



PowerShell

55039: Windows PowerShell Scripting and Toolmaking

\$2,995.00

- 5 Days

Upcoming Dates

Oct 28 - Nov 01

Course Description

This 5-day instructor-led class is intended for IT professionals who are interested in furthering their skills in Windows PowerShell and administrative automation. The course assumes a basic working knowledge of PowerShell as an interactive command-line shell, and teaches students the correct patterns and practices for building reusable, tightly scoped units of automation.

Course Outline

Module 1: Tool Design

This module explains how to design tools and units of automation that comply with native PowerShell usage patterns.

Lessons

- Tools do one thing
- Tools are flexible
- Tools look native

Lab : Designing a Tool

- Design a tool

After completing this module, students will be able to:

- Describe the native shell patterns that a good tool design should exhibit

Module 2: Start with a Command

This module explains how to start the scripting process by beginning in the interactive shell console.

Lessons

- Why start with a command?
- Discovery and experimentation

Lab : Designing a Tool

- Start with a command

After completing this module, students will be able to:

- Describe the benefits of discovery and experimentation in the console
- Discover and experiment with existing commands in the console

Module 3: Build a Basic Function and Module

This module explains how to build a basic function and module, using commands already experimented with in the shell.

Lessons

- Start with a basic function
- Create a script module
- Check prerequisites
- Run the new command

Lab : Designing a Tool

- Build a basic function and module

After completing this module, students will be able to:

- Build a basic function
- Create a script module
- Run a command from a script module

Module 4: Adding CmdletBinding and Parameterizing

This module explains how to extend the functionality of a tool, parameterize input values, and use CmdletBinding.

Lessons

- About CmdletBinding and common parameters
- Accepting pipeline input
- Mandatory-ness
- Parameter validation
- Parameter aliases

Lab : Designing a Tool

- Adding CmdletBinding and Parameterizing

After completing this module, students will be able to:

- Describe the purpose of CmdletBinding and list common parameters
- Parameterize a script's input
- Define parameters as mandatory
- Define parameters as accepting pipeline input
- Define parameter validation

Module 5: Emitting Objects as Output

This module explains how to create tools that produce custom objects as output.

Lessons

- Assembling information

- Constructing and emitting output
- Quick tests

Lab : Designing a Tool

- Emitting objects as output

After completing this module, students will be able to:

- Describe the purpose of object-based output
- Create and output custom objects from a function

Module 6: An Interlude: Changing Your Approach

This module explains how to re-think tool design, using concrete examples of how it's often done wrong.

Lessons

- Examining a script
- Critiquing a script
- Revising the script

After completing this module, students will be able to:

- Describe the native patterns that a good tool design should exhibit
- Redesign a script to meet business requirements and conform to native patterns

Module 7: Using Verbose, Warning, and Informational Output

This module explains how to use additional output pipelines for better script behaviors.

Lessons

- Knowing the six channels
- Adding verbose and warning output
- Doing more with verbose output
- Informational output

Lab : Designing a Tool

- Using Verbose, Warning, and Informational Output

After completing this module, students will be able to:

- Describe the six output channels in the shell
- Write commands that use verbose, warning, and informational output
- Run commands with extra output enabled

Module 8: Comment-Based Help

This module explains how to add comment-based help to tools.

Lessons

- Where to put your help
- Getting started
- Going further with comment-based help
- Broken help

Lab : Designing a Tool

- Comment-based help

After completing this module, students will be able to:

- Describe the purpose and construction of comment-based help
- Add comment-based help to a function
- Identify causes of broken comment-based help

Module 9: Handling Errors

This module explains how to create tools that deal with anticipated errors.

Lessons

- Understanding errors and exceptions
- Bad handling
- Two reasons for exception handling
- Handling exceptions in our tool
- Capturing the actual exception
- Handling exceptions for non-commands
- Going further with exception handling
- Deprecated exception handling

Lab : Designing a Tool

- Handling errors

After completing this module, students will be able to:

- Describe the native patterns for handling errors in a command
- Add error handling to a command
- Run a command and observe error handling behaviors

Module 10: Basic Debugging

This module explains how to use native PowerShell script debugging tools.

Lessons

- Two kinds of bugs
- The ultimate goal of debugging
- Developing assumptions
- Write-Debug
- Set-PSBreakpoint
- The PowerShell ISE

Lab : Designing a Tool

- Basic debugging

After completing this module, students will be able to:

- Describe the tools used for debugging in PowerShell
- Debug a broken script

Module 11: Going Deeper with Parameters

This module explains how to further define parameter attributes in a PowerShell command.

Lessons

- Parameter positions
- Validation
- Multiple parameter sets
- Value from remaining arguments
- Help messages
- Aliases
- More CmdletBinding

After completing this module, students will be able to:

- Describe the use of positional parameters
- Describe additional parameter validation methods
- Describe how to define multiple parameter sets
- Describe other parameter definition options

Module 12: Writing Full Help

This module explains how to create external help for a command.

Lessons

- External help
- Using PlatyPs
- Supporting online help
- "About" topics
- Making your help updatable

Lab : Designing a Tool

- Writing full help

After completing this module, students will be able to:

- Describe the advantages of external help
- Create external help using PlatyPS and Markdown

Module 13: Unit Testing Your Code

This module explains how to use Pester to perform basic unit testing.

Lessons

- Sketching out the test
- Making something to test
- Expanding the test

- Going further with Pester

Lab : Designing a Tool

- Unit testing your code

After completing this module, students will be able to:

- Describe the purpose of unit testing
- Write basic unit tests for PowerShell functions

Module 14: Extending Output Types

This module explains how to extend objects with additional capabilities.

Lessons

- Understanding types
- The Extensible Type System
- Extending an object
- Using Update-TypeInfo

After completing this module, students will be able to:

- Describe the purpose of the ETS
- Extend an existing object type

Module 15: Analyzing Your Script

This module explains how to use Script Analyzer to support best practices and prevent common problems.

Lessons

- Performing a basic analysis
- Analyzing the analysis

Lab : Designing a Tool

- Analyzing your script

After completing this module, students will be able to:

- Describe the use of Script Analyzer
- Perform a basic script analysis

Module 16: Publishing Your Tools

This module explains how to publish tools to public and private repositories.

Lessons

- Begin with a manifest
- Publishing to PowerShell Gallery
- Publishing to private repositories

Lab : Designing a Tool

- Publishing your tools

After completing this module, students will be able to:

- Describe the tool publishing process and requirements
- Publish a tool to a repository

Module 17: Basic Controllers: Automation Scripts and Menus

This module explains how to create controller scripts that put tools to use.

Lessons

- Building a menu
- Using UIChoice
- Writing a process controller

Lab : Designing a Tool

- Basic controllers

After completing this module, students will be able to:

- Describe the purpose of basic controller scripts
- Write a simple controller script

Module 18: Adding a Graphical interface

This module explains the concepts and techniques used to apply a graphical interface on top of PowerShell code. The goal is to provide non-technical users with the advantages of PowerShell while utilizing the familiar graphical interface.

Lessons:

- Introduction to PowerShell Studio
- Building a basic form.
- Exploring common form controls
- Responding to events.
- Building a GUI from code you already have.

Module 19: Introduction to Desired State Configuration

This module will cover the basics of DSC and move into more advanced topics. We will configure our servers and DSC and then watch them auto correct misconfigurations. We will also create our own DSC Resources to handle customized configurations.

Lessons:

- What is DSC
- DSC terminology
- Deploying a basic configuration and correcting drift
- Deploying a configuration with resources from the PowerShell Gallery
- Creating our own DSC Resources for customized requirements

Module 20: Working with JSON Data

This module explains how to using JSON data in PowerShell.

Lessons

- Converting to JSON
- Converting from JSON

Lab : Designing a Tool

- Working with JSON data

After completing this module, students will be able to:

- Describe the use of JSON data within PowerShell
- Use JSON data within a PowerShell function

Module 21: Working with SQL Server Data

This module explains how to use SQL Server from within a PowerShell script.

Lessons

- SQL Server terminology and facts
- Connecting to the server and database
- Writing a query
- Running a query
- Invoke-SqlCmd
- Thinking about tool design patterns

After completing this module, students will be able to:

- Describe the use of SQL Server from within PowerShell
- Write and run SQL Server queries
- Design tools that use SQL Server for data storage

Module 22: Final Exam

This module provides a chance for students to use everything they have learned in this course within a practical example.

Lessons

- Lab problem
- Break down the problem
- Do the design
- Test the commands
- Code the tool

Lab : Final Exam

- Lab one

Lab : Final Exam

- Lab two

After completing this module, students will be able to:

- Create PowerShell tools, using native design patterns, from business requirements.

Audience

This course is intended for administrators in a Microsoft-centric environment who want to build reusable units of automation, automate business processes, and enable less-technical colleagues to accomplish administrative tasks.

Prerequisites

Before attending this course, students must have:

- Experience at basic Windows administration
- Experience using Windows PowerShell to query and modify system information
- Experience using Windows PowerShell to discover commands and their usage
- Experience using WMI and/or CIM to query system information

What You Will Learn

After completing this course, students will be able to:

- Describe the correct patterns for building modularized tools in Windows PowerShell
- Build highly modularized functions that comply with native PowerShell patterns
- Build controller scripts that expose user interfaces and automate business processes
- Manage data in a variety of formats
- Write automated tests for tools
- Debug tools