

Show, Get, Start and Restart Commands

List of commands:

1. **Show-Command:** Show-Command opens a **graphical window** that displays the PowerShell cmdlets.
 - **Show-Command**

The screenshot shows a Windows PowerShell ISE window with several tabs open: Untitled1.ps1*, All_Cmdlets.ps1, Untitled3.ps1, and Day5_assignment_Cmdlets. The main pane displays the following PowerShell command:

```
PS C:\Users\admin> Show-Command
```

To the right of the main pane, there is a 'Commands' window with a search bar and two dropdown menus: 'Modules:' and 'Name:'. The 'Modules:' dropdown is set to 'All'. The 'Name:' dropdown is empty. Below these dropdowns is a large list of PowerShell cmdlet names, starting with 'A' and including 'Add-ADEdgeExtension', 'Add-ADSessionClosingCallback', 'Add-ADSessionFinishingCallback', 'Add-ADSessionOpeningCallback', 'Add-ADSessionStartingCallback', 'Add-AppClientConnectionGroup', 'Add-AppClientPackage', 'Add-AppPublishingServer', 'Add-AppxPackage', 'Add-AppProvisionedPackage', 'Add-AppVolume', 'Add-BCDataCacheExtension', 'Add-BitLockerKeyProtector', 'Add-BitsFile', 'Add-CertificateEnrollmentPolicyServer', 'Add-CMAppv5XDeploymentType', 'Add-CMAppvDeploymentType', 'Add-CMAssetIntelligenceSynchronizationPoint', 'Add-CMBoundaryToGroup', and 'Add-CMCIDetectionMethod'. At the bottom of the 'Commands' window are 'Run', 'Copy', and 'Cancel' buttons.

2. **Get-Service:** The Get-Service cmdlet in PowerShell retrieves the **services installed on a local or remote computer**, including their status, name, and display name.
 - **Get-Service**

The screenshot shows a Windows PowerShell ISE window with several tabs open: Untitled1.ps1*, All_Cmdlets.ps1, Untitled3.ps1, and Day5_assignment_Cmdlets_and_code (1).ps1*. The main pane displays the following PowerShell command:

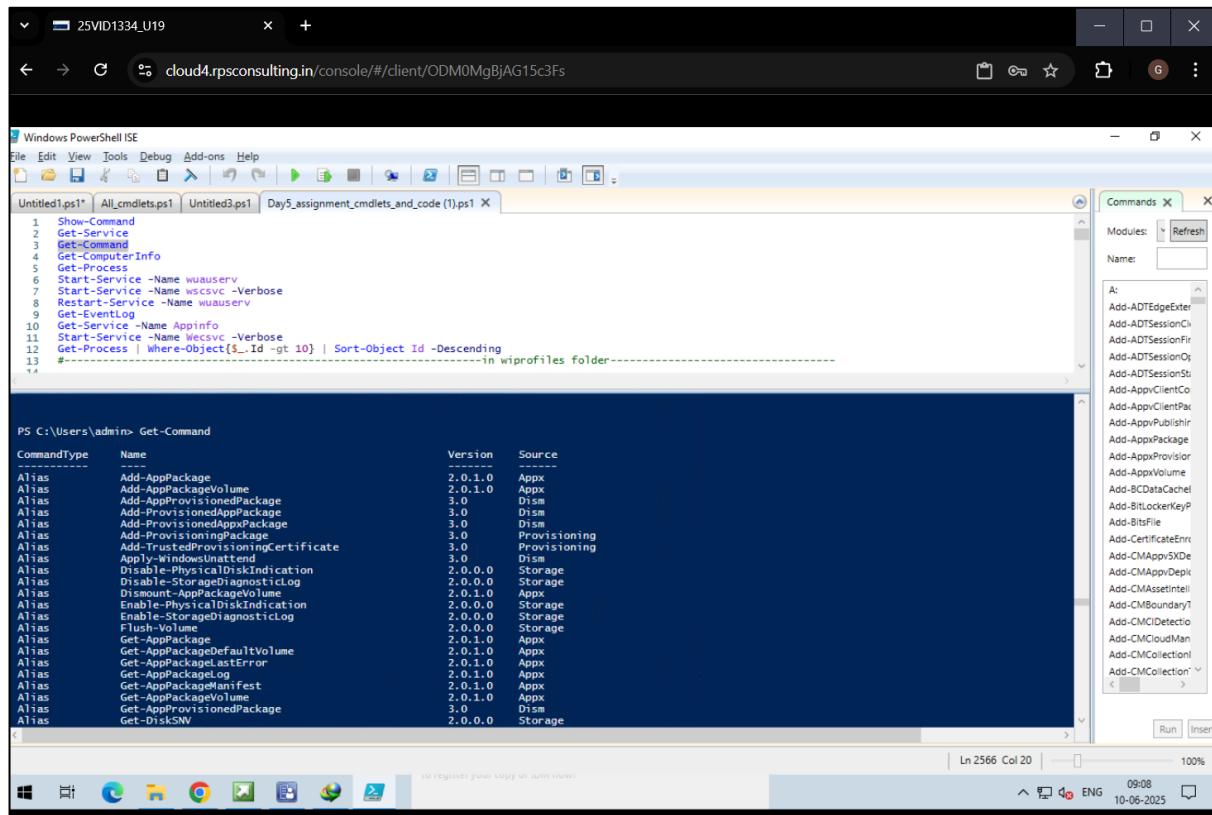
```
PS C:\Users\admin> Get-Service
```

The output shows a table of service details:

Status	Name	DisplayName
Stopped	AarSvc_6c015	Agent Activation Runtime_6c015
Stopped	AIRouter	Application Layer Gateway Service
Running	ALG	Application Layer Gateway
Running	AppIDSvc	App ID Service
Running	AppInfo	Application Information
Running	AppMgmt	Application Management
Stopped	AppReadiness	App Readiness
Stopped	Apprendent	Apprendent-V Client
Running	AppSvc	App Deployment Service (AppSVC)
Stopped	AssignedAccessM...	AssignedAccessManager Service
Running	AutoEndpointBu...	Windows Auto Endpoint Builder
Running	Autovpn	Autovpn
Stopped	autotimesvc	Cellular Time
Stopped	AxInstSV	ActiveX Installer (AxInstSV)
Running	BFE	Base Filtering Engine
Stopped	BeastIIS	BitLocker Drive Encryption Service
Running	BFE	Base Filtering Engine
Stopped	BluetoothHUserSe...	Bluetooth User Support Service_6c015
Running	BrokerInfrstru...	Background Tasks Infrastructure Ser...
Stopped	BTAGService	Bluetooth Audio Gateway Service
Running	BTMvctpSvc	AVCTP Service
Running	BTmon	Bluetooth Connect Service

3. Get-command: The Get-Command cmdlet retrieves **list of all available commands**.

- **Get-command**

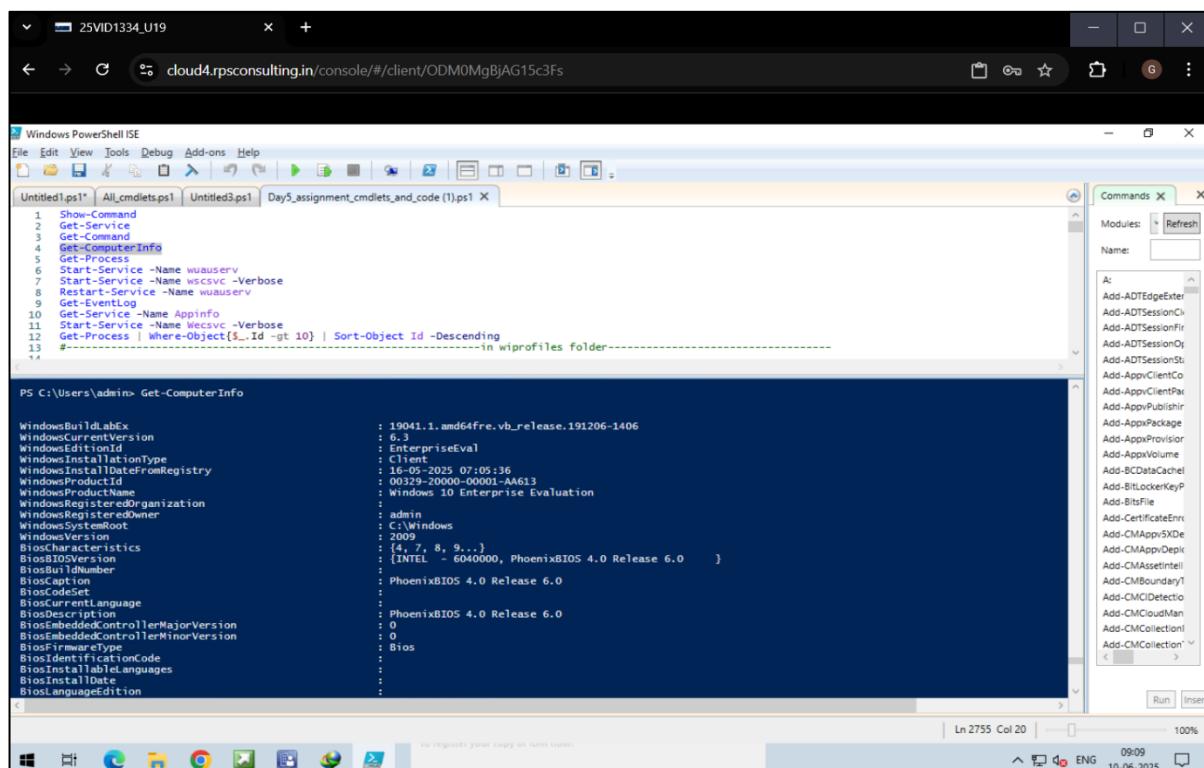


The screenshot shows the Windows PowerShell ISE interface. In the center, the command `Get-Command` is run, displaying a table of available cmdlets. On the right, a 'Commands' pane lists various cmdlets starting with 'Add-'.

CommandType	Name	Version	Source
Alias	Add-AppPackage	2.0.1.0	Appx
Alias	Add-AppPackageVolume	2.0.1.0	Appx
Alias	Add-AppProvisionedPackage	3.0	Dism
Alias	Add-ProvisionedAppPackage	3.0	Dism
Alias	Add-AppVolume	3.0	Dism
Alias	Add-ProvisioningPackage	3.0	Provisioning
Alias	Add-TrustedProvisioningCertificate	3.0	Provisioning
Alias	Apply-WindowsUnattend	3.0	Dism
Alias	Disable-PhysicalDiskIndication	2.0.0.0	Storage
Alias	Disable-StorageDiagnosticLog	2.0.0.0	Storage
Alias	Disable-StorageVolume	2.0.0.0	Appx
Alias	Enable-PhysicalDiskIndication	2.0.0.0	Storage
Alias	Enable-StorageDiagnosticLog	2.0.0.0	Storage
Alias	Flush-Volume	2.0.0.0	Storage
Alias	Get-AppPackage	2.0.1.0	Appx
Alias	Get-AppPackageDefaultVolume	2.0.1.0	Appx
Alias	Get-AppPackageLastError	2.0.1.0	Appx
Alias	Get-AppPackageLog	2.0.1.0	Appx
Alias	Get-AppPackageManifest	2.0.1.0	Appx
Alias	Get-AppPackageVolume	2.0.1.0	Appx
Alias	Get-AppProvisionedPackage	3.0	Dism
Alias	Get-DiskSNV	2.0.0.0	Storage

4. Get-ComputerInfo: The Get-ComputerInfo cmdlet in PowerShell retrieves a **consolidated** object containing system and operating system properties.

- **Get-ComputerInfo**



The screenshot shows the Windows PowerShell ISE interface. The command `Get-ComputerInfo` is run, displaying detailed system information. The output includes fields like WindowsBuildLabEx, WindowsEdition, WindowsEditionId, WindowsInstallType, WindowsInstallDateFromRegistry, WindowsProductId, WindowsName, WindowsRegisteredOrganization, WindowsRegisteredOwner, WindowsSystemRoot, WindowsVersion, BiosArchitecture, BiosBuildNumber, BiosCaption, BiosCodeSet, BiosCommentLanguage, BiosDescription, BiosEmbeddedControllerMajorVersion, BiosEmbeddedControllerMinorVersion, BiosFirmwareType, BiosIdentificationCode, BiosInstallableLanguages, BiosInstallDate, and BiosLanguageEdition.

```
PS C:\Users\admin> Get-ComputerInfo

WindowsBuildLabEx : 19041.1.amd64fre.vb_release.191206-1406
WindowsEdition : Enterprise
WindowsEditionId : EnterpriseEval
WindowsInstallType : Client
WindowsInstallDateFromRegistry : 16-05-2025 07:05:36
WindowsProductId : 00329-20000-00001-AA613
WindowsName : Windows 10 Enterprise Evaluation
WindowsRegisteredOrganization :
WindowsRegisteredOwner : admin
WindowsSystemRoot : C:\Windows
WindowsVersion : 2009
BiosArchitecture : {4,7, 8, 9,...}
BiosBuildNumber : [INTEL - 6040000, PhoenixBIOS 4.0 Release 6.0]
BiosCaption : PhoenixBIOS 4.0 Release 6.0
BiosCodeSet :
BiosCommentLanguage :
BiosDescription : PhoenixBIOS 4.0 Release 6.0
BiosEmbeddedControllerMajorVersion : 0
BiosEmbeddedControllerMinorVersion : 0
BiosFirmwareType : Bios
BiosIdentificationCode :
BiosInstallableLanguages :
BiosInstallDate :
BiosLanguageEdition :
```

5. **Start-Service:** Starts a specific service with the name parameter.

- **Start-Service -Name AjRouter**

The screenshot shows a Windows PowerShell ISE window with the title bar "Administrator: Windows PowerShell ISE". The script file "Untitled1.ps1" contains the following code:

```
1 Show-Command
2 Get-Service
3 Get-Command
4 Get-ComputerInfo
5 Get-Process
6 Start-Service -Name AjRouter
7 Stop-Service -Name wscsvc -Verbose
8 Restart-Service -Name wuauserv
9 Get-EventLog
10 Get-Service -Name Appinfo
11 Start-Service -Name Wecsvc -Verbose
12 Get-Process | Where-Object{$_._Id -gt 10} | Sort-Object Id -Descending
13 #-----in wiprofiles folder
```

The command `Start-Service -Name AjRouter` is executed, followed by a `Get-Service` command to list all services. The output shows the AjRouter service is running. The status, name, and display name for each service are listed.

Status	Name	DisplayName
Stopped	AarSvc_6c015	Agent Activation Runtime_6c015
Running	AJRouter	AllJoyn Router Service
Stopped	ALG	Application Layer Gateway Service
Running	AppIDSvc	Application Identity
Running	AppInfo	Application Information
Running	AppReadiness	Application Management
Stopped	AppClient	Microsoft App-V Client
Running	AppXSvc	AppX Deployment Service (AppXVC)
Stopped	AssignedAccessM... AssignedAccessPointBu...	AssignedAccessManager Service AssignedAccessPointBuilder
Running	audiosrv	Windows Audio
Stopped	autotimesvc	Cellular Time
Stopped	AxInstSV	ActiveX Installer (AxInstSV)
Running	BalloonService	BalloonService
Stopped	BcastDVRUserSer...	Background and Broadcast User Service...
Stopped	BFE	BitLocker Drive Encryption Service
Stopped	BITS	Background Intelligent Transfer Ser...
Stopped	BluetoothUserSe...	Bluetooth User Support Service_6c015
Running	BrokerInfrastru...	Background Tasks Infrastructure Ser...
Stopped	BTAGService	Bluetooth Audio Gateway Service
Running	BTMUserSvc	AVG to remove

6. **Stop-Service:** Stops the service with the name parameter.

- **Stop-Service -Name AjRouter**

The screenshot shows a Windows PowerShell ISE window with the title bar "Administrator: Windows PowerShell ISE". The script file "Untitled1.ps1" contains the following code:

```
1 Show-Command
2 Get-Service
3 Get-Command
4 Get-ComputerInfo
5 Get-Process
6 Start-Service -Name AjRouter
7 Stop-Service -Name AjRouter
8 Start-Service -Name wscsvc -Verbose
9 Restart-Service -Name wuauserv
10 Get-EventLog
11 Get-Service -Name Appinfo
12 Start-Service -Name Wecsvc -Verbose
13 Get-Process | Where-Object{$_._Id -gt 10} | Sort-Object Id -Descending
14 #-----in winnfanter folder
```

The command `Stop-Service -Name AjRouter` is executed, followed by a `Get-Service` command to list all services. The output shows the AjRouter service is stopped. The status, name, and display name for each service are listed.

Status	Name	DisplayName
Stopped	AarSvc_6c015	Agent Activation Runtime_6c015
Stopped	AJRouter	AllJoyn Router Service
Stopped	ALG	Application Layer Gateway Service
Running	AppIDSvc	Application Identity
Running	AppInfo	Application Information
Running	AppReadiness	Application Management
Stopped	AppClient	Microsoft App-V Client
Running	AppXSvc	AppX Deployment Service (AppXVC)
Stopped	AssignedAccessM... AssignedAccessPointBu...	AssignedAccessManager Service AssignedAccessPointBuilder
Running	audiosrv	Windows Audio
Stopped	autotimesvc	Cellular Time
Stopped	AxInstSV	ActiveX Installer (AxInstSV)
Running	BalloonService	BalloonService
Stopped	BcastDVRUserSer...	Background and Broadcast User Service...
Stopped	BFE	BitLocker Drive Encryption Service
Stopped	BITS	Background Intelligent Transfer Ser...
Stopped	BluetoothUserSe...	Bluetooth User Support Service_6c015
Running	BrokerInfrastru...	Background Tasks Infrastructure Ser...
Stopped	BTAGService	Bluetooth Audio Gateway Service

7. **Restart-Service:** Restarts the service with the name parameter

- **Restart-Service -Name AjRouter**

The screenshot shows a Windows PowerShell ISE window with the following content:

```
Administrator: Windows PowerShell ISE
Untitled1.ps1 Day5_assignment_cmdlets_and_code (1).ps1* X

1 Show-Command
2 Get-Service
3 Get-ComputerInfo
4 Get-Process
5 Start-Service -Name AjRouter
6 Stop-Service -Name AjRouter
7 Start-Service -Name wscsvc -Verbose
8 Restart-Service -Name AjRouter
9 Get-EventLog
10 Get-Service -Name Appinfo
11 Start-Service -Name Wecsvc -Verbose
12 Get-Process | Where-Object {$_.Id -gt 10} | Sort-Object Id -Descending
13 ...
```

PS C:\Windows\system32> Restart-Service -Name AjRouter

PS C:\Windows\system32> Get-Service

Status	Name	DisplayName
Stopped	AarSvc_6c015	Agent Activation Runtime_6c015
Running	AjRouter	AllJoyn Router Service
Stopped	ALG	Application Layer Gateway Service
Running	AppIDSvc	Application ID Service
Running	AppInfo	Application Information
Running	AppMgmt	Application Management
Stopped	AppReadiness	App Readiness
Stopped	AppVClient	Microsoft App-V Client
Running	AppDeployService	App Deployment Service (AppXSV)
Stopped	AssignedAccessM...	AssignedAccessManager Service
Running	AudioEndpointBu...	Windows Audio Endpoint Builder
Running	Audiosrv	Windows Audio
Stopped	autotimesvc	Cellular Time
Stopped	AxInstS...	ActiveX Installer (AxInstS)
Running	BalService	Balancer Service
Stopped	BcastDVRUserSer...	GameDV and Broadcast User Service...
Stopped	BDESVC	BitLocker Drive Encryption Service
Running	BFE	Base Filtering Engine
Stopped	BITS	Background Intelligent Transfer Ser...
Stopped	BluetoothUserSe...	Bluetooth User Support Service_6c015
Running	BrokerInfrast...	Background Tasks Infrastructure Ser...
Stopped	BTAGService	Bluetooth Audio Gateway Service

Completed

Ln 9 Col 1 | 100% 09:16 10-06-2025

8. **Get-EventLog:** Get-EventLog retrieves event log entries from Windows Event Viewer useful for checking system, application, and security logs.(in the LogName give system)

- **Get-EventLog**

The screenshot shows a Windows PowerShell ISE window with the following content:

```
Administrator: Windows PowerShell ISE
Untitled1.ps1 Day5_assignment_cmdlets_and_code (1).ps1* X

1 Show-Command
2 Get-Service
3 Get-ComputerInfo
4 Get-Process
5 Start-Service -Name AjRouter
6 Stop-Service -Name AjRouter
7 Start-Service -Name wscsvc -Verbose
8 Restart-Service -Name AjRouter
9 Get-EventLog
10 Get-Service -Name Appinfo
11 ...
```

PS C:\Windows\system32> Get-EventLog

cmdlet Get-EventLog at command pipeline position 1
Supply values for the following parameters:
LogName: system

Index	Time	EntryType	Source	InstanceID	Message
7009	Jun 10 09:28	Information	Service Control M...	1073748864	The start type of the Background Intelligent Transfer Service service was changed from auto... .
7008	Jun 10 09:26	Information	Service Control M...	1073748864	The start type of the Background Intelligent Transfer Service service was changed from de... .
7007	Jun 10 06:38	Information	Microsoft-Windows...	1073748864	37 The time provider NtpClient is currently receiving valid time data from time.windows.com... .
7006	Jun 10 05:01	Information	Service Control M...	3221229797	The start type of the Windows Update service was changed from demand start to disabled.
7005	Jun 10 04:19	Information	Service Control M...	3221229797	The start type of the Windows Update service was changed from demand start to disabled.
7003	Jun 10 00:19	Error	Server	3221229793	The server could not bind to the transport {Device\NetB.Tcpip_{[0DA33E65-86F7-4EC9-AA69-1...}}
7002	Jun 09 22:18	Error	NetBT	3221229793	The name "DESKTOP-066LAAS:20" could not be registered on the interface with IP address 17... .
7001	Jun 09 22:18	Error	Server	3221229797	The server could not bind to the transport {Device\NetB.Tcpip_{[0DA33E65-86F7-4EC9-AA69-1...}}
7000	Jun 09 22:17	Error	Tcpip	3221229671	The system detected an address conflict for IP address 172.20.0.30 with the system... .
6999	Jun 09 21:52	Error	NetBT	3221229791	A duplicate name has been detected on the network. The IP address of... .
6997	Jun 09 21:32	Information	Microsoft-Windows...	3221229791	37 The time provider NtpClient is currently receiving valid time data from time.windows.com... .
6996	Jun 09 20:01	Information	Service Control M...	1073748864	The start type of the Update Orchestrator Service service was changed from auto start to... .
6995	Jun 09 20:01	Information	Service Control M...	1073748864	The start type of the Windows Update service was changed from demand start to disabled.
6994	Jun 09 18:17	Information	Service Control M...	1073748864	37 The start type of the Background Intelligent Transfer Service service was changed from au... .
6993	Jun 09 18:17	Information	Service Control M...	1073748864	The start type of the Background Intelligent Transfer Service service was changed from de... .
6992	Jun 09 18:17	Error	Server	3221229797	The server could not bind to the transport {Device\NetB.Tcpip_{[0DA33E65-86F7-4EC9-AA69-1...}}
6991	Jun 09 18:17	Error	Server	3221229797	The server could not bind to the transport {Device\NetB.Tcpip_{[0DA33E65-86F7-4EC9-AA69-1...}}
6990	Jun 09 18:17	Error	Tcpip	3221229671	The system detected an address conflict for IP address 172.20.0.30 with the system... .
6989	Jun 09 18:17	Error	NetBT	3221229793	37 The time provider NtpClient is currently receiving valid time data from time.windows.com... .
6988	Jun 09 18:17	Error	Server	3221229793	The name "DESKTOP-066LAAS:20" could not be registered on the interface with IP address 17... .
6987	Jun 09 18:17	Error	DCOM	10028	The description for Event ID '10028' in Source 'DCOM' cannot be found. The local computer... .
6986	Jun 09 18:17	Error	DCOM	10028	The description for Event ID '10028' in Source 'DCOM' cannot be found. The local computer... .

Ln 2095 Col 25 | 100% 10:23 10-06-2025

9. Get-Process: Retrieves the currently running process and **where-object** helps to filter the process and **sort-object** helps to sort the process in descending.

- **Get-Process | Where-Object{\$_.Id -gt 10} | Sort-Object Id -Descending**

```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1 Day5_assignment_cmdlets_and_code (1).ps1*
4 Get-ComputerInfo
5 Get-Process
6 Start-Service -Name AJRouter
7 Start-Service -Name AJRouter
8 Start-Service -Name wscsvc -Verbose
9 Restart-Service -Name AJRouter
10 Get-EventLog
11 Get-Service -Name Appinfo
12 Get-Process | Where-Object{$_.Id -gt 10} | Sort-Object Id -Descending
13 #-----in wiprofiles folder-----
14

PS C:\Windows\system32> Get-Process | Where-Object{$_.Id -gt 10} | Sort-Object Id -Descending
Handles NPM(K) PM(K) WS(K) CPU(s) Id SI ProcessName
---- -- -- -- -- --
114 7 1284 6064 0.31 11276 0 svchost
115 35 2932 2614 1.05 10732 0 powershell_settings
200 10 2444 9524 7.16 10732 0 svchost
958 68 154416 207188 72.92 10724 1 powershell_ise
186 8 2068 7524 14.42 10688 0 svchost
543 32 12812 39352 42.70 10592 1 mshta
544 30 8252 30924 57.55 10140 1 IDMan
575 12 2532 1176 4.16 10052 0 taskhostw
402 22 7316 32960 162.03 9864 1 taskhostw
1864 66 105184 213972 4,820.59 9840 1 chrome
1057 30 16056 46924 8.61 9808 1 ShellExperienceHost
247 13 5036 11672 4.94 9472 0 svchost
118 6 4136 7344 0.89 9230 0 svchost
33 65 8556 48248 73.05 9234 0 explorer
241 19 2628 8364 748.75 9144 0 svchost
267 18 4324 16428 775.95 9000 1 vmtoolsd
449 18 6584 19740 72.22 8928 0 SecurityHealthService
169 10 2052 10044 3.41 8860 1 SecurityHealthSystem
386 12 2760 12236 2.38 8860 0 svchost
553 23 9392 38360 24.63 8668 0 svchost
749 59 69276 106236 11,859.28 8664 1 Microsoft.ConfigurationManagement
309 15 5004 18744 6.42 8628 0 svchost
357 21 12272 31396 9.55 8564 1 ApplicationFrameHost
255 15 12916 19824 9.09 8516 1 chrome
195 15 6548 10160 1.73 8412 0 svchost

```

10. Out-File: The Out-File cmdlet in PowerShell is used to send output to a file.

- **Get-Process | Out-File "E:\wiprofiles\processfile1"**

```

Administrator: Windows PowerShell ISE
File Edit Format View Help
Untitled1.ps1 Day5_assignment_cmdlets_and_code (1).ps1*
13 #
14
15 Get-Process | Out-File E:\wiprofiles\processfile1
16 Get-NetIPConfiguration | Out-File E:\wiprofiles\configfile
17 Get-Service | Out-File E:\wiprofiles\configfile -Append
18 Get-PSDrive -PSProvider File | Out-File E:\wiprofiles\datafile -Append
19 Get-Command | Out-File E:\wiprofiles\datafile.txt -Append
20 Get-Command | Out-File E:\demo\commandsfile.txt
21 Get-PSDrive -PSProvider FileSystem | Out-File E:\demo\datafile.txt
22 Get-Process | Out-File E:\wiprofiles\processfile1
23 Get-Process | Out-File E:\wiprofiles\processfile1 -Append
24 Get-Process | Out-File E:\wiprofiles\processfile1 -Append
25 Get-Process | Out-File E:\wiprofiles\processfile1 -Append
26 Get-Process | Out-File E:\wiprofiles\processfile1 -Append
27 Get-Process | Out-File E:\wiprofiles\processfile1 -Append
28 Get-Process | Out-File E:\wiprofiles\processfile1 -Append
29 Get-Process | Out-File E:\wiprofiles\processfile1 -Append
30

PS C:\Windows\system32> Get-Process | Out-File E:\wiprofiles\processfile1
PS C:\Windows\system32>

Completed

```

- **Get-NetIPConfiguration | Out-File E:\wiprofiles\configfiles**

```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1 Day5_assignment_cmdlets_and_code (1).ps1* X
13 #-----
14
15
16 Get-Process | Out-File E:\wiprofiles\processfile1
17 Get-NetIPConfiguration | Out-File E:\wiprofiles\configfile1
18
19 Get-Service | Out-File E:\wiprofiles\configfile -Append
20
21 Get-PSDrive -PSProvider FileSystem | Out-File E:\wiprofiles\da
22 Get-Command | Out-File E:\wiprofiles\datafile.txt -Append
23
24 #-----
25 Get-Command |Out-File E:\demo\commandsfile.txt
26
27 Get-PSDrive -PSProvider FileSystem | Out-File E:\demo\datafil
28
29 Get-PSDrive -PSProvider FileSystem | Out-File E:\demo\datafil
30

6025 Jun 08 19:51 Error 7SPnP 3221229793
6012 Jun 08 19:51 Error NetBT 3221229793
6010 Jun 08 19:51 Error Server 3221229793
6009 Jun 08 19:51 Error Server 3221229793
6008 Jun 08 19:51 Error NetBT 3221229793
6007 Jun 08 19:51 Error Server 3221229793
6006 Jun 08 19:50 Error Server 3221229793
6005 Jun 08 19:50 Error Server 3221229793
6004 Jun 08 19:50 Error NetBT 3221229793
6003 Jun 08 19:49 Error Server 3221229793
6002 Jun 08 19:49 Error NetBT 3221229793
6001 Jun 08 19:49 Error Server 3221229793
6000 Jun 08 19:49 Error Server 3221229793
5999 Jun 08 19:49 Error Server 3221229793
5998 Jun 08 19:49 Error Server 3221229793
5997 Jun 08 19:49 Error NetBT 3221229793
5996 Jun 08 19:49 Error Server 3221229793
5995 Jun 08 19:49 Error Server 3221229793

InterfaceAlias      : VMware Network Adapter VMnet1
InterfaceIndex     : 12
InterfaceDescription: VMware Virtual Ethernet Adapter for VMnet1
IPv4Address        : 192.168.245.1
IPv6DefaultGateway :
IPv4DefaultGateway :
DNSServer          : fec0:0:0:ffff::1
                      fec0:0:0:ffff::2
                      fec0:0:0:ffff::3

InterfaceAlias      : Ethernet0
InterfaceIndex     : 3
InterfaceDescription: Intel(R) 82574L Gigabit Network Connection
NetProfile.Name    : Network 6
IPv4Address        : 172.20.0.19
IPv6DefaultGateway :
IPv4DefaultGateway : 172.20.0.1
DNSServer          : 172.20.0.1

InterfaceAlias      : VMware Network Adapter VMnet8
InterfaceIndex     : 5
InterfaceDescription: VMware Virtual Ethernet Adapter for VMnet8
NetProfile.Name    : Network 6
IPv4Address        : 192.168.162.1
IPv6DefaultGateway :
IPv4DefaultGateway :
DNSServer          : fec0:0:0:ffff::1
                      fec0:0:0:ffff::2
                      fec0:0:0:ffff::3

PS C:\Windows\system32> Get-Process | Out-File E:\wiprofiles\processfile1
PS C:\Windows\system32> Get-NetIPConfiguration | Out-File E:\wiprofiles\configfile1
PS C:\Windows\system32>

```

Completed

Ln 1, Col 1 100% Windows (CRLF) UTF-16 LE
Ln 2101 Col 23
11:25 10-06-2025

- **Get-Service |Out-File E:\wiprofiles\configfile1 -Append**

```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1 Day5_assignment_cmdlets_and_code (1).ps1* X
13 #-----
14
15
16 Get-Process | Out-File E:\wiprofiles\processfile1
17 Get-NetIPConfiguration | Out-File E:\wiprofiles\configfile1
18
19 Get-Service | Out-File E:\wiprofiles\configfile -Append
20
21 Get-PSDrive -PSProvider FileSystem | Out-File E:\wiprofiles\da
22 Get-Command | Out-File E:\wiprofiles\datafile.txt -Append
23
24 #-----
25 Get-Command |Out-File E:\demo\commandsfile.txt
26
27 Get-PSDrive -PSProvider FileSystem | Out-File E:\demo\datafil
28
29 Get-PSDrive -PSProvider FileSystem | Out-File E:\demo\datafil
30

6010 Jun 08 19:51 Error Server 3221229793
6009 Jun 08 19:51 Error Server 3221229793
6008 Jun 08 19:51 Error NetBT 3221229793
6007 Jun 08 19:51 Error Server 3221229793
6006 Jun 08 19:50 Error Server 3221229793
6005 Jun 08 19:50 Error Server 3221229793
6004 Jun 08 19:50 Error NetBT 3221229793
6003 Jun 08 19:49 Error Server 3221229793
6002 Jun 08 19:49 Error NetBT 3221229793
6001 Jun 08 19:49 Error Server 3221229793
6000 Jun 08 19:49 Error Server 3221229793
5999 Jun 08 19:49 Error Server 3221229793
5998 Jun 08 19:49 Error Server 3221229793
5997 Jun 08 19:49 Error NetBT 3221229793
5996 Jun 08 19:49 Error Server 3221229793
5995 Jun 08 19:49 Error Server 3221229793

Status   Name           DisplayName
----- 
Stopped  AarSvc_6c015  Agent Activation Runtime_6c015
Stopped  ALRrouter    AllJoyn Router Service
Stopped  ALG           Application Layer Gateway Service
Running   AppIDSvc    Application Identity
Running   AppInfo      Application Information
Running   AppMgmt     Application Management
Stopped  AppReadiness  App Readiness

PS C:\Windows\system32> Get-Process | Out-File E:\wiprofiles\processfile1
PS C:\Windows\system32> Get-NetIPConfiguration | Out-File E:\wiprofiles\configfile1
PS C:\Windows\system32> Get-Service | Out-File E:\wiprofiles\configfile1
PS C:\Windows\system32>

```

Completed

Ln 1, Col 1 100% Windows (CRLF) UTF-16 LE
Ln 2101 Col 23
11:26 10-06-2025

- **Get-PSDrive -PSProvider FileSystem | OutFile E:\wiprofiles\datafile1.txt**

The screenshot shows a Windows PowerShell ISE window with a script named `Untitled1.ps1`. The script contains the following commands:

```

8 Start-Service -Name wscsvc -Verbose
9 Restart-Service -Name A2Router
10 Get-EventLog
11 Get-Service -Name Appinfo
12 Get-Process | Where-Object{$_.Id -gt 10} | Sort-Object Id -Descending
13 #-----in wiprof
14
15
16 Get-Process | Out-File E:\wiprofiles\processfile1
17
18 Get-NetIPConfiguration | Out-File E:\wiprofiles\configfile1
19
20 Get-Service | Out-File E:\wiprofiles\configfile1 -Append
21
22 Get-PSDrive -PSProvider FileSystem | Out-File E:\wiprofiles\datafile1.txt
23
24 Get-Command | Out-File E:\wiprofiles\datafile.txt -Append
25

```

After running the script, the terminal shows the command `Get-PSDrive -PSProvider FileSystem | Out-File E:\wiprofiles\datafile1.txt` being executed. To the right, a Notepad window titled "datafile1 - Notepad" displays the contents of the file, which includes a table of disk drives and their properties:

Name	Used (GB)	Free (GB)	Provider	Root
C	111.39	87.51	FileSystem	C:\
D			FileSystem	D:\
E	32.87	167.11	FileSystem	E:\

- **Get-ChildItem | OutFile E:\wiprofiles\datafile1.txt -Append**

The screenshot shows a Windows PowerShell ISE window with a script named `Untitled1.ps1`. The script contains the following commands:

```

29 Get-PSDrive -PSProvider FileSystem | Out-File E:\demo\datafile.txt
30 Get-NetIPConfiguration | Out-File E:\demo\datafile.txt -Append
31
32 Get-ChildItem | Out-File E:\wiprofiles\datafile1.txt -Append
33
34 Get-Service | Out-File E:\demo\opencommands.txt -Append
35
36 #-----formatting-----
37 #Format-Table
38 Get-Process | Format-Table -Property Name , CPU
39
40 Get-Process | Format-Table -Property Name , CPU , StartTime
41
42 #Format-List
43 Get-Service | Format-List -Property Name , Status , DisplayName
44
45 #format-Wide
46

```

After running the script, the terminal shows the command `Get-ChildItem | Out-File E:\wiprofiles\datafile1.txt -Append` being executed. To the right, a Notepad window titled "datafile1 - Notepad" displays the contents of the file, which includes a table of disk drives and their properties, followed by a list of files in the `C:\Windows\system32` directory:

Name	Used (GB)	Free (GB)	Provider	Root
C	111.39	87.51	FileSystem	C:\
D			FileSystem	D:\
E	32.87	167.11	FileSystem	E:\

Below the table, the terminal shows the output of the `Get-ChildItem` command:

```

Directory: C:\Windows\system32

Mode LastWriteTime Length Name
---- --:-- --:-- ----
d---- 07-12-2019 15:19 0409
d---- 05-03-2025 17:37 AdvancedInstallers
d---- 07-12-2019 14:44 am-et
d---- 02-06-2025 15:09 AppLocker
d---- 05-03-2025 17:37 appraiser
d---- 05-03-2025 17:37 AppV
d---- 05-03-2025 17:37 ar-SA
d---- 05-03-2025 17:37 bg-BG
d---- 05-03-2025 18:23 Boot
d---- 07-12-2019 14:44 Bthprops
d---- 05-03-2025 17:39 CatRoot
d---- 24-05-2025 13:53 catroot2
d---- 05-03-2025 17:37 CodeIntegrity
d---- 05-03-2025 17:37 Com
d---- 05-03-2025 17:37 compatrel
d---- 30-05-2025 16:15 config
d----s- 07-12-2019 15:01 Configuration
d----s- 07-12-2019 14:44 ContainerSettingsProviders
d---- 05-03-2025 17:37 cs-CZ

```

11. Format-Table: The Format-Table cmdlet in PowerShell formats the output of a command as a table with the selected properties of the object in each column.

- `Get-Process | Format-table -Property Name,CPU,StartTime`

The screenshot shows a Windows PowerShell ISE window with the title bar "25VID1334_U19" and the address bar "cloud4.rpsconsulting.in/console/#/client/ODM0MgBjAG15c3Fs". The menu bar includes File, Edit, View, Tools, Debug, Add-ons, Help. The toolbar has icons for New, Open, Save, Run, Stop, and others. A tab bar at the bottom shows "Untitled1.ps1" and "Day5_assignment_cmdlets_and_code (1).ps1*".

The code in the editor is as follows:

```
38 #-----
39 #Format-Table
40 Get-Process | Format-Table -Property Name , CPU
41
42 Get-Process | Format-Table -Property Name , CPU , StartTime
43
44 #Format-List
45 Get-Service | Format-List -Property Name , Status , DisplayName
46
47 #format-Wide
48 Get-ChildItem | Format-Wide -Column 3
49
50 #-----
```

The code is annotated with comments: "formatting--" above the first two Get-Process lines, "-get content" below the final Get-ChildItem line.

The command PS C:\Windows\system32> Get-Process | Format-Table -Property Name , CPU , StartTime is run in the console, and its output is displayed as a table:

Name	CPU	StartTime
AggregatorHost	0. 9375	24-05-2025 13:43:05
ApplicationFrameHost	9.546875	24-05-2025 13:43:55
bInsvr	1.625	24-05-2025 13:43:00
chrome	168.484375	28-05-2025 16:25:23
chrome	14. 640625	29-05-2025 11:12:35
chrome	65.796875	29-05-2025 11:06:47
chrome	2. 140625	09-06-2025 14:33:46
chrome	0.9375	28-05-2025 11:44:14
chrome	118. 390625	29-05-2025 11:06:48
chrome	400. 09375	28-05-2025 16:44:12
chrome	737. 234375	28-05-2025 16:25:15
chrome	180. 53125	09-06-2025 14:33:39
chrome	909. 03125	28-05-2025 16:25:15
chrome	24. 040625	28-05-2025 13:43:05
chrome	3. 1875	28-05-2025 16:37:53
chrome	4835. 40625	28-05-2025 16:25:14
conhost	5. 609375	24-05-2025 13:43:01
csrss	29.796875	24-05-2025 13:42:58
csrss	211. 03125	24-05-2025 13:42:58
ctfmon	256.453125	24-05-2025 13:43:06
dashHost	2656.1875	24-05-2025 13:43:35
dllHost	5.375	24-05-2025 13:43:01
dllHost	2. 46875	24-05-2025 13:43:02
dllHost	5. 46875	26-05-2025 06:01:10

12. Format-List: The Format-List cmdlet in PowerShell formats the output of a command as a list of properties, displaying each property on a separate line.

- `Get-Service | Format-List -Property Name,Status,DisplayName`

The screenshot shows a Windows PowerShell ISE window with the following details:

- Title Bar:** 25VID1334_U19
- Address Bar:** cloud4.rpsconsulting.in/console/#/client/ODM0MgBjAG15c3Fs
- Toolbar:** Standard file operations like Save, Open, Print, etc.
- Menu Bar:** File, Edit, View, Tools, Debug, Add-ons, Help
- Code Editor:** Untitled1.ps1 (Day5_assignment_cmdlets_and_code.ps1)
- Script Content:**

```
38 #-----formatting-----
39 #Format-Table
40 Get-Process | Format-Table -Property Name , CPU
41
42 Get-Process | Format-Table -Property Name , CPU , StartTime
43
44 #Format-List
45 Get-Service | Format-List -Property Name , Status , DisplayName
46
47 #format-Wide
48 Get-ChildItem | Format-Wide -Column 3
49
50 #-----get content-----
```
- Output Window:**

```
PS C:\Windows\system32> #Format-List
Get-Service | Format-List -Property Name , Status , DisplayName

Name      : AarSvc_6c015
Status    : Stopped
DisplayName: Agent Activation Runtime_6c015

Name      : AJRouter
Status    : Stopped
DisplayName: AllJoyn Router Service

Name      : ALG
Status    : Stopped
DisplayName: Application Layer Gateway Service

Name      : AppIDSvc
Status    : Running
DisplayName: Application Identity

Name      : AppInfo
Status    : Running
DisplayName: Application Information

Name      : AppMgmt
Status    : Running
DisplayName: Application Management
```

13. Format-Wide: The Format-Wide cmdlet in PowerShell formats objects as a wide table that displays only one property of each object.

- **Get-ChildItem | Format-Wide -Column 3**

```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1 Day5_assignment_cmdlets_and_code(1).ps1* X
38 #_--format-table-----formatting-----
39 #Format-Table
40 Get-Process | Format-Table -Property Name , CPU
41
42 Get-Process | Format-Table -Property Name , CPU , StartTime
43
44 #Format-List
45 Get-Service | Format-List -Property Name , Status , DisplayName
46
47 #Format-Wide
48 Get-ChildItem | Format-Wide -Column 3
49
50 #-----get content-----
PS C:\Windows\system32> #Format-Wide
Get-ChildItem | Format-Wide -Column 3

Directory: C:\Windows\system32

0409 AdvancedInstallers am-et
AppLocker appraiser AppV
ar-SA bg-BG Boot
Bthprops CatRoot catroot2
CodeIntegrity Com [compatrel]
config da-DK ContainerSettingsProviders
cs-CZ DiagSvcs DDFs
de-DE drivers Dism
downlevel DRVSTORE DriverState
DriverStore en dsc
e1-GR es-ES en-GB
en-CA F12 es-ESX
et-EE fr-CA ff-Adm-SN
F1-FI GroupPolicyUsers fr-FR
Fxstmp hr-HR he-IL
ias hu-HU Hydrogen
inetresv icxsm1 IME
InputMethod ja-JP Ipmi
l1-LT ja-Jpanor Keywords
lg-kr

```

14. Get-Content: Retrieves the information from the file provided in the -Path parameter

- **Get-Content -Path "E:\wiprofiles\configfile1"**

```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1 Day5_assignment_cmdlets_and_code(1).ps1* X
41
42 Get-Process | Format-Table -Property Name , CPU , StartTime
43
44 #Format-List
45 Get-Service | Format-List -Property Name , Status , DisplayName
46
47 #Format-Wide
48 Get-ChildItem | Format-Wide -Column 3
49
50 #-----get content-----
51 $con=Get-Content -Path "E:\wiprofiles\configfile1"
52 Write-Host "$con"
53

PS C:\Windows\system32> #-----get content-----
$con=Get-Content -Path "E:\wiprofiles\configfile1"
Write-Host "$con"
InterfaceAlias : VMware Network Adapter VMnet1 InterfaceIndex : 12 InterfaceDescription : VMware Virtual Ethernet Adapter for VMnet1 IPv4Address
: 192.168.245.1 IPv6DefaultGateway : IPv4DefaultGateway : fe00::ffff::1 fe00::0::ffff::2
fe00::0::ffff::3 InterfaceAlias : Ethernet0 InterfaceIndex : 3 InterfaceDescription : Intel(R) 82574L Gigabit Network Connection NetProfile
e.Name : Network 6 IPv4Address : 172.0.19 IPv6DefaultGateway : IPv4DefaultGateway : 172.20.0.1 DNSServer : 172.20.0.1 InterfaceA
Int : VMware Network Adapter VMnet8 InterfaceIndex : 5 InterfaceDescription : VMware Virtual Ethernet Adapter for VMnet8 IPv4Address : 192.
68.162.1 IPv6DefaultGateway : IPv4DefaultGateway : fe00::ffff::1 fe00::0::ffff::2
fe00::0::ffff::3 Status Name DisplayName
AarSvc_6c015 Agent Activation Runtime_6c015 Stopped AJRouter All1Join Router Service Stopped ALG Application
on Layer Gateway Service Running AppIDSvc Application Identity Running AppInfo Application Information
Running AppMgmt Application Management Stopped AppReadiness App Readiness Stopped AppClient
Microsoft App-V Client Stopped AppXSvc AppX Deployment Service (AppXSVC) Stopped AssignedAccessM... AssignedAccessManager Serv
ice Running AudioEndpointBu... Windows Audio Endpoint Builder Running Audiosrv Windows Audio
mesvc Cellular Timer Stopped ActiveX ActiveX Instance (AxInstSV) Running BalloonService BalloonService
Stopped BroadcastDVRUserSer... GameDVR and Broadcast User Service... Stopped BDESVC Bitlocker Drive Encryption Service Runn
ing BFE Base Filtering Engine Stopped BITS Background Intelligent Transfer Ser... Stopped BTAGService Bluetooth Audio Gateway Service
tooth User Support Service_6c015 Running BrokerInfrastruc... Background Tasks Infrastructure Ser... Stopped BTAGService Bluetooth Support Service Stopped cansvc
Running BthavctpSvc AVCTP service Stopped bthserv Running ClickToRunSvc Microsoft Office Click-to-Run Service Stopped ClipSVC Clipboard User Servic
e_6c015 Capability Access Manager Service Stopped CaptureService... CaptureService_6c015 Running cbdhsvc_6c015 Clipboard User Service
ertPropSvc Certificate Propagation Connected Devices Platform Service Running CDUserSvc_6c015 Connected Devices Platform User Ser... Running C
ense Service (ClipSVC) Stopped Cloudbase-init cloudbase-init Stopped Clouddsvc Microsoft Cloud Identity Service
Running COMSysApp COM+ System Application Stopped ConsentUserSv... ConsentUX_6c015 Running CryptSvc Cryptographic Services
CoreMessaging Stopped CredentialEnrol... CredentialEnrollmentManagerUserSv... Running DcomLaunch DCOM Server Process Launcher
Declared Configuration(DC) service Stopped defragsvc Optimized drives Stopped dicsvc Stopped DeviceAssociati... DeviceAssociatio

```

Objects, Arrays, Variables, Scripting Constraints

➤ Variable

Variables are used to store values. They are denoted by \$ symbol followed by name. They are not case sensitive. They can include letters, numbers and underscore. There is no need to declare datatypes of variables. However, we can use type casting if needed.

Code Example:

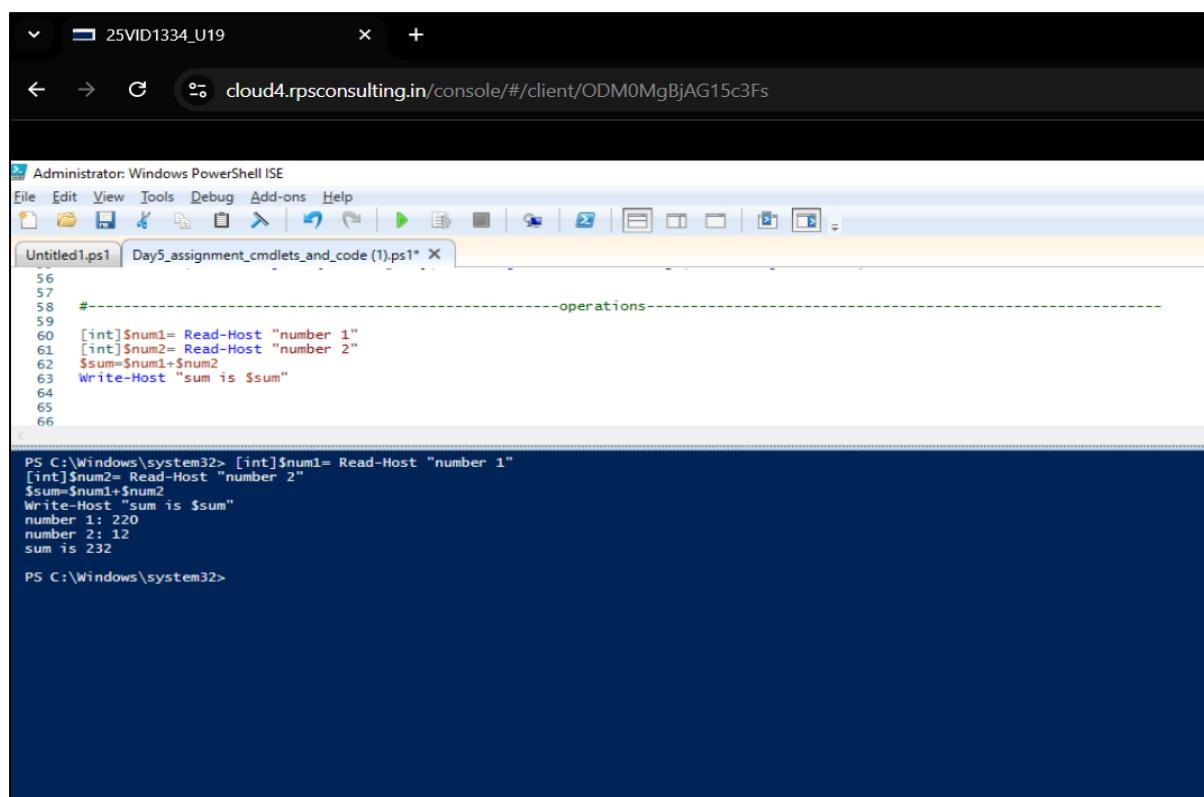
```
$name="Ghanshyam"  
$age=22  
$isTrue=$true  
[int]$number="123"
```

➤ Input and Output Formatting

1. **Read-Host:** This prompts the user for input and stores it as a string.
2. **Write-Host:** Displays output directly to the console.

Code Example:

```
[int]$num1= Read-Host "number 1"  
[int]$num2= Read-Host "number 2"  
  
$sum=$num1+$num2  
  
Write-Host "Sum is $sum"
```



The screenshot shows the Windows PowerShell ISE interface. The title bar says "Administrator: Windows PowerShell ISE". The menu bar includes File, Edit, View, Tools, Debug, Add-ons, Help. The toolbar has various icons for file operations. A tab bar shows "Untitled1.ps1" and "Day5_assignment_cmdlets_and_code (1).ps1". The main editor area contains the following PowerShell script:

```
56  
57  
58 #-----operations-----  
59 [int]$num1= Read-Host "number 1"  
60 [int]$num2= Read-Host "number 2"  
61 $sum=$num1+$num2  
62 Write-Host "sum is $sum"  
63  
64  
65  
66
```

Below the editor, the PowerShell command prompt shows the execution of the script:

```
PS C:\Windows\system32> [int]$num1= Read-Host "number 1"  
[int]$num2= Read-Host "number 2"  
$sum=$num1+$num2  
Write-Host "sum is $sum"  
number 1: 220  
number 2: 12  
sum is 232
```

- **String Formatting:** -f format operator allows for composite formatting.

Code Example:

```
$name="Ghanshyam"  
$age=22  
"My name is {0} and I am {1} years old" -f $name,$age
```

The screenshot shows a Windows PowerShell ISE window titled "Administrator: Windows PowerShell ISE". The script file is "Day5_assignment_cmdlets_and_code (1).ps1". The code includes a section for string formatting:

```
59  
60 [int]$num1= Read-Host "number 1"  
61 [int]$num2= Read-Host "number 2"  
62 $sum=$num1+$num2  
63 Write-Host "sum is $sum"  
64 #-----String Formatting -----  
65 $name = "Ghanshyam"  
66 $age =22  
67 "My name is {0} and i am {1} years old" -f $name, $age  
68  
69
```

The output window shows the execution of the script:

```
PS C:\Windows\system32> $arr=@(1,"gh",3,4,5)  
$n=$arr[1]  
Write-Host "Array second element $n"  
Write-Host "Entire array $arr"  
Array second element gh  
Entire array 1 gh 3 4 5  
PS C:\Windows\system32> $name = "Ghanshyam"  
$age =22  
"My name is {0} and i am {1} years old" -f $name, $age  
My name is Ghanshyam and i am 22 years old  
PS C:\Windows\system32> |
```

- **Array**

Arrays are used to store collections of items. An array can hold multiple objects of the same or different types. We can also declare them directly with @(elements).

Code Example:

```
$arr=@(1,2,3,"gh")  
$first=$arr[0]  
$last=$arr[-1]  
Write-Host "Array first element is $first"  
Write-Host "Array last element is $last"  
Write-Host "Entire array $arr"
```

The screenshot shows a Windows PowerShell ISE window titled "25VID1334_U19". The URL in the address bar is "cloud4.rpsconsulting.in/console/#/client/ODM0MgBjAG15c3Fs". The code in the editor is:

```
71
72
73 #-----Array-----
74 $arr=@(1,"gh",3,4,5)
75 $first=$arr[0]
76 $last=$arr[-1]
77 Write-Host "Array first element $first"
78 Write-Host "Array last element $last"
79 |
80 Write-Host "Entire array $arr"
81
```

The output window shows the execution of the script:

```
My name is Ghanshyam and i am 22 years old
PS C:\Windows\system32> $arr=@(1,"gh",3,4,5)
$first=$arr[0]
$last=$arr[-1]
Write-Host "Array first element $first"
Write-Host "Array last element $last"

Write-Host "Entire array $arr"
Array first element 1
Array last element 5
Entire array 1 gh 3 4 5
PS C:\Windows\system32>
```

Size of an array can be adjusted. We can add elements **using += operator**.

Code Example:

```
$AddArray=@()

for($i=0;$i -le 3; $i++){
    $a=Read-Host "Enter anything"
    $AddArray+=$a
}
```

The screenshot shows a Windows PowerShell ISE window titled "25VID1334_U19". The URL in the address bar is "cloud4.rpsconsulting.in/console/#/client/ODM0MgBjAG15c3Fs". The code in the editor is:

```
83 $AddArray=@()
84
85 for($i=0;$i -le 3; $i++){
86     $a = Read-host "enter anything "
87     $AddArray+=$a
88 }
89
90 write-Host "$AddArray"
91
92
93
```

The output window shows the execution of the script:

```
PS C:\Windows\system32> $AddArray=@()
for($i=0;$i -le 3; $i++){
$a = Read-host "enter anything "
$AddArray+=$a
}
enter anything : 1
enter anything : 4
enter anything : 6
enter anything : gh
PS C:\Windows\system32> Write-Host "$AddArray"
1 4 6 gh
PS C:\Windows\system32>
```

- **Conditional Statements: if, elseif, else:** These are used to execute code blocks based on conditions.

Code Example:

```
$age=25
if($age -ge 18){Write-Host "Adult"}
elseif($age -ge 13){Write-Host "Teenager"}
else{Write-Host "Child"}
```

The screenshot shows the Windows PowerShell ISE interface. The script file 'Untitled1.ps1' contains the following code:

```
98 Write-Host "$1"
99 }
100 #-----conditional statement -----
101
102 $age=22
103 if($age -gt 18)
104 {
105     Write-Host "Adult"
106 }
107 elseif($age -gt 13)
108 {
109     Write-Host "Teen"
110 }
111 else
112 {
113     Write-Host "child"
114 }
115 #-----switch-----
116 #<#>"sun"
```

The output window shows the command PS C:\Windows\system32> \$age=22 followed by the execution of the script. The output is:

```
PS C:\Windows\system32> $age=22
if($age -gt 18)
{
    Write-Host "Adult"
}
elseif($age -gt 13)
{
    Write-Host "Teen"
}
else
{
    Write-Host "child"
}
Adult
```

- **Switch:** Switch case efficiently handles multiple conditions.

Code Example:

```
$day="sun"

switch($day){
    "mon" {Write-Host "Start week"}
    "sun" {Write-Host "holi day"}
    "fri" {Write-Host "end week"}
    default {Write-Host "nothing...."}
}
```

```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1 Day5_assignment_cmdlets_and_code (1).ps1*
113     Write-Host "child"
114 }
115
116 #-----switch-----
117 $day="sun"
118 switch($day)
119 {
120     "mon" {Write-Host "start week"}
121     "sun" {Write-Host "holi day"}
122     "fri" {Write-Host "end week "}
123     default {Write-Host "nothing ..."}
124 }
125
126 #-----function-----
127 function greet{
128     param($name )
129     Write-Host "Hello, $name"
130 }
131
132 function greet2{
133 }

PS C:\Windows\system32> $day="sun"
switch($day)
{
    "mon" {Write-Host "start week"}
    "sun" {Write-Host "holi day"}
    "fri" {Write-Host "end week "}
    default {Write-Host "nothing ..."}
}
holi day

PS C:\Windows\system32> |

```

➤ Looping Statements

1. **for:** For loop iterates a specific number of times.

Code Example:

```

for($i=-5;$i -le 0;$i++){
    Write-Host "$i"
}

```

```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1 Day5_assignment_cmdlets_and_code (1).ps1*
92
93
94
95
96 #-----for-----
97 for($i=-5;$i -le 0 ; $i++)
98 {
99     Write-Host "$i"
100 }
101
102
103
104
105
106

PS C:\Windows\system32> for($i=-5;$i -le 0 ; $i++)
{
    Write-Host "$i"
}

-5
-4
-3
-2
-1
0

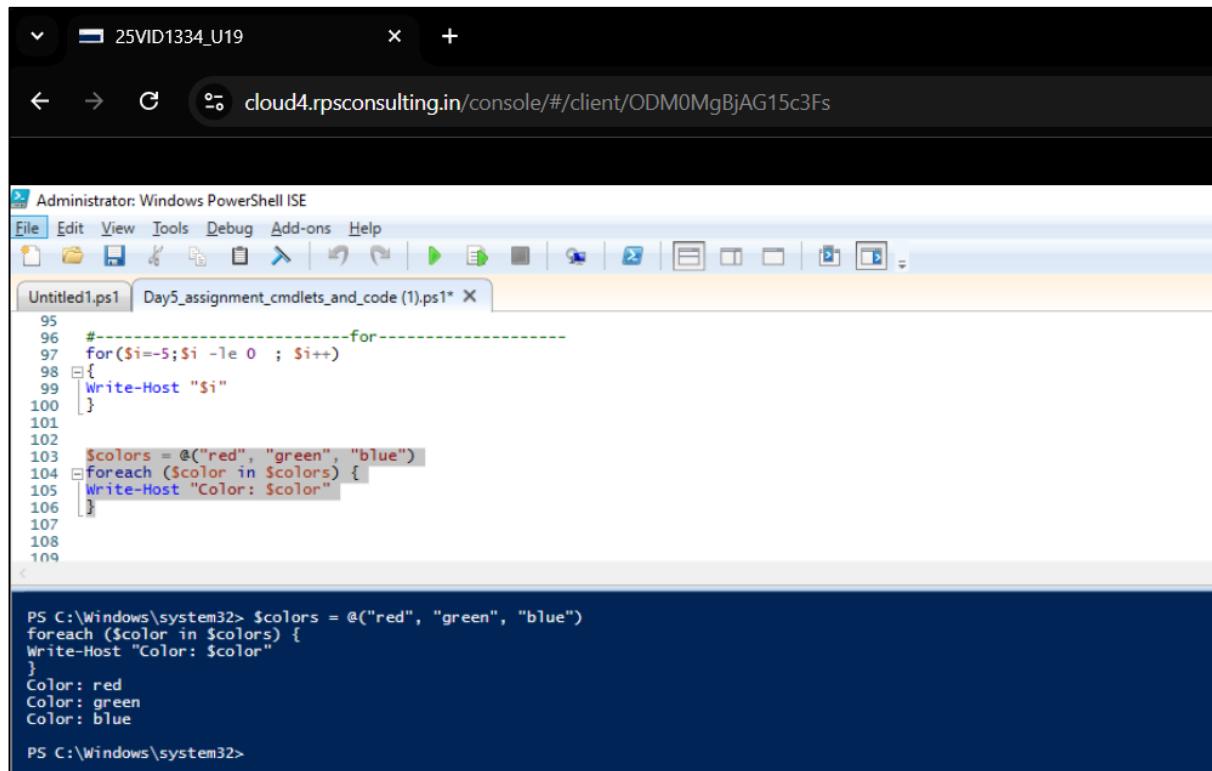
PS C:\Windows\system32>

```

2. foreach: Foreach loop iterates through a collection.

Code Example:

```
$colors="red","green","blue"  
  
foreach($elem in $colors){  
  
    Write-Host "$elem"  
  
}
```



The screenshot shows the Windows PowerShell ISE interface. The top navigation bar includes tabs for 'File', 'Edit', 'View', 'Tools', 'Debug', 'Add-ons', and 'Help'. Below the tabs is a toolbar with various icons. The main area contains two panes: a script editor on the left and a terminal window on the right. The script editor pane shows a file named 'Untitled1.ps1' with the following content:

```
95  #-----for  
96  for($i=-5;$i -le 0 ; $i++)  
97  {  
98      Write-Host "$i"  
99  }  
100  
101  
102  
103  $colors = @("red", "green", "blue")  
104  foreach ($color in $colors) {  
105      Write-Host "Color: $color"  
106  }  
107  
108  
109
```

The terminal window below shows the execution of the script:

```
PS C:\Windows\system32> $colors = @("red", "green", "blue")  
foreach ($color in $colors) {  
    Write-Host "Color: $color"  
}  
Color: red  
Color: green  
Color: blue  
PS C:\Windows\system32>
```

3. while: While loop repeats as long as a condition is true.

Code Example:

```
$count=0  
  
while($count -lt 3){  
  
    Write-Host "Count: $count"  
  
    $count++  
  
}
```

```
Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1 Day5_assignment_cmdlets_and_code (1).ps1*
99     Write-Host $i
100    }
101
102
103 $colors = @("red", "green", "blue")
104 foreach ($color in $colors) {
105     Write-Host "Color: $color"
106 }
107
108 $count = 0
109 while ($count -lt 3) {
110     Write-Host "Count: $count"
111     $count++
112 }
113

PS C:\Windows\system32> $count = 0
while ($count -lt 3) {
Write-Host "Count: $count"
$count++
}

Count: 0
Count: 1
Count: 2

PS C:\Windows\system32>
```

4. **do-while & do-until:** They are similar to while but the condition is checked at the end.

Code Example:

```
$a=9
do{
    "Starting Loop $a"
    $a
    $a++
    "Now '$a is $a'"
}while($a -le 5)

#-----do-until

$a=0
do{
    "Starting Loop $a"
    $a
    $a++
    "Now '$a is $a'"
}until($a -le 5)
```

```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1 | Day5_assignment_cmdlets_and_code (1).ps1* ×
114 #-----do-while
115 $a=9
116 do{
117 "Starting Loop $a"
118 $a++
119 "Now '$a is $a"
120 }while($a -le 5)
122 #-----do-until
123 $a=0
124 do{
125 "Starting Loop $a"
126 $a++
127 $a++
128 "Now '$a is $a"
129 }until($a -le 5)
130

PS C:\Windows\system32> $a=9
do{
"Starting Loop $a"
$a
$a++
"Now '$a is $a"
}while($a -le 5)

Starting Loop 9
9
Now $a is 10
PS C:\Windows\system32> #-----do-until
$a=0
do{
"Starting Loop $a"
$a
$a++
"Now '$a is $a"
}until($a -le 5)
Starting Loop 0
0
Now $a is 1

```

Completed | Ln 125 Col 1

➤ **Functions:** PowerShell functions are blocks of code designed to perform specific tasks making scripts more modular, reusable and easier to maintain. To define a basic function, we use the function keyword followed by the function name and a block of code enclosed in curly braces { }. To invoke or call a function, simply use the function's name. Parameters can be added to the function declaration to make them more flexible. The keyword param is used to declare all parameters in the first function statement. To invoke a function with arguments, the values are passed to the function as parameters or list arguments after the function call.

Code Example:

```

function greet {
    param($name)
    Write-Host "Hello, $name"

}

• greet -name "ghanshyam"

function greet2 {
    param($name, $age)
    Write-Host "Hello, $name your age is $age"

}

• greet -name "ghanshyam" -age 21

```

```
Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1 Day5_assignment cmdlets and code (1).ps1* ×
156 [ ]
157 #
158 #-----function-----
159 function greet{
160     param($name)
161     Write-Host "Hello, $name"
162 }
163
164 function greet2{
165     param($name, $age)
166     Write-Host "Hello, $name your age is $age"
167 }
168
169
170 greet -name "ghanshyam"
171 greet2 -name "ghanshyam" -age "21"
172

Now $a is 1
PS C:\Windows\system32> function greet{
param($name )
Write-Host "Hello, $name"
}

function greet2{
param($name , $age )
Write-Host "Hello, $name your age is $age "
}

greet -name "ghanshyam"
greet2 -name "ghanshyam" -age "21"
Hello, ghanshyam
Hello, ghanshyam your age is 21
PS C:\Windows\system32>
```

➤ **Comment:** PowerShell allows for two types of comments: single-line and multi-line (block) comments. Single-line comments are created using the hash symbol (#) whereas Multi-line comments are created using the <#.....#> delimiters.

Code Example:

Single-line comment

```
#heyyyyyyyyyyyyyyyy
```

Multi-line comment

```
<#
```

First line

Second line

N lines

```
#>
```

The screenshot shows the Windows PowerShell ISE interface. In the top navigation bar, it says "25VID1334_U19" and the address bar shows "cloud4.rpsconsulting.in/console/#/client/ODM0MgBjAG15c3Fs". Below the toolbar, there's a tab bar with "Untitled1.ps1" and "Day5_assignment_cmdlets_and_code (1).ps1". The main code editor area contains the following PowerShell script:

```
196
197
198
199
200
201 #-----single line comment
202
203 <#
204 # multiple lines
205 # are
206 # commented
207
208 #
209
210
211
212

Hello, ghanshyam your age is 21
PS C:\Windows\system32> #-----single line comment

<#
multiple lines
are
commented

#>

PS C:\Windows\system32>
```

➤ **Error Handling:** try, catch, finally blocks to manage exceptions

Code Example:

```
try {
    # Code that might throw an error
    Get-Content "nonexistent_file.txt" -ErrorAction Stop
}
catch {
    Write-Host "Error: $($_.Exception.Message)"
}
finally {
    #This block will run even if error occurs
    Write-Host "Cleanup actions"
}
```

25VID1334_U19

cloud4.rpsconsulting.in/console/#/client/ODM0MgBjAG15c3Fs

Administrator: Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

Untitled1.ps1 Day5_assignment_cmdlets_and_code (1).ps1*

```
211
212
213 try {
214     # Code that might throw an error
215     Get-Content "nonexistent_file.txt" -ErrorAction Stop
216 } catch {
217     Write-Host "Error: $($_.Exception.Message)"
218 } finally {
219     #this part will run even if error occurs
220     Write-Host "Cleanup actions"
221 }
222
223
224
225
226
227
```

```
PS C:\Windows\system32> try {
# Code that might throw an error
Get-Content "nonexistent_file.txt" -ErrorAction Stop
} catch {
Write-Host "Error: $($_.Exception.Message)"
} finally {
#this part will run even if error occurs
Write-Host "Cleanup actions"
}

Error: Cannot find path 'C:\Windows\system32\nonexistent_file.txt' because it does not exist.
Cleanup actions

PS C:\Windows\system32>
```