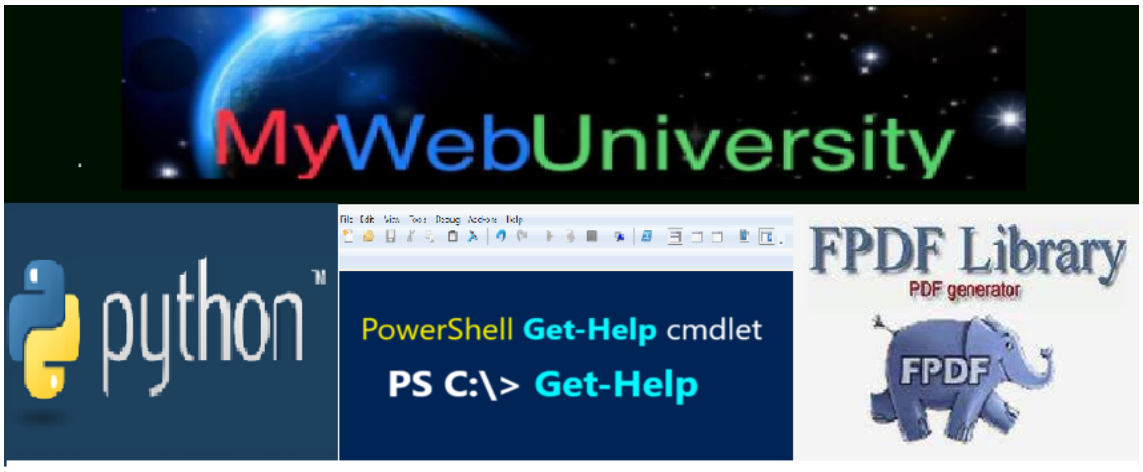

Full credit is given to all the above companies including the Operating System that this PDF file was generated!

Windows PowerShell Get-Help on Cmdlet 'wildcard'

PS:\>Get-HELP wildcard

TOPIC

Should

SYNOPSIS

Provides assertion convenience methods for comparing objects and throwing test failures when test expectations fail.

DESCRIPTION

Should is an Extension of System.Object and can be used as a native type inside Describe blocks. The various Should member methods can be invoked directly from an object being compared. It is typically used in individual It blocks to verify the results of an expectation. The Should method is typically called from the "actual" object being compared and takes the "expected" object as a parameter. Should includes several members that perform various comparisons of objects and will throw a PesterFailure when the objects do not evaluate to be comparable.

SHOULD MEMBERS

Be

Compares one object with another for equality and throws if the two

objects are not the same.

```
$actual="Actual value"
$actual | Should Be "actual value" # Test will pass
$actual | Should Be "not actual value"  # Test will fail
```

BeExactly

Compares one object with another for equality and throws if the two objects are not the same. This comparison is case sensitive.

```
$actual="Actual value"
$actual | Should BeExactly "Actual value" # Test will pass
$actual | Should BeExactly "actual value" # Test will fail
```

BeLike

Users a wildcard to compare two objects. The comparision is not case sensitive.

```
"I am a value" | Should BeLike "*value"  # Test will pass
"I am a value" | Should BeLike "value"  # Test will fail
```

Tip: use [Management.Automation.WildcardPattern]::Escape() to escape wildcard-characters.

BeLikeExactly

Users a wildcard to compare two objects. The comparision is case sensitive.

```
"I am a value" | Should BeLike "*value"  # Test will pass
"I am a value" | Should BeLike "*VALUE"  # Test will fail
```

Tip: use [Management.Automation.WildcardPattern]::Escape() to escape wildcard-characters.

BeGreaterThan

Asserts that a number is greater than an expected value. Uses PowerShell's -gt operator to compare the two values.

```
$Error.Count | Should BeGreaterThan 0
```

BeLessThan

Asserts that a number is less than an expected value. Uses PowerShell's -gt operator to compare the two values.

```
$Error.Count | Should BeLessThan 1
```

Exist

Does not perform any comparison but checks if the object calling Exist
is present in a PS Provider. The object must have valid path syntax. It
essentially must pass a Test-Path call.

```
$actual=(Dir . )[0].FullName
Remove-Item $actual
$actual | Should Exist # Test will fail
```

Contain

Checks to see if a file contains the specified text. This search is not case sensitive and uses regular expressions.

```
Set-Content -Path TestDrive:\file.txt -Value 'I am a file.'
'TestDrive:\file.txt' | Should Contain 'I Am' # Test will pass
'TestDrive:\file.txt' | Should Contain '^I.*file$' # Test will pass

'TestDrive:\file.txt' | Should Contain 'I Am Not' # Test will fail
```

Tip: Use [regex]::Escape("pattern") to match the exact text.

```
Set-Content -Path TestDrive:\file.txt -Value 'I am a file.'
'TestDrive:\file.txt' | Should Contain 'I.am.a.file' # Test will pass
'TestDrive:\file.txt' | Should Contain ([regex]::Escape('I.am.a.file')) # Test will fail
```

ContainExactly

Checks to see if a file contains the specified text. This search is case sensitive and uses regular expressions to match

the text.

```
Set-Content -Path TestDrive:\file.txt -Value 'I am a file.'
'TestDrive:\file.txt' | Should Contain 'I am' # Test will pass
'TestDrive:\file.txt' | Should Contain 'I Am' # Test will fail
```

Match

Uses a regular expression to compare two objects. This comparison is not case sensitive.

```
"I am a value" | Should Match "I Am" # Test will pass
"I am a value" | Should Match "I am a bad person" # Test will fail
```

Tip: Use [regex]::Escape("pattern") to match the exact text.

```
"Greg" | Should Match ".reg" # Test will pass
"Greg" | Should Match ([regex]::Escape(".reg")) # Test will fail
```

MatchExactly

Uses a regular expression to compare two objects. This comparison is case sensitive.

```
"I am a value" | Should MatchExactly "I am" # Test will pass
"I am a value" | Should MatchExactly "I Am" # Test will fail
```

Throw

Checks if an exception was thrown in the input ScriptBlock.

```
{ foo } | Should Throw # Test will pass
{ $foo = 1 } | Should Throw # Test will fail
{ foo } | Should Not Throw # Test will fail
{ $foo = 1 } | Should Not Throw # Test will pass
```

Warning: The input object must be a ScriptBlock, otherwise it is processed outside of the assertion.

Get-Process -Name "process" -ErrorAction Stop | Should Throw # Should pass, but the exception thrown by Get-Process causes the test to fail.

BeNullOrEmpty

Checks values for null or empty (strings). The static [String]::IsNullOrEmpty() method is used to do the comparison.

```
$null | Should BeNullOrEmpty # Test will pass
$null | Should Not BeNullOrEmpty # Test will fail
@()   | Should BeNullOrEmpty # Test will pass
""    | Should BeNullOrEmpty # Test will pass
```

USING SHOULD IN A TEST

```
function Add-Numbers($a, $b) {
    return $a + $b
}

Describe "Add-Numbers" {

    It "adds positive numbers" {
        $sum = Add-Numbers 2 3
        $sum | should be 3
    }
}
```

This test will fail since 3 will not be equal to the sum of 2 and 3.

SEE ALSO

Describe

Context

It