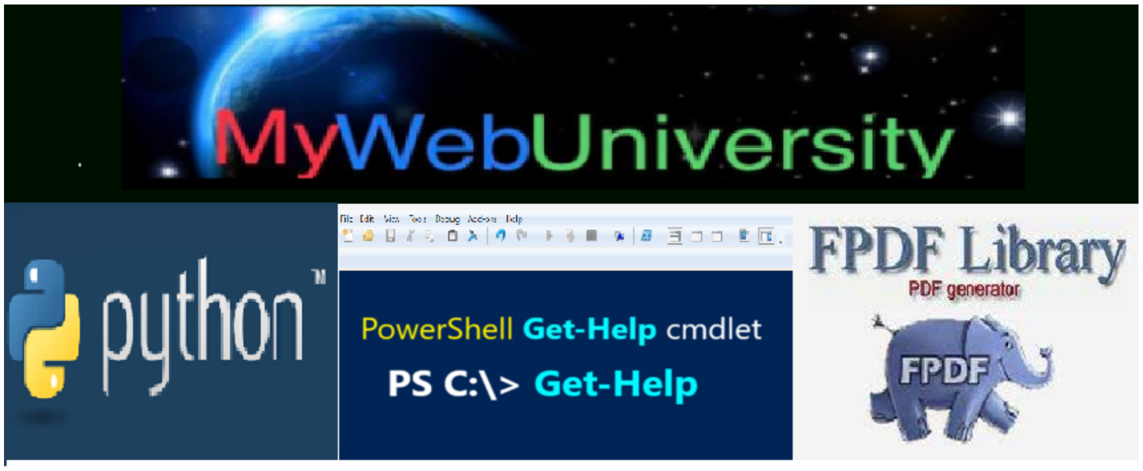

Windows PowerShell Get-Help on Cmdlet 'Invoke-Expression'

PS:\>Get-HELP Invoke-Expression -Full

NAME

Invoke-Expression

SYNOPSIS

Runs commands or expressions on the local computer.

SYNTAX

Invoke-Expression [-Command] <System.String> [<CommonParameters>]

DESCRIPTION

The `Invoke-Expression` cmdlet evaluates or runs a specified string as a command and returns the results of the expression or command. Without `Invoke-Expression`, a

string submitted at the command line is returned (echoed) unchanged.

Expressions are evaluated and run in the current scope. For more information, see about_Scopes (../Microsoft.PowerShell.Core/About/about_Scopes.md).

> [!CAUTION] > Take reasonable precautions when using the `Invoke-Expression` cmdlet in scripts. When using >

`Invoke-Expression` to run a command that the user

enters, verify that the command is safe to run > before running it. In general, it is best to design your script with predefined input options, > rather than allowing

freeform input.

PARAMETERS

-Command <System.String>

Specifies the command or expression to run. Type the command or expression or enter a variable that contains the command or expression. The Command parameter is

required.

```
Required?                    true
Position?                    0
Default value                None
Accept pipeline input?       True (ByValue)
Accept wildcard characters?  false
```

<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.String

You can pipe a string representing the expression to invoke to this cmdlet. Use the `$Input` automatic variable to represent the input objects in the command.

System.Management.Automation.PSObject

You can pipe an object representing the expression to invoke to this cmdlet. Use the `$Input` automatic variable to represent the input objects in the command.

OUTPUTS

None

This cmdlet returns no output of its own, but the invoked command may return output.

NOTES

Windows PowerShell includes the following aliases for `Invoke-Expression`:

- `iex`

In most cases, you invoke expressions using PowerShell's call operator and achieve the same results. The call operator is a safer method. For more information,

see about_Operators (../microsoft.powershell.core/about/about_operators.md#call-operator-).

-------------- Example 1: Evaluate an expression --------------

```
$Command = "Get-Process"
$Command

Get-Process

Invoke-Expression $Command

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id   ProcessName
-------  ------    -----      ----- -----   ------     --   -----------
296       4       1572       1956    20       0.53     1348   AdtAgent
270       6       1328       800     34       0.06     2396   alg
67        2       620        484     20       0.22     716    ati2evxx
1060      15      12904      11840   74       11.48    892    CcmExec
1400      33      25280      37544   223      38.44    2564   communicator
```

...

This example demonstrates the use of `Invoke-Expression` to evaluate an expression. Without `Invoke-Expression`, the expression is printed, but not evaluated.

The first command assigns a value of `Get-Process` (a string) to the `$Command` variable.

The second command shows the effect of typing the variable name at the command line. PowerShell echoes the string.

The third command uses `Invoke-Expression` to evaluate the string.

-------- Example 2: Run a script on the local computer --------

```
Invoke-Expression -Command "C:\ps-test\testscript.ps1"
"C:\ps-test\testscript.ps1" | Invoke-Expression
```

These commands use `Invoke-Expression` to run a script, TestScript.ps1, on the local computer. The two commands are equivalent. The first uses the Command parameter
to specify the command to run. The second uses a pipeline operator (`|`) to send the command string to `Invoke-Expression`.

------------ Example 3: Run a command in a variable ------------

```
$Command = 'Get-Process | where {$_.cpu -gt 1000}'
Invoke-Expression $Command
```

This example runs a command string that is saved in the `$Command` variable.

The command string is enclosed in single quotation marks because it includes a variable, `$_`, which represents the current object. If it were enclosed in double
quotation marks, the `$_` variable would be replaced by its value before it was saved in the `$Command` variable.

--------- Example 4: Get and run a cmdlet Help example ---------

```
$Cmdlet_name = "Get-ComputerInfo"
$Example_number = 1
```

```
    $Example_code = (Get-Help $Cmdlet_name).examples.example[($Example_number-1)].code
    Invoke-Expression $Example_code
```

This command retrieves and runs the first example in the `Get-EventLog` cmdlet Help topic.

To run an example of a different cmdlet, change the value of the `$Cmdlet_name` variable to the name of the cmdlet. And, change the `$Example_number` variable to the

example number you want to run. The command fails if the example number is not valid.

> [!NOTE] > If the example code from the help file has output in the example, PowerShell attempts to run the > output along with the code and an error will be thrown.

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/invoke-expression?view=powershell-5.1&WT.mc_id=ps-gethelp

Invoke-Command

about_Scopes