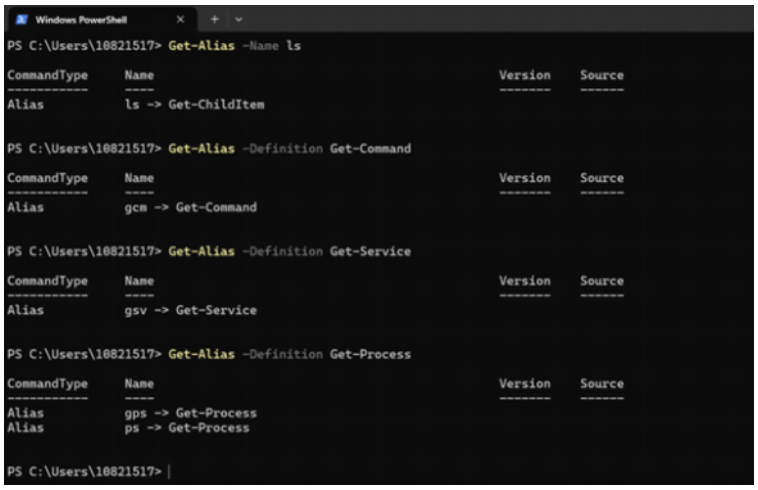
Get-Alias(All the aliases)

I know the alias but don't know the cmd

Get-Alias -Name cls
Get-Alias -Name ls

I know the cmd but don't know the alias

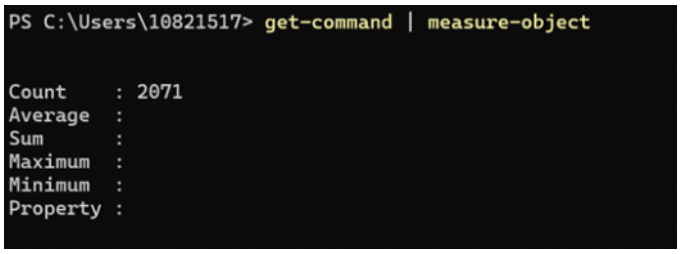
Get-Alias -Definition Get-Command
Get-Alias -Definition Get-Service
Get-Alias -Definition Get-Process



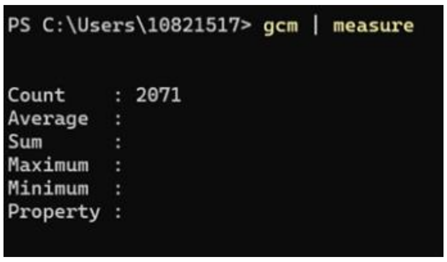
```
PS C:\Users\10821517> Get-Alias -Name ls

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Alias           ls -> Get-ChildItem


PS C:\Users\10821517> Get-Alias -Definition Get-Command

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Alias           gcm -> Get-Command


PS C:\Users\10821517> Get-Alias -Definition Get-Service

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Alias           gsv -> Get-Service


PS C:\Users\10821517> Get-Alias -Definition Get-Process

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Alias           gps -> Get-Process
Alias           ps -> Get-Process


PS C:\Users\10821517> |
```

To count all the lines or commands

Get-Command (now try to count how many cmds are there)



```
PS C:\Users\10821517> get-command | measure-object


Count    : 2071
Average  :
Sum      :
Maximum  :
Minimum  :
Property :
```

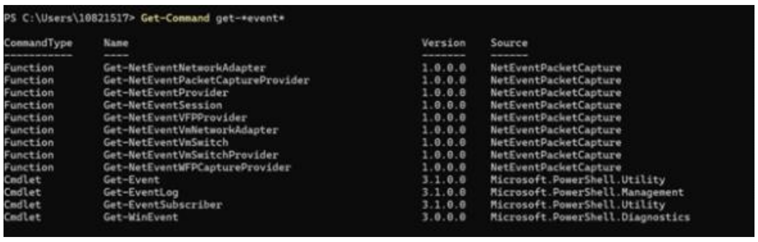


```
PS C:\Users\10821517> gcm | measure


Count    : 2071
Average  :
Sum      :
Maximum  :
Minimum  :
Property :
```

TASK
List top 5 system event logs
Now you have to know only just 2 cmds to find how the command will work and where to use it

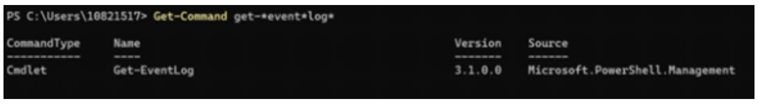
1. Get-Command -> Which cmdlet is helpful
2. Get-help -> how this cmdlet works

Get-Command get-*event*



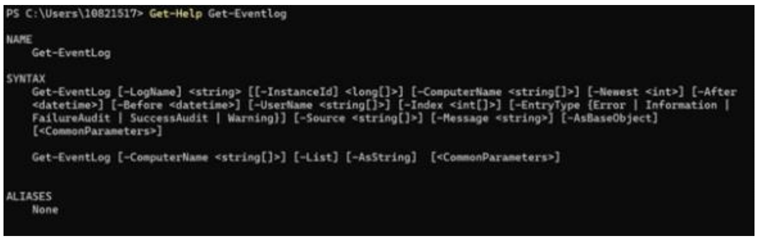
```
PS C:\Users\10821517> Get-Command get-*event*

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Function        Get-NetEventNetworkAdapter                         1.0.0.0    NetEventPacketCapture
Function        Get-NetEventPacketCaptureProvider                  1.0.0.0    NetEventPacketCapture
Function        Get-NetEventProvider                               1.0.0.0    NetEventPacketCapture
Function        Get-NetEventSession                                1.0.0.0    NetEventPacketCapture
Function        Get-NetEventVFPProvider                            1.0.0.0    NetEventPacketCapture
Function        Get-NetEventVmNetworkAdapter                       1.0.0.0    NetEventPacketCapture
Function        Get-NetEventVmSwitch                               1.0.0.0    NetEventPacketCapture
Function        Get-NetEventVmSwitchProvider                       1.0.0.0    NetEventPacketCapture
Function        Get-NetEventWFPCaptureProvider                     1.0.0.0    NetEventPacketCapture
Cmdlet          Get-Event                                          3.1.0.0    Microsoft.PowerShell.Utility
Cmdlet          Get-EventLog                                       3.1.0.0    Microsoft.PowerShell.Management
Cmdlet          Get-EventSubscriber                                3.1.0.0    Microsoft.PowerShell.Utility
Cmdlet          Get-WinEvent                                       3.0.0.0    Microsoft.PowerShell.Diagnostics
```

Get-Command get-*event*log*



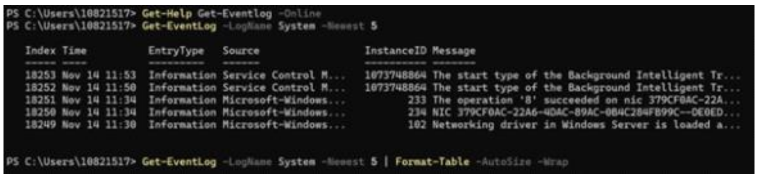
```
PS C:\Users\10821517> Get-Command get-*event*log*

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Cmdlet          Get-EventLog                                       3.1.0.0    Microsoft.PowerShell.Management
```

Get-Help Get-Eventlog



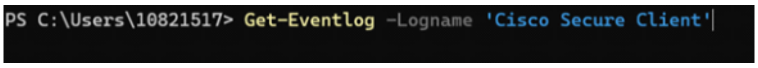
```
PS C:\Users\10821517> Get-Help Get-Eventlog

NAME
    Get-EventLog

SYNTAX
    Get-EventLog [-LogName] <string> [[-InstanceId] <long[]>] [-ComputerName <string[]>] [-Newest <int>] [-After
    <datetime>] [-Before <datetime>] [-UserName <string[]>] [-Index <int[]>] [-EntryType {Error | Information |
    FailureAudit | SuccessAudit | Warning}] [-Source <string[]>] [-Message <string>] [-AsBaseObject]
    [<CommonParameters>]

    Get-EventLog [-ComputerName <string[]>] [-List] [-AsString]  [<CommonParameters>]


ALIASES
    None
```

Get-Help Get-Eventlog
Get-Help Get-Eventlog -Online
Get-Eventlog -Logname System -Newest 5
Get-Eventlog -Logname System -Newest 5 | Format-Table -Autosize -Wrap



```
PS C:\Users\10821517> Get-Help Get-Eventlog -Online
PS C:\Users\10821517> Get-EventLog -LogName System -Newest 5

   Index Time          EntryType   Source                 InstanceID Message
   ----- ----          ---------   ------                 ---------- -------
   18253 Nov 14 11:53  Information Service Control M...   1073748864 The start type of the Background Intelligent Tr...
   18252 Nov 14 11:50  Information Service Control M...   1073748864 The start type of the Background Intelligent Tr...
   18251 Nov 14 11:34  Information Microsoft-Windows...          233 The operation '8' succeeded on nic 379CF0AC-22A...
   18250 Nov 14 11:34  Information Microsoft-Windows...          234 NIC 379CF0AC-22A6-4DAC-89AC-0B4C284FB99C--DE0ED...
   18249 Nov 14 11:30  Information Microsoft-Windows...          102 Networking driver in Windows Server is loaded a...


PS C:\Users\10821517> Get-EventLog -LogName System -Newest 5 | Format-Table -AutoSize -Wrap
```

Get-Eventlog -Logname <your_log_name>



```
PS C:\Users\10821517> Get-Eventlog -Logname 'Cisco Secure Client'
```

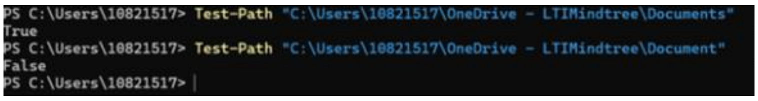
Powershell cmd to ping a computer is

Test-Connection google.com

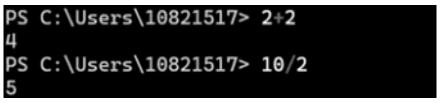
Test-Connection google.com -Count 1 [How many times we want to ping]

To check A file or dir is exist or not

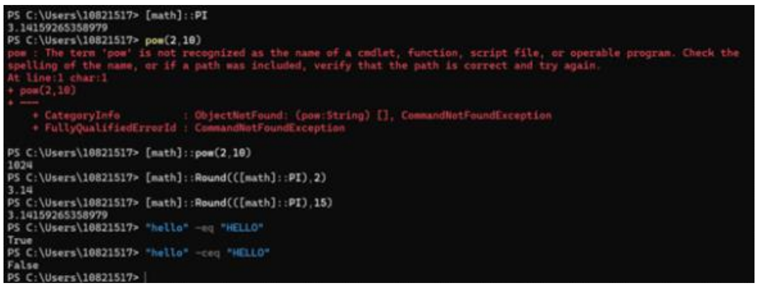
Test-Path



```
PS C:\Users\10821517> Test-Path "C:\Users\10821517\OneDrive - LTIMindtree\Documents"
True
PS C:\Users\10821517> Test-Path "C:\Users\10821517\OneDrive - LTIMindtree\Document"
False
PS C:\Users\10821517> |
```



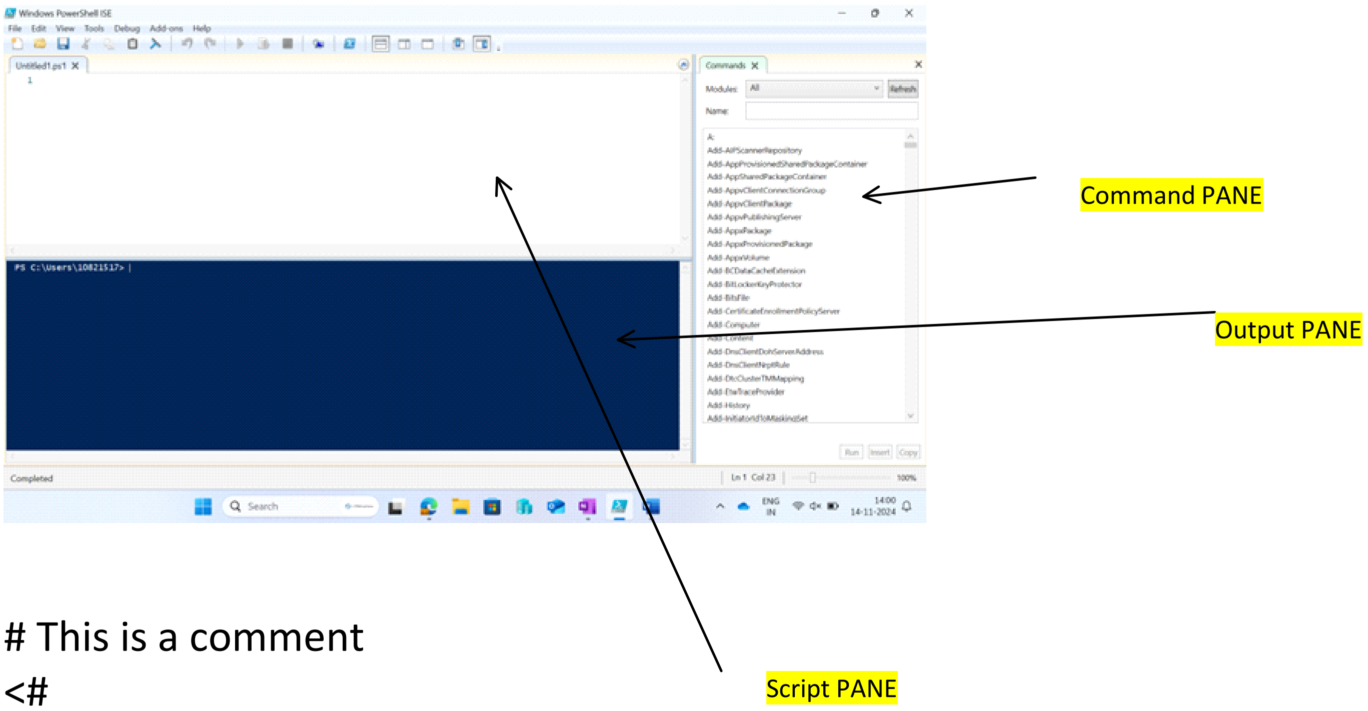
```
PS C:\Users\10821517> 2+2
4
PS C:\Users\10821517> 10/2
5
```



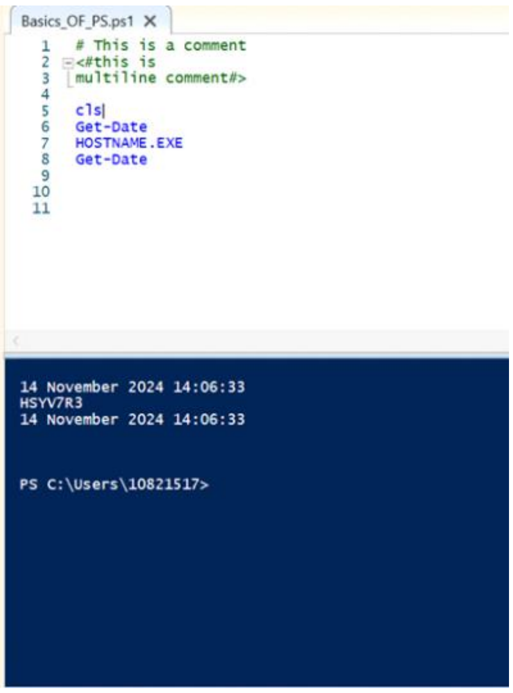
```
PS C:\Users\10821517> [math]::PI
3.14159265358979
PS C:\Users\10821517> pow(2,10)
pow : The term 'pow' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the
spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ pow(2,10)
+ ~~~
    + CategoryInfo          : ObjectNotFound: (pow:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\10821517> [math]::pow(2,10)
1024
PS C:\Users\10821517> [math]::Round(([math]::PI),2)
3.14
PS C:\Users\10821517> [math]::Round(([math]::PI),15)
3.14159265358979
PS C:\Users\10821517> "hello" -eq "HELLO"
True
PS C:\Users\10821517> "hello" -ceq "HELLO"
False
PS C:\Users\10821517>
```

POWERSHELL IN ISE

In run type powershell_ise

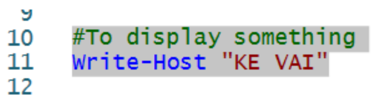


# This is a comment
<#
this is
multiline comment
#>


```
Basics_OF_PS.ps1
# This is a comment
<#this is
multiline comment#>

cls
Get-Date
HOSTNAME.EXE
Get-Date
```

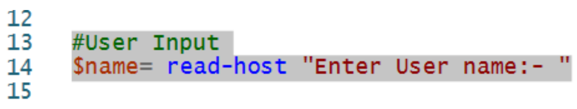
```
14 November 2024 14:06:33
HSYV7R3
14 November 2024 14:06:33


PS C:\Users\10821517>
```

#To display something
Write-Host "KE VAI"



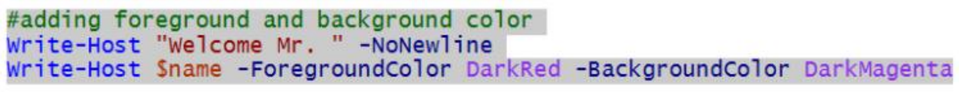
```
#To display something
Write-Host "KE VAI"
```

#User Input
$name= read-host "Enter User name:- "



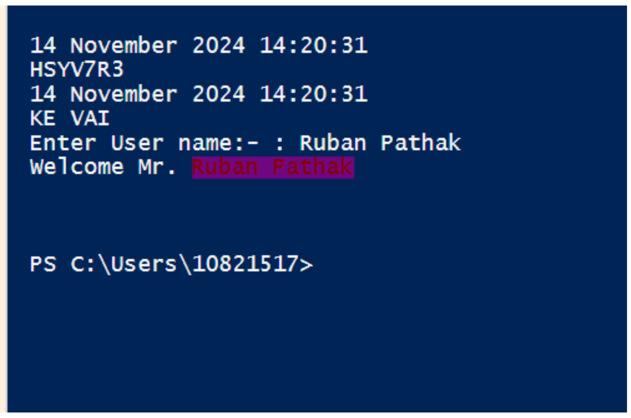
```
#User Input
$name= read-host "Enter User name:- "
```

#adding foreground and background color
Write-Host "Welcome Mr. " -NoNewline
Write-Host $name -ForegroundColor DarkRed -BackgroundColor DarkMagenta



```
#adding foreground and background color
Write-Host "Welcome Mr. " -NoNewline
Write-Host $name -ForegroundColor DarkRed -BackgroundColor DarkMagenta
```

Output

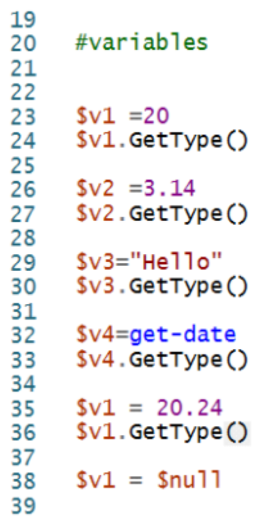


```
14 November 2024 14:20:31
HSYV7R3
14 November 2024 14:20:31
KE VAI
Enter User name:- : Ruban Pathak
Welcome Mr. Ruban Pathak



PS C:\Users\10821517>
```

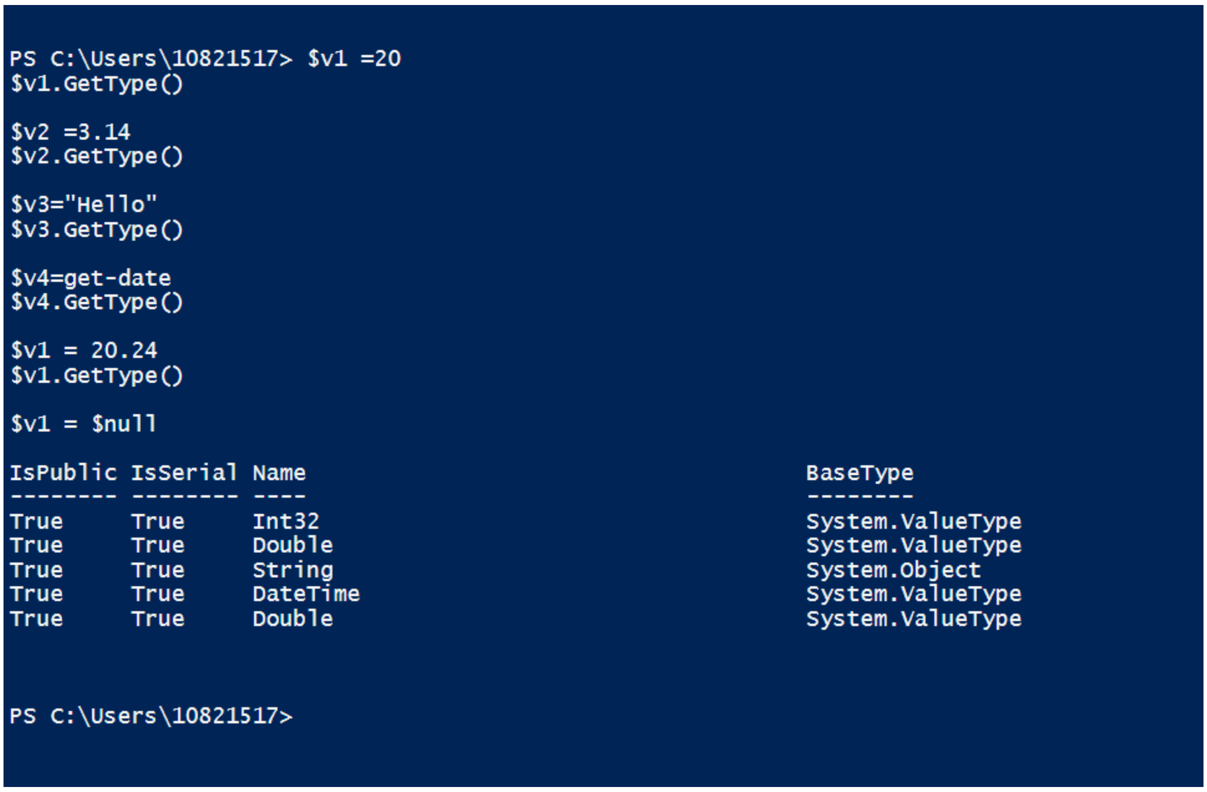
Data Structures

->Variable
 ->Array
->Hashtable

```
#variables
$v1 = 20
$v1.GetType()

$v2=3.14
$v2.GetType()

$v3  = "Hello"
$v3.GetType()

$v4 = Get-Date
$v4.GetType()

$v1 = 20.24
$v1.Gettype()

V1=$null
```



```
#variables

$v1 =20
$v1.GetType()

$v2 =3.14
$v2.GetType()

$v3="Hello"
$v3.GetType()

$v4=get-date
$v4.GetType()

$v1 = 20.24
$v1.GetType()

$v1 = $null
```

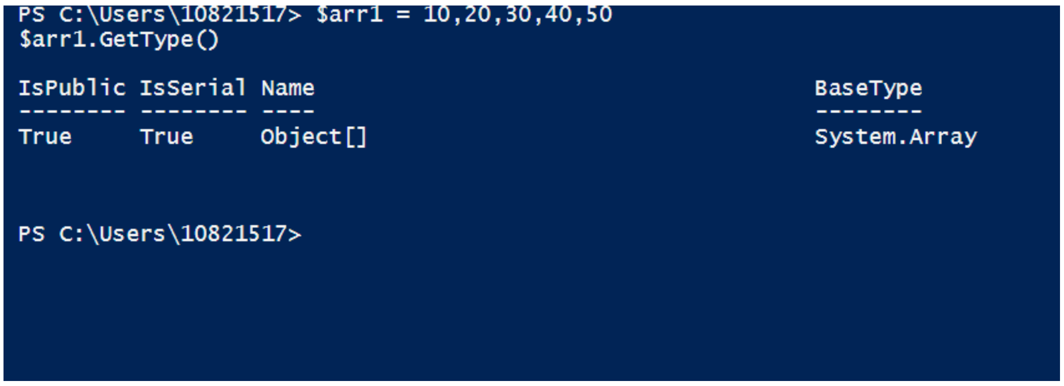


```
PS C:\Users\10821517> $v1 =20
$v1.GetType()

$v2 =3.14
$v2.GetType()

$v3="Hello"
$v3.GetType()

$v4=get-date
$v4.GetType()

$v1 = 20.24
$v1.GetType()

$v1 = $null

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     Int32                                    System.ValueType
True     True     Double                                   System.ValueType
True     True     String                                   System.Object
True     True     DateTime                                 System.ValueType
True     True     Double                                   System.ValueType


PS C:\Users\10821517>
```

Arrays

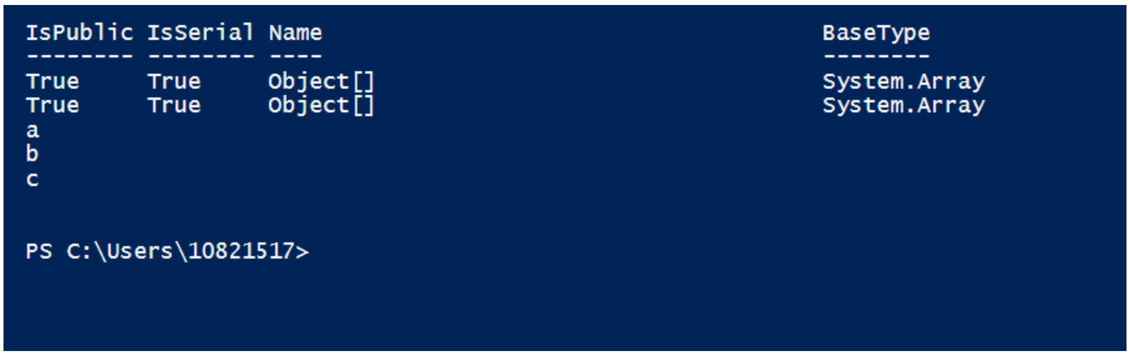
Method 01 ~ Old Method

$arr1 = 10,20,30,40,50
$arr1.GetType()


```
PS C:\Users\10821517> $arr1 = 10,20,30,40,50
$arr1.GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     Object[]                                 System.Array


PS C:\Users\10821517>
```

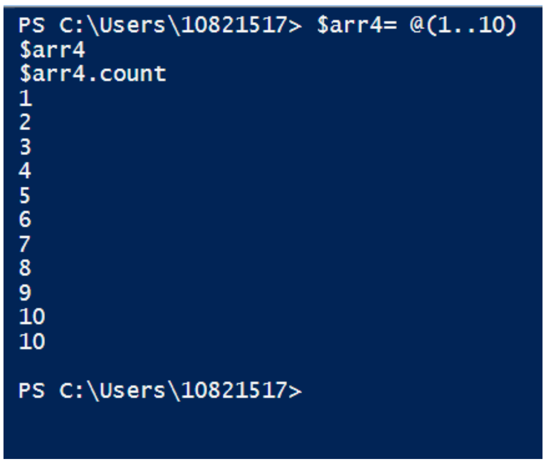
Method 02 ~ New Method

```
$arr2 = @()
$arr2.GetType()

$arr3=@("a","b","c")
$arr3.GetType()
$arr3
```


```
IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     Object[]                                 System.Array
True     True     Object[]                                 System.Array
a
b
c

PS C:\Users\10821517>
```

Add 1 to N in a fraction of second

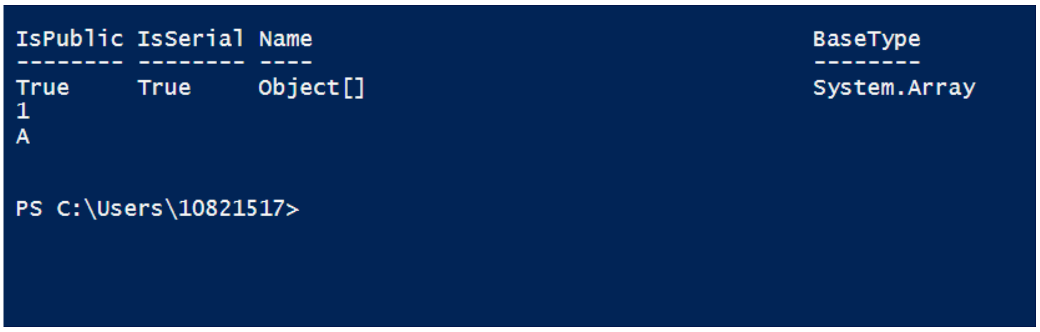
```
$arr4= @(1..100)
$arr4
$arr4.count
$arr4.Length
```



```
PS C:\Users\10821517> $arr4= @(1..10)
$arr4
$arr4.count
1
2
3
4
5
6
7
8
9
10
10

PS C:\Users\10821517>
```

Multidimensional Array

```
$arr5= @(
        @(1,2,3),
        @("A","B","C"),
        @(Get-Process)
)

$arr5.GetType()
$arr5[0][0]
$arr5[-2][-1]
```



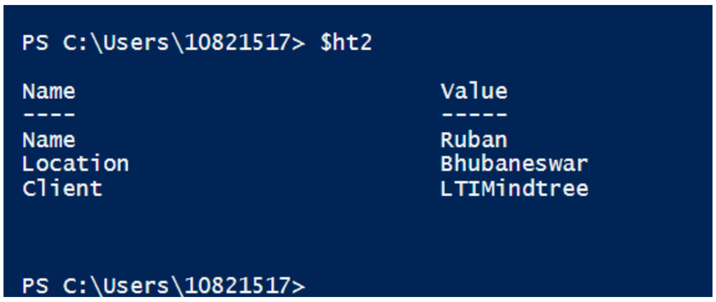
```
IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     Object[]                                 System.Array
1
A


PS C:\Users\10821517>
```

Hash Table

Key Value pair

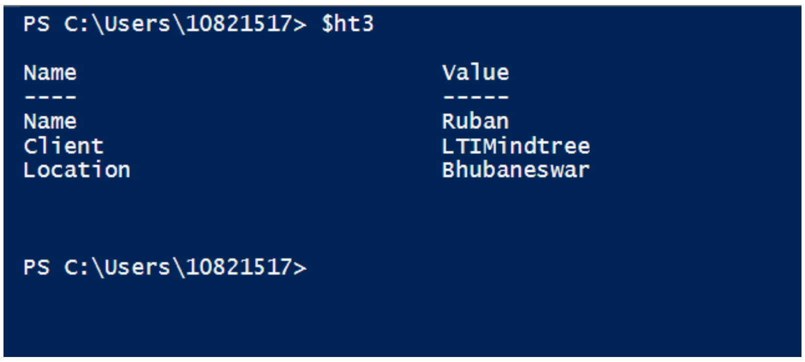
```
$ht1 = @{}
$ht1.GetType()

$ht2 = @{"Name" = "Ruban" ; "Client" = "LTIMindtree" ; "Location" = "Bhubaneswar"}
$ht2
```

(Unordered Hash Table)


```
PS C:\Users\10821517> $ht2

Name                           Value
----                           -----
Name                           Ruban
Location                       Bhubaneswar
Client                         LTIMindtree


PS C:\Users\10821517>
```

(Ordered Hash Table)

$ht3 = [ordered]@{"Name" = "Ruban" ; "Client" = "LTIMindtree" ; "Location" = "Bhubaneswar"}
$ht3


```
PS C:\Users\10821517> $ht3

Name                           Value
----                           -----
Name                           Ruban
Client                         LTIMindtree
Location                       Bhubaneswar


PS C:\Users\10821517>
```

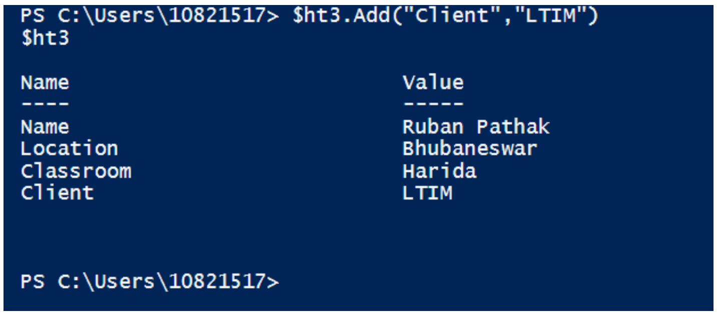
Day2~Practical PowerShell

15 November 2024 08:56

Hash Table (Continuation)

Adding key-value pair

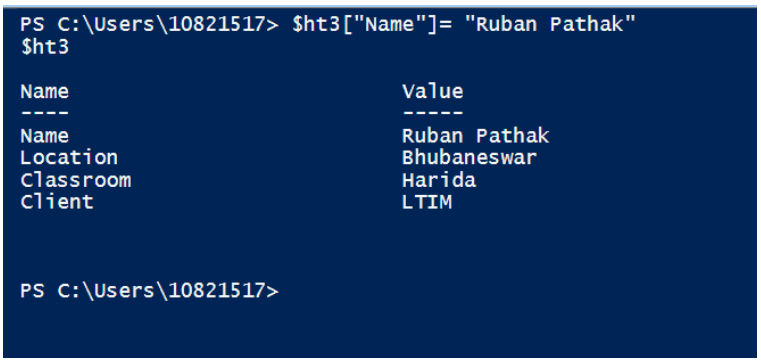
$ht3.Add("Classroom","Harida")
$ht3



```
PS C:\Users\10821517> $ht3.Add("Client","LTIM")
$ht3

Name                           Value
----                           -----
Name                           Ruban Pathak
Location                       Bhubaneswar
Classroom                      Harida
Client                         LTIM


PS C:\Users\10821517>
```

Modify key-value pair

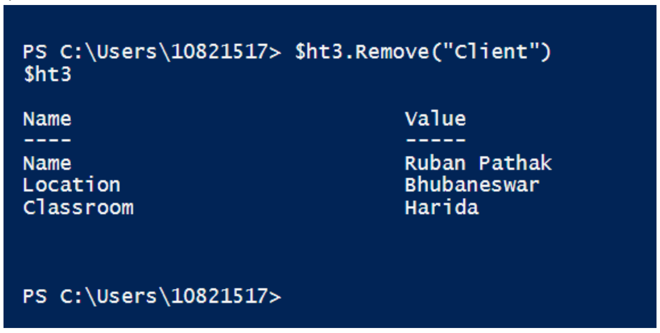
$ht3["Name"]= "Ruban Pathak"
$ht3



```
PS C:\Users\10821517> $ht3["Name"]= "Ruban Pathak"
$ht3

Name                           Value
----                           -----
Name                           Ruban Pathak
Location                       Bhubaneswar
Classroom                      Harida
Client                         LTIM


PS C:\Users\10821517>
```

Delete key-value pair

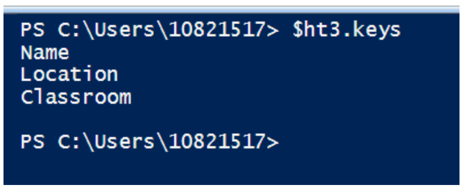
$ht3.Remove("Client")
$ht3



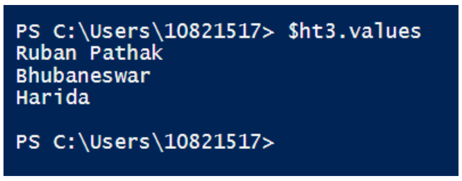
```
PS C:\Users\10821517> $ht3.Remove("Client")
$ht3

Name                           Value
----                           -----
Name                           Ruban Pathak
Location                       Bhubaneswar
Classroom                      Harida


PS C:\Users\10821517>
```

Other Operations

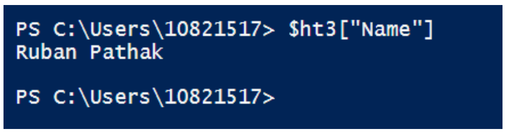
$ht3.keys


```
PS C:\Users\10821517> $ht3.keys
Name
Location
Classroom

PS C:\Users\10821517>
```

$ht3.values


```
PS C:\Users\10821517> $ht3.values
Ruban Pathak
Bhubaneswar
Harida

PS C:\Users\10821517>
```

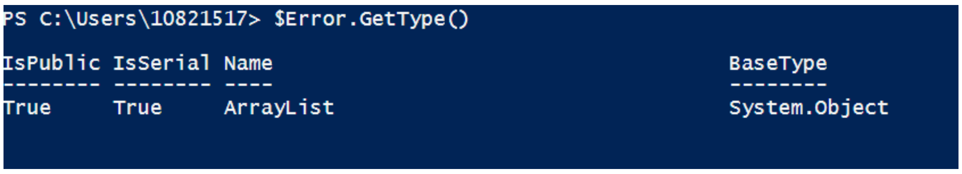
$ht3["Name"]


```
PS C:\Users\10821517> $ht3["Name"]
Ruban Pathak

PS C:\Users\10821517>
```

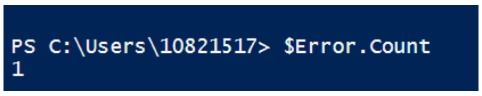
Variables (Continuation)

There are some system defined variables are there into PS
Suppose you run a script and have some error and after that you wipe out the terminal
But someone comes and ask to show the error.

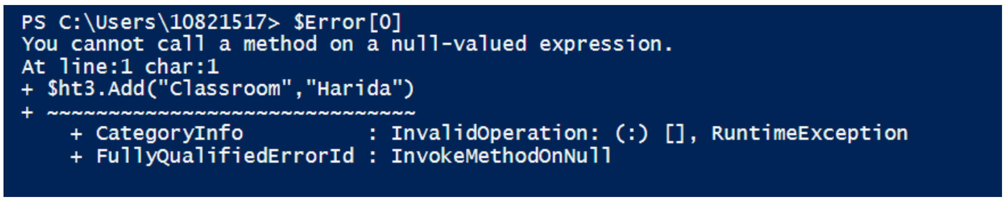
$Error.Gettype()


```
PS C:\Users\10821517> $Error.GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     ArrayList                                System.Object
```

$Error.Count


```
PS C:\Users\10821517> $Error.Count
1
```

$Error[0]

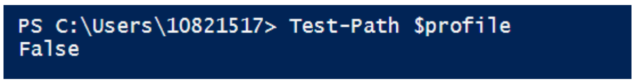

```
PS C:\Users\10821517> $Error[0]
You cannot call a method on a null-valued expression.
At line:1 char:1
+ $ht3.Add("Classroom","Harida")
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [], RuntimeException
    + FullyQualifiedErrorId : InvokeMethodOnNull
```

Hhhhhh
$error[1]

Display something whenever you open powershell

$profile [It Stores the file path inside it]
Test-Path $profile

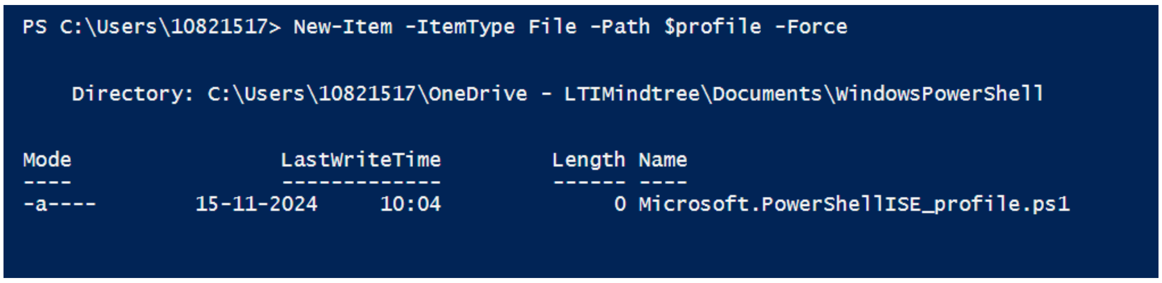
It's the file that is loaded (if present) by default by PowerShell



```
PS C:\Users\10821517> Test-Path $profile
False
```

Create Profile

New-Item -ItemType File -Path $profile -Force



```
PS C:\Users\10821517> New-Item -ItemType File -Path $profile -Force

    Directory: C:\Users\10821517\OneDrive - LTIMindtree\Documents\WindowsPowerShell

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        15-11-2024     10:04              0 Microsoft.PowerShellISE_profile.ps1
```

To open that profile file

Notepad.exe $profile

In that file
Cls
Write-host "Hello Ruban" -ForegroundColor yellow
Cd C:\Powershell

Now Test the profile path in powershell not in ise it will give false
Because powershell_ise works on other path and powershell works on other file
So write this cmd once again in powershell

New-Item -ItemType File -Path $profile -Force

Now check the $profile path

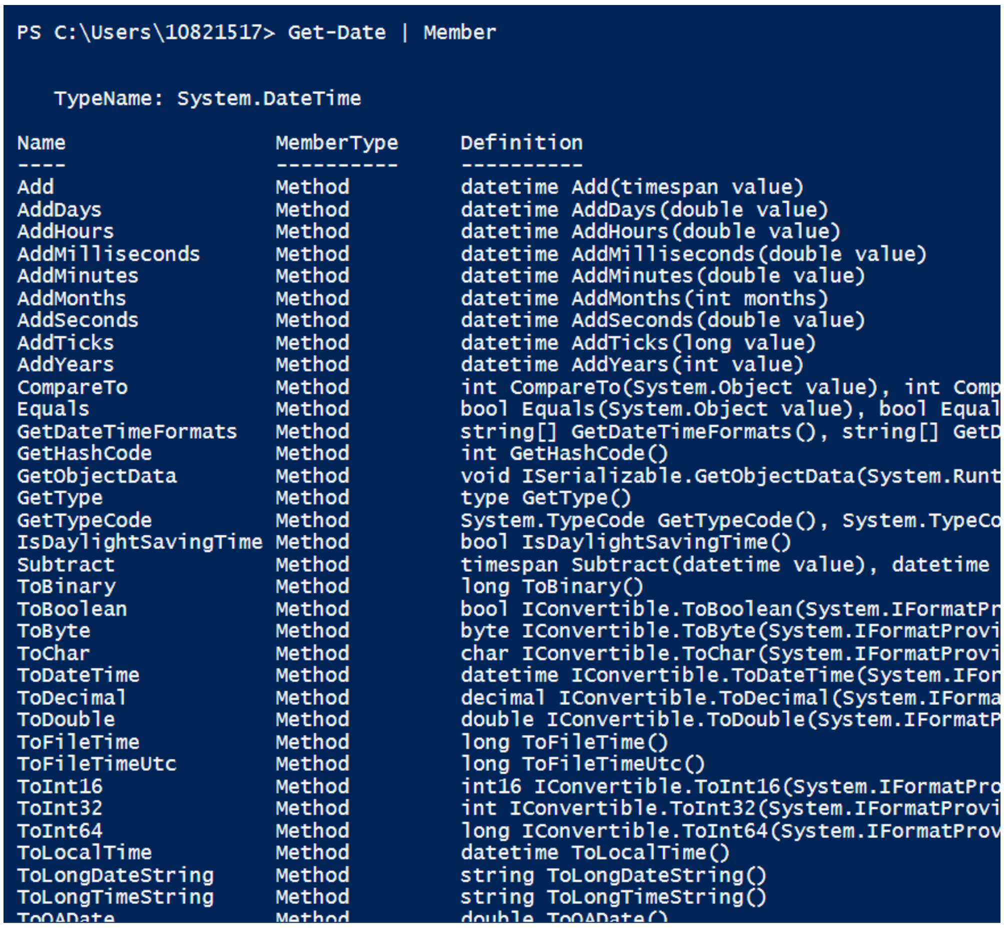
Objects & Members

Cls
$d = Get-Date
$d.Date
$d.Hour
$d.Minute
$d.DayOfYear

(get-date).DayOfWeek

Get-Date | Get-Member
Get-Process | Get-Member

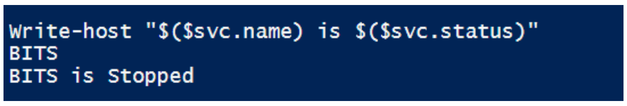
How many members we get inside a cmdlets


```
PS C:\Users\10821517> Get-Date | Member


   TypeName: System.DateTime

Name                 MemberType     Definition
----                 ----------     ----------
Add                  Method         datetime Add(timespan value)
AddDays              Method         datetime AddDays(double value)
AddHours             Method         datetime AddHours(double value)
AddMilliseconds      Method         datetime AddMilliseconds(double value)
AddMinutes           Method         datetime AddMinutes(double value)
AddMonths            Method         datetime AddMonths(int months)
AddSeconds           Method         datetime AddSeconds(double value)
AddTicks             Method         datetime AddTicks(long value)
AddYears             Method         datetime AddYears(int value)
CompareTo            Method         int CompareTo(System.Object value), int Comp
Equals               Method         bool Equals(System.Object value), bool Equal
GetDateTimeFormats   Method         string[] GetDateTimeFormats(), string[] GetD
GetHashCode          Method         int GetHashCode()
GetObjectData        Method         void ISerializable.GetObjectData(System.Runt
GetType              Method         type GetType()
GetTypeCode          Method         System.TypeCode GetTypeCode(), System.TypeCo
IsDaylightSavingTime Method         bool IsDaylightSavingTime()
Subtract             Method         timespan Subtract(datetime value), datetime 
ToBinary             Method         long ToBinary()
ToBoolean            Method         bool IConvertible.ToBoolean(System.IFormatPr
ToByte               Method         byte IConvertible.ToByte(System.IFormatProvi
ToChar               Method         char IConvertible.ToChar(System.IFormatProvi
ToDateTime           Method         datetime IConvertible.ToDateTime(System.IFor
ToDecimal            Method         decimal IConvertible.ToDecimal(System.IForma
ToDouble             Method         double IConvertible.ToDouble(System.IFormatP
ToFileTime           Method         long ToFileTime()
ToFileTimeUtc        Method         long ToFileTimeUtc()
ToInt16              Method         int16 IConvertible.ToInt16(System.IFormatPro
ToInt32              Method         int IConvertible.ToInt32(System.IFormatProvi
ToInt64              Method         long IConvertible.ToInt64(System.IFormatProv
ToLocalTime          Method         datetime ToLocalTime()
ToLongDateString     Method         string ToLongDateString()
ToLongTimeString     Method         string ToLongTimeString()
ToOADate             Method         double ToOADate()
```

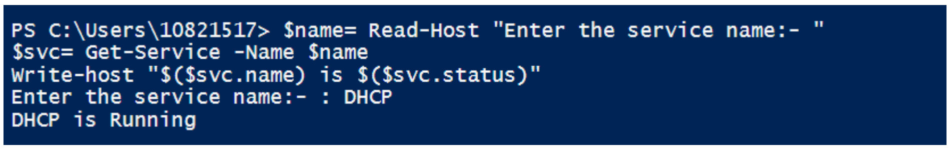
$svc= Get-Service -Name BITS
-Name is use to target something

$svc.Name(We can use it to show inside the output)

Write-host "$($svc.name) is $($svc.status)"


```
Write-host "$($svc.name) is $($svc.status)"
BITS
BITS is Stopped
```

$name= Read-Host "Enter the service name:- "
$svc= Get-Service -Name $name
Write-host "$($svc.name) is $($svc.status)"

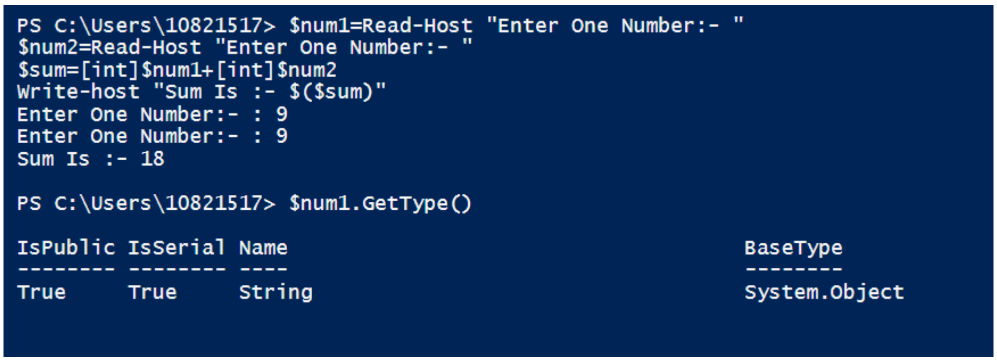


```
PS C:\Users\10821517> $name= Read-Host "Enter the service name:- "
$svc= Get-Service -Name $name
Write-host "$($svc.name) is $($svc.status)"
Enter the service name:- : DHCP
DHCP is Running
```

To run powershell scripts in your powershell write this cmds
./<your_script_name>.ps1

Task:- Add 2 numbers in powershell (Typecasting)

$num1=Read-Host "Enter One Number:- "(convert the input in string)
$num2=Read-Host "Enter One Number:- "
$sum=[int]$num1+[int]$num2
Write-host "Sum Is :- $($sum)"

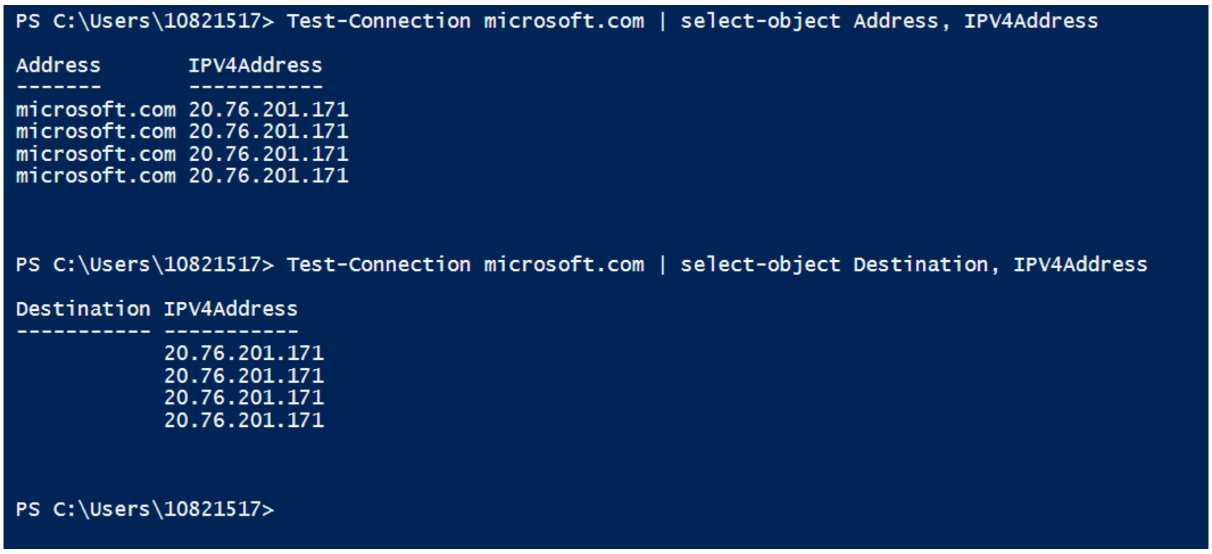


```
PS C:\Users\10821517> $num1=Read-Host "Enter One Number:- "
$num2=Read-Host "Enter One Number:- "
$sum=[int]$num1+[int]$num2
Write-host "Sum Is :- $($sum)"
Enter One Number:- : 9
Enter One Number:- : 9
Sum Is :- 18

PS C:\Users\10821517> $num1.GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     String                                   System.Object
```

Object is an instance of a class

Get-Process| Get-Member
Get-Process | select-object ProcessName, ID

Test-Connection microsoft.com
Test-Connection microsoft.com | select-object Destination, IPV4Address
Test-Connection microsoft.com | select-object Address, IPV4Address
Test-Connection microsoft.com | Get-Member



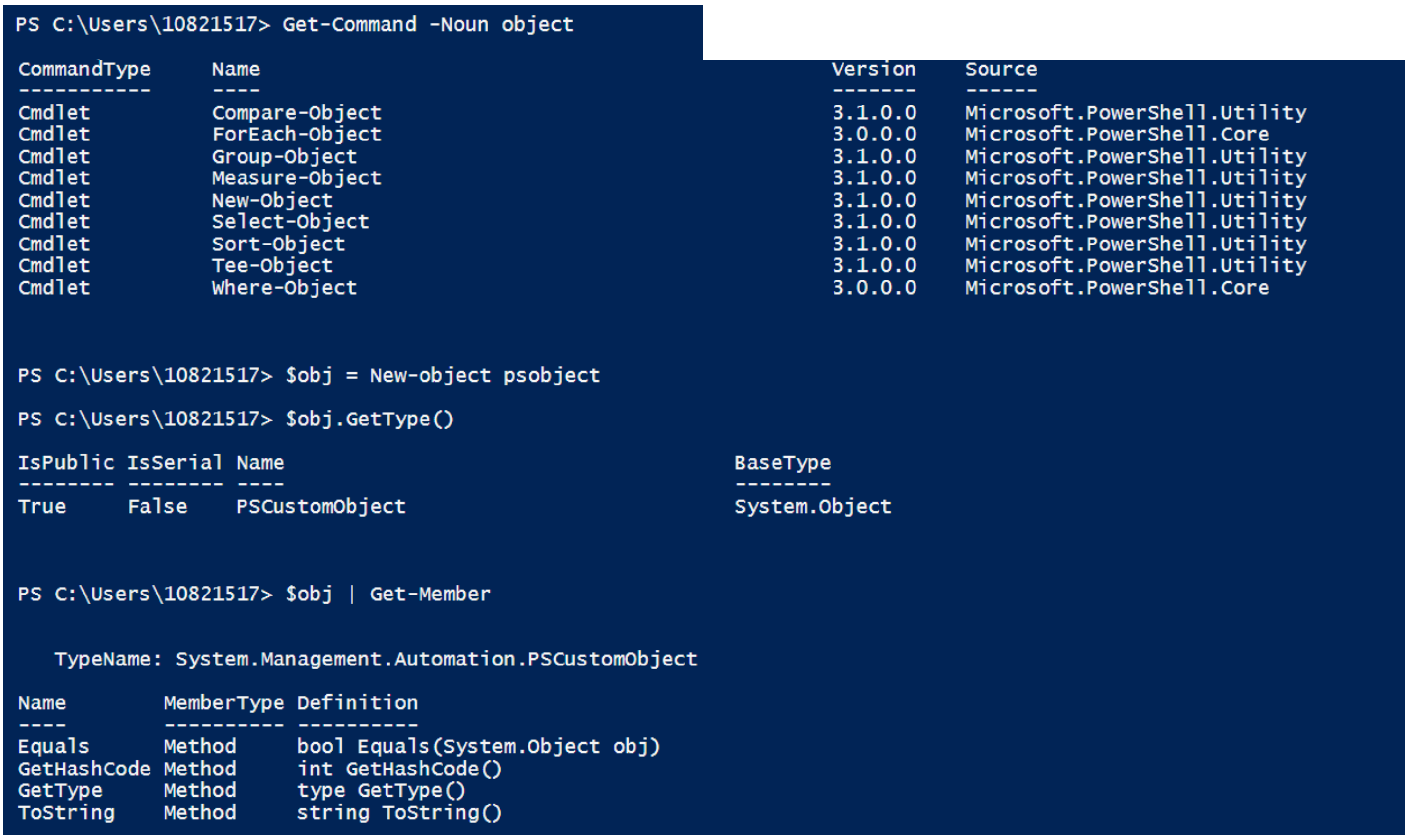
```
PS C:\Users\10821517> Test-Connection microsoft.com | select-object Address, IPV4Address

Address       IPV4Address
-------       -----------
microsoft.com 20.76.201.171
microsoft.com 20.76.201.171
microsoft.com 20.76.201.171
microsoft.com 20.76.201.171


PS C:\Users\10821517> Test-Connection microsoft.com | select-object Destination, IPV4Address

Destination IPV4Address
----------- -----------
            20.76.201.171
            20.76.201.171
            20.76.201.171
            20.76.201.171


PS C:\Users\10821517>
```

If the member is not present but you type it then it will not be give u some error basically it will Filter out some columns inside various columns

Creating a new object

#list all the cmdlets that has object in it
Get-Command -Noun object
Get-Command *-object

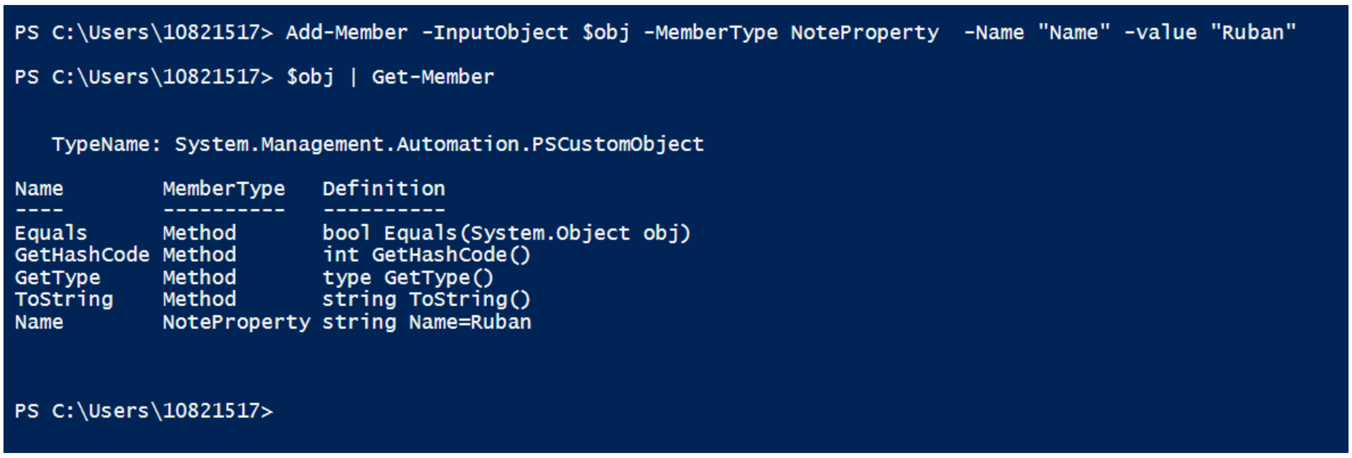
#create a new object
$obj = New-object psobject #(it will be a power shell based custom object)
$obj.GetType()
$obj | Get-Member #(listing all the default members)



```
PS C:\Users\10821517> Get-Command -Noun object

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Cmdlet          Compare-Object                                     3.1.0.0    Microsoft.PowerShell.Utility
Cmdlet          ForEach-Object                                     3.0.0.0    Microsoft.PowerShell.Core
Cmdlet          Group-Object                                       3.1.0.0    Microsoft.PowerShell.Utility
Cmdlet          Measure-Object                                     3.1.0.0    Microsoft.PowerShell.Utility
Cmdlet          New-Object                                         3.1.0.0    Microsoft.PowerShell.Utility
Cmdlet          Select-Object                                      3.1.0.0    Microsoft.PowerShell.Utility
Cmdlet          Sort-Object                                        3.1.0.0    Microsoft.PowerShell.Utility
Cmdlet          Tee-Object                                         3.1.0.0    Microsoft.PowerShell.Utility
Cmdlet          Where-Object                                       3.0.0.0    Microsoft.PowerShell.Core


PS C:\Users\10821517> $obj = New-object psobject

PS C:\Users\10821517> $obj.GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     False    PSCustomObject                           System.Object


PS C:\Users\10821517> $obj | Get-Member

   TypeName: System.Management.Automation.PSCustomObject

Name        MemberType Definition
----        ---------- ----------
Equals      Method     bool Equals(System.Object obj)
GetHashCode Method     int GetHashCode()
GetType     Method     type GetType()
ToString    Method     string ToString()
```

#Add new member to that object

Add-Member -InputObject $obj -MemberType NoteProperty [Because no linking required just pass Key and value] -Name "Name" -value "Ruban"



```
PS C:\Users\10821517> Add-Member -InputObject $obj -MemberType NoteProperty  -Name "Name" -value "Ruban"
PS C:\Users\10821517> $obj | Get-Member


   TypeName: System.Management.Automation.PSCustomObject

Name        MemberType   Definition
----        ----------   ----------
Equals      Method       bool Equals(System.Object obj)
GetHashCode Method       int GetHashCode()
GetType     Method       type GetType()
ToString    Method       string ToString()
Name        NoteProperty string Name=Ruban


PS C:\Users\10821517>
```

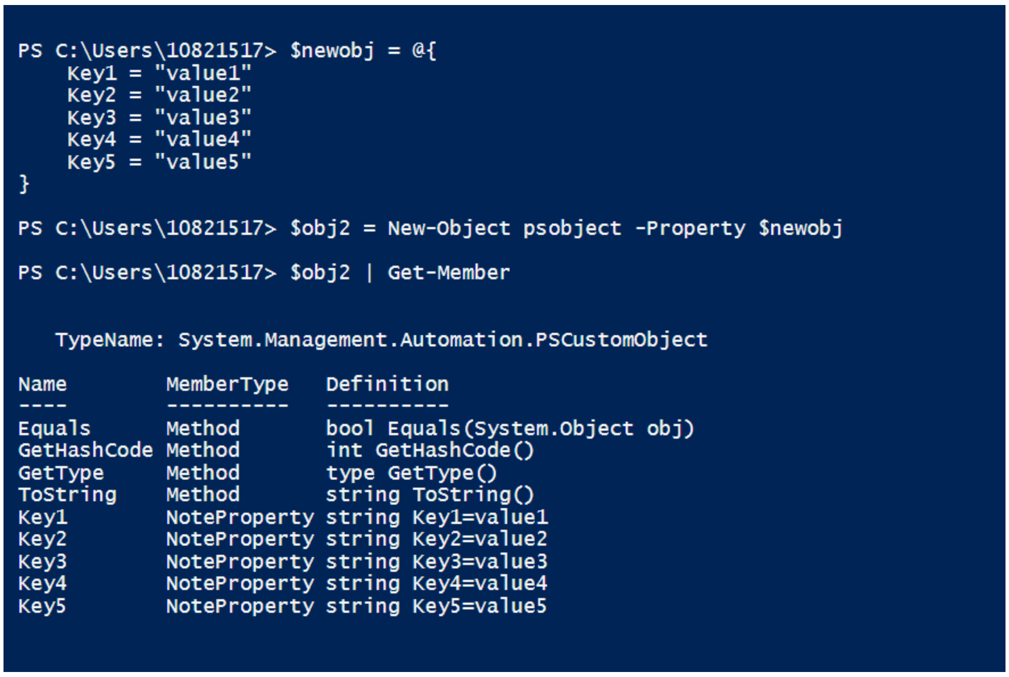
If we want to add 100 of object members at a time here the hash table concept will appear

#create all the members using hashtable

```
$newobj = @{
	Key1 = "value1"
	Key2 = "value2"
	Key3 = "value3"
	Key4 = "value4"
	Key5 = "value5"
}
```

#creating a new object

```
$obj2 = New-Object psobject -Property $newobj
$obj2 | Get-Member
```

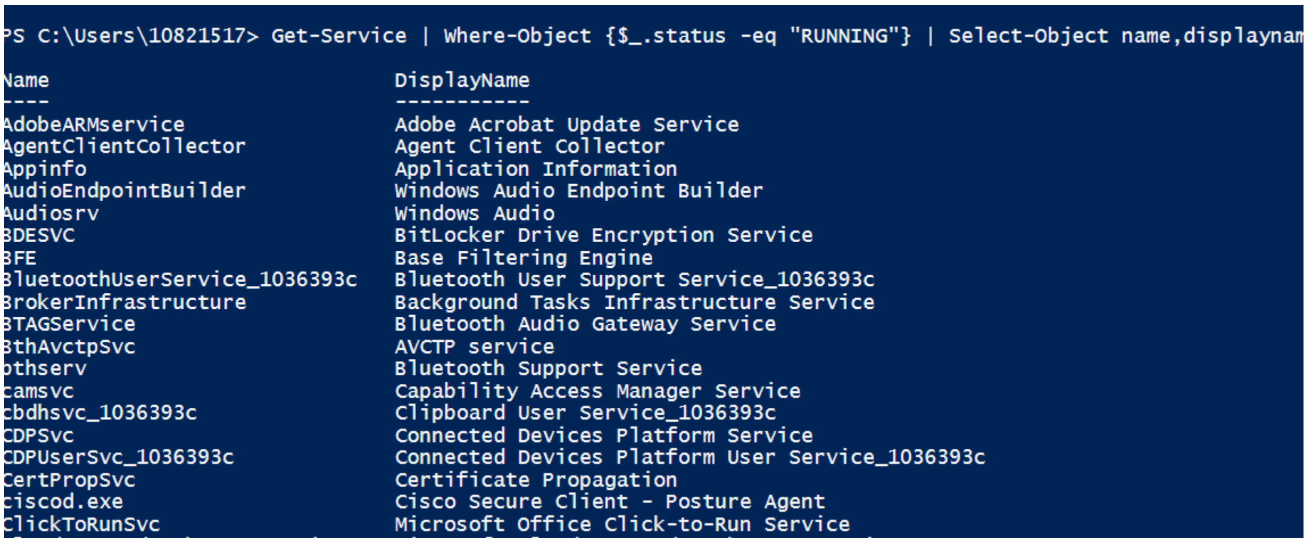


```
PS C:\Users\10821517> $newobj = @{
    Key1 = "value1"
    Key2 = "value2"
    Key3 = "value3"
    Key4 = "value4"
    Key5 = "value5"
}
PS C:\Users\10821517> $obj2 = New-Object psobject -Property $newobj
PS C:\Users\10821517> $obj2 | Get-Member


   TypeName: System.Management.Automation.PSCustomObject

Name        MemberType   Definition
----        ----------   ----------
Equals      Method       bool Equals(System.Object obj)
GetHashCode Method       int GetHashCode()
GetType     Method       type GetType()
ToString    Method       string ToString()
Key1        NoteProperty string Key1=value1
Key2        NoteProperty string Key2=value2
Key3        NoteProperty string Key3=value3
Key4        NoteProperty string Key4=value4
Key5        NoteProperty string Key5=value5
```

Task Display only the running service

Get-Service | Where-Object {$_.status -eq "RUNNING"} | Select-Object name,displayname

[$_.status] it makes it as member otherwise it assumes that it is a command
Select-Object name,displayname filtering only the name and display name.



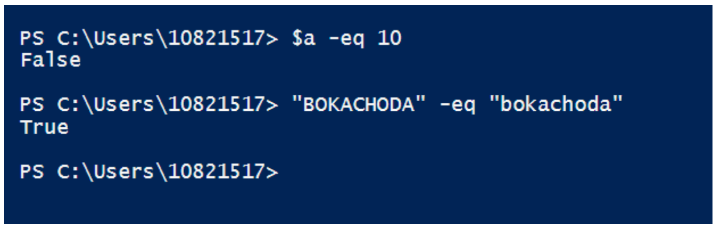
```
PS C:\Users\10821517> Get-Service | Where-Object {$_.status -eq "RUNNING"} | Select-Object name,displaynam

Name                            DisplayName
----                            -----------
AdobeARMservice                 Adobe Acrobat Update Service
AgentClientCollector            Agent Client Collector
Appinfo                         Application Information
AudioEndpointBuilder            Windows Audio Endpoint Builder
Audiosrv                        Windows Audio
BDESVC                          BitLocker Drive Encryption Service
BFE                             Base Filtering Engine
BluetoothUserService_1036393c   Bluetooth User Support Service_1036393c
BrokerInfrastructure            Background Tasks Infrastructure Service
BTAGService                     Bluetooth Audio Gateway Service
BthAvctpSvc                     AVCTP service
bthserv                         Bluetooth Support Service
camsvc                          Capability Access Manager Service
cbdhsvc_1036393c                Clipboard User Service_1036393c
CDPSvc                          Connected Devices Platform Service
CDPUserSvc_1036393c             Connected Devices Platform User Service_1036393c
CertPropSvc                     Certificate Propagation
ciscod.exe                      Cisco Secure Client - Posture Agent
ClickToRunSvc                   Microsoft Office Click-to-Run Service
```

Operators

-eq
-ne
-lt
-gt
-ge
-and
-or

"hello" -eq "HELLO" ~ True
"hello" -ceq "HELLO" ~ False

$a=10
$a -eq 10



```
PS C:\Users\10821517> $a -eq 10
False

PS C:\Users\10821517> "BOKACHODA" -eq "bokachoda"
True

PS C:\Users\10821517>
```

Using Backticks (it will allow you to break the cmds into separate line)

Get-Service | `
Where-Object {$_.status -eq "RUNNING"} | `

Select-Object name,displayname

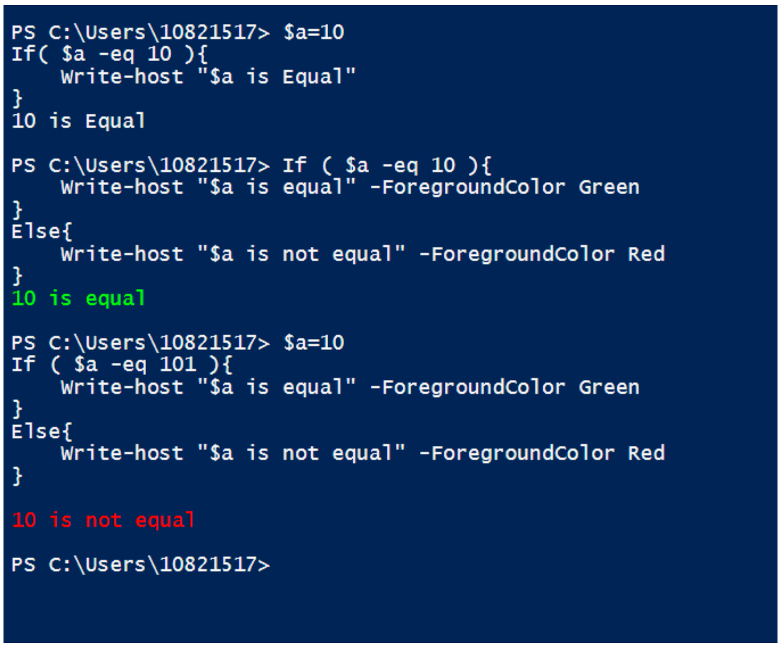
Conditional Statement

->if
->if-else
Nested if-else
Switch

```
$a=10
If( $a -eq 10 ){
	Write-host "$a is Equal"
}
```

#if-else statement

```
$a=10
If ( $a -eq 10 ){
	Write-host "$a is equal" -ForegroundColor Green
}
Else{
	Write-host "$a is not equal" -ForegroundColor Red
}
```

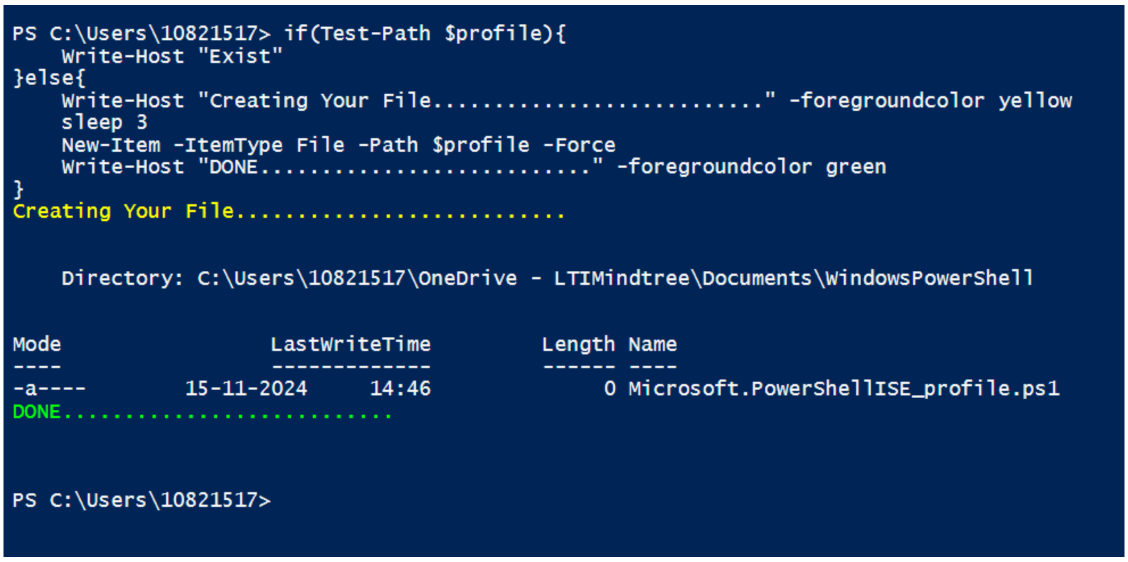


```
PS C:\Users\10821517> $a=10
If( $a -eq 10 ){
    Write-host "$a is Equal"
}
10 is Equal

PS C:\Users\10821517> If ( $a -eq 10 ){
    Write-host "$a is equal" -ForegroundColor Green
}
Else{
    Write-host "$a is not equal" -ForegroundColor Red
}
10 is equal

PS C:\Users\10821517> $a=10
If ( $a -eq 101 ){
    Write-host "$a is equal" -ForegroundColor Green
}
Else{
    Write-host "$a is not equal" -ForegroundColor Red
}

10 is not equal

PS C:\Users\10821517>
```

Task go to documents folder and remove Windows PowerShell Dir manually, write a Ps script to create $PROFILE, if it doesn't exists

```
if(Test-Path $profile){
   Write-Host "Exist"
}else{
   Write-Host "Creating Your File.........................." -foregroundcolor yellow
```

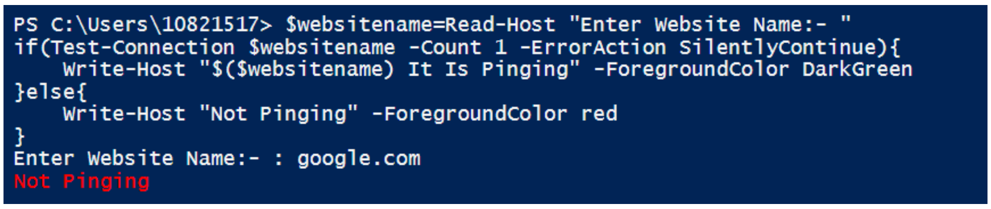
```
    sleep 3
    New-Item -ItemType File -Path $profile -Force
    Write-Host "DONE.........................." -foregroundcolor green
}
```



```
PS C:\Users\10821517> if(Test-Path $profile){
    Write-Host "Exist"
}else{
    Write-Host "Creating Your File..........................." -foregroundcolor yellow
    sleep 3
    New-Item -ItemType File -Path $profile -Force
    Write-Host "DONE..........................." -foregroundcolor green
}
Creating Your File...........................


    Directory: C:\Users\10821517\OneDrive - LTIMindtree\Documents\WindowsPowerShell


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        15-11-2024     14:46              0 Microsoft.PowerShellISE_profile.ps1
DONE...........................



PS C:\Users\10821517>
```

Task to check any website is pinging or not?

```
$websitename=Read-Host "Enter Website Name:- "
if(Test-Connection $websitename -Count 1 -ErrorAction SilentlyContinue){
   Write-Host "$($websitename) It Is Pinging" -ForegroundColor DarkGreen
}else{
   Write-Host "Not Pinging" -ForegroundColor red
}
```



```
PS C:\Users\10821517> $websitename=Read-Host "Enter Website Name:- "
if(Test-Connection $websitename -Count 1 -ErrorAction SilentlyContinue){
    Write-Host "$($websitename) It Is Pinging" -ForegroundColor DarkGreen
}else{
    Write-Host "Not Pinging" -ForegroundColor red
}
Enter Website Name:- : google.com
Not Pinging
```

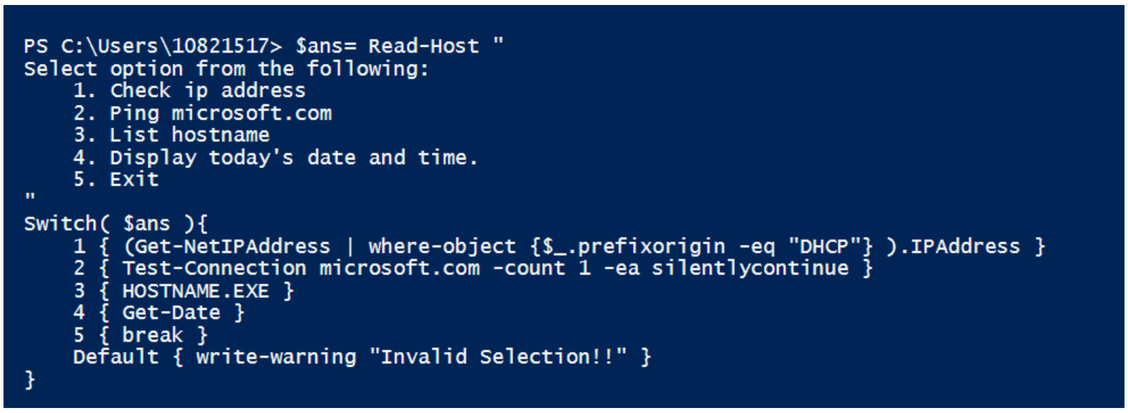
Nested If-Else

```
If(){

}elseif(){

}elseif(){

}elseif(){

}else{

}
```

```
[int]$age= Read-Host "Enter Your Age:- "
If( $age -lt 18 ){
	Write-host "You can't Vote"
}elseif(($age -gt 18) -or ($age -lt 60)){
	Write-host "Enough-Take rest"
}
elseif(($age -gt 60) -or ($age -lt 100)){
	Write-host "Enough-Take rest"
}else{
	Write-warning "Invalid"
}
```

Switch Stmt

```
$ans= Read-Host "
Select option from the following:
  1. Check ip address
  2. Ping microsoft.com
  3. List hostname
  4. Display today's date and time.
  5. Exit
"
Switch( $ans ){
	1 { (Get-NetIPAddress | where-object {$_.prefixorigin -eq "DHCP"} ).IPAddress }
	2 { Test-Connection microsoft.com -count 1 -ea silentlycontinue }
	3 { HOSTNAME.EXE }
	4 { Get-Date }
	5 { break }
	Default { write-warning "Invalid Selection!!" }
}
```

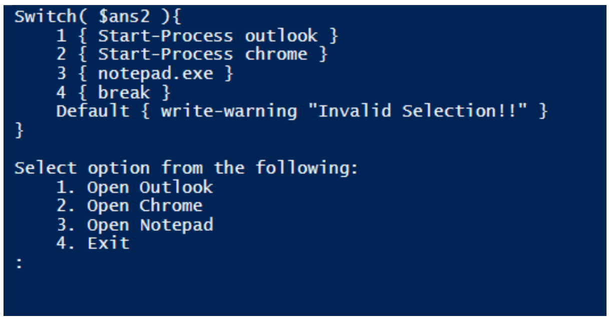


```
PS C:\Users\10821517> $ans= Read-Host "
Select option from the following:
    1. Check ip address
    2. Ping microsoft.com
    3. List hostname
    4. Display today's date and time.
    5. Exit
"
Switch( $ans ){
    1 { (Get-NetIPAddress | where-object {$_.prefixorigin -eq "DHCP"} ).IPAddress }
    2 { Test-Connection microsoft.com -count 1 -ea silentlycontinue }
    3 { HOSTNAME.EXE }
    4 { Get-Date }
    5 { break }
    Default { write-warning "Invalid Selection!!" }
}
```

TASK1

```
$ans2= Read-Host "
Select option from the following:
	1. Open Outlook
	2. Open Chrome
	3. Open Notepad
```

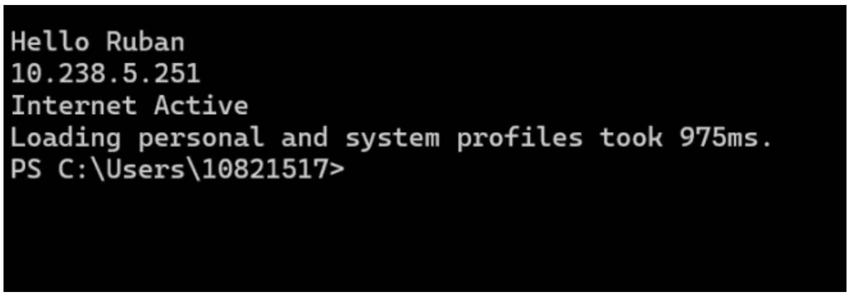
```
        4. Exit
"
Switch( $ans2 ){
        1 { Start-Process outlook }
        2 { Start-Process chrome }
        3 { notepad.exe }
        4 { break }
        Default { write-warning "Invalid Selection!!" }
}
```



```
Switch( $ans2 ){
    1 { Start-Process outlook }
    2 { Start-Process chrome }
    3 { notepad.exe }
    4 { break }
    Default { write-warning "Invalid Selection!!" }
}

Select option from the following:
    1. Open Outlook
    2. Open Chrome
    3. Open Notepad
    4. Exit
:
```

TASK2

```
write-host "Hello Ruban"
write-host (Get-NetIPAddress | where-object {$_.prefixorigin -eq "DHCP"} ).IPAddress
if( Test-Connection microsoft.com -ea silentlycontinue -count 1){
        write-host "Internet Active"
}
else{
        write-host "Internet Not Active"
}
```



```
Hello Ruban
10.238.5.251
Internet Active
Loading personal and system profiles took 975ms.
PS C:\Users\10821517>
```

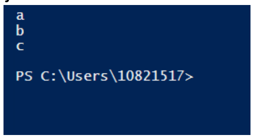
Day3~Practical PowerShell

16 November 2024 08:55

LOOPS

1. **While Loop**

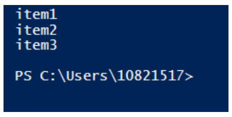
```
$arr1 = @("a","b","c")
$c=0
cls
while ( $c -lt $arr1.Length){
   $arr1[$c]
   $c+=1
   sleep 0.90
}
```



```
a
b
c

PS C:\Users\10821517>
```

2. **Do-While Loop**

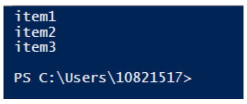
```
cls
$array= @("item1","item2","item3")
$counter = 0

do{
   $array[$counter]
   $counter  += 1
   sleep 1
}while($counter -lt $array.Length)
```



```
item1
item2
item3

PS C:\Users\10821517>
```

3. **For Loop**

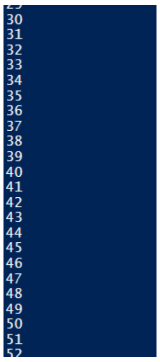
```
cls
$array1 = @("item1","item2","item3")
for($i=0; $i -lt $array1.Length ;$i++){
   $array1[$i]
   sleep 1
}
```



```
item1
item2
item3

PS C:\Users\10821517>
```

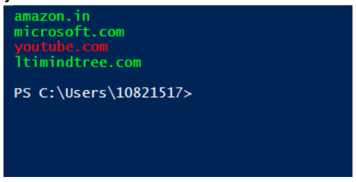
4. **ForEach Loop**

```
cls
$array2 = @(1..93)

foreach( $a in $array2) {
   $a
   sleep 0.60
}
```



Task to check any website is pinging or not if pinging make it green or else make it red

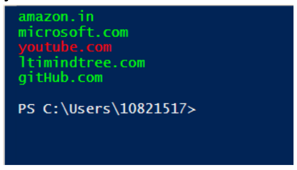
```
cls
$web=@("amazon.in","microsoft.com","youtube.com","ltimindtree.com")
foreach($w in $web){
   if(Test-Connection $w -Count 1 -ea SilentlyContinue){
      Write-Host $w -ForegroundColor Green

   }else{
      Write-Host $w -ForegroundColor red
   }
}
```



```
amazon.in
microsoft.com
youtube.com
ltimindtree.com

PS C:\Users\10821517>
```

Task to Access all the site name from a file and ping them

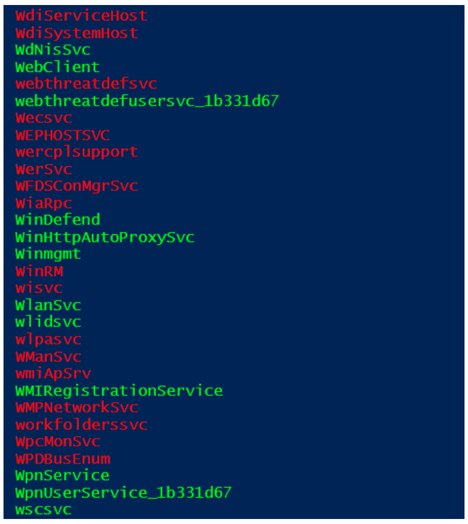
```
cls
notepad.exe sites.txt
$web= Get-Content "C:\Users\10821517\sites.txt"
foreach($w in $web){
   if(Test-Connection $w -Count 1 -ea SilentlyContinue){
      Write-Host $w -ForegroundColor Green

   }else{
      Write-Host $w -ForegroundColor red
   }
}
```



```
amazon.in
microsoft.com
youtube.com
ltimindtree.com
gitHub.com

PS C:\Users\10821517>
```

Task put the running services in green colour and stopped in red colour

```
cls
foreach($svc in Get-Service){

if ($svc.status -eq "Running"){
   Write-Host $svc.name -ForegroundColor Green
}else{
   Write-Host $svc.name -ForegroundColor Red
}

}
```



```
WdiServiceHost
WdiSystemHost
WdNisSvc
WebClient
webthreatdefsvc
webthreatdefusersvc_1b331d67
Wecsvc
WEPHOSTSVC
wercplsupport
WerSvc
WFDSConMgrSvc
WiaRpc
WinDefend
WinHttpAutoProxySvc
Winmgmt
WinRM
wisvc
WlanSvc
wlidsvc
wlpasvc
WManSvc
wmiApSrv
WMIRegistrationService
WMPNetworkSvc
workfolderssvc
WpcMonSvc
WPDBusEnum
WpnService
WpnUserService_1b331d67
wscsvc
```

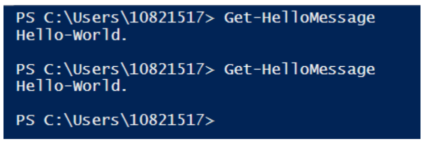
Functions

Types -> Basic Functions
          -> Advanced Functions = basic + advance function parameters

Basic Function

Function verb-noun ->>>>>> Syntax [in future if u want to this function into your own cmdlets]

```
function Get-HelloMessage {
       Write-host "Hello-World."
}
Cls
Get-HelloMessage
```

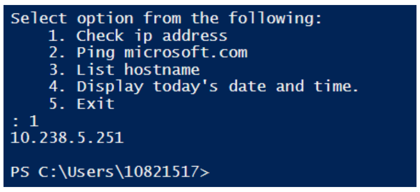


```
PS C:\Users\10821517> Get-HelloMessage
Hello-World.

PS C:\Users\10821517> Get-HelloMessage
Hello-World.

PS C:\Users\10821517>
```

Task Use Case Of Basic Functions

```
function Get-IpAddress{
   (Get-NetIPAddress | where-object {$_.prefixorigin -eq "DHCP"} ).IPAddress
}
function Test-Microsoft {
   Test-Connection microsoft.com -count 1 -ea silentlycontinue
}
function Get-Hostname {
   HOSTNAME.EXE
}
function Get-TodaysDate {
   Get-Date
}

$ans= Read-Host "
Select option from the following:
       1. Check ip address
       2. Ping microsoft.com
       3. List hostname
       4. Display today's date and time.
       5. Exit
"
Switch( $ans ){
       1 { Get-IpAddress }
       2 { Test-Microsoft }
```

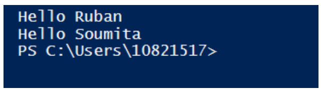
```
        3 { Get-Hostname }
        4 { Get-TodaysDate }
        5 { break }
        Default { write-warning "Invalid Selection!!" }
}
```



```
Select option from the following:
     1. Check ip address
     2. Ping microsoft.com
     3. List hostname
     4. Display today's date and time.
     5. Exit
: 1
10.238.5.251

PS C:\Users\10821517>
```

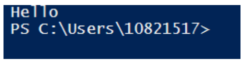
Task write a function that accepts username and prints "Hello yourname"

```
cls
function Get-Message{
    param(
        [string]$name
    )

    Write-Host "Hello $($name)"
}

Get-Message -name "Ruban"
Get-Message -name "Soumita"
```



```
Hello Ruban
Hello Soumita
PS C:\Users\10821517>
```

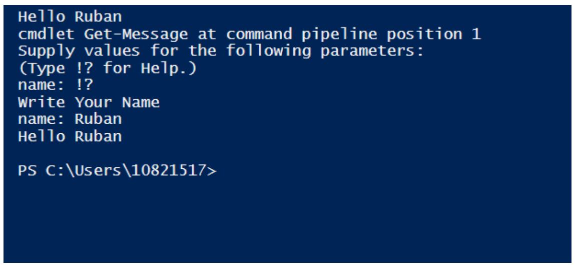
Get-Message
If you forget to provide name param then it will happen



```
Hello
PS C:\Users\10821517>
```

To overcome from this problem use this

```
param(
        [Parameter(Mandatory=$true,HelpMessage="Write Your Name")] [string]$name
)
```



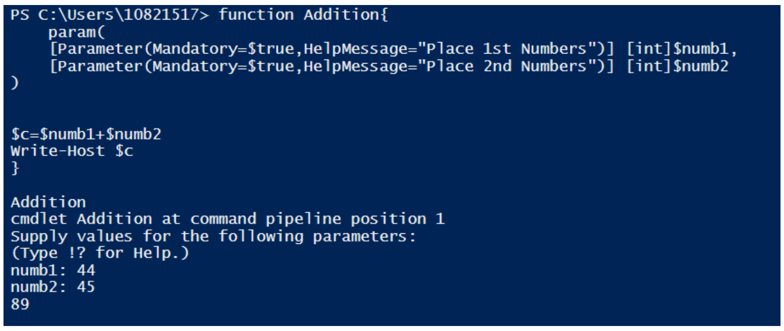
```
Hello Ruban
cmdlet Get-Message at command pipeline position 1
Supply values for the following parameters:
(Type !? for Help.)
name: !?
Write Your Name
name: Ruban
Hello Ruban

PS C:\Users\10821517>
```

Task Add 2 numbers by using function

```
function Addition{
    param(
        [Parameter(Mandatory=$true,HelpMessage="Place 1st Numbers")] [int]$numb1,
        [Parameter(Mandatory=$true,HelpMessage="Place 2nd Numbers")] [int]$numb2
)


$c=$numb1+$numb2
Write-Host $c
```

}

Addition



```
PS C:\Users\10821517> function Addition{
     param(
     [Parameter(Mandatory=$true,HelpMessage="Place 1st Numbers")] [int]$numb1,
     [Parameter(Mandatory=$true,HelpMessage="Place 2nd Numbers")] [int]$numb2
)


$c=$numb1+$numb2
Write-Host $c
}

Addition
cmdlet Addition at command pipeline position 1
Supply values for the following parameters:
(Type !? for Help.)
numb1: 44
numb2: 45
89
```

Note : nothing can be written between param and function name

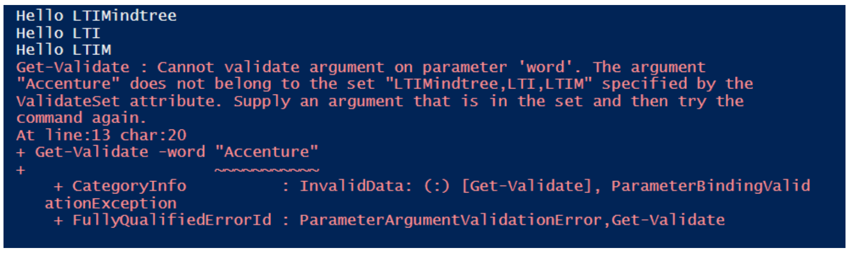
To see all the functions we used till now

Get-ChildItem function:\
(Built-in and custom functions are running into the system right now!!)

CmdletBinding attribute Is used for converting the basic fun to adv function

To allow user which keyword they can put

```
function Get-Validate{
[cmdletbinding()]
param(
    [validateSet("LTIMindtree","LTI","LTIM")][string]$word
    )
    Write-Host "Hello $word"
}

cls
Get-Validate -word "LTIMindtree"
Get-Validate -word "LTI"
Get-Validate -word "LTIM"
Get-Validate -word "Accenture"
```



```
Hello LTIMindtree
Hello LTI
Hello LTIM
Get-Validate : Cannot validate argument on parameter 'word'. The argument
"Accenture" does not belong to the set "LTIMindtree,LTI,LTIM" specified by the
ValidateSet attribute. Supply an argument that is in the set and then try the
command again.
At line:13 char:20
+ Get-Validate -word "Accenture"
+                    ~~~~~~~~~~~
    + CategoryInfo          : InvalidData: (:) [Get-Validate], ParameterBindingValid
   ationException
    + FullyQualifiedErrorId : ParameterArgumentValidationError,Get-Validate
```

Handle with password

```
$pwd=Get-Credential -UserName trainee\Ruban -Message "Type only ur PWD"
$pwd
```

Loading Screen

```
for ( $i = 1; $i -le 100; $i++ ){
        Write-progress -Activity "Search in progress" -status "$i% complete:" -PercentComplete $i
        Start-Sleep -Milliseconds 250
}
```

Search in progress.
23% complete:.

Open your vm

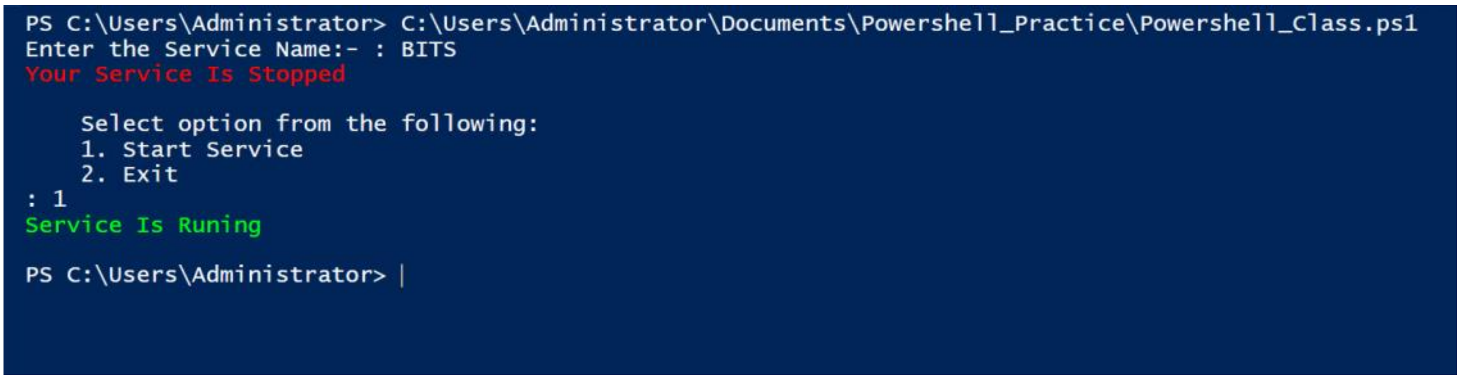
<#Task
Ask user to input a service name(BITS) and check it the service is running or stopped.

if the services is running, then ask user to
1.stop the service
2.exit(no changes)

if the service is stoppped ,then ask the user to
1.start the sevice
2.exit (no changes)

#>

```
$svc=read-host "Enter the Service Name:- "
$s= Get-service -Name $svc
if($s.status -eq "Running"){
   Write-Host "Your Service Is Running" -ForegroundColor Green
   $ans= Read-Host "
   Select option from the following:
       1. Stop Service
       2. Exit
"
Switch( $ans ){
       1 { Stop-Service $svc
       Write-Host "Service Is Stopped" -ForegroundColor Red

          }
       2 { break }
       Default { write-warning "Invalid Selection!!" }
}
}else{
Write-Host "Your Service Is Stopped" -ForegroundColor Red
$ans2= Read-Host "
   Select option from the following:
       1. Start Service
       2. Exit
"
Switch( $ans2 ){
       1 { Start-Service $svc
          Write-Host "Service Is Runing" -ForegroundColor Green}
       2 { break }
       Default { write-warning "Invalid Selection!!" }
}

}
```


```
PS C:\Users\Administrator> C:\Users\Administrator\Documents\Powershell_Practice\Powershell_Class.ps1
Enter the Service Name:- : BITS
Your Service Is Stopped

    Select option from the following:
    1. Start Service
    2. Exit
: 1
Service Is Runing

PS C:\Users\Administrator> |
```

Conversion

Converting to HTML page

 get-command -verb

#listing all the commands for conversion

```
Get-Command -Verb convertto
```

#display all the service full name that start with "a"

```
Get-Service -DisplayName a* | `
Select-Object Displayname, status, StartType | Out-GridView
```

#converting to HTML page

```
Get-Service -DisplayName a* | `
Select-Object DisplayName, status, starttype | `
ConvertTo-Html |`
Out-File a-services.html

.\a-services.html
```

#converting to JSON

```
Get-Service -DisplayName a* | `
Select-Object DisplayName, status, starttype | `
ConvertTo-Json |`
Out-File a-services.json

.\a-services.json
```

#converting to CSV

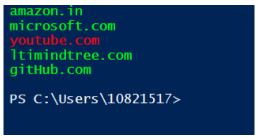
```
Get-Service -DisplayName a* | `
Select-Object DisplayName, status, starttype | `
ConvertTo-Csv |`
Out-File a-services.csv

.\a-services.csv
```

Day4~Practical PowerShell

18 November 2024 08:59

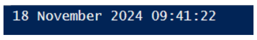
->functions + pipeline + foreach

```
function get-ping{
    param([string] $sites)
    if(Test-Connection $sites -Count 1 -ea SilentlyContinue){
        Write-Host $sites -ForegroundColor Green

    }else{
        Write-Host $sites -ForegroundColor red
    }
}

$websites= Get-Content "C:\Users\10821517\sites.txt"

$websites | foreach {
    get-ping -sites $_
}
```



```
amazon.in
microsoft.com
youtube.com
ltimindtree.com
gitHub.com

PS C:\Users\10821517>
```

Create a clock like this



```
18 November 2024 09:41:22
```

```
function get-clock{
    cls
    Get-Date
    sleep 1
    get-clock
}

get-clock
```

Script Execution

->Either you will be allowed to run the script or denied

Ways to execute a script

1. Local
2. Remote

To check the current exe policy

get-ExecutionPolicy

Execution Policy

It is 4 types

1. **Unrestricted**
   - It will allow local and remote user to run any script
   - The most unsecured policy, not to be used in production
2. **Restricted**
   - It will not allow local or remote user to run ANY script.
   - It used in domain controller.
3. **Remote-Signed**
   - Local user is not required to run script with digital certificate, but
   - Remote users must have a Digital Certificate binded with the script .
4. **All-signed**
   - Both local and remote users must have digital certificate.

How to create a Digital Certificate ?

-> makecert.exe (deprecated)
-> IIS web server
-> LINUX ----> openssl
-> New-selfsignedCertificate (v5.0)

To change the ExecutionPolicy ------> as admin

Set-ExecutionPolicy <policy_name>

To create a digitally signed script:

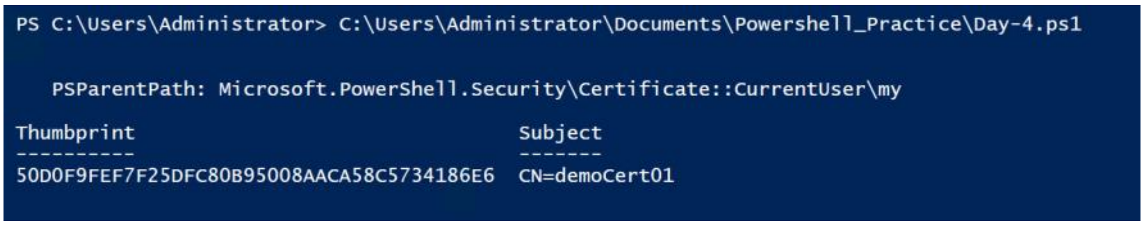
- Create a script ----------> make sure you save that script
- Create Certificate ----------------> New-SelfSignedcertificate
- Export the certificate---------------> user cmdlets, GUI
- Import/Provisioning the certificate ->>>> cmdlets, GUI
- Bind the certificate with script------> set-Authenticodesignature

STEP 01:- CREATE A SCRIPT

```
Get-date
```

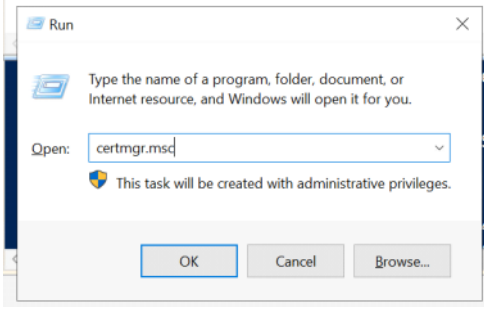
STEP 02:- Creating the certificate

```
New-SelfSignedCertificate -CertStoreLocation Cert:\currentuser\my `
-subject "CN=demoCert01" `
-KeyAlgorithm RSA `
-KeyLength 1024 `
-Provider "Microsoft Enhanced RSA and AES Cryptographic Provider" `
-KeyExportPolicy Exportable `
-KeyUsage DigitalSignature `
-Type CodeSigningCert
```



```
PS C:\Users\Administrator> C:\Users\Administrator\Documents\Powershell_Practice\Day-4.ps1

   PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\my

Thumbprint                                Subject
----------                                -------
50D0F9FEF7F25DFC80B95008AACA58C5734186E6  CN=demoCert01
```

Check certificates In GUI



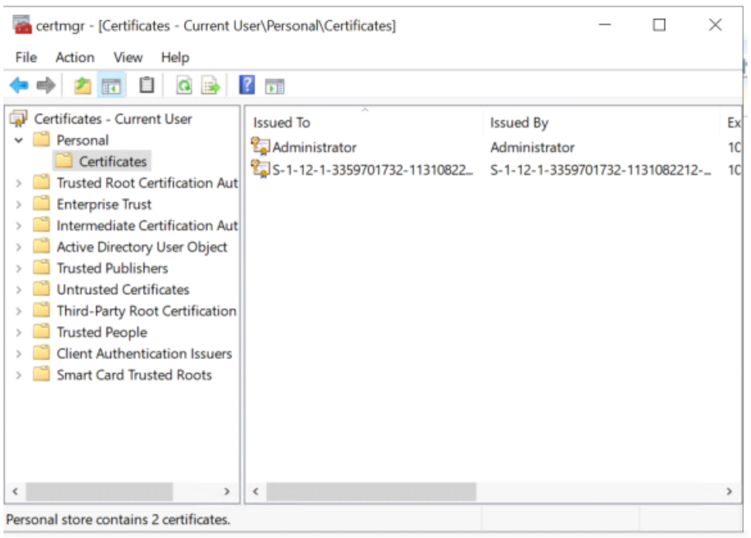
Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: certmgr.msc

This task will be created with administrative privileges.

Screen clipping taken: 18-11-2024 10:40



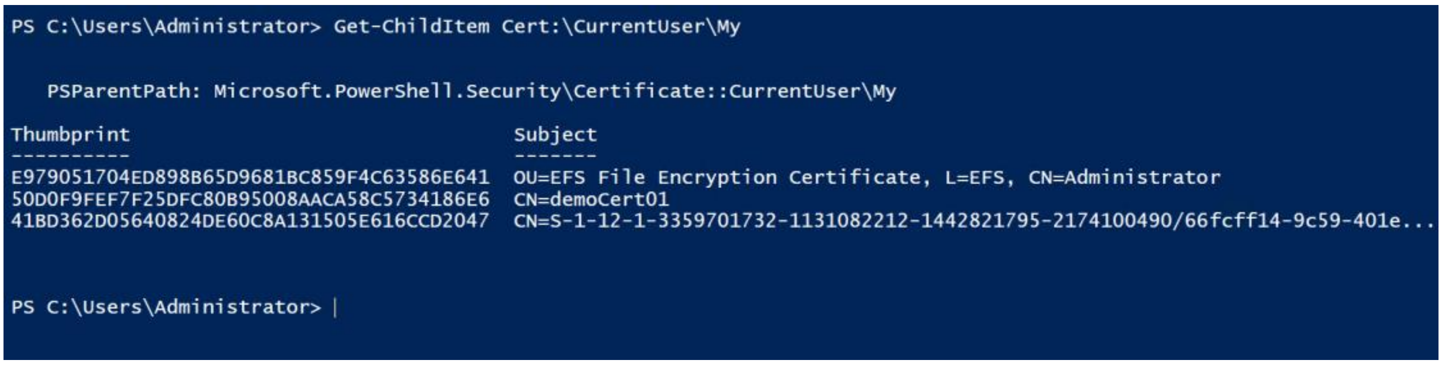
certmgr - [Certificates - Current User\Personal\Certificates]

File
Action
View
Help
Certificates - Current User
Personal
Certificates
Trusted Root Certification Aut
Enterprise Trust
Intermediate Certification Aut
Active Directory User Object
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification
Trusted People
Client Authentication Issuers
Smart Card Trusted Roots

Issued To	Issued By	Ex
Administrator	Administrator	1(
S-1-12-1-3359701732-11310822...	S-1-12-1-3359701732-1131082212-...	1(

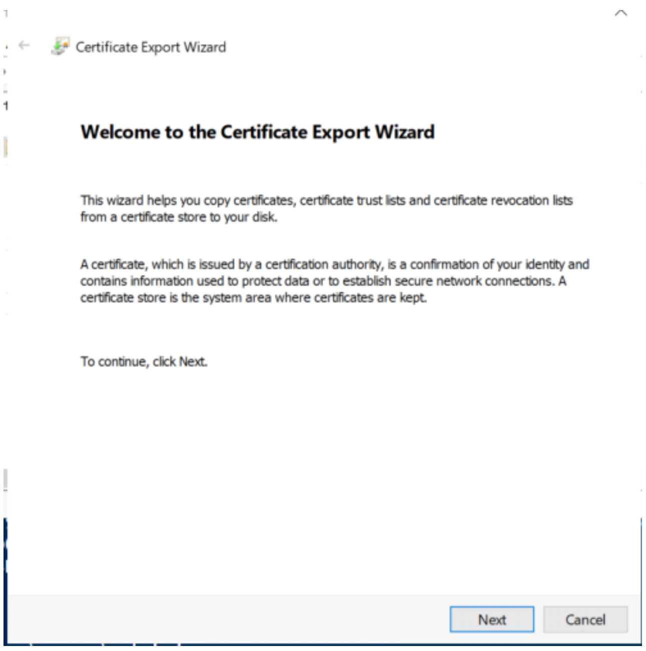
Personal store contains 2 certificates.

Check certificates In Console



```
PS C:\Users\Administrator> Get-ChildItem Cert:\CurrentUser\My


   PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My

Thumbprint                                Subject
----------                                -------
E979051704ED898B65D9681BC859F4C63586E641  OU=EFS File Encryption Certificate, L=EFS, CN=Administrator
50D0F9FEF7F25DFC80B95008AACA58C5734186E6  CN=demoCert01
41BD362D05640824DE60C8A131505E616CCD2047  CN=S-1-12-1-3359701732-1131082212-1442821795-2174100490/66fcff14-9c59-401e...


PS C:\Users\Administrator> |
```

STEP 03:- Export the certificate
#export the pfxCertificate
Win+R -----> certmgr.msc -----> personal ------>certificate



Certificate Export Wizard

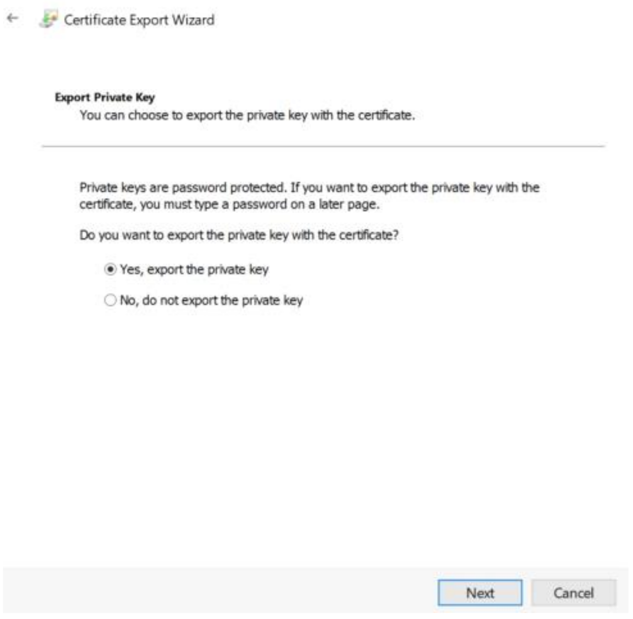
Welcome to the Certificate Export Wizard

This wizard helps you copy certificates, certificate trust lists and certificate revocation lists from a certificate store to your disk.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

Screen clipping taken: 18-11-2024 11:24



Certificate Export Wizard

Export Private Key
You can choose to export the private key with the certificate.

Private keys are password protected. If you want to export the private key with the certificate, you must type a password on a later page.

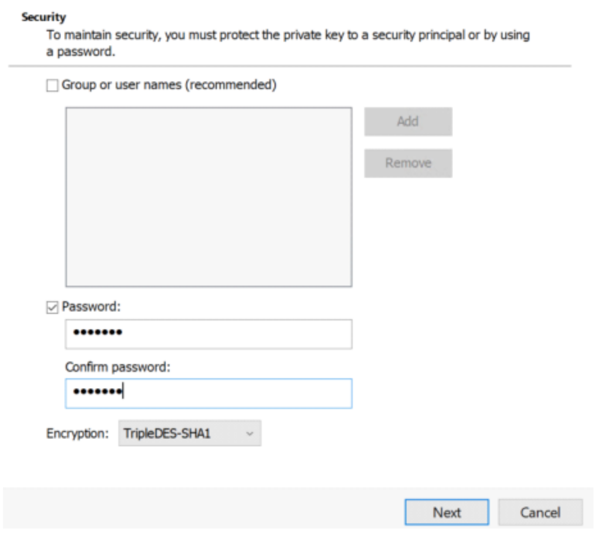
Do you want to export the private key with the certificate?

[✓ Yes, export the private key]

No, do not export the private key

Screen clipping taken: 18-11-2024 11:25

Next



Security

To maintain security, you must protect the private key to a security principal or by using a password.

☐ Group or user names (recommended)

☑ Password:

•••••••

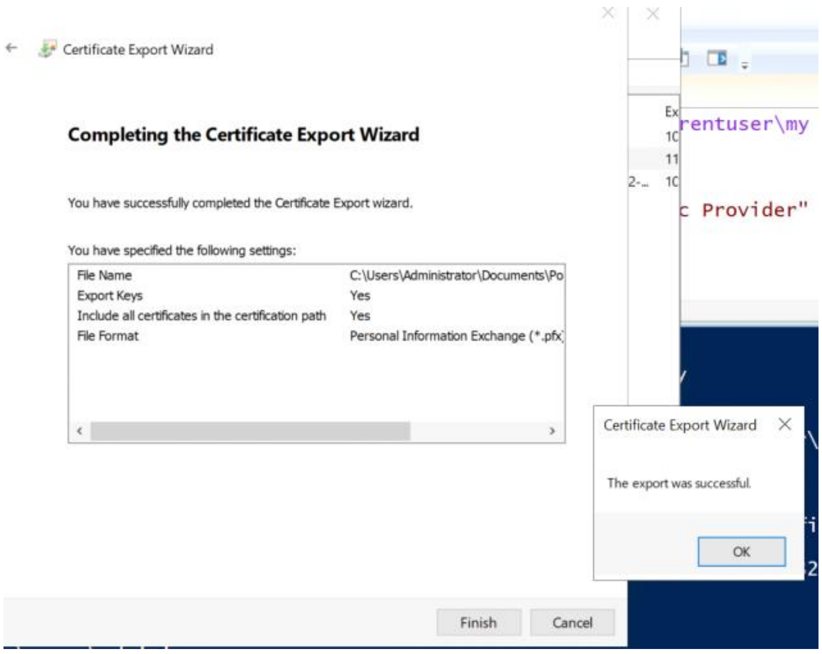
Confirm password:

•••••••

Encryption: TripleDES-SHA1

Screen clipping taken: 18-11-2024 11:27

Now select the folder and save the file



Certificate Export Wizard

Completing the Certificate Export Wizard

You have successfully completed the Certificate Export wizard.

You have specified the following settings:

File Name	C:\Users\Administrator\Documents\Po
Export Keys	Yes
Include all certificates in the certification path	Yes
File Format	Personal Information Exchange (*.pfx)

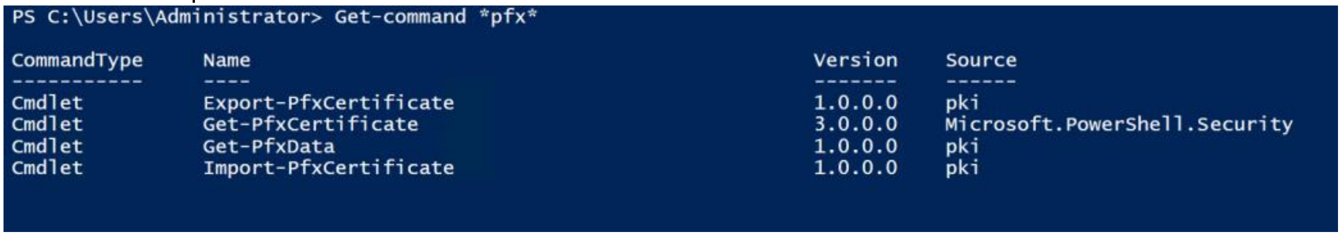
Certificate Export Wizard

The export was successful.

Screen clipping taken: 18-11-2024 11:27

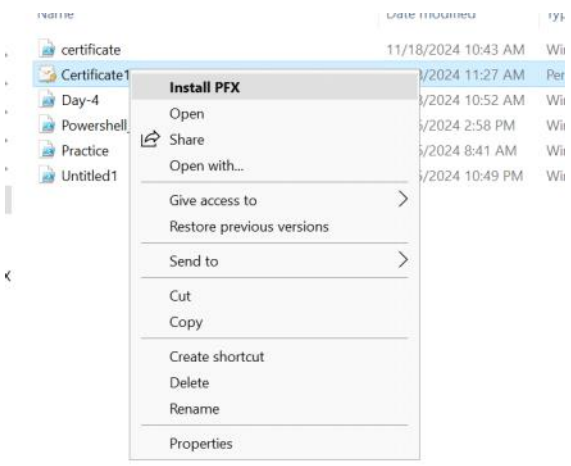
#cmdlets for the certificates

Get-command *pfx*

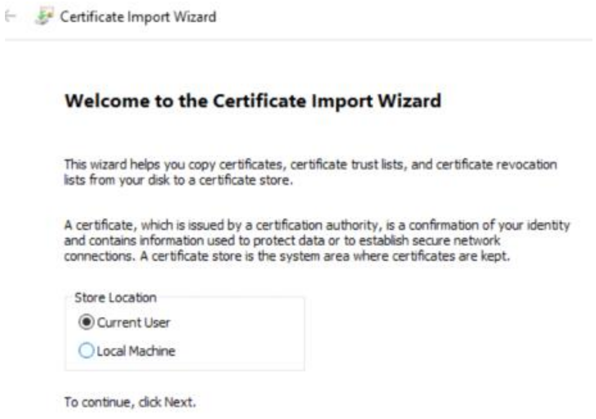


```
PS C:\Users\Administrator> Get-command *pfx*

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Cmdlet          Export-PfxCertificate                              1.0.0.0    pki
Cmdlet          Get-PfxCertificate                                 3.0.0.0    Microsoft.PowerShell.Security
Cmdlet          Get-PfxData                                        1.0.0.0    pki
Cmdlet          Import-PfxCertificate                              1.0.0.0    pki
```

STEP 04:- Provisioning The Certificate



Screen clipping taken: 18-11-2024 11:37


Certificate Import Wizard

Welcome to the Certificate Import Wizard

This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

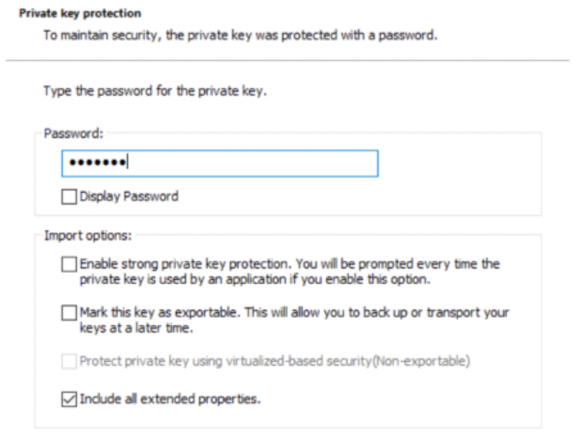
A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

Store Location
[✓ Current User]
Local Machine

To continue, click Next.

Screen clipping taken: 18-11-2024 11:38

Next


Private key protection
To maintain security, the private key was protected with a password.

Type the password for the private key.

Password:
•••••••

☐ Display Password

Import options:

☐ Enable strong private key protection. You will be prompted every time the private key is used by an application if you enable this option.

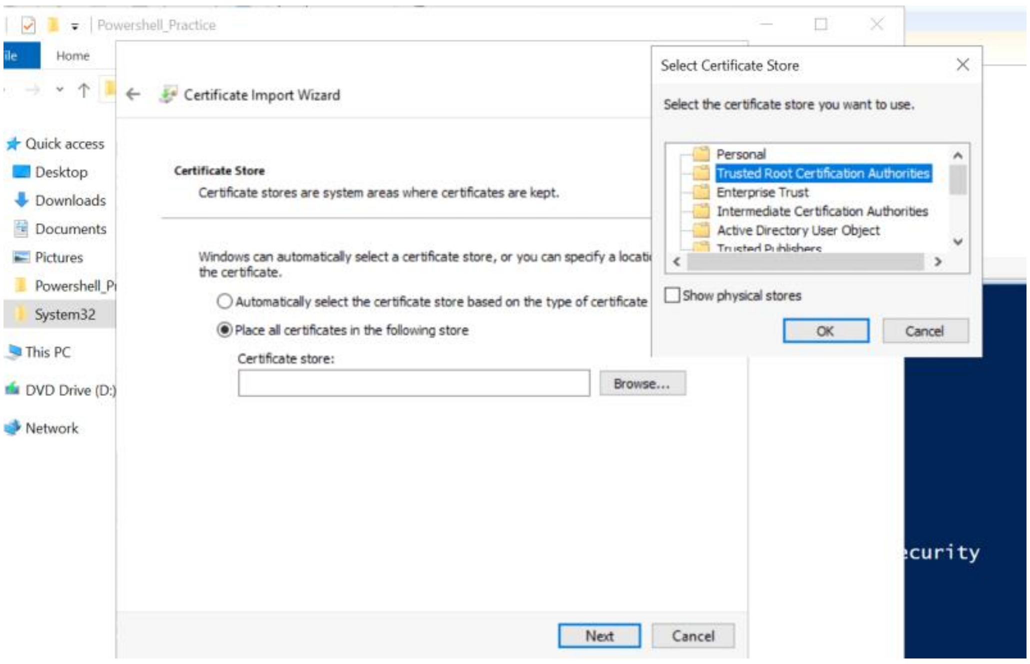
☐ Mark this key as exportable. This will allow you to back up or transport your keys at a later time.

☐ Protect private key using virtualized-based security(Non-exportable)

☑ Include all extended properties.

Screen clipping taken: 18-11-2024 11:38

Type the same passwd which you are created


Certificate Import Wizard

Certificate Store

Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a locati the certificate.

( ) Automatically select the certificate store based on the type of certificate

(•) Place all certificates in the following store

Certificate store:

Select Certificate Store

Select the certificate store you want to use.

- Personal
- Trusted Root Certification Authorities
- Enterprise Trust
- Intermediate Certification Authorities
- Active Directory User Object
- Trusted Publishers

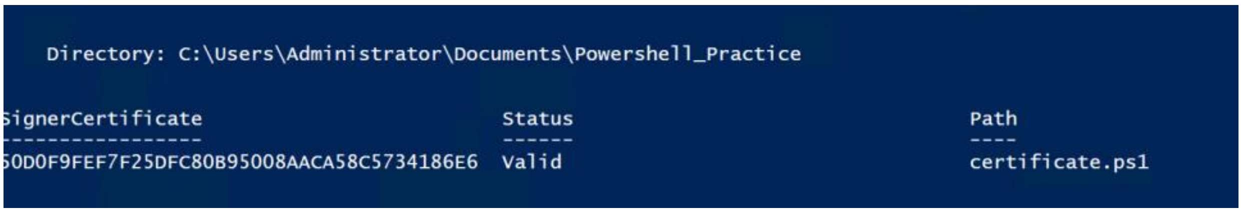
[ ] Show physical stores

Screen clipping taken: 18-11-2024 11:39

NEXT->FINISH->YES

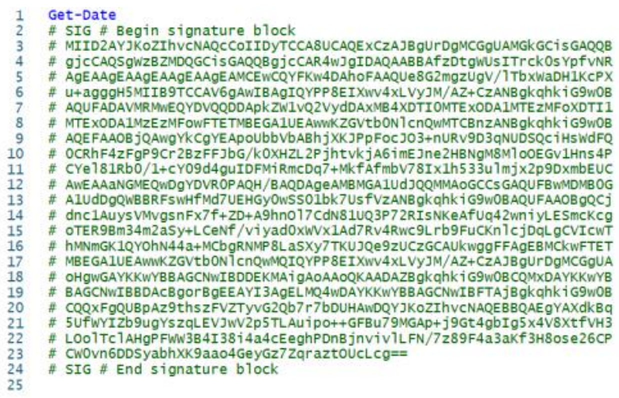
STEP 05:- Binding The Certificate in scripts

$file= "C:\Users\Administrator\Documents\Powershell_Practic\certificate.ps1"
$cert= Get-ChildItem Cert:\CurrentUser\My
\50D0F9FEF7F25DFC80B95008AACA58C5734186E6
Set-AuthenticodeSignature -Certificate $cert -FilePath $file


```
    Directory: C:\Users\Administrator\Documents\Powershell_Practice

SignerCertificate                         Status                                   Path
-----------------                         ------                                   ----
50D0F9FEF7F25DFC80B95008AACA58C5734186E6  Valid                                    certificate.ps1
```

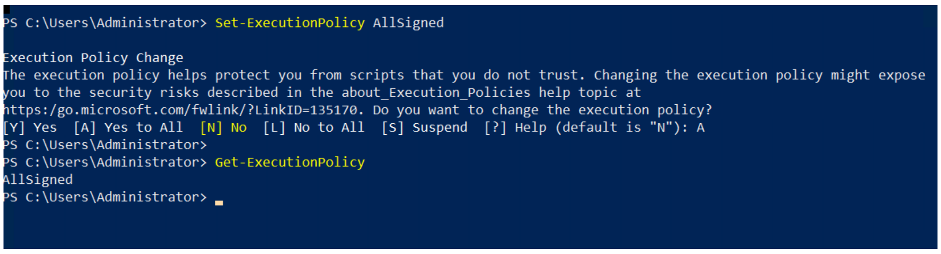
Screen clipping taken: 18-11-2024 12:08

Reopen the file to check it


```
Get-Date
# SIG # Begin signature block
# MIID2AYJKoZIhvcNAQcCoIIDyTCCA8UCAQExCzAJBgUrDgMCGgUAMGkGCisGAQQB
# gjcCAQSgWzBZMDQGCisGAQQBgjcCAR4wJgIDAQAABBAfzDtgWUsITrck0sYpfvNR
# AgEAAgEAAgEAAgEAAgEAMCEwCQYFKw4DAhoFAAQUe8G2mgzUgV/lTbxWaDH1KcPX
# u+agggH5MIIB9TCCAV6gAwIBAgIQYPP8EIXwv4xLVyJM/AZ+CzANBgkqhkiG9w0B
# AQUFADAVMRMwEQYDVQQDDApkZW1vQ2VydDAxMB4XDTI0MTExODA1MTEzMFoXDTI1
# MTExODA1MzEzMFowFTETMBEGA1UEAwwKZGVtb0NlcnQwMTCBnzANBgkqhkiG9w0B
# AQEFAAOBjQAwgYkCgYEApoUbbVbABhjXKJPpFocJO3+nURv9D3qNUDSQciHsWdFQ
# OCRhF4zFgP9Cr2BzFFJbG/kOXHZL2PjhtvkjA6imEJne2HBNgM8MloOEGv1Hns4P
# CYel81Rb0/1+cY09d4guIDFMiRmcDq7+MkfAfmbV78Ix1h533ulmjx2p9DxmbEUC
# AwEAAaNGMEQwDgYDVR0PAQH/BAQDAgeAMBMGA1UdJQQMMAoGCCsGAQUFBwMDMB0G
# A1UdDgQWBBRFswHfMd7UEHGy0wSS01bk7UsfVzANBgkqhkiG9w0BAQUFAAOBgQCj
# dnc1AuysVMvgsnFx7f+ZD+A9hnOl7CdN81UQ3P72RIsNKeAfUq42wniyLESmcKcg
# oTER9Bm34m2aSy+LCeNf/viyad0xWVx1Ad7Rv4Rwc9Lrb9FuCKnlcjDqLgCVIcwT
# hMNmGK1QYOhN44a+MCbgRNMP8LaSXy7TKUJQe9zUCzGCAUkwggFFAgEBMCkwFTET
# MBEGA1UEAwwKZGVtb0NlcnQwMQIQYPP8EIXwv4xLVyJM/AZ+CzAJBgUrDgMCGgUA
# oHgwGAYKKwYBBAGCNwIBDDEKMAigAoAAoQKAADAZBgkqhkiG9w0BCQMxDAYKKwYB
# BAGCNwIBBDAcBgorBgEEAYI3AgELMQ4wDAYKKwYBBAGCNwIBFTAjBgkqhkiG9w0B
# CQQxFgQUBpAz9thszFVZTyvG2Qb7r7bDUHAwDQYJKoZIhvcNAQEBBQAEgYAXdkBq
# 5UfWYIZb9ugYszqLEVJwV2p5TLAuipo++GFBu79MGAp+j9Gt4gbIg5x4V8XtfVH3
# LOolTclAHgPFWW3B4I38i4a4cEeghPDnBjnvivlLFN/7z89F4a3aKf3H8ose26CP
# CWOvn6DDSyabhXK9aao4GeyGz7ZqraztOUcLcg==
# SIG # End signature block

```

Screen clipping taken: 18-11-2024 12:08

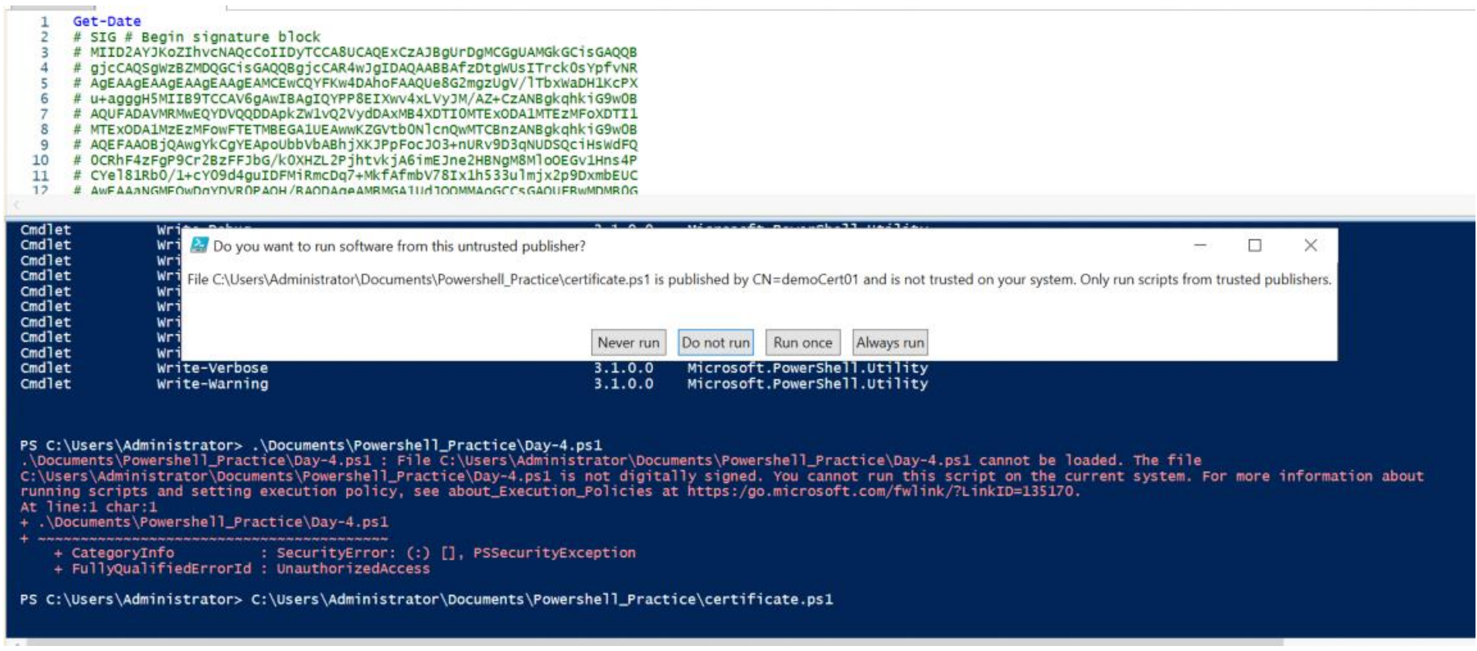
Set-ExecutionPolicy AllSigned



```
PS C:\Users\Administrator> Set-ExecutionPolicy AllSigned

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): A
PS C:\Users\Administrator>
PS C:\Users\Administrator> Get-ExecutionPolicy
AllSigned
PS C:\Users\Administrator>
```

Screen clipping taken: 18-11-2024 12:24

To check this



```
Get-Date
# SIG # Begin signature block
# MIID2AYJKoZIhvcNAQcCoIIDyTCCA8UCAQExCzAJBgUrDgMCGgUAMGkGCisGAQQB
# gjcCAQSgWzBZMDQGCisGAQQBgjcCAR4wJgIDAQAABBAfzDtgWUsITrck0sYpfvNR
# AgEAAgEAAgEAAgEAAgEAMCEwCQYFKw4DAhoFAAQUe8G2mgzUgV/lTbxWaDH1KcPX
# u+agggH5MIIB9TCCAV6gAwIBAgIQYPP8EIXwv4xLVyJM/AZ+CzANBgkqhkiG9w0B
# AQUFADAVMRMwEQYDVQQDDApkZW1vQ2VydDAxMB4XDTI0MTExODA1MTEzMFoXDTI1
# MTExODA1MzEzMFowFTETMBEGA1UEAwwKZGVtb0NlcnQwMTCBnzANBgkqhkiG9w0B
# AQEFAAOBjQAwgYkCgYEApoUbbVbABhjXKJPpFocJO3+nURv9D3qNUDSQciHsWdFQ
# OCRhF4zFgP9Cr2BzFFJbG/k0XHZL2PjhtvkjA6imEJne2HBNgM8Ml0OEGv1Hns4P
# CYel81Rb0/1+cY09d4guIDFMiRmcDq7+MkfAfmbV78Ix1h533ulmjx2p9DxmbEUC
# AwEAAaNGMEQwDgYDVR0PAQH/BAQDAgeAMBMGA1UdJQQMMAoGCCsGAQUFBwMDMB0G
```

Do you want to run software from this untrusted publisher?
File C:\Users\Administrator\Documents\Powershell_Practice\certificate.ps1 is published by CN=demoCert01 and is not trusted on your system. Only run scripts from trusted publishers.
Never run
Do not run
Run once
Always run

```
Cmdlet          Wri
Cmdlet          Wri
Cmdlet          Wri
Cmdlet          Wri
Cmdlet          Wri
Cmdlet          Wri
Cmdlet          Wri
Cmdlet          Wri
Cmdlet          Wri
Cmdlet          Write-Verbose                                      3.1.0.0    Microsoft.PowerShell.Utility
Cmdlet          Write-Warning                                      3.1.0.0    Microsoft.PowerShell.Utility


PS C:\Users\Administrator> .\Documents\Powershell_Practice\Day-4.ps1
.\Documents\Powershell_Practice\Day-4.ps1 : File C:\Users\Administrator\Documents\Powershell_Practice\Day-4.ps1 cannot be loaded. The file
C:\Users\Administrator\Documents\Powershell_Practice\Day-4.ps1 is not digitally signed. You cannot run this script on the current system. For more information about
running scripts and setting execution policy, see about_Execution_Policies at https:/go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:1
+ .\Documents\Powershell_Practice\Day-4.ps1
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess

PS C:\Users\Administrator> C:\Users\Administrator\Documents\Powershell_Practice\certificate.ps1
```

Screen clipping taken: 18-11-2024 12:26

XML

Creating XML and storing in file

Get-service -DisplayName A* | `
Select-object Name, DisplayName, Status, StartType | `
Export-clixml a-services.xml

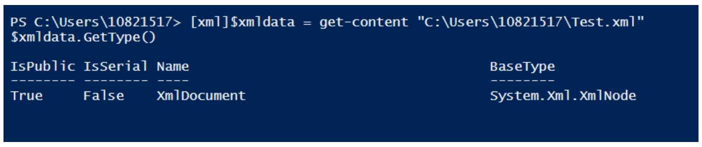
./a-services.xml

Go to Sample XML File (books.xml) | Microsoft Learn this website

Copy and paste all the codes to notepad and check

Calling the data on the PowerShell

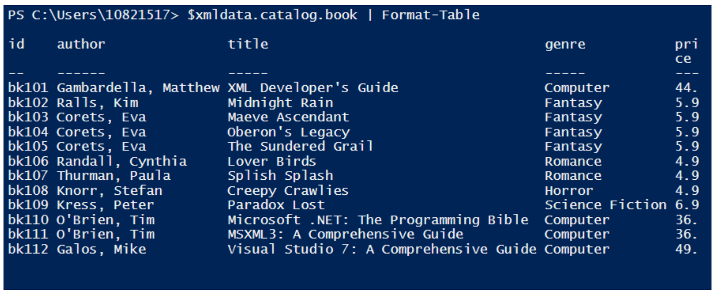
[xml]$xmldata = get-content "C:\Users\10821517\Test.xml"
$xmldata.GetType()



```
PS C:\Users\10821517> [xml]$xmldata = get-content "C:\Users\10821517\Test.xml"
$xmldata.GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     False    XmlDocument                              System.Xml.XmlNode
```

Screen clipping taken: 18-11-2024 12:51

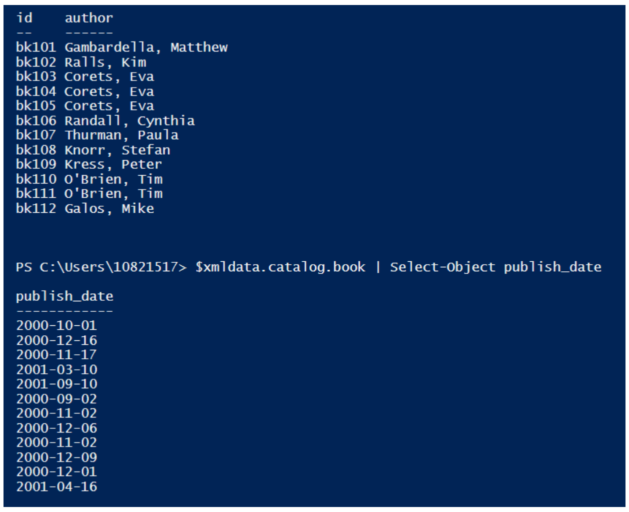
$xmldata.catalog.book | Format-Table [Make it in tabular form



```
PS C:\Users\10821517> $xmldata.catalog.book | Format-Table

id    author               title                                genre           pri
                                                                                ce
--    ------               -----                                -----           ---
bk101 Gambardella, Matthew XML Developer's Guide                Computer        44.
bk102 Ralls, Kim           Midnight Rain                        Fantasy         5.9
bk103 Corets, Eva          Maeve Ascendant                      Fantasy         5.9
bk104 Corets, Eva          Oberon's Legacy                      Fantasy         5.9
bk105 Corets, Eva          The Sundered Grail                   Fantasy         5.9
bk106 Randall, Cynthia     Lover Birds                          Romance         4.9
bk107 Thurman, Paula       Splish Splash                        Romance         4.9
bk108 Knorr, Stefan        Creepy Crawlies                      Horror          4.9
bk109 Kress, Peter         Paradox Lost                         Science Fiction 6.9
bk110 O'Brien, Tim         Microsoft .NET: The Programming Bible Computer       36.
bk111 O'Brien, Tim         MSXML3: A Comprehensive Guide        Computer        36.
bk112 Galos, Mike          Visual Studio 7: A Comprehensive Guide Computer      49.
```

Screen clipping taken: 18-11-2024 16:30

$xmldata.catalog.book | Select-Object publish_date **[Select an object from the data mentioned in the xml]**



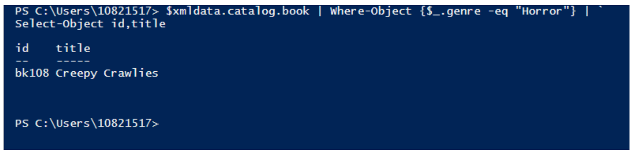
```
id    author
--    ------
bk101 Gambardella, Matthew
bk102 Ralls, Kim
bk103 Corets, Eva
bk104 Corets, Eva
bk105 Corets, Eva
bk106 Randall, Cynthia
bk107 Thurman, Paula
bk108 Knorr, Stefan
bk109 Kress, Peter
bk110 O'Brien, Tim
bk111 O'Brien, Tim
bk112 Galos, Mike


PS C:\Users\10821517> $xmldata.catalog.book | Select-Object publish_date

publish_date
------------
2000-10-01
2000-12-16
2000-11-17
2001-03-10
2001-09-10
2000-09-02
2000-11-02
2000-12-06
2000-11-02
2000-12-09
2000-12-01
2001-04-16
```

Screen clipping taken: 18-11-2024 16:31

#list book title with genre name only

$xmldata.catalog.book | Where-Object {$_.genre -eq "Horror"} | `
Select-Object id,title

[EQ Means it will need the full sentence]

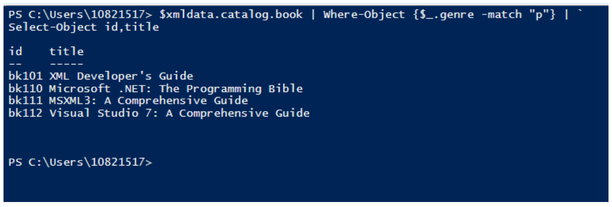


```
PS C:\Users\10821517> $xmldata.catalog.book | Where-Object {$_.genre -eq "Horror"} | `
Select-Object id,title

id    title
--    -----
bk108 Creepy Crawlies


PS C:\Users\10821517>
```

Screen clipping taken: 18-11-2024 16:59

$xmldata.catalog.book | Where-Object {$_.genre -match "p"} | `
Select-Object id,title

[Match means It will pick any matching word]

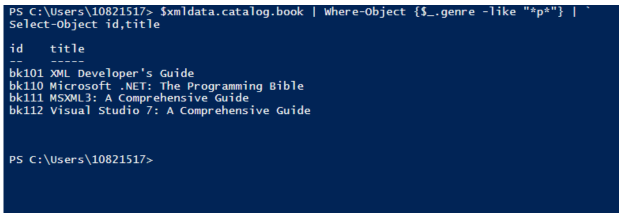


```
PS C:\Users\10821517> $xmldata.catalog.book | Where-Object {$_.genre -match "p"} | `
Select-Object id,title

id    title
--    -----
bk101 XML Developer's Guide
bk110 Microsoft .NET: The Programming Bible
bk111 MSXML3: A Comprehensive Guide
bk112 Visual Studio 7: A Comprehensive Guide


PS C:\Users\10821517>
```

Screen clipping taken: 18-11-2024 17:00

$xmldata.catalog.book | Where-Object {$_.genre -like "*p*"} | `
Select-Object id,title

[It will pick the pattern]

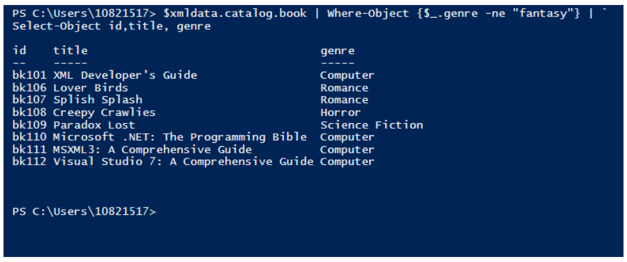


```
PS C:\Users\10821517> $xmldata.catalog.book | Where-Object {$_.genre -like "*p*"} | `
Select-Object id,title

id    title
--    -----
bk101 XML Developer's Guide
bk110 Microsoft .NET: The Programming Bible
bk111 MSXML3: A Comprehensive Guide
bk112 Visual Studio 7: A Comprehensive Guide


PS C:\Users\10821517>
```

Screen clipping taken: 18-11-2024 17:00

$xmldata.catalog.book | Where-Object {$_.genre -ne "fantasy"} | `
Select-Object id,title, genre

[Not equals]

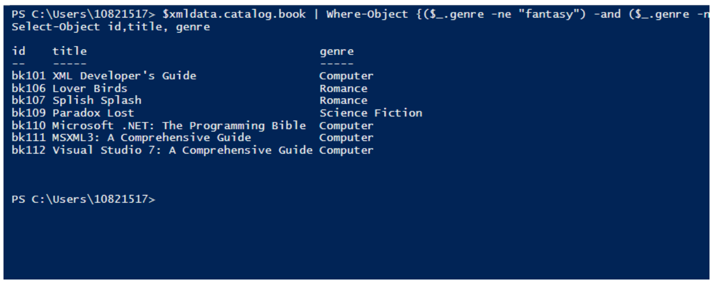


```
PS C:\Users\10821517> $xmldata.catalog.book | Where-Object {$_.genre -ne "fantasy"} | `
Select-Object id,title, genre

id    title                                  genre
--    -----                                  -----
bk101 XML Developer's Guide                  Computer
bk106 Lover Birds                            Romance
bk107 Splish Splash                          Romance
bk108 Creepy Crawlies                        Horror
bk109 Paradox Lost                           Science Fiction
bk110 Microsoft .NET: The Programming Bible  Computer
bk111 MSXML3: A Comprehensive Guide          Computer
bk112 Visual Studio 7: A Comprehensive Guide Computer


PS C:\Users\10821517>
```

Screen clipping taken: 18-11-2024 17:01

$xmldata.catalog.book | Where-Object {($_.genre -ne "fantasy") -and
($_.genre -ne "Horror")} | `
Select-Object id,title, genre

[And operator use]



```
PS C:\Users\10821517> $xmldata.catalog.book | Where-Object {($_.genre -ne "fantasy") -and ($_.genre -n
Select-Object id,title, genre

id    title                                  genre
--    -----                                  -----
bk101 XML Developer's Guide                  Computer
bk106 Lover Birds                            Romance
bk107 Splish Splash                          Romance
bk109 Paradox Lost                           Science Fiction
bk110 Microsoft .NET: The Programming Bible  Computer
bk111 MSXML3: A Comprehensive Guide          Computer
bk112 Visual Studio 7: A Comprehensive Guide Computer


PS C:\Users\10821517>
```

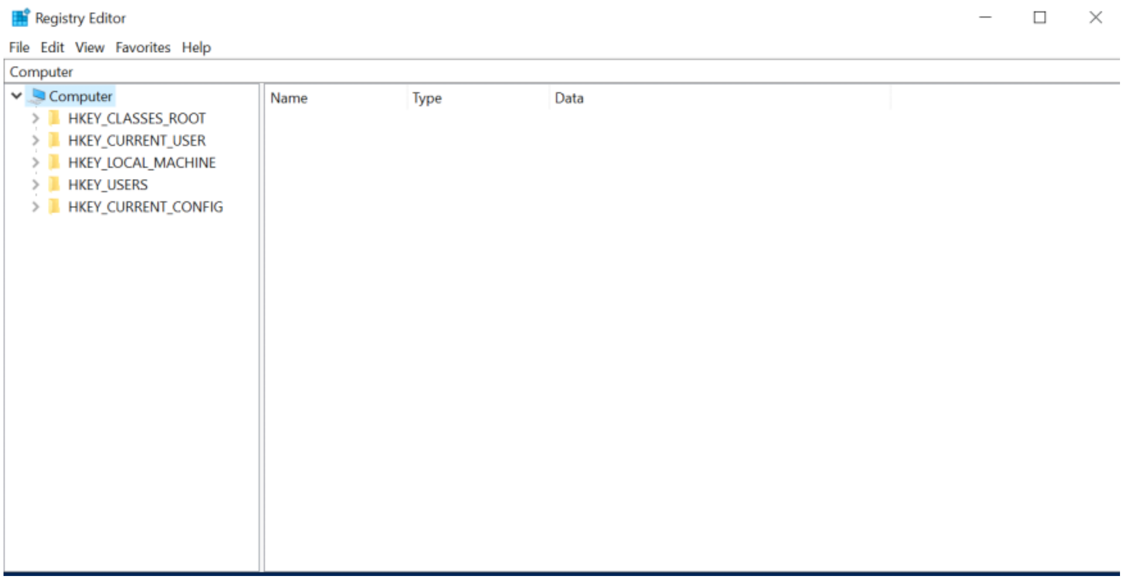
Screen clipping taken: 18-11-2024 17:01

Windows Registry

It is the database of your win os

Without it windows is useless

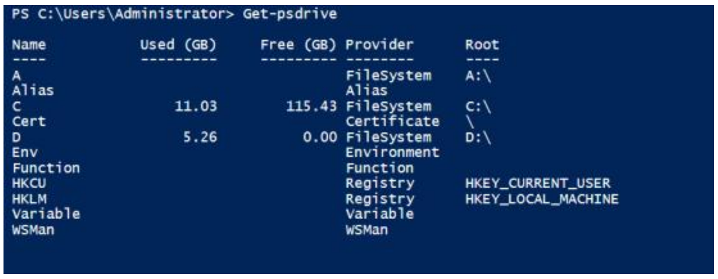
Regedit [write in your run]



Screen clipping taken: 18-11-2024 17:50

HKCU(Hash key Current User)
HKLM(Hash Key Local Machine)

Shows PS Drives
Get-psdrive

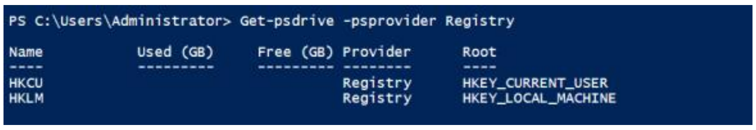


```
PS C:\Users\Administrator> Get-psdrive

Name           Used (GB)     Free (GB) Provider      Root
----           ---------     --------- --------      ----
A                                      FileSystem    A:\
Alias                                  Alias
C                  11.03        115.43 FileSystem    C:\
Cert                                   Certificate   \
D                   5.26          0.00 FileSystem    D:\
Env                                    Environment
Function                               Function
HKCU                                   Registry      HKEY_CURRENT_USER
HKLM                                   Registry      HKEY_LOCAL_MACHINE
Variable                               Variable
WSMan                                  WSMan
```

Screen clipping taken: 18-11-2024 17:59

Cd cert:\ [Accessed via powershell not by cmdprompt]

List Registry Provider
Get-psdrive -psprovider Registry



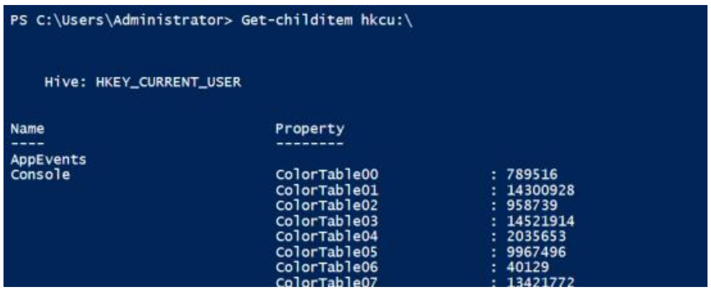
```
PS C:\Users\Administrator> Get-psdrive -psprovider Registry

Name           Used (GB)     Free (GB) Provider      Root
----           ---------     --------- --------      ----
HKCU                                   Registry      HKEY_CURRENT_USER
HKLM                                   Registry      HKEY_LOCAL_MACHINE
```

Screen clipping taken: 18-11-2024 17:59

[Targetting a provider]

TO access the reg key of HKCU

Get-childitem hkcu:\

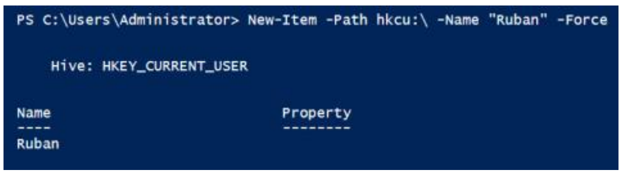


```
PS C:\Users\Administrator> Get-childitem hkcu:\


    Hive: HKEY_CURRENT_USER


Name                           Property
----                           --------
AppEvents
Console                        ColorTable00             : 789516
                               ColorTable01             : 14300928
                               ColorTable02             : 958739
                               ColorTable03             : 14521914
                               ColorTable04             : 2035653
                               ColorTable05             : 9967496
                               ColorTable06             : 40129
                               ColorTable07             : 13421772
```

Screen clipping taken: 18-11-2024 17:59

#create registry key folder

New-Item -Path hkcu:\ -Name "Ruban" -Force

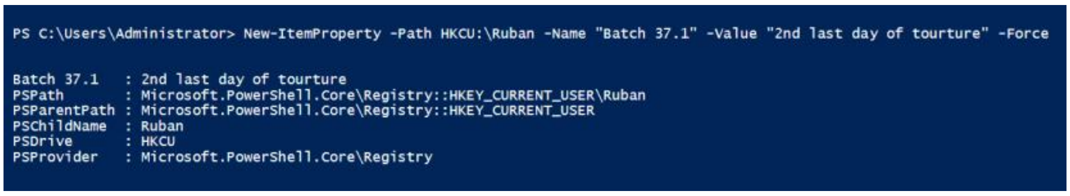


```
PS C:\Users\Administrator> New-Item -Path hkcu:\ -Name "Ruban" -Force

    Hive: HKEY_CURRENT_USER


Name                           Property
----                           --------
Ruban
```

Screen clipping taken: 18-11-2024 18:16

#create registry key inside folder

New-ItemProperty -Path HKCU:\Ruban -Name "Batch 37.1" -Value "2nd last day of tourture" -Force



```
PS C:\Users\Administrator> New-ItemProperty -Path HKCU:\Ruban -Name "Batch 37.1" -Value "2nd last day of tourture" -Force


Batch 37.1   : 2nd last day of tourture
PSPath       : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Ruban
PSParentPath : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER
PSChildName  : Ruban
PSDrive      : HKCU
PSProvider   : Microsoft.PowerShell.Core\Registry
```

Screen clipping taken: 18-11-2024 18:16

#modify the key's value

Set-ItemProperty -Path HKCU:\Ruban -Name "Batch 37.1" -Value "Powershell Murdabad" -Force

#remove the folder

Remove-Item -Path HKCU:\Ruban -Force

# remove the key

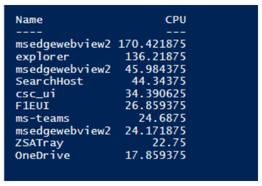
Remove-ItemProperty -Path HKCU:\Ruban -Name "Batch 37.1" -Force

Day5~Practical PowerShell

19 November 2024 09:20

Task :- List top 10 unique processes (Name, CPU(in Descending order)) with high CPU utilization

Get-Process | Sort-Object CPU -Descending | Select-Object -Unique -First 10 name,cpu

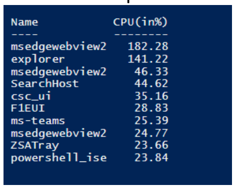


```
Name                   CPU
----                   ---
msedgewebview2 170.421875
explorer        136.21875
msedgewebview2  45.984375
SearchHost       44.34375
csc_ui          34.390625
F1EUI           26.859375
ms-teams          24.6875
msedgewebview2  24.171875
ZSATray             22.75
OneDrive        17.859375
```

Screen clipping taken: 19-11-2024 09:23

Task :- Make the CPU utilization to 2 decimal place

cls
Get-Process | `
Sort-Object CPU -Descending | `
Select-Object Name, `
@{l = "CPU(in%)" ; e={[math]::Round(($_.CPU),2)}} `
-First 10 -Unique



```
Name            CPU(in%)
----            --------
msedgewebview2    182.28
explorer          141.22
msedgewebview2     46.33
SearchHost         44.62
csc_ui             35.16
F1EUI              28.83
ms-teams           25.39
msedgewebview2     24.77
ZSATray            23.66
powershell_ise     23.84
```

Screen clipping taken: 19-11-2024 09:37

Open your DC

WMI (Windows Management Instrumentation)

->It fetches the information of local machine and remote machine too
->It fetches information like

- -Hardware
- -Software
- -service
- -process
- -firmware

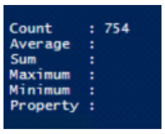
CIM (Common Information Model)
->It is open source
->It fetches the information of local machine and remote machine too
->It fetches information like

- -Hardware
- -Software
- -service
- -process
- -firmware

Cmdlets #Get-WMIObject
Alias: # gwmi
Classes:
- -bios
- -operatingsystem
- -physicalmemory
- -Logicaldisk
- -computersystem

List the all WMI Object

Get-WmiObject -List | Where-Object {$_.name -match '^win32_'}|Measure-Object



```
Count    : 754
Average  :
Sum      :
Maximum  :
Minimum  :
Property :
```

Screen clipping taken: 19-11-2024 09:54

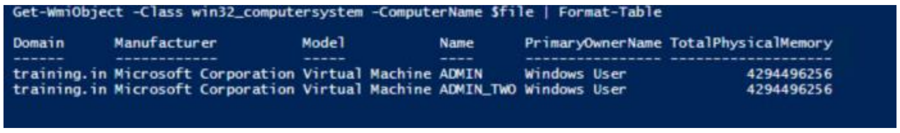
#win32 ~ WMI
#CIM ~ CIM

Fetch the info of bios

Get-WmiObject -Class win32_bios

Fetch The Data From Remote Machine

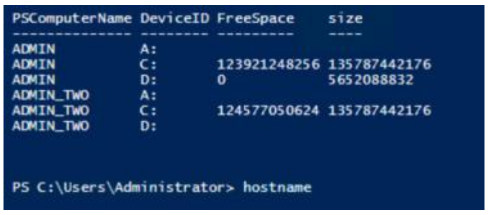
notepad.exe comp.txt [Here please add all the name of your hosts]
$file= Get-Content .\comp.txt
Get-WmiObject -Class win32_computersystem -ComputerName $file | Format-Table



```
Get-WmiObject -Class win32_computersystem -ComputerName $file | Format-Table

Domain      Manufacturer          Model           Name      PrimaryOwnerName TotalPhysicalMemory
------      ------------          -----           ----      ---------------- -------------------
training.in Microsoft Corporation Virtual Machine ADMIN     Windows User              4294496256
training.in Microsoft Corporation Virtual Machine ADMIN_TWO Windows User              4294496256
```

Screen clipping taken: 19-11-2024 10:28

Get-WmiObject -Class win32_logicaldisk -ComputerName $file | Select-Object PScomputerName, DeviceID, FreeSpace, size



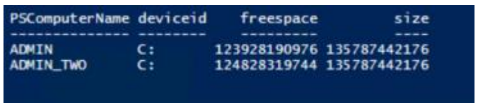
```
PSComputerName DeviceID FreeSpace    size
-------------- -------- ---------    ----
ADMIN          A:
ADMIN          C:       123921248256 135787442176
ADMIN          D:       0            5652088832
ADMIN_TWO      A:
ADMIN_TWO      C:       124577050624 135787442176
ADMIN_TWO      D:


PS C:\Users\Administrator> hostname
```

Screen clipping taken: 19-11-2024 10:28

Get-WmiObject -Class win32_logicaldisk|Get-Member [getting the member]

#Task:- to fetch the c drive size of all machines

Get-WmiObject -Class win32_logicaldisk -ComputerName $file| Where-Object DeviceID -EQ "C:"| Select-Object pscomputername, deviceid, freespace, size| Format-Table

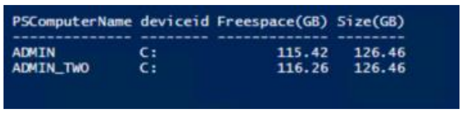


```
PSComputerName deviceid    freespace         size
-------------- --------    ---------         ----
ADMIN          C:       123928190976 135787442176
ADMIN_TWO      C:       124828319744 135787442176
```

Screen clipping taken: 19-11-2024 10:48

#Task:- to fetch the c drive size of all machines and convert it into GB

Get-WmiObject -Class win32_logicaldisk -ComputerName $file `
| Where-Object DeviceID -EQ "C:" `
| Select-Object pscomputername, deviceid, `
@{l="Freespace(GB)" ; e={[math]::Round((($_.freespace)/1GB),2)}}, `
@{l="Size(GB)" ; e={[math]::Round((($_.size)/1GB),2)}} `
| Format-Table



```
PSComputerName deviceid Freespace(GB) Size(GB)
-------------- -------- ------------- --------
ADMIN          C:              115.42   126.46
ADMIN_TWO      C:              116.26   126.46
```

Screen clipping taken: 19-11-2024 10:53

Remote Execution

Taking another machine on remote and do some job on it or

Accessing a remote system to perform task(s)

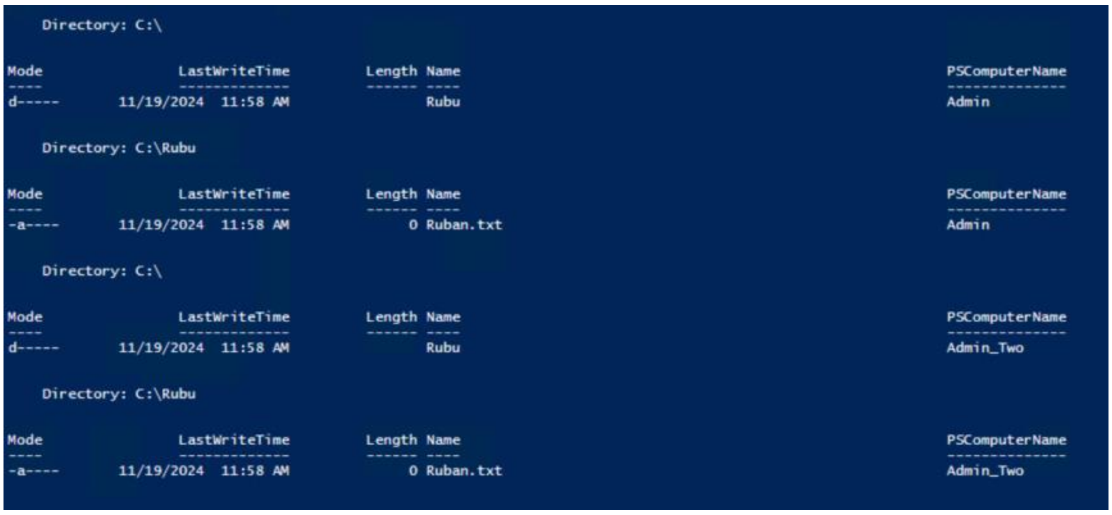
2 ways it will be done

1. Non persistent remoting
   ->Connection is for limited time
   ->1 to many connection
   ->upto 64 systems can be connected
   ->cmdlet (invoke-command)
   It will do the job and disconnected after ending the job

2. Persistent Remoting
   When my job is over I will be destroy the tunnel
   ->1 to 1 connection
   ->cmdlet: get-command -Noun pssession

1. Non Persistent Remoting Code

```
$file = Get-Content .\comp.txt
Invoke-Command -ComputerName $file -ScriptBlock{
   New-Item -ItemType Directory -Path "c:\" -Name "Rubu" -Force
   New-Item -ItemType File -Path "c:\Rubu" -Name "Ruban.txt" -Force
   Set-Content -Path "c:\Rubu\Ruban.txt" -Value "Ruban Subhadip Bhai Assistent" -Force
}
```

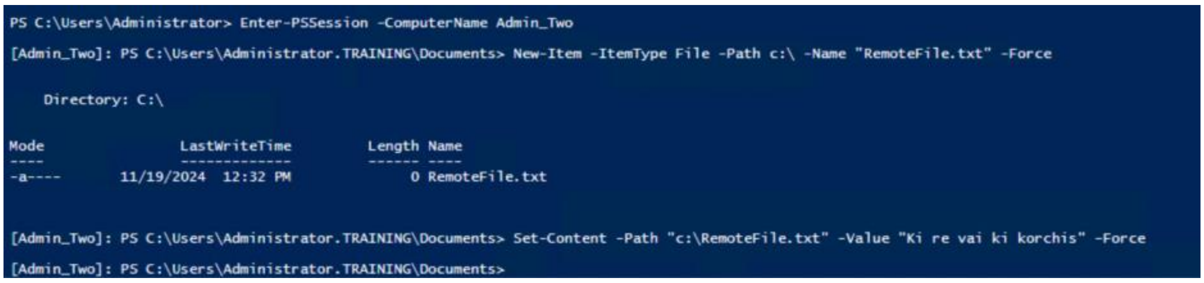


```
    Directory: C:\

Mode                 LastWriteTime         Length Name                                                           PSComputerName
----                 -------------         ------ ----                                                           --------------
d-----        11/19/2024  11:58 AM                Rubu                                                           Admin

    Directory: C:\Rubu

Mode                 LastWriteTime         Length Name                                                           PSComputerName
----                 -------------         ------ ----                                                           --------------
-a----        11/19/2024  11:58 AM              0 Ruban.txt                                                      Admin

    Directory: C:\

Mode                 LastWriteTime         Length Name                                                           PSComputerName
----                 -------------         ------ ----                                                           --------------
d-----        11/19/2024  11:58 AM                Rubu                                                           Admin_Two

    Directory: C:\Rubu

Mode                 LastWriteTime         Length Name                                                           PSComputerName
----                 -------------         ------ ----                                                           --------------
-a----        11/19/2024  11:58 AM              0 Ruban.txt                                                      Admin_Two
```

Screen clipping taken: 19-11-2024 12:00

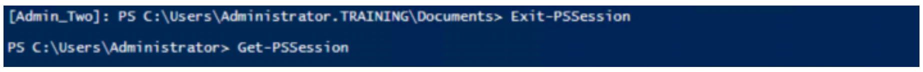
2. Persistent Remoting Code

```
Get-Command -Noun psssession
Get-PSSession
New-PSSession -ComputerName Admin_Two -Name "Admin_Two_Tunnel"
Enter-PSSession -ComputerName Admin_Two

New-Item -ItemType File -Path c:\ -Name "RemoteFile.txt" -Force
Set-Content -Path "c:\RemoteFile.txt" -Value "Ki re vai ki korchis" -Force
```



```
PS C:\Users\Administrator> Enter-PSSession -ComputerName Admin_Two
[Admin_Two]: PS C:\Users\Administrator.TRAINING\Documents> New-Item -ItemType File -Path c:\ -Name "RemoteFile.txt" -Force

    Directory: C:\

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        11/19/2024  12:32 PM              0 RemoteFile.txt


[Admin_Two]: PS C:\Users\Administrator.TRAINING\Documents> Set-Content -Path "c:\RemoteFile.txt" -Value "Ki re vai ki korchis" -Force
[Admin_Two]: PS C:\Users\Administrator.TRAINING\Documents>
```

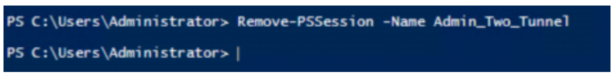
Exit-PSSession



```
[Admin_Two]: PS C:\Users\Administrator.TRAINING\Documents> Exit-PSSession
PS C:\Users\Administrator> Get-PSSession
```

Screen clipping taken: 19-11-2024 12:38

Get-PSSession
Remove-PSSession



```
PS C:\Users\Administrator> Remove-PSSession -Name Admin_Two_Tunnel
PS C:\Users\Administrator> |
```

Screen clipping taken: 19-11-2024 12:39

DSC (Desired state configuration)

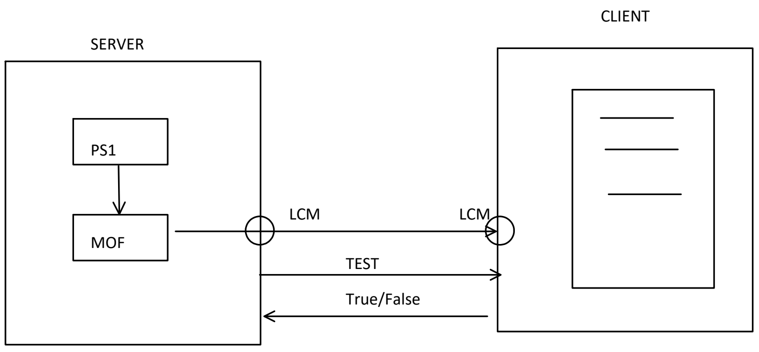
Firewall-> on
WinRM service -> on
BITS service -> off
file: c:\ruban.txt ---> present
IIS web server -------> ON

If you want to add this config in 45 computer manually it will be very time consuming

Mode of DSC:

1. **Push Mode**
   ->server will push updates to client
   ->Advantage: easy to deploy
   ->Disadvantage: if client is off/unreachable, updates will be missed.
2. **Pull Mode**
   ->client will pull updates from server
   ->Advantage: client will never miss any updates
   ->Disadvantage: very complicated

Microsoft object file
Local Config Manager



We create a ps file and it will be converted to MOF it will convert the PS file in
Human readable manner and sent via LCM . Meanwhile the client's LCM got the MOF
And read it line by line and it make the change and the server sent the test msg
And it will be acknowledge by client by using boolean value.

#import the DSC module

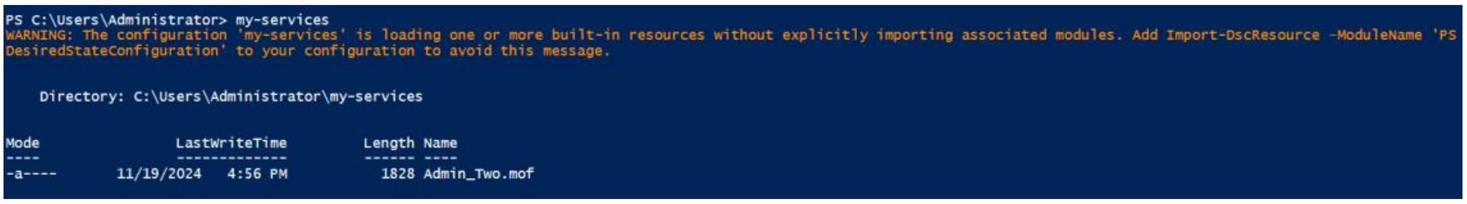
Import-Module -Name PSDesiredStateConfiguration

#DSC push mode config for member machine

```
configuration my-services{
   Node Admin_Two{
      service bits{
         Name = "BITS"
         State = "Running"
      }
   }
}
```

#generate the MOF file

my-services

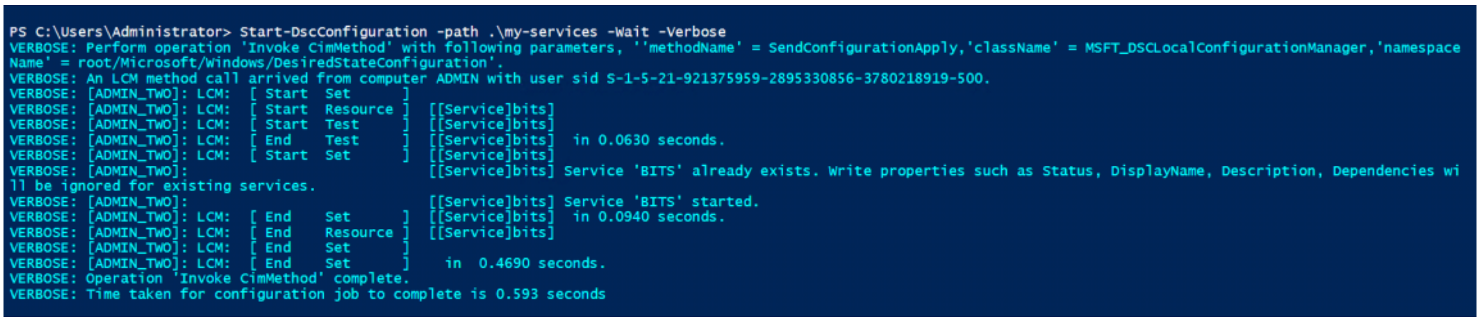


```
PS C:\Users\Administrator> my-services
WARNING: The configuration 'my-services' is loading one or more built-in resources without explicitly importing associated modules. Add Import-DscResource -ModuleName 'PS
DesiredStateConfiguration' to your configuration to avoid this message.


    Directory: C:\Users\Administrator\my-services


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        11/19/2024   4:56 PM           1828 Admin_Two.mof
```

Screen clipping taken: 19-11-2024 16:56

#apply-config

Start-DscConfiguration -path .\my-services -Wait -Verbose

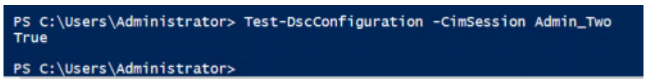


```
PS C:\Users\Administrator> Start-DscConfiguration -path .\my-services -Wait -Verbose
VERBOSE: Perform operation 'Invoke CimMethod' with following parameters, ''methodName' = SendConfigurationApply,'className' = MSFT_DSCLocalConfigurationManager,'namespace
Name' = root/Microsoft/Windows/DesiredStateConfiguration'.
VERBOSE: An LCM method call arrived from computer ADMIN with user sid S-1-5-21-921375959-2895330856-3780218919-500.
VERBOSE: [ADMIN_TWO]: LCM:  [ Start  Set      ]
VERBOSE: [ADMIN_TWO]: LCM:  [ Start  Resource ]  [[Service]bits]
VERBOSE: [ADMIN_TWO]: LCM:  [ Start  Test     ]  [[Service]bits]
VERBOSE: [ADMIN_TWO]: LCM:  [ End    Test     ]  [[Service]bits]  in 0.0630 seconds.
VERBOSE: [ADMIN_TWO]: LCM:  [ Start  Set      ]  [[Service]bits]
VERBOSE: [ADMIN_TWO]:                            [[Service]bits] Service 'BITS' already exists. Write properties such as Status, DisplayName, Description, Dependencies wi
ll be ignored for existing services.
VERBOSE: [ADMIN_TWO]:                            [[Service]bits] Service 'BITS' started.
VERBOSE: [ADMIN_TWO]: LCM:  [ End    Set      ]  [[Service]bits]  in 0.0940 seconds.
VERBOSE: [ADMIN_TWO]: LCM:  [ End    Resource ]  [[Service]bits]
VERBOSE: [ADMIN_TWO]: LCM:  [ End    Set      ]
VERBOSE: [ADMIN_TWO]: LCM:  [ End    Set      ]    in  0.4690 seconds.
VERBOSE: Operation 'Invoke CimMethod' complete.
VERBOSE: Time taken for configuration job to complete is 0.593 seconds
```

Screen clipping taken: 19-11-2024 16:57

#testing the config

Test-DscConfiguration -CimSession Admin_Two



```
PS C:\Users\Administrator> Test-DscConfiguration -CimSession Admin_Two
True

PS C:\Users\Administrator>
```

Screen clipping taken: 19-11-2024 16:58

Workflow

Serial exe ~ top to bottom left to right
Decide how you want to execute the program

Sequential Execution
Parallel Execution

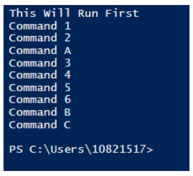
Workflow is a mechanism to execute your script either sequentially and pararaly

Syntax:-

```
Workflow my-wf3{
      Write-host "This Is Workflow" //it is the error
}
Cls
my-wf3
```

```
Workflow my-wf3{
      "This Is Workflow" //it is the correct
}
Cls
my-wf3
```

```
Workflow Test-workflow{
   "This Will Run First"
   parallel {
      "Command 1"
      "Command 2"
   sequence{
      "Command A"
      "Command B"
      "Command C"
   }
      "Command 3"
      "Command 4"
      "Command 5"
      "Command 6"
   }

}

cls
Test-workflow
```

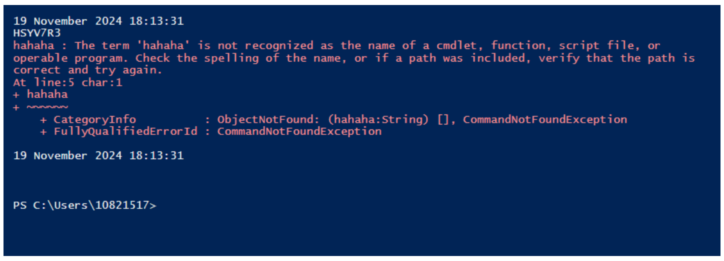


```
This Will Run First
Command 1
Command 2
Command A
Command 3
Command 4
Command 5
Command 6
Command B
Command C

PS C:\Users\10821517>
```

Screen clipping taken: 19-11-2024 17:49

Error Handling

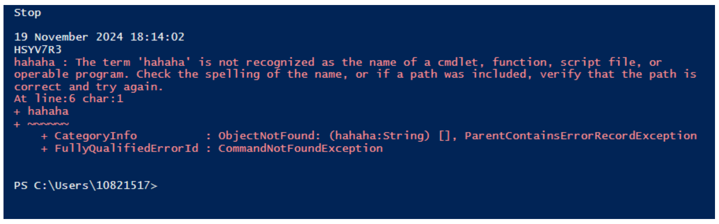
```
cls
Test-workflow

$Error
$Error.Count
$Error | Out-File .\err.txt
notepad.exe .\err.txt

$ErrorActionPreference
cls
Get-Date
hostname
hahaha
Get-Date
```



```
19 November 2024 18:13:31
HSYV7R3
hahaha : The term 'hahaha' is not recognized as the name of a cmdlet, function, script file, or
operable program. Check the spelling of the name, or if a path was included, verify that the path is
correct and try again.
At line:5 char:1
+ hahaha
+ ~~~~~~
    + CategoryInfo          : ObjectNotFound: (hahaha:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

19 November 2024 18:13:31


PS C:\Users\10821517>
```

Screen clipping taken: 19-11-2024 18:13

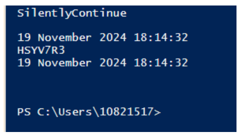
```
$ErrorActionPreference = "stop"
cls
$ErrorActionPreference
Get-Date
hostname
hahaha
Get-Date
```



```
Stop

19 November 2024 18:14:02
HSYV7R3
hahaha : The term 'hahaha' is not recognized as the name of a cmdlet, function, script file, or
operable program. Check the spelling of the name, or if a path was included, verify that the path is
correct and try again.
At line:6 char:1
+ hahaha
+ ~~~~~~
    + CategoryInfo          : ObjectNotFound: (hahaha:String) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : CommandNotFoundException


PS C:\Users\10821517>
```

Screen clipping taken: 19-11-2024 18:14

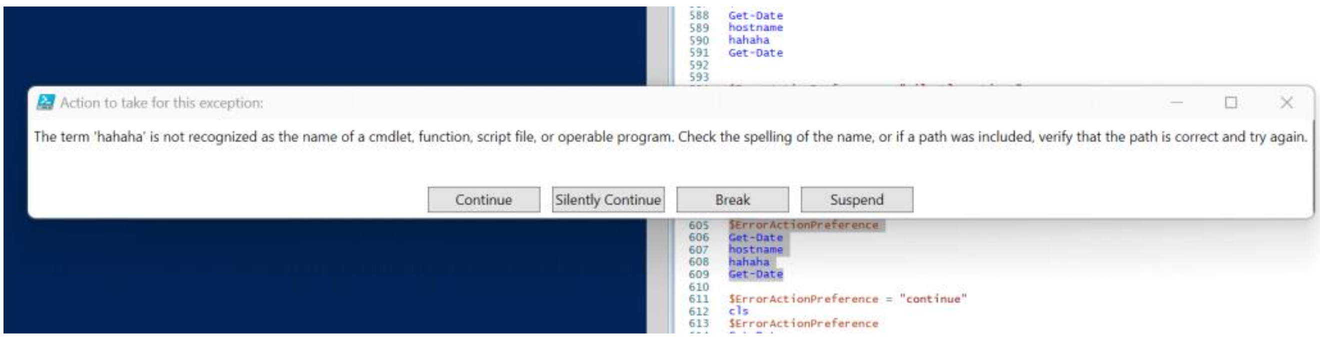
```
$ErrorActionPreference = "silentlycontinue"
cls
$ErrorActionPreference
Get-Date
hostname
hahaha
Get-Date
```



```
SilentlyContinue

19 November 2024 18:14:32
HSYV7R3
19 November 2024 18:14:32


PS C:\Users\10821517>
```

Screen clipping taken: 19-11-2024 18:14

```
$ErrorActionPreference = "inquire"
cls
$ErrorActionPreference
Get-Date
hostname
hahaha
Get-Date
```



```
Get-Date
hostname
hahaha
Get-Date
```

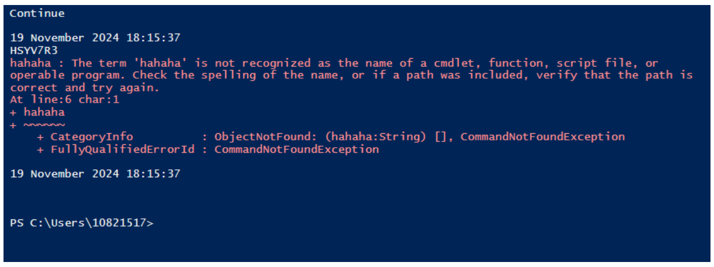
Action to take for this exception:

The term 'hahaha' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.

```
$ErrorActionPreference
Get-Date
hostname
hahaha
Get-Date

$ErrorActionPreference = "continue"
cls
$ErrorActionPreference
```

Screen clipping taken: 19-11-2024 18:15

```
$ErrorActionPreference = "continue"
cls
$ErrorActionPreference
Get-Date
hostname
hahaha
Get-Date
```

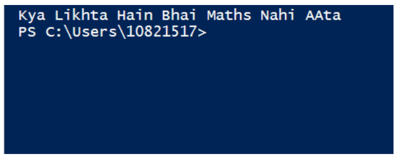


```
Continue

19 November 2024 18:15:37
HSYV7R3
hahaha : The term 'hahaha' is not recognized as the name of a cmdlet, function, script file, or
operable program. Check the spelling of the name, or if a path was included, verify that the path is
correct and try again.
At line:6 char:1
+ hahaha
+ ~~~~~~
    + CategoryInfo          : ObjectNotFound: (hahaha:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

19 November 2024 18:15:37


PS C:\Users\10821517>
```

Screen clipping taken: 19-11-2024 18:15

Try Catch Block

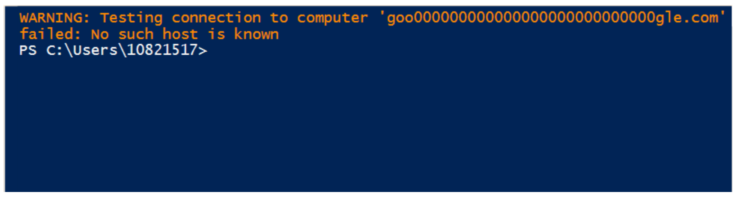
```
cls
$one=1
$zero=0

try{
   $one/$zero
}catch [System.DivideByZeroException]{
   "Kya Likhta Hain Bhai Maths Nahi AAta"
}catch{
   "Paka mat pata nahi kya error hain!!!!"
}
```



```
Kya Likhta Hain Bhai Maths Nahi AAta
PS C:\Users\10821517>
```

Screen clipping taken: 19-11-2024 18:23

```
function get-ping{
   param(
      [string]$site
      )
      try{
         Test-Connection $site -Count 1 -ErrorAction Stop| `
         Select-Object Address, IPv4Address
      }catch{
         Write-Warning $_
      }
}

cls
get-ping -site goo00000000000000000000000000000gle.com
```



```
WARNING: Testing connection to computer 'goo00000000000000000000000000000gle.com'
failed: No such host is known
PS C:\Users\10821517>
```

Screen clipping taken: 19-11-2024 18:35

**Task create a powershell script that asks for a file name (with extension) and returns the file path.
Also ensures that same file will be searched in all available partitions**

```
cls
$file=Read-Host "Enter the file:-"
$a=Get-PSDrive -PSProvider FileSystem | Select-Object name
foreach($i in $a){
   $i = $i.Name + ":" + "\"
   $items=Get-ChildItem $i -Recurse -ea SilentlyContinue

   foreach($j in $items){
      if($j.Name -eq $file){
         $b=$j.Directory
         if($b -eq $i){
            Write-Host $b.Name
```

```
            }
            write-host $i$($b.name)
        }
    }


}
```

----------------------------------x----------------------------------