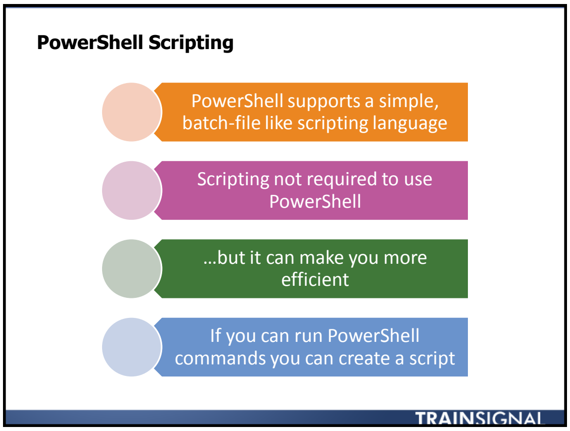
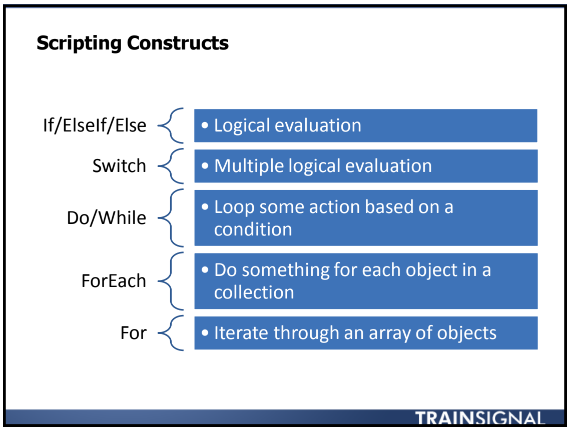
PowerShell Scripts and Functions



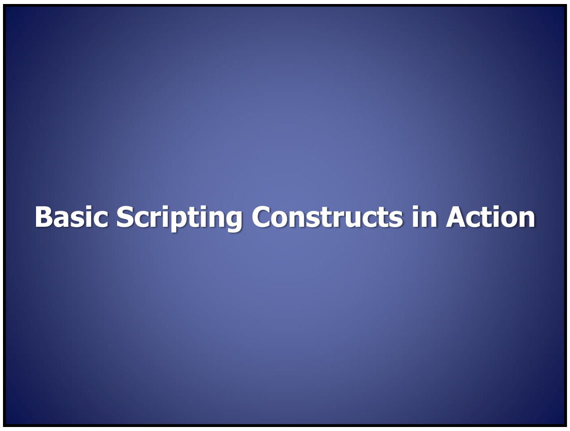
PowerShell Scripting

- PowerShell supports a simple, batch-file like scripting language
- Scripting not required to use PowerShell
- ...but it can make you more efficient
- If you can run PowerShell commands you can create a script

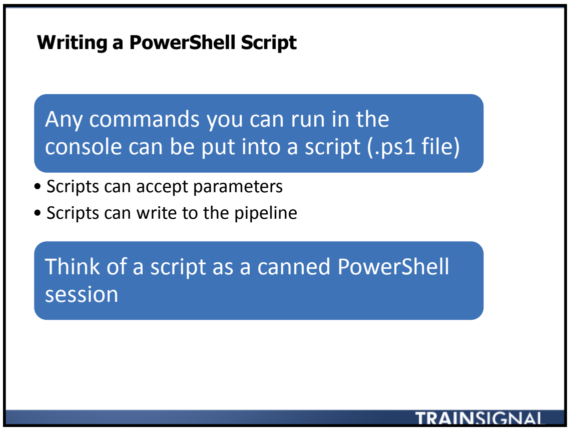


Scripting Constructs

If/ElseIf/Else	• Logical evaluation
Switch	• Multiple logical evaluation
Do/While	• Loop some action based on a condition
ForEach	• Do something for each object in a collection
For	• Iterate through an array of objects



Basic Scripting Constructs in Action

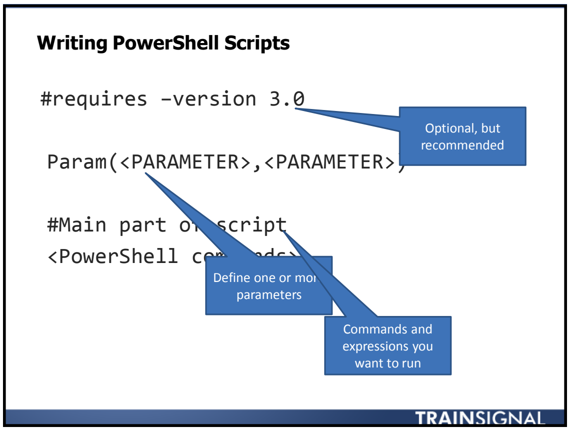


Writing a PowerShell Script

Any commands you can run in the console can be put into a script (.ps1 file)

- Scripts can accept parameters
- Scripts can write to the pipeline

Think of a script as a canned PowerShell session



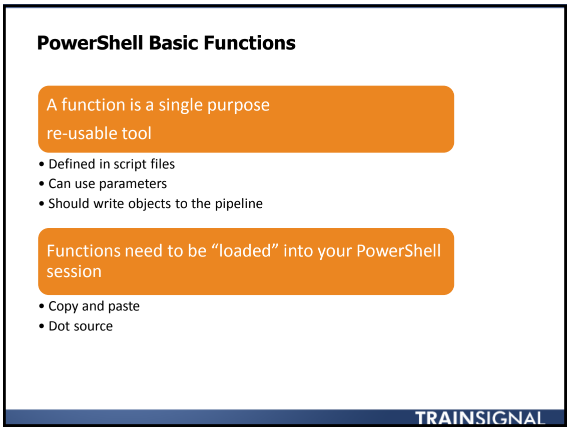
Writing PowerShell Scripts

```
#requires -version 3.0

 Param(<PARAMETER>,<PARAMETER>)

 #Main part of script
 <PowerShell commands>
```

Optional, but recommended

Define one or more parameters

Commands and expressions you want to run



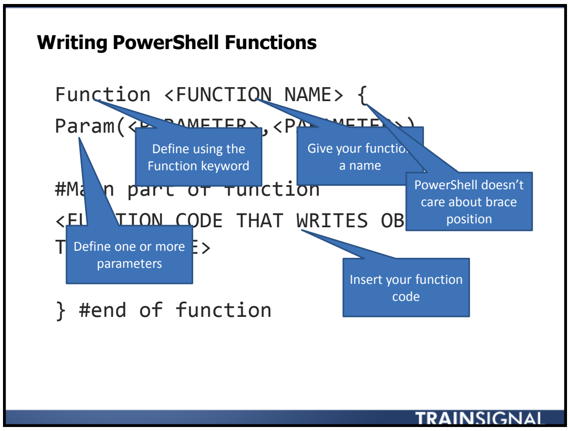
PowerShell Basic Functions

A function is a single purpose re-usable tool

- Defined in script files
- Can use parameters
- Should write objects to the pipeline

Functions need to be “loaded” into your PowerShell session

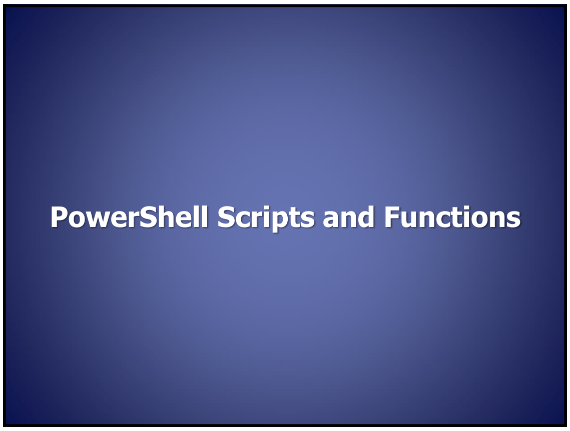
- Copy and paste
- Dot source



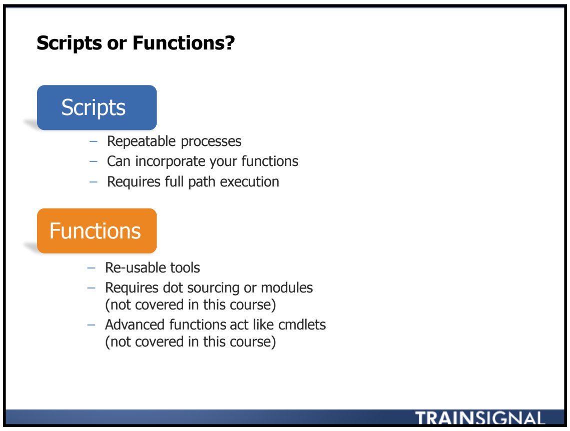
Writing PowerShell Functions

```
Function <FUNCTION NAME> {
Param(<PARAMETER>,<PARAMETER>)

#Main part of function
<FUNCTION CODE THAT WRITES OB
T                          E>

} #end of function
```

Define using the Function keyword
Give your function a name
PowerShell doesn’t care about brace position
Define one or more parameters
Insert your function code



PowerShell Scripts and Functions



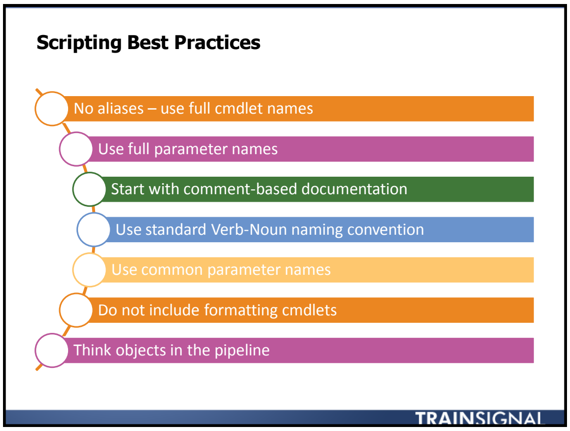
Scripts or Functions?

Scripts

- Repeatable processes
- Can incorporate your functions
- Requires full path execution

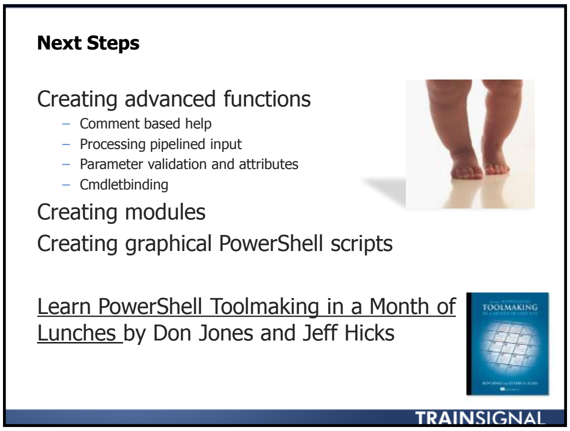
Functions

- Re-usable tools
- Requires dot sourcing or modules (not covered in this course)
- Advanced functions act like cmdlets (not covered in this course)



Scripting Best Practices

- No aliases – use full cmdlet names
- Use full parameter names
- Start with comment-based documentation
- Use standard Verb-Noun naming convention
- Use common parameter names
- Do not include formatting cmdlets
- Think objects in the pipeline



Next Steps

Creating advanced functions

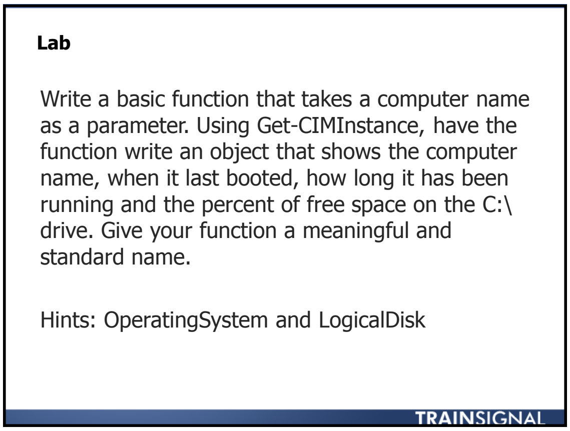
- Comment based help
- Processing pipelined input
- Parameter validation and attributes
- Cmdletbinding

Creating modules

Creating graphical PowerShell scripts

Learn PowerShell Toolmaking in a Month of Lunches by Don Jones and Jeff Hicks

TOOLMAKING



Lab

Write a basic function that takes a computer name as a parameter. Using Get-CIMInstance, have the function write an object that shows the computer name, when it last booted, how long it has been running and the percent of free space on the C:\ drive. Give your function a meaningful and standard name.

Hints: OperatingSystem and LogicalDisk