

# PowerShell Do While Loop Explained

---

 [lazyadmin.nl/powershell/do-while-loop](https://lazyadmin.nl/powershell/do-while-loop)

September 8, 2024

Loops are one of the basic functions in any programming or scripting language. One of those loops is the Do While loop, which executes a piece of code while the condition is true.

In PowerShell, we can use a While loop and a Do-While loop. They may look the same, but they do have a difference that is important to understand.

In this article, I will explain the difference between the two loops and show how to use the Do-While Loop in your script.

## PowerShell Do While Loop

---

When writing a script you sometimes need to execute a piece of code an x amount of time until a condition is met. We can do this with a loop which is one of the control flow statements that we can use.

Now in PowerShell, we actually have three different Loop types that we can use:

- While Loop
- Do While Loop
- Do Until Loop

The While loop will only execute the code inside the loop if the condition is true. The Do While and Do Until, however, will execute the code in the loop at least once. This is an important difference to keep in mind.

The difference between Do-While and Do-Until is that the first will execute the code in the loop as long as the condition is true. Whereas Do-Until we execute the code as long as the condition is false.

Also good to know is that PowerShell won't continue with the execution of your script until the loop is completed.

## Using the Do-While Loop

---

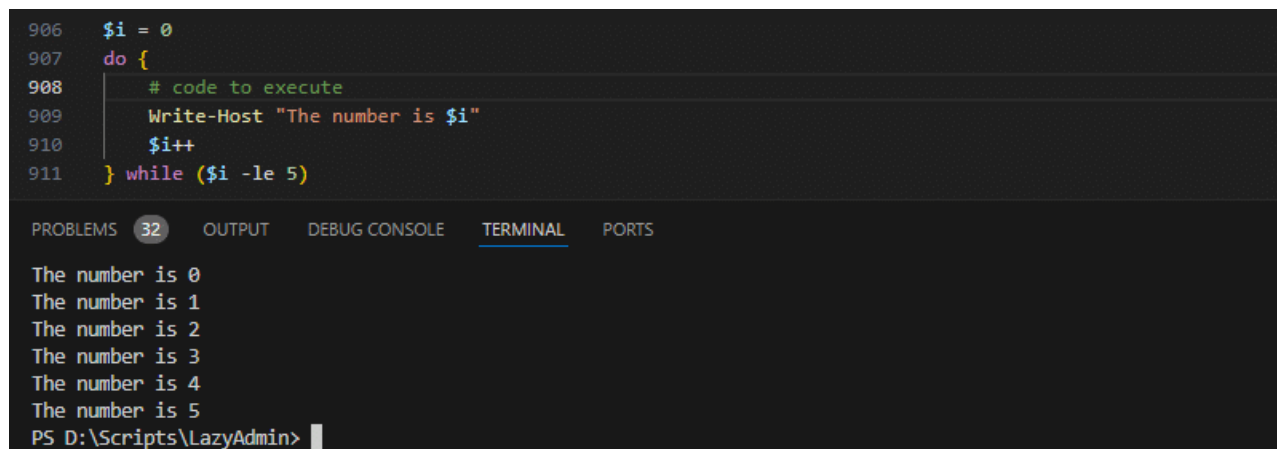
To use the Do-While loop inside our scripts, we will need to define the code that is going to be executed in the Do part and add a condition in the While part.

Let's take a look at a simple example. The code below will write the current number (**\$i**), and add 1 to it, as long as **\$i** is less than 5:

```
$i = 0
```

```
do {
# code to execute
Write-Host "The number is $i"
$i++
} while ($i -le 5)
```

Try out the example above, simply copy and paste it into Windows Terminal. You will see that the code is being executed while the condition is true.



```
906 $i = 0
907 do {
908     # code to execute
909     Write-Host "The number is $i"
910     $i++
911 } while ($i -le 5)

PROBLEMS 32 OUTPUT DEBUG CONSOLE TERMINAL PORTS

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
PS D:\Scripts\LazyAdmin>
```

## Continue and Break

When working with loops in PowerShell, we can use **Continue** and **Break** to control, or interfere with, the flow of the loop. When the Break statement is encountered inside a loop, then the script jumps out of the loop and continues below the loop with the next line of code. This is done regardless of the condition that is specified in the While statement.

Continue, on the other hand, will stop processing the current item in the loop, and jump to the next iteration. The continue statement will re-evaluate the while condition. So if the condition is met before the continue statement, then the Do-While loop is finished.

Let's take a look at a Continue example inside a loop. The code below will get all the files from the given directory. If we find a hidden file, then the Continue statement is triggered, making it go to the next one:

```
$files = Get-ChildItem "C:\Temp"
$i = 0
Do {
$file = $files[$i]
if ($file.Attributes -match "Hidden") {
$i++
continue # Skip hidden files
}
Write-Host "Processing file: $file.Name"
$i++
} While ($i -lt $files.Count)
```

The Break statement can for example be used to check if we still want to continue with the loop. The example below will ask the user if they want to continue, and if the answer is no, we will exit the loop:

```
Do {
$response = Read-Host "Do you want to continue? (yes/no)"
if ($response -eq "no") {
Write-Host "Exiting loop."
break # Exit the loop if the user chooses "no"
}
Write-Host "Continuing..."
} While ($response -ne "no")
Write-Host "Loop ended."
```

## Do-While Examples

---

Personally, I find it easier to learn programming or scripting with examples. So I have created a couple of realistic Do-While examples to get you started.

The first one we are going to take a look at is a small script that will wait until the specified service is running. In this example, we are going to wait until the Windows Update Service is running. Keep in mind that PowerShell won't execute the code below the Do While loop until the While condition is met:

```
$serviceName = "wuauserv" # Windows Update service
$status = (Get-Service -Name $serviceName).Status
Do {
Write-Host "Waiting for the service '$serviceName' to start..."
Start-Sleep -Seconds 5
$status = (Get-Service -Name $serviceName).Status
} While ($status -ne "Running")
Write-Host "The service '$serviceName' is now running."
```

For the example above, you would actually start with an If-Else statement, checking if the service is running at all or not. If the service is not running, you can start it, and then use the code above to wait until it's up and running.

## Validating User Input

---

Another common use case is when you need to validate a user input in your script. We can use the Do-While loop to keep asking for new input until it matches our defined condition. For example, we ask for an email address, and will check if it matches the email address pattern before we continue:

```
$emailPattern = "^[w\.-]+@[w\.-]+\.\w+$"
$email = ""
Do {
```

```
$email = Read-Host "Enter a valid email address"
```

```
} While ($email -notmatch $emailPattern)
```

```
Write-Host "Valid email entered: $email"
```

In the While condition, we are not limited to one condition. We can have multiple conditions and combine them using the **-And** or **-Or** operators. So for example, we can set a max under the number of guesses as user can make:

```
$maxAttempts = 3
```

```
$attempts = 0
```

```
do {
```

```
$input = Read-Host "Enter a number between 1 and 10"
```

```
$attempts++
```

```
} while (($input -lt 1 -or $input -gt 10) -and ($attempts -ne $maxAttempts))
```

## Infinite Loop

---

If you set the condition to true, then you can create an infinite loop in PowerShell. The code inside the loop will be executed until the **\$condition** inside the loop is true. Then we can use the **Break** statement to exit the loop.

```
Do {
```

```
# Do stuff forever
```

```
Write-Host 'looping'
```

```
# Use Break to exit the loop
```

```
If ($condition) {
```

```
Break
```

```
}
```

```
} While ($true)
```

## Download File Attempt

---

Do-while loops are also a great option when you want to try something. In the example below, we will try to download a file. Using a Try-Catch block, we can check if the download attempt was successful or not.

We will keep trying to download the file until it's successful or the maximum attempts are reached.

```
$url = "https://example.com/file.zip"
```

```
$destination = "C:\Downloads\file.zip"
```

```
$attempt = 0
```

```
$maxAttempts = 5
```

```
Do {
```

```
$attempt++
```

```
Write-Host "Attempt $attempt: Downloading file..."
```

```
try {
```

```
Invoke-WebRequest -Uri $url -OutFile $destination
```

```
$success = $true
} catch {
Write-Host "Download failed, retrying..."
Start-Sleep -Seconds 5
$success = $false
}
} While (-not $success -and $attempt -lt $maxAttempts)
if ($success) {
Write-Host "File downloaded successfully!"
} else {
Write-Host "Failed to download file after $maxAttempts attempts."
}
```

## Wrapping Up

---

Do-While loops are a great control flow statement to use in PowerShell. Keep in mind that the code inside the Do loop is always executed at least one time. If you only want to execute the code when the condition is met, you will need to use the normal While loop.

If you want to learn more about PowerShell, then make sure you check out the [other PowerShell articles](#) and of course the [Cheat Sheet](#). And if you have any questions, just drop a comment below.

Did you **Liked** this **Article**?

Get the latest articles like this **in your mailbox**  
or share this article

I hate spam to, so you can unsubscribe at any time.