Here's how to catch return codes within PowerShell:

1. Using $LASTEXITCODE

- This automatic variable stores the exit code of the last external command or executable that was run.
- It's most useful when you're calling non-PowerShell programs (like .exe files or batch scripts).
- A value of 0 typically indicates success, while non-zero values indicate failure or specific error conditions.

```
   Start-Process "my_program.exe" -Wait
   $exitCode = $LASTEXITCODE
   if ($exitCode -ne 0) {
       Write-Host "Error: my_program.exe failed with exit code
$($exitCode)"
   }
```

Important: $LASTEXITCODE only captures the exit code of external commands. PowerShell cmdlets themselves do not typically set this variable.

2. Using try...catch blocks

- This is the primary method for handling errors within PowerShell scripts.
- The try block encloses code that might generate an error.
- The catch block specifies what action to take when an error occurs.

```
   try {
       # Code that might throw an error
       Get-Content "nonexistent_file.txt"
   }
   catch {
       Write-Host "An error occurred: $($_.Exception.Message)"
   }
```

- **ErrorActionPreference:** Controls how PowerShell handles errors. Setting it to "Stop" will make non-terminating errors act like terminating errors, allowing them to be caught by try...catch.

```
   $ErrorActionPreference = "Stop"
   try {
       Get-ChildItem "C:\nonexistent_folder"
```

```
}
catch {
   Write-Host "Error: $($_.Exception.Message)"
}
```

3. Returning Custom Exit Codes

- You can set a custom exit code for your PowerShell script using the exit statement.
- The exit statement takes an integer as an argument, which will be the exit code returned by the script.
- You can use $LASTEXITCODE to return the exit code of the last command in the script.

```
if ($someCondition -eq $true) {
    exit 1 # Indicate failure
}
else {
    exit 0 # Indicate success
}
```

Key Points

- PowerShell's error handling is object-oriented. Errors are represented as objects, not just numeric codes.
- Use $ErrorActionPreference to control how errors are handled.
- Use try...catch for robust error handling within your scripts.
- Use $LASTEXITCODE to capture exit codes from external executables.
- Use the exit statement to return custom exit codes from your PowerShell scripts.