



python



PowerShell

FPDF Library
PDF generator

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

PowerShell Get-Help on command 'It'

PS C:\Users\wahid> Get-Help It

NAME

It

SYNOPSIS

Validates the results of a test inside of a Describe block.

SYNTAX

```
It [-name] <String> [[-test] <ScriptBlock>] [-TestCases <IDictionary[]>]  
[<CommonParameters>]
```

```
It [-name] <String> [[-test] <ScriptBlock>] [-TestCases <IDictionary[]>]  
[-Pending] [<CommonParameters>]
```

```
It [-name] <String> [[-test] <ScriptBlock>] [-TestCases <IDictionary[]>]  
[-Skip] [<CommonParameters>]
```

DESCRIPTION

The It command is intended to be used inside of a Describe or Context Block.

If you are familiar with the AAA pattern (Arrange-Act-Assert), the body of

the It block is the appropriate location for an assert. The convention is to assert a single expectation for each It block. The code inside of the It block should throw a terminating error if the expectation of the test is not met and thus cause the test to fail. The name of the It block should expressively state the expectation of the test.

In addition to using your own logic to test expectations and throw exceptions, you may also use Pester's Should command to perform assertions in plain language.

PARAMETERS

`-name <String>`

An expressive phrase describing the expected test outcome.

`-test <ScriptBlock>`

The script block that should throw an exception if the expectation of the test is not met. If you are following the AAA pattern (Arrange-Act-Assert), this typically holds the Assert.

`-TestCases <IDictionary[]>`

Optional array of hashtable (or any IDictionary) objects. If this parameter is used,

Pester will call the test script block once for each table in the TestCases array,

splating the dictionary to the test script block as input. If you want the name of

the test to appear differently for each test case, you can embed tokens into the Name

parameter with the syntax 'Adds numbers <A> and ' (assuming you have keys named A and B

in your TestCases hashtables.)

-Pending [<SwitchParameter>]

Use this parameter to explicitly mark the test as work-in-progress/not implemented/pending when you need to distinguish a test that fails because it is not finished yet from a tests that fail as a result of changes being made in the code base. An empty test, that is a test that contains nothing except whitespace or comments is marked as Pending by default.

-Skip [<SwitchParameter>]

Use this parameter to explicitly mark the test to be skipped. This is preferable to temporarily commenting out a test, because the test remains listed in the output. Use the Strict parameter of Invoke-Pester to force all skipped tests to fail.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see [about_CommonParameters \(https://go.microsoft.com/fwlink/?LinkID=113216\)](https://go.microsoft.com/fwlink/?LinkID=113216).

----- EXAMPLE 1 -----

```
PS C:\>function Add-Numbers($a, $b) {
```

```
    return $a + $b
```

```
}
```

```
Describe "Add-Numbers" {
```

```
    It "adds positive numbers" {
```

```
$sum = Add-Numbers 2 3
$sum | Should Be 5
}
```

```
It "adds negative numbers" {
    $sum = Add-Numbers (-2) (-2)
    $sum | Should Be (-4)
}
```

```
It "adds one negative number to positive number" {
    $sum = Add-Numbers (-2) 2
    $sum | Should Be 0
}
```

```
It "concatenates strings if given strings" {
    $sum = Add-Numbers two three
    $sum | Should Be "twothree"
}
}
```

----- EXAMPLE 2 -----

```
PS C:\>function Add-Numbers($a, $b) {
    return $a + $b
}
```

```
Describe "Add-Numbers" {
    $testCases = @(
        @{ a = 2;    b = 3;    expectedResult = 5 }
```

```
@{ a = -2; b = -2;   expectedResult = -4 }
@{ a = -2; b = 2;   expectedResult = 0 }
@{ a = 'two'; b = 'three'; expectedResult = 'twothree' }
)
```

It 'Correctly adds <a> and to get <expectedResult>' -TestCases

```
$testCases {
    param ($a, $b, $expectedResult)

    $sum = Add-Numbers $a $b
    $sum | Should Be $expectedResult
}
}
```

REMARKS

To see the examples, type: "get-help It -examples".

For more information, type: "get-help It -detailed".

For technical information, type: "get-help It -full".

For online help, type: "get-help It -online"