Creating an object-oriented PowerShell script involves defining classes, methods, and properties while ensuring clear documentation using comment-based help.

# Define a PowerShell class

class Person {

# Properties

[string]$FirstName

[string]$LastName

[int]$Age

# Constructor

Person([string]$firstName, [string]$lastName, [int]$age) {

$this.FirstName = $firstName

$this.LastName = $lastName

$this.Age = $age

}

# Method to get full name

[string] GetFullName() {

return "$($this.FirstName) $($this.LastName)"

}

# Method to check if the person is an adult

[bool] IsAdult() {

return $this.Age -ge 18

}

}

# Instantiate the object

$person1 = [Person]::new("John", "Doe", 30)

# Use the methods

$fullName = $person1.GetFullName()

$isAdult = $person1.IsAdult()

Write-Host "Full Name: $fullName"

Write-Host "Is Adult: $isAdult"

# Define a PowerShell class with documentation

class Person {

<#

.SYNOPSIS

Represents a person with a first name, last name, and age.

.DESCRIPTION

This class allows you to create a person object, get their full name, and check if they are an adult.

.PROPERTY [string]$FirstName

The first name of the person.

.PROPERTY [string]$LastName

The last name of the person.

.PROPERTY [int]$Age

The age of the person.

.METHOD GetFullName

Returns the full name of the person in "First Last" format.

.METHOD IsAdult

Checks if the person is 18 or older, indicating adulthood.

.EXAMPLE

$person1 = [Person]::new("John", "Doe", 30)

$fullName = $person1.GetFullName()

Write-Host $fullName

.EXAMPLE

$isAdult = $person1.IsAdult()

Write-Host $isAdult

.NOTES

Author: Your Name

Date: 2025-03-05

#>

# Properties

[string]$FirstName

[string]$LastName

[int]$Age

# Constructor

Person([string]$firstName, [string]$lastName, [int]$age) {

$this.FirstName = $firstName

$this.LastName = $lastName

$this.Age = $age

}

# Method to get full name

[string] GetFullName() {

return "$($this.FirstName) $($this.LastName)"

}

# Method to check if the person is an adult

[bool] IsAdult() {

return $this.Age -ge 18

}

}

Class documentation

Let's create a simple **Library** class with **Book** objects. The script will include object-oriented concepts like classes, properties, methods, and constructor functions, along with documentation.

# Class to represent a Book

class Book {

<#

.SYNOPSIS

Represents a book with a title, author, and ISBN.

.DESCRIPTION

This class defines a book with properties for the title, author, and ISBN.

It includes methods to get a brief description of the book and to check if the book is a best-seller based on a rating.

.PROPERTY [string]$Title

The title of the book.

.PROPERTY [string]$Author

The author of the book.

.PROPERTY [string]$ISBN

The ISBN number of the book.

.PROPERTY [double]$Rating

The rating of the book (on a scale of 1 to 5).

.METHOD GetBookDescription

Returns a string describing the book's title, author, and ISBN.

.METHOD IsBestSeller

Checks if the book is a best-seller (rating greater than or equal to 4).

.EXAMPLE

$book = [Book]::new("The Great Gatsby", "F. Scott Fitzgerald", "9780743273565", 4.5)

$description = $book.GetBookDescription()

Write-Host $description

.EXAMPLE

$isBestSeller = $book.IsBestSeller()

Write-Host "Is Best Seller: $isBestSeller"

.NOTES

Author: Your Name

Date: 2025-03-05

#>

# Properties of the Book class

[string]$Title

[string]$Author

[string]$ISBN

[double]$Rating

# Constructor to initialize a new Book

Book([string]$title, [string]$author, [string]$isbn, [double]$rating) {

$this.Title = $title

$this.Author = $author

$this.ISBN = $isbn

$this.Rating = $rating

}

# Method to get a description of the book

[string] GetBookDescription() {

return "Title: $($this.Title), Author: $($this.Author), ISBN: $($this.ISBN)"

}

# Method to check if the book is a best-seller

[bool] IsBestSeller() {

return $this.Rating -ge 4.0

}

}

# Class to represent a Library

class Library {

<#

.SYNOPSIS

Represents a collection of books in a library.

.DESCRIPTION

This class allows you to create a library object and add books to it.

You can retrieve all books in the library and find the best-sellers.

.PROPERTY [array]$Books

A collection of Book objects in the library.

.METHOD AddBook

Adds a new book to the library collection.

.METHOD GetAllBooks

Returns a list of all books in the library.

.METHOD GetBestSellers

Returns a list of books that are considered best-sellers.

.EXAMPLE

$library = [Library]::new()

$book1 = [Book]::new("The Great Gatsby", "F. Scott Fitzgerald", "9780743273565", 4.5)

$library.AddBook($book1)

$bestSellers = $library.GetBestSellers()

.NOTES

Author: Your Name

Date: 2025-03-05

#>

# Properties of the Library class

[array]$Books = @()

# Constructor to initialize an empty library

Library() {}

# Method to add a book to the library

[void] AddBook([Book]$book) {

$this.Books += $book

}

# Method to get a list of all books in the library

[array] GetAllBooks() {

return $this.Books

}

# Method to get a list of best-sellers from the library

[array] GetBestSellers() {

return $this.Books | Where-Object { $\_.IsBestSeller() }

}

}

# Example of creating a Library and adding Books to it

$library = [Library]::new()

# Create some books

$book1 = [Book]::new("The Great Gatsby", "F. Scott Fitzgerald", "9780743273565", 4.5)

$book2 = [Book]::new("1984", "George Orwell", "9780451524935", 3.9)

$book3 = [Book]::new("Moby-Dick", "Herman Melville", "9781851244422", 4.2)

# Add books to the library

$library.AddBook($book1)

$library.AddBook($book2)

$library.AddBook($book3)

# Get all books in the library

$allBooks = $library.GetAllBooks()

$allBooks | ForEach-Object { Write-Host ($\_.GetBookDescription()) }

# Get best-sellers in the library

$bestSellers = $library.GetBestSellers()

Write-Host "Best-Sellers:"

$bestSellers | ForEach-Object { Write-Host ($\_.GetBookDescription()) }