Windows PowerShell

Pester testing

Pester testing

Pester is a framework for writing unit tests in PowerShell. This allows for Behavior Driven Development based tests. For example, you need a function that multiplies any number by 2.

```
function double-up ($number)
{
    return $number + 2
}
```

The function we wrote might be wrong, but how will we ever know for sure?

Testing the function

```
Describe 'double-up' {
    It "Try number 2" {
        double-up 2 | Should -Be 4
    }

    It "Try number 4" {
        double-up 4 | Should -Be 8
    }
}
```

Result:

```
Describing double-up
  [+] Try number 2 471ms
  [-] Try number 4 194ms
    Expected 8, but got 6.
    12:         double-up 4 | Should -Be 8
```

What are unit tests used for?

Test driven development. You start with a simple function that does one thing, and passes all the tests. Then you want that function to keep on doing the same, but also something else and add this to the test. This way you make sure not only the new functionality works, but the old one stays intact as well.

- Write a function that doubles the parameter
- Add an optional second parameter, if it's present, multiply both parameters

```
Describe 'double-up' {
    It "Try number 2" {
        double-up 2 | Should -Be 4 }

    It "Try number 4" {
        double-up 4 | Should -Be 8 }

    It "Multiply 3 and 4" {
        double-up 3 4 | Should -Be (3*4) }
}
```

Function mark 1:

```
function double-up ($number)
{
    return $number * 2
}
```

Returns:

```
Describing double-up
  [+] Try number 2 4ms
  [+] Try number 4 4ms
  [-] Multiply 3 and 4 6ms
    Expected 12, but got 6.
    18:         double-up 3 4 | Should -Be (3*4)
```

Function mark 2

```
function double-up ($number, $multiplier = 2)
{
    return $number * $multiplier
}
```

Returns:

```
Describing double-up
  [+] Try number 2 3ms
  [+] Try number 4 8ms
  [+] Multiply 3 and 4 5ms
```

But I'm not a programmer

Good point: Unit testing and test driven development is a thing among programmers.
As scripters we'll never have enough code to necessitate pester testing on our scripts.

But who ever claimed testing should only be written for code?

```
Describe 'Test network' {
    It "Ping localhost" {
        $ping = Test-NetConnection -ComputerName localhost
        $ping.PingSucceeded | Should -Be True }
    It "IP-connection to the internet" {
        $ping = Test-NetConnection -ComputerName 8.8.8.8
        $ping.PingSucceeded | Should -Be True }
    It "Name resolution" {
        $ping = Test-NetConnection -ComputerName www.google.com
        $ping.PingSucceeded | Should -Be True }
    It "VPN up and running" {
        $ping = Test-NetConnection -ComputerName 1.0.128.125
        $ping.PingSucceeded | Should -Be True }
}
```

Result

```
Describing Test network
  [+] Ping localhost 1.15s
  [+] Ipconnection to the internet 9.18s
  [+] Name resolution 172ms
WARNING: Ping to 1.0.128.125 failed with status: TimedOut
  [-] VPN up and running 14.05s
    Expected 'True', but got $false.
    19:         $ping.PingSucceeded | Should -Be True
```

This test would quickly give an overview of what is wrong with a certain computer. A replacement for the "Let's check the basics before moving on" that we always do when a server is acting up.