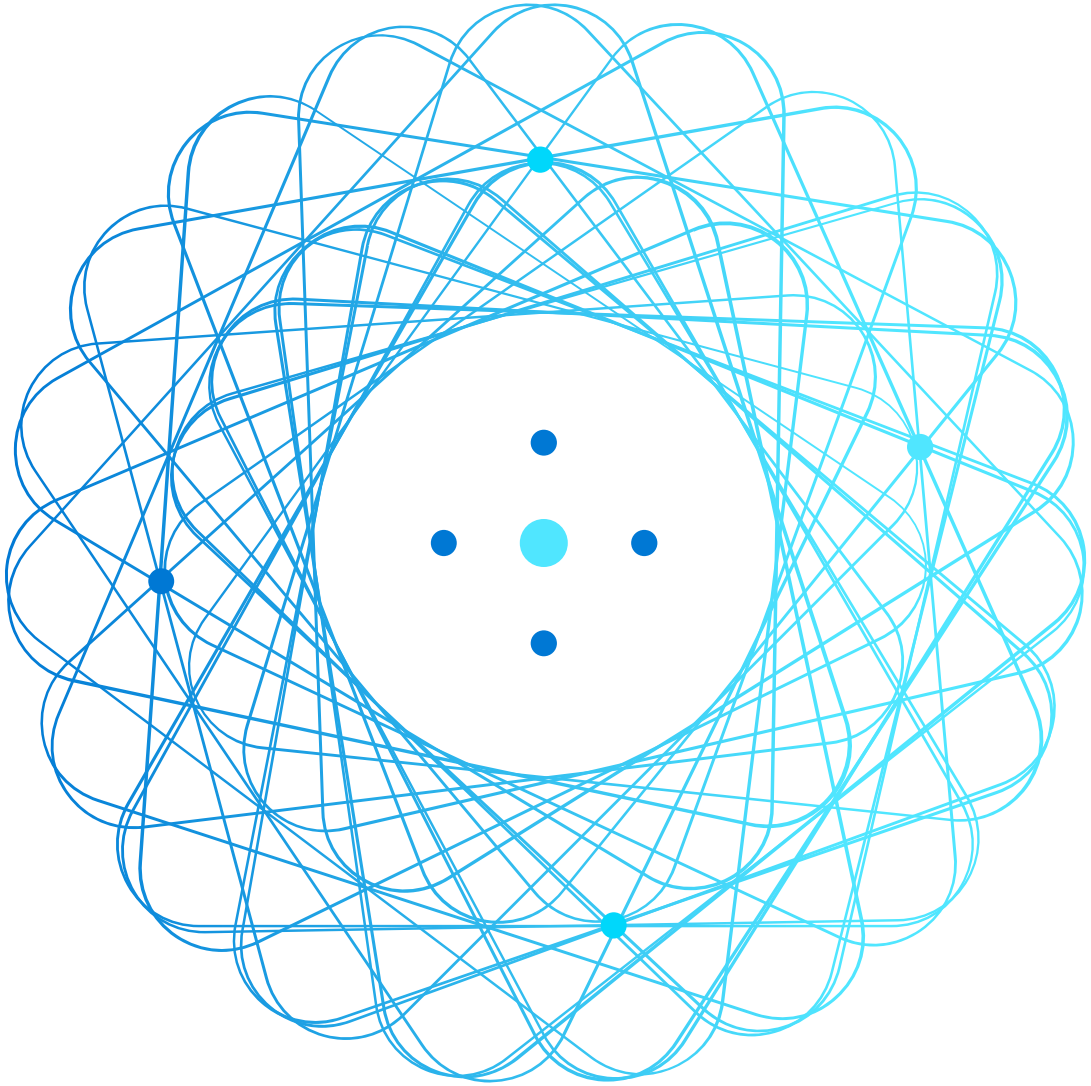
Microsoft Azure

AZ-040 Automating Administration with PowerShell



Course outline

Learning Path 1: Getting started with Windows PowerShell
Learning Path 2: Windows PowerShell for local systems administration
Learning Path 3: Working with the Windows PowerShell pipeline
Learning Path 4: Using PSProviders and PSDrives
Learning Path 5: Querying management information by using CIM and WMI
Learning Path 6: Working with variables, arrays, and hash tables
Learning Path 7: Windows PowerShell scripting
Learning Path 8: Administering remote computers with Windows PowerShell
Learning Path 9: Managing Azure resources with PowerShell
Learning Path 10: Managing Microsoft 365 services with PowerShell
Learning Path 11: Using background jobs and scheduled jobs

Learning Path 6: Working with variables, arrays, and hash tables

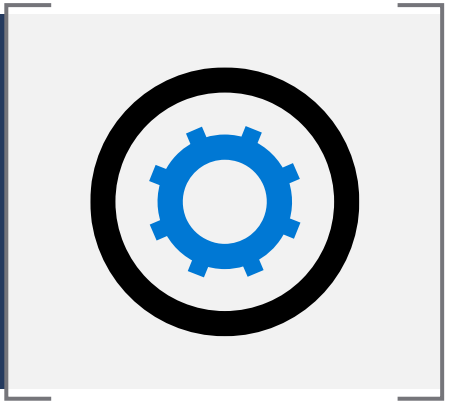
Overview

As an essential component of scripts, variables enable you to accomplish complex tasks that you can't complete using a single command. This Learning Path explains how to work with variables, arrays, and hash tables as steps in learning how to create Windows PowerShell scripts.

Modules:

- Use variables in Window PowerShell scripts
- Manipulate variables in Window PowerShell scripts
- Work with arrays and hash tables in Window PowerShell scripts

Section break 1



Use variables in Window PowerShell scripts

Module overview

This Module explains some rules for using variables, which help ensure that the data you store in variables is in the correct format and easily accessible. It's important to name variables appropriately and to assign them the correct data type.

Units:

- What are variables?
- Variable naming
- Assigning a value to a variable
- Variable types
- Demonstration: Assigning a variable type

What are variables?

- A variable stores a value or object in memory.
- Some things you can do with a variable:
  - Store the name of a log file that you write data to multiple times.
  - Derive and store an email address based on the name of a user account.
  - Calculate and store the date representing the beginning of the most recent 30-day period, to identify whether computer accounts have authenticated during that time.
- You can access object properties stored in a variable.
- Variables and their values can be reviewed in the PSDrive named **Variable**.

Variable naming

- Variable names:
  - Should be easily understandable.
  - Can contain spaces if enclosed in braces.
  - Should contain only alphanumeric characters.
  - Are not case-sensitive.
- A common convention for variable names uses capital letters to separate words:
  - **$LogFile**
  - **$StartDate**
  - **$ipAddress**

Assigning a value to a variable

- Use standard mathematical operators when working with variables.
- To assign a value to a variable, use the **=** operator:
  - **$num1 = 5**
  - **$logFile = "C:\Logs\Log.txt"**
  - **$user = Get-ADUser Administrator**
  - **$service = Get-Service W32Time**
- To display the value of a variable, enter the variable name or use **Write-Host**:
  - **$num1**
  - **Write-Host "The log location is $logfile"**
- To clear a variable, use **$null**:
  - **$num1 = $null**

$num1 = ''

$abc

Variable types

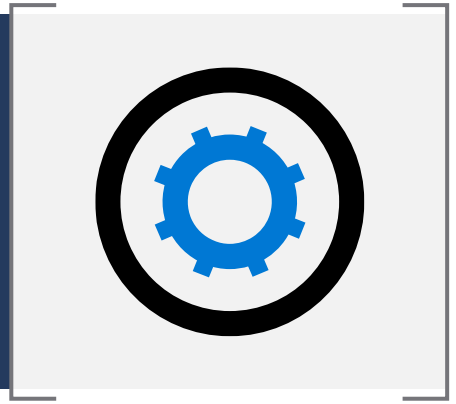
- The variable type determines the data that can be stored in it:
  - String. Stores text, including special characters.
  - Int32. Stores integers without decimals.
  - Double. Stores numbers with decimals.
  - DateTime. Stores date and time.
  - Bool. Stores true or false.
- Windows PowerShell can automatically assign the type based on a value.
- Specify the type if data is going to be ambiguous.

Demonstration: Assigning a variable type

In this demonstration, you will learn how to:

1. Set the value for variables.
2. Display the contents of a variable.
3. Review the properties of a variable.
4. Review variables in memory.
5. Use the **GetType** method to review variable types.
6. Force variable types when assigning values.
7. Add variables together.

Section break 2



Manipulate variables

Module overview

In this Module, you'll learn how to identify the properties and methods for use in your scripts and manipulate the contents of variables. Each variable type has a unique set of properties that you can access and each variable has a unique set of methods that you can use to manipulate it.

Units:

- Identifying methods and properties
- Working with strings
- Demonstration: Manipulating strings
- Working with dates
- Demonstration: Manipulating dates

Identifying methods and properties

- A variable's properties and methods are based on the variable type.
- To identify a variable's properties and methods, use:
  - **Get-Member**
  - Tab completion
- Documentation for properties and methods is available in the Microsoft .NET Framework Class Library.

Working with strings

- The only property available for strings is Length.
- Some commonly used methods for strings are:
  - **Contains(string value)**
  - **Insert(int startindex,string value)**
  - **Remove(int startindex,int count)**
  - **Replace(string value,string value)**
  - **Split(char separator)**
  - **ToLower()**
  - **ToUpper()**

Demonstration: Manipulating strings

In this demonstration, you will learn how to:

1. Use the **Contains** method.
2. Use the **Insert** method.
3. Use the **Replace** method.
4. Use the **Split** method.
5. Use the **ToUpper** method.
6. Use the **ToLower** method.

Working with dates

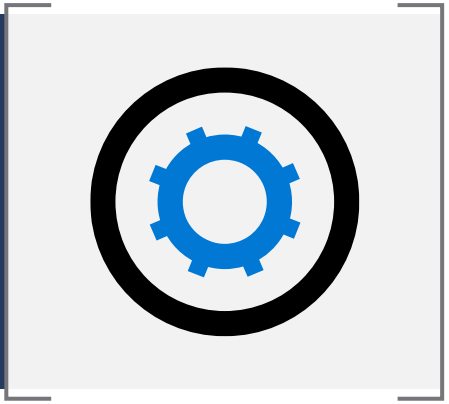
- Commonly used DateTime properties:
  - **Hour**
  - **Date**
  - **Minute**
  - **DayOfWeek**
  - **Second**
  - **Month**
- Commonly used DateTime methods:
  - **AddDays(double value)**
  - **ToLongDateString()**
  - **AddHours(double value)**
  - **ToShortDateString()**
  - **AddMinutes(double value)**
  - **ToLongTimeString()**
  - **AddMonths(int value)**
  - **ToShortTimeString()**

Demonstration: Manipulating dates

In this demonstration, you will learn how to:

1. Put output from **Get-Date** into a variable.
2. Review date properties.
3. Use date methods.

Section break 3



Manipulate arrays and hash tables

Module overview

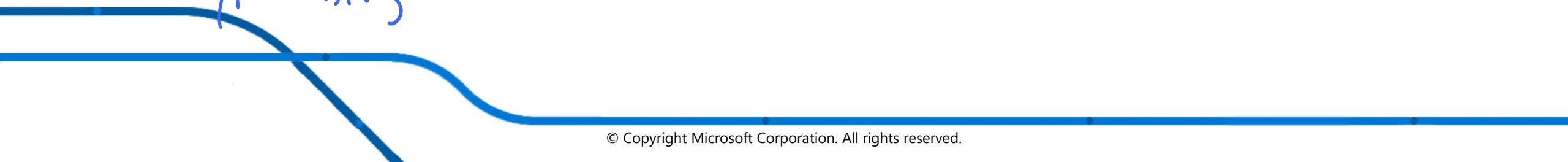
This Module covers arrays and hash tables, which you can use to store data that's more complex than simple variables. You can also use them to complete more complex tasks with your scripts.

Units:

- What is an array?
- Working with arrays
- Working with array lists
- Demonstration: Manipulating arrays and array lists
- What is a hash table?
- Working with hash tables
- Demonstration: Manipulating hash tables

What is an array?

- An array contains multiple values or objects.
- Values can be assigned by:
  - Providing a list:
    $computers = "LON-DC1","LON-SRV1","LON-CL1"
  - Using command output:
    $users = Get-ADUser -Filter *
- You can create an empty array:
  $newUsers = @()
- You can force an array to be created when adding only a single item:
  [Array]$computers = "LON-DC1"



Working with arrays

- To display all items in an array:

  $users

- To display specific items in an array by using an index number:

  $users[0]

- To add items to an array:

  $users = $users + $user1

  $users += $user1 ✓

  ~= geht nicht

- To pipe the array to **Get-Member** to identify what you can do with the array contents:

  $files | Get-Member

Working with array lists

- Arrays are fixed-size, which limits performance and makes removing items difficult.
- Arraylists are variable-sized.
- To create an arraylist:
  - **$computers = New-Object System.Collections.ArrayList**
  - **[System.Collections.ArrayList]$computers = "LON-DC1","LON-SRV1"**
- To add or remove items from an arraylist:
  - **$computers.Add("LON-SVR2")**
  - **$computers.Remove("LON-CL1")**
  - **$computers.RemoveAt(1)**

nicht verwenden

+=
+=
-=

Demonstration: Manipulating arrays and array lists

In this demonstration, you will learn how to:

1. Create an array.
2. Review the contents of an array.
3. Add items to an array.
4. Create an array list.
5. Review the contents of an array list.
6. Remove an item from an array list.

What is a hash table?

@{ }

- A *hash table* is a list of names and values.
- To refer to a value in the hash table, you provide the key:
  - **$servers.'LON-DC1'**
  - **$servers['LON-DC1']**

Array
ArrayList
@( )

New-Object

Key Name	IP address
LON-DC1	172.16.0.10
LON-SRV1	172.16.0.11
LON-SRV2	172.16.0.12

Key Value Pair

Working with hash tables

@{ n='size in GB' ; e={$_.size / 1GB} } Calculated prop.

- To define a hash table:

  $servers = @{"LON-DC1"="172.16.0.10"; "LON-SRV1"="172.16.0.11"}

- To add an item to a hash table:

  $servers.Add("LON-SRV2","172.16.0.12")

- To remove an item from a hash table:

  $servers.Remove("LON-DC1")

- To update a value for an item in a hash table:

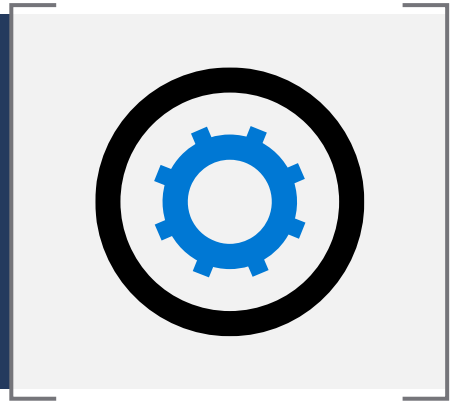
  $servers.'LON-SRV2'="172.16.0.100"

Demonstration: Manipulating hash tables

In this demonstration, you will learn how to:

1. Create a hash table.
2. Review the contents of a hash table.
3. Add an item to a hash table.
4. Remove an item from a hash table.
5. Create a hash table for a calculated property.

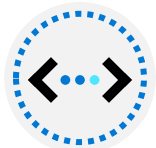
Section break 4



Lab: Using variables, arrays, and hash tables in PowerShell

Exercise 1: Working with variable types

Exercise 2: Using arrays



Exercise 3: Using hash tables

Sign-in information for the exercises:

Virtual machines: **AZ-040T00A-LON-DC1** and **AZ-040T00A-LON-CL1**

Username: **Adatum\Administrator**

Password: **Pa55w.rd**

Lab scenario

You’re preparing to create scripts to automate server administration in your organization. Before you begin creating scripts, you want to practice working with variables, arrays, and hash tables.

Lab-review questions

In the “Using arrays” exercise, why did user objects in **$mktgUsers** not update with the new department name?

How do you reference individual items in a hash table?

In Exercise 1, you replaced **C:** with **D:** in the variable **$logPath**. Why is it better to include a colon rather than simply replace **C** with **D**?

Lab-review answers

In the “Using arrays” exercise, why did user objects in **$mktgUsers** not update with the new department name?

When you query data and place it in a variable, the data in the variable is a snapshot in time. If you want the data in the variable to be updated, you need to query the data again.

How do you reference individual items in a hash table?

You reference hash table items based on the key name rather than an index number.

In Exercise 1, you replaced **C:** with **D:** in the variable **$logPath**. Why is it better to include colon than simply replace **C** with **D**?

In a file path, the colon character only appears once. By including the colon as part of the text that is replaced, you ensure that only the drive letter is updated. If you did not include the colon and if the path contained the character C at any other location, it would be replaced as well.

References

[about_Assignment_Operators]

[System Namespace]