MyWebUniversity
python
PowerShell Get-Help cmdlet
PS C:\> Get-Help
FPDF Library
PDF generator
FPDF
Full credit is given to all the above companies including the Operating System that this PDF file was generated!

Windows PowerShell Get-Help on Cmdlet 'Where-Object'

PS:\>Get-HELP Where-Object -Full

NAME

Where-Object

SYNOPSIS

Selects objects from a collection based on their property values.

SYNTAX

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -CContains [-InputObject <System.Management.Automation.PSObject>]

[<CommonParameters>]

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -CEQ [-InputObject <System.Management.Automation.PSObject>]

[<CommonParameters>]

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -CGE [-InputObject <System.Management.Automation.PSObject>]

[<CommonParameters>]

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -CGT [-InputObject <System.Management.Automation.PSObject>]

[<CommonParameters>]

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -CIn [-InputObject <System.Management.Automation.PSObject>]

[<CommonParameters>]

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -CLE [-InputObject <System.Management.Automation.PSObject>]

[<CommonParameters>]

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -CLike [-InputObject <System.Management.Automation.PSObject>]

[<CommonParameters>]

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -CLT [-InputObject <System.Management.Automation.PSObject>]

[<CommonParameters>]

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -CMatch [-InputObject <System.Management.Automation.PSObject>]

[<CommonParameters>]

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -CNE [-InputObject <System.Management.Automation.PSObject>]

[<CommonParameters>]

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -CNotContains [-InputObject <System.Management.Automation.PSObject>]

[<CommonParameters>]

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -CNotIn [-InputObject

```
<System.Management.Automation.PSObject>]
    [<CommonParameters>]

    Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -CNotLike
[-InputObject <System.Management.Automation.PSObject>]
    [<CommonParameters>]

    Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -CNotMatch
[-InputObject <System.Management.Automation.PSObject>]
    [<CommonParameters>]

    Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -Contains
[-InputObject <System.Management.Automation.PSObject>]
    [<CommonParameters>]

    Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] [-EQ] [-InputObject
<System.Management.Automation.PSObject>]
    [<CommonParameters>]

    Where-Object [-FilterScript] <System.Management.Automation.ScriptBlock> [-InputObject
<System.Management.Automation.PSObject>] [<CommonParameters>]

    Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -GE [-InputObject
<System.Management.Automation.PSObject>]
    [<CommonParameters>]

    Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -GT [-InputObject
<System.Management.Automation.PSObject>]
    [<CommonParameters>]

    Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] -In [-InputObject
<System.Management.Automation.PSObject>]
    [<CommonParameters>]
```

```
        Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] [-InputObject
<System.Management.Automation.PSObject>] -Is
    [<CommonParameters>]

        Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] [-InputObject
<System.Management.Automation.PSObject>] -IsNot
    [<CommonParameters>]

        Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] [-InputObject
<System.Management.Automation.PSObject>] -LE
    [<CommonParameters>]

        Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] [-InputObject
<System.Management.Automation.PSObject>] -Like
    [<CommonParameters>]

        Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] [-InputObject
<System.Management.Automation.PSObject>] -LT
    [<CommonParameters>]

        Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] [-InputObject
<System.Management.Automation.PSObject>] -Match
    [<CommonParameters>]

        Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] [-InputObject
<System.Management.Automation.PSObject>] -NE
    [<CommonParameters>]

        Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] [-InputObject
<System.Management.Automation.PSObject>] -NotContains
    [<CommonParameters>]
```

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] [-InputObject <System.Management.Automation.PSObject>] -NotIn
[<CommonParameters>]

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] [-InputObject <System.Management.Automation.PSObject>] -NotLike
[<CommonParameters>]

Where-Object [-Property] <System.String> [[-Value] <System.Management.Automation.PSObject>] [-InputObject <System.Management.Automation.PSObject>] -NotMatch
[<CommonParameters>]

DESCRIPTION

The `Where-Object` cmdlet selects objects that have particular property values from the collection of objects that are passed to it. For example, you can use the
`Where-Object` cmdlet to select files that were created after a certain date, events with a particular ID, or computers that use a particular version of Windows.

Starting in Windows PowerShell 3.0, there are two different ways to construct a `Where-Object` command.

- Script block . You can use a script block to specify the property name, a comparison operator, and a property value. `Where-Object` returns all objects for which
the script block statement is true.

For example, the following command gets processes in the `Normal` priority class, that is, processes where the value of the PriorityClass property equals `Normal`.

`Get-Process | Where-Object {$_.PriorityClass -eq "Normal"}`

All PowerShell comparison operators are valid in the script block format. For more information, see about_Comparison_Operators
(./About/about_Comparison_Operators.md).

- Comparison statement . You can also write a comparison statement, which is much more like natural language. Comparison statements were introduced in Windows
PowerShell 3.0.

For example, the following commands also get processes that have a priority class of `Normal`. These commands are equivalent and you can use them interchangeably.

`Get-Process | Where-Object -Property PriorityClass -EQ -Value "Normal"`

`Get-Process | Where-Object PriorityClass -EQ "Normal"`

Starting in Windows PowerShell 3.0, `Where-Object` adds comparison operators as parameters in a `Where-Object` command. Unless specified, all operators are
case-insensitive. Before Windows PowerShell 3.0, the comparison operators in the PowerShell language were only usable in script blocks.

When you provide a single Property to `Where-Object`, the cmdlet treats the value of the property as a boolean expression. When the value of the property's Length
isn't zero, the expression evaluates to `$true`. For example: `('hi', '', 'there') | Where-Object Length`

The previous example is functionally equivalent to:

- `('hi', '', 'there') | Where-Object Length -GT 0`

- `('hi', '', 'there') | Where-Object { $_.Length -gt 0 }`

For more information about how PowerShell evaluates booleans, see about_Booleans (about/about_Booleans.md).

-CContains <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects from a collection if the property value of the object is an exact match for the specified value. This operation is
case-sensitive.

For example: `Get-Process | Where-Object ProcessName -CContains "svchost"` CContains refers to a collection of values and is true if the collection contains an
item that is an exact match for the specified value. If the input is a single object, PowerShell converts it to a collection of one object.

This parameter was introduced in Windows PowerShell 3.0.

```
Required?                    true
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-CEQ <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value is the same as the specified value. This operation is case-sensitive.

This parameter was introduced in Windows PowerShell 3.0.

```
Required?                    true
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-CGE <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value is greater than or equal to the specified value. This operation is case-sensitive.

This parameter was introduced in Windows PowerShell 3.0.

Required?	true
Position?	named
Default value	False
Accept pipeline input?	False
Accept wildcard characters?	false

-CGT <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value is greater than the specified value. This operation is case-sensitive.

This parameter was introduced in Windows PowerShell 3.0.

Required?	true
Position?	named
Default value	False
Accept pipeline input?	False
Accept wildcard characters?	false

-CIn <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value includes the specified value. This operation is case-sensitive.

For example: `Get-Process | Where-Object -Value "svchost" -CIn ProcessName` CIn resembles CContains , except that the property and value positions are reversed.

For example, the following statements are both true.

`"abc", "def" -CContains "abc"`

`"abc" -CIn "abc", "def"`

This parameter was introduced in Windows PowerShell 3.0.

Required?   true

Position?   named

Default value   False

Accept pipeline input?   False

Accept wildcard characters? false

-CLE <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value is less-than or equal to the specified value. This operation is case-sensitive.

This parameter was introduced in Windows PowerShell 3.0.

Required?   true

Position?   named

Default value   False

Accept pipeline input?   False

Accept wildcard characters? false

-CLike <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value matches a value that includes wildcard characters (`*`). This operation is case-sensitive.

For example: `Get-Process | Where-Object ProcessName -CLike "*host"`

This parameter was introduced in Windows PowerShell 3.0.

Required?   true

Position?   named

Default value   False

Accept pipeline input?   False

Accept wildcard characters? false

-CLT <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value is less-than the specified value. This operation is case-sensitive.

This parameter was introduced in Windows PowerShell 3.0.

```
Required?                    true
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-CMatch <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value matches the specified regular expression. This operation is case-sensitive. When the input is a

single object, the matched value is saved in the `$Matches` automatic variable.

For example: `Get-Process | Where-Object ProcessName -CMatch "Shell"`

This parameter was introduced in Windows PowerShell 3.0.

```
Required?                    true
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-CNE <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value is different than the specified value. This operation is case-sensitive.

This parameter was introduced in Windows PowerShell 3.0.

Required?   true

Position?   named

Default value   False

Accept pipeline input?   False

Accept wildcard characters? false

-CNotContains <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value of the object isn't an exact match for the specified value. This operation is case-sensitive.

For example: `Get-Process | Where-Object ProcessName -CNotContains "svchost"` NotContains and CNotContains refer to a collection of values and are true when the

collection doesn't contain any items that are an exact match for the specified value. If the input is a single object, PowerShell converts it to a collection of

one object.

This parameter was introduced in Windows PowerShell 3.0.

Required?   true

Position?   named

Default value   False

Accept pipeline input?   False

Accept wildcard characters? false

-CNotIn <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value isn't an exact match for the specified value. This operation is case-sensitive.

For example: `Get-Process | Where-Object -Value "svchost" -CNotIn -Property ProcessName` NotIn and CNotIn operators resemble NotContains and CNotContains , except

that the property and value positions are reversed. For example, the following statements are true.

`"abc", "def" -CNotContains "Abc"`

`"abc" -CNotIn "Abc", "def"`

```
Required?                    true
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-CNotLike <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value doesn't match a value that includes wildcard characters. This operation is case-sensitive.

For example: `Get-Process | Where-Object ProcessName -CNotLike "*host"`

This parameter was introduced in Windows PowerShell 3.0.

```
Required?                    true
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-CNotMatch <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value doesn't match the specified regular expression. This operation is case-sensitive. When the input is

a single object, the matched value is saved in the `$Matches` automatic variable.

For example: `Get-Process | Where-Object ProcessName -CNotMatch "Shell"`

This parameter was introduced in Windows PowerShell 3.0.

Required?   true

Position?   named

Default value   False

Accept pipeline input?   False

Accept wildcard characters? false

-Contains <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if any item in the property value of the object is an exact match for the specified value.

For example: `Get-Process | Where-Object ProcessName -Contains "Svchost"`

If the input is a single object, PowerShell converts it to a collection of one object.

This parameter was introduced in Windows PowerShell 3.0.

Required?   true

Position?   named

Default value   False

Accept pipeline input?   False

Accept wildcard characters? false

-EQ <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value is the same as the specified value.

This parameter was introduced in Windows PowerShell 3.0.

Required?   false

Position?   named

Default value   False

Accept pipeline input?   False

Accept wildcard characters? false

```
-FilterScript <System.Management.Automation.ScriptBlock>
    Specifies the script block that's used to filter the objects. Enclose the script block in braces (`{}`).


    The parameter name, FilterScript , is optional.


    Required?                    true
    Position?                    0
    Default value                None
    Accept pipeline input?       False
    Accept wildcard characters?  false


-GE <System.Management.Automation.SwitchParameter>
    Indicates that this cmdlet gets objects if the property value is greater than or equal to the specified value.


    This parameter was introduced in Windows PowerShell 3.0.


    Required?                    true
    Position?                    named
    Default value                False
    Accept pipeline input?       False
    Accept wildcard characters?  false


-GT <System.Management.Automation.SwitchParameter>
    Indicates that this cmdlet gets objects if the property value is greater than the specified value.


    This parameter was introduced in Windows PowerShell 3.0.


    Required?                    true
    Position?                    named
    Default value                False
    Accept pipeline input?       False
    Accept wildcard characters?  false
```

-In <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value matches any of the specified values. For example:

`Get-Process | Where-Object -Property ProcessName -in -Value "Svchost", "TaskHost", "WsmProvHost"`

If the input is a single object, PowerShell converts it to a collection of one object.

If the property value of an object is an array, PowerShell uses reference equality to determine a match. `Where-Object` returns the object only if the value of
the Property parameter and any value of Value are the same instance of an object.

This parameter was introduced in Windows PowerShell 3.0.

```
Required?                    true
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-InputObject <System.Management.Automation.PSObject>

Specifies the objects to filter. You can also pipe the objects to `Where-Object`.

When you use the InputObject parameter with `Where-Object`, instead of piping command results to `Where-Object`, the cmdlet treats the InputObject as a single
object. This is true even if the value is a collection that's the result of a command, such as `-InputObject (Get-Process)`.

Because InputObject can't return individual properties from an array or collection of objects, we recommend that, if you use `Where-Object` to filter a collection
of objects for those objects that have specific values in defined properties, you use `Where-Object` in the pipeline, as shown in the examples in this topic.

```
Required?                    false
```

Position?   named

Default value   None

Accept pipeline input?   True (ByValue)

Accept wildcard characters? false

-Is <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value is an instance of the specified .NET type. Enclose the type name in square brackets.

For example, `Get-Process | Where-Object StartTime -Is [DateTime]`

This parameter was introduced in Windows PowerShell 3.0.

Required?   true

Position?   named

Default value   False

Accept pipeline input?   False

Accept wildcard characters? false

-IsNot <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value isn't an instance of the specified .NET type.

For example, `Get-Process | where StartTime -IsNot [DateTime]`

This parameter was introduced in Windows PowerShell 3.0.

Required?   true

Position?   named

Default value   False

Accept pipeline input?   False

Accept wildcard characters? false

-LE <System.Management.Automation.SwitchParameter>

```
    Indicates that this cmdlet gets objects if the property value is less than or equal to the specified value.


    This parameter was introduced in Windows PowerShell 3.0.


    Required?                    true
    Position?                    named
    Default value                False
    Accept pipeline input?       False
    Accept wildcard characters?  false


-Like <System.Management.Automation.SwitchParameter>
    Indicates that this cmdlet gets objects if the property value matches a value that includes wildcard characters (`*`).


    For example: `Get-Process | Where-Object ProcessName -Like "*host"`


    This parameter was introduced in Windows PowerShell 3.0.


    Required?                    true
    Position?                    named
    Default value                False
    Accept pipeline input?       False
    Accept wildcard characters?  false


-LT <System.Management.Automation.SwitchParameter>
    Indicates that this cmdlet gets objects if the property value is less than the specified value.


    This parameter was introduced in Windows PowerShell 3.0.


    Required?                    true
    Position?                    named
    Default value                False
    Accept pipeline input?       False
    Accept wildcard characters?  false
```

-Match <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value matches the specified regular expression. When the input is a single object, the matched value is

saved in the `$Matches` automatic variable.

For example: `Get-Process | Where-Object ProcessName -Match "shell"`

This parameter was introduced in Windows PowerShell 3.0.

```
Required?                    true
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-NE <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value is different than the specified value.

This parameter was introduced in Windows PowerShell 3.0.

```
Required?                    true
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-NotContains <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if none of the items in the property value is an exact match for the specified value.

For example: `Get-Process | Where-Object ProcessName -NotContains "Svchost"` NotContains refers to a collection of values and is true if the collection doesn't contain any items that are an exact match for the specified value. If the input is a single object, PowerShell converts it to a collection of one object.

This parameter was introduced in Windows PowerShell 3.0.

```
Required?                    true
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-NotIn <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value isn't an exact match for any of the specified values.

For example: `Get-Process | Where-Object -Value "svchost" -NotIn -Property ProcessName`

If the value of Value is a single object, PowerShell converts it to a collection of one object.

If the property value of an object is an array, PowerShell uses reference equality to determine a match. `Where-Object` returns the object only if the value of

Property and any value of Value aren't the same instance of an object.

This parameter was introduced in Windows PowerShell 3.0.

```
Required?                    true
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-NotLike <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects if the property value doesn't match a value that includes wildcard characters (`*`).

For example: `Get-Process | Where-Object ProcessName -NotLike "*host"`

This parameter was introduced in Windows PowerShell 3.0.

```
Required?                    true
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-NotMatch <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets objects when the property value doesn't match the specified regular expression. When the input is a single object, the matched

value is saved in the `$Matches` automatic variable.

For example: `Get-Process | Where-Object ProcessName -NotMatch "PowerShell"`

This parameter was introduced in Windows PowerShell 3.0.

```
Required?                    true
Position?                    named
Default value                False
Accept pipeline input?       False
Accept wildcard characters?  false
```

-Property <System.String>

Specifies the name of a property of the input object. The property must be an instance property, not a static property. This is a positional parameter, so the

name, Property , is optional.

This parameter was introduced in Windows PowerShell 3.0.

```
Required?                    true
```

Position?   0

Default value   None

Accept pipeline input?   False

Accept wildcard characters? false

-Value <System.Management.Automation.PSObject>

Specifies a property value. The parameter name, Value , is optional. This parameter accepts wildcard characters when used with the following comparison parameters:

- CLike - CNotLike - Like - NotLike This parameter was introduced in Windows PowerShell 3.0.

Required?   false

Position?   1

Default value   None

Accept pipeline input?   False

Accept wildcard characters? true

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug,

ErrorAction, ErrorVariable, WarningAction, WarningVariable,

OutBuffer, PipelineVariable, and OutVariable. For more information, see

about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

INPUTS

System.Management.Automation.PSObject

You can pipe any object to this cmdlet.

OUTPUTS

System.Object

This cmdlet returns the selected items from the input object set.

NOTES

Windows PowerShell includes the following aliases for `Where-Object`:

- `?`   - `where`

Starting in Windows PowerShell 4.0, `Where` and `ForEach` methods were added for use with collections.

You can read more about these methods here about_arrays (./About/about_Arrays.md)

--------------- Example 1: Get stopped services ---------------

```
Get-Service | Where-Object { $_.Status -eq "Stopped" }
Get-Service | Where-Object Status -EQ "Stopped"
```

-------- Example 2: Get processes based on working set --------

```
Get-Process | Where-Object { $_.WorkingSet -GT 250MB }
Get-Process | Where-Object WorkingSet -GT 250MB
```

-------- Example 3: Get processes based on process name --------

```
Get-Process | Where-Object { $_.ProcessName -Match "^p.*" }
Get-Process | Where-Object ProcessName -Match "^p.*"
```

-------- Example 4: Use the comparison statement format --------

```
Get-Process | Where-Object -Property Handles -GE -Value 1000
Get-Process | where Handles -GE 1000
```

}

--------- Example 5: Get commands based on properties ---------

```
# Use Where-Object to get commands that have any value for the OutputType
# property of the command. This omits commands that do not have an OutputType
# property and those that have an OutputType property, but no property value.
Get-Command | Where-Object OutputType
Get-Command | Where-Object { $_.OutputType }

# Use Where-Object to get objects that are containers. This gets objects that
# have the **PSIsContainer** property with a value of $True and excludes all
# others.
Get-ChildItem | Where-Object PSIsContainer
Get-ChildItem | Where-Object { $_.PSIsContainer }

# Finally, use the -not operator (!) to get objects that are not containers.
# This gets objects that do have the **PSIsContainer** property and those
# that have a value of $False for the **PSIsContainer** property.
Get-ChildItem | Where-Object { !$_.PSIsContainer }
# You cannot use the -not operator (!) in the comparison statement format
# of the command.
Get-ChildItem | Where-Object PSIsContainer -eq $False
```

-------------- Example 6: Use multiple conditions --------------

```
Get-Module -ListAvailable | Where-Object {
   ($_.Name -notlike "Microsoft*" -and $_.Name -notlike "PS*") -and $_.HelpInfoUri
}
```

This example shows how to create a `Where-Object` command with multiple conditions.

This command gets non-core modules that support the Updatable Help feature. The command uses the ListAvailable parameter of the `Get-Module` cmdlet to get all modules

on the computer. A pipeline operator (`|`) sends the modules to the `Where-Object` cmdlet, which gets modules whose names don't begin with `Microsoft` or `PS`, and

have a value for the HelpInfoURI property, which tells PowerShell where to find updated help files for the module. The `-and` logical operator connects the comparison

statements.

The example uses the script block command format. Logical operators, such as `-and`,`-or`, and `-not` are valid only in script blocks. You can't use them in the

comparison statement format of a `Where-Object` command.

- For more information about PowerShell logical operators, see about_Logical_Operators (./About/about_logical_operators.md). - For more information about the

Updatable Help feature, see about_Updatable_Help (./About/about_Updatable_Help.md).

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.core/where-object?view=powershell-5.1&WT.mc_id=ps-gethelp

Compare-Object

ForEach-Object

Group-Object

Measure-Object

New-Object

Select-Object

Sort-Object

Tee-Object

about_Booleans