Document Generated: 10/13/2026

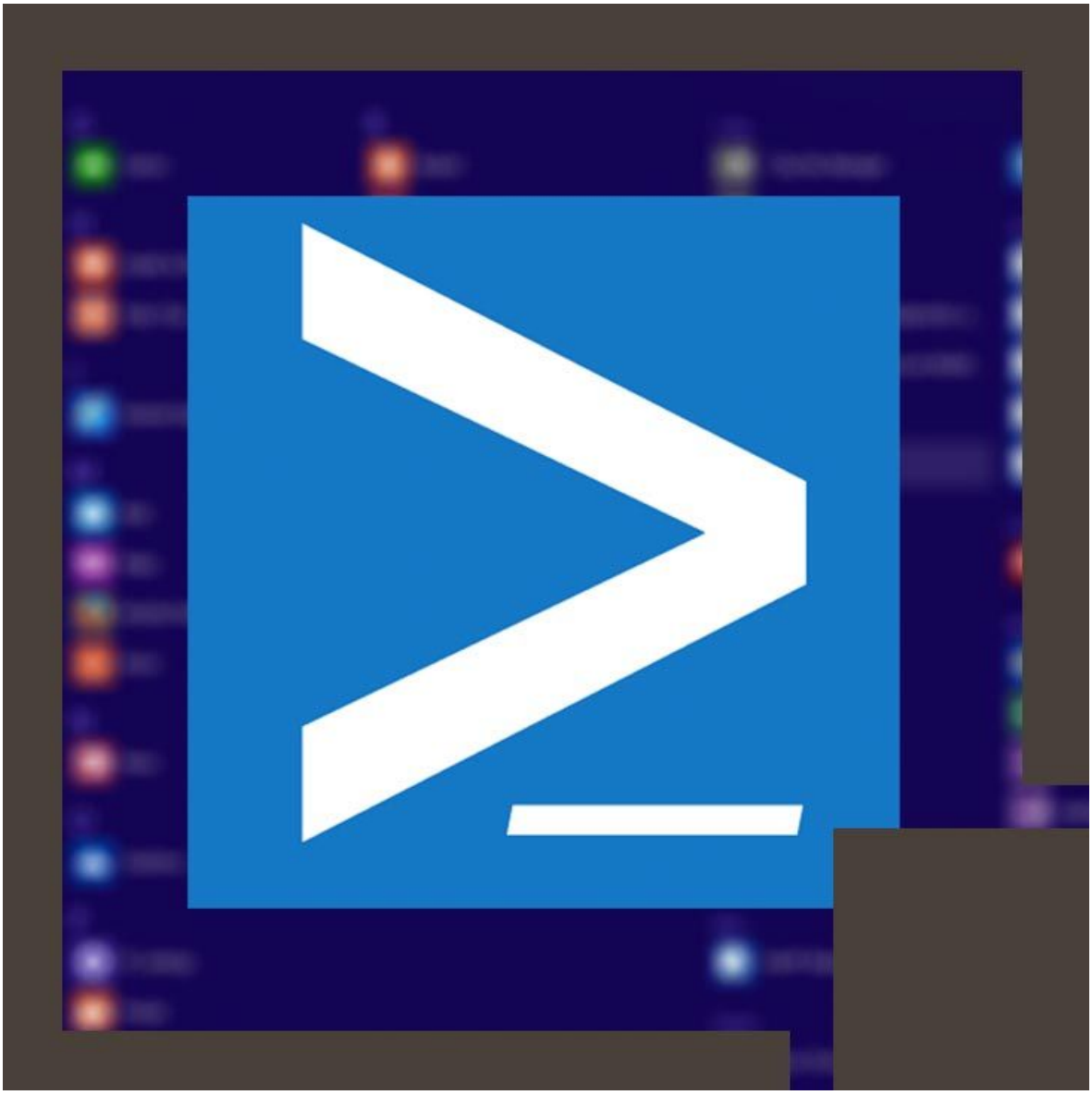
Learning Style: Virtual Classroom

Technology: Microsoft

Difficulty: Intermediate

Course Duration: 5 Days

Windows PowerShell Scripting and Toolmaking (MS-55039)



About this Course:

This instructor-led, 5-day course targets the IT students and professionals who desire to extend their skillset in Windows PowerShell and administrative automation. The course demands that the student must possess a fundamental practical knowledge in PowerShell as an interactive command-line shell. Students will learn the right practices and patterns for creating reusable, tightly scoped automation units.

The average salary of a Systems Administrator at national level is around **$72,762** year.

Course Objectives:

After the course completion, students should be able to:

- Explain the right patterns for creating modularized tools in Windows PowerShell
- Create highly modularized functions that work well with native PowerShell patterns
- Create controller scripts to uncover user interfaces and facilitate business processes automation
- Data management for various formats
- Write automated tests papers for tools
- Debug tools

Audience:

The course aims for a target audience, administrators working in a Microsoft-based environment, who are looking to create reusable units of automation, enable business processes automation, and help less-technically sound users to complete administrative tasks.

Prerequisites:

Prior to starting this course, students must have:

- Experience regarding fundamental Windows administration
- Experience using Windows PowerShell to inquire and change system information
- Experience using Windows PowerShell to learn the commands and their

utilization

- Experience using WMI and/or CIM to inquire the system information

Prerequisites Courses:

Prior to taking this course, you may want to look into the following prerequisite courses:

- Automating Administration with Windows PowerShell (MS-10961)
- Advanced Automated Administration with Windows PowerShell (MS-10962)

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
- Parmeter 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-TypeData

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: Proxy Functions

This module explains how to create and use proxy functions.

Lessons

- A proxy example
- Creating the proxy base
- Modifying the proxy
- Adding or removing parameters

Lab : Designing a Tool

- Proxy functions

After completing this module, students will be able to:

- Describe the purpose of proxy functions
- Create a simple proxy function

Module 19: Working with XML Data

This module explains how to work with XML data in PowerShell.

Lessons

- Simple: CliXML
- Importing native XML
- ConvertTo-XML
- Creating native XML from scratch

Lab : Designing a Tool

- Working with XML

After completing this module, students will be able to:

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

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

Credly Badge:

Display your Completion Badge And Get The Recognition You Deserve.

Add a completion and readiness badge to your Linkedin profile, Facebook page, or Twitter account to validate your professional and technical expertise. With badges issued and validated by Credly, you can:

- Let anyone verify your completion and achievement by clicking on the badge
- Display your hard work and validate your expertise
- Display each badge's details about specific skills you developed.

Badges are issued by QuickStart and verified through Credly.

Find Out More or See List Of Badges