

PowerShell

O PODER HACKER E SEUS TENTÁCULOS



GENIVAN SOUZA

01

INTRODUÇÃO



Introdução

Programação com PowerShell

Você já se perguntou como os programas de computador funcionam? A chave está na lógica de programação. Este ebook é um guia prático e simples para ajudá-lo a dominar a lógica de programação usando PowerShell, uma linguagem poderosa e fácil de aprender.

O Que é Lógica de Programação?

A lógica de programação é a base de qualquer software. É o processo de planejar e ordenar instruções para que um computador execute tarefas específicas. Aprender lógica de programação ajuda a resolver problemas de maneira estruturada e eficiente.

Por Que PowerShell?

PowerShell é uma linguagem de script da Microsoft que permite automatizar tarefas e gerenciar sistemas. É amplamente usada por administradores de sistemas e desenvolvedores pela sua simplicidade e poder. Neste ebook, você aprenderá os conceitos fundamentais de programação com exemplos práticos em PowerShell.

Prepare-se para aprender e aplicar a lógica de programação de uma maneira divertida e eficaz com PowerShell. Vamos juntos nessa jornada rumo ao domínio da programação!

02

COMO ABRIR O POWERSHELL

Aprender a abrir o PowerShell no Sistema Operacional Windows é o primeiro passo essencial para explorar suas poderosas capacidades.

Como Abrir o PowerShell

Alguns Métodos para Abrir o PowerShell

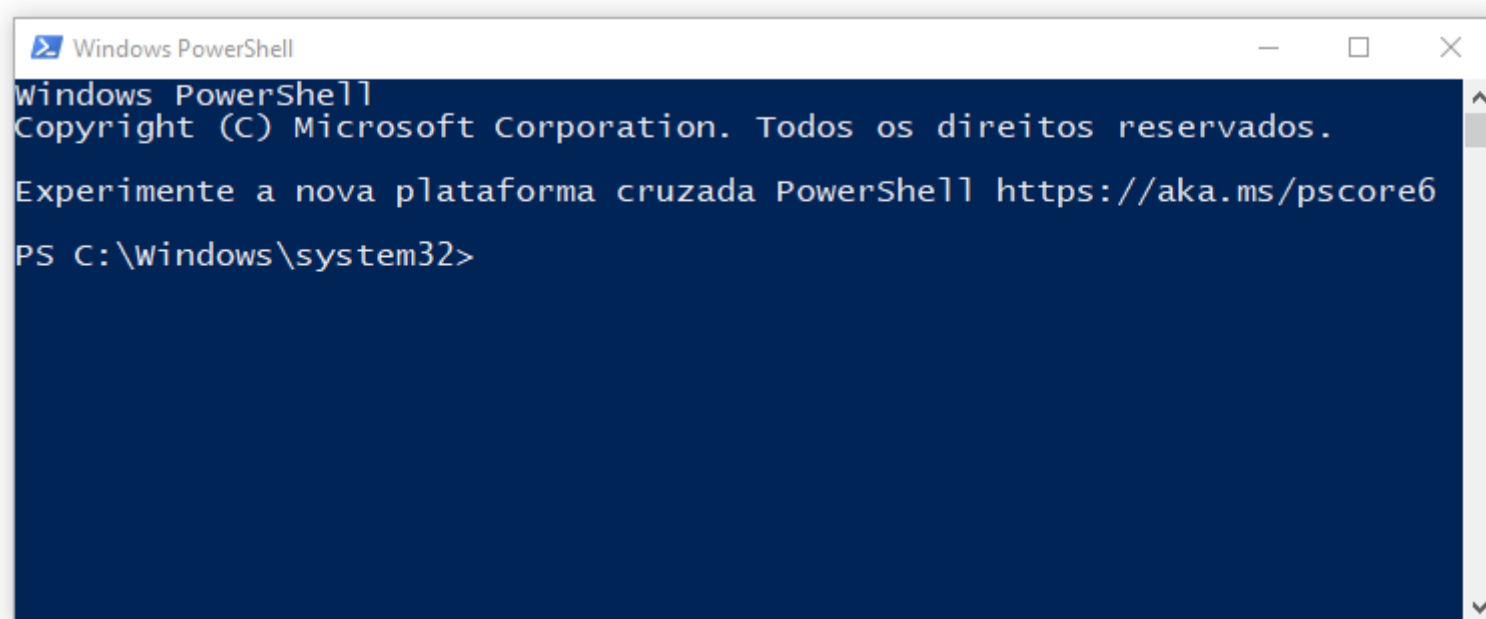
Método 1: Menu Iniciar

1. Clique no botão Iniciar no canto inferior esquerdo da tela.
2. Digite "PowerShell" na barra de pesquisa.
3. Clique em "Windows PowerShell" nos resultados da pesquisa.

Método 2: Executar (Run)

1. Pressione as teclas Win + R no teclado.
2. Digite "powershell" na janela que aparecer.
3. Pressione Enter.

Após executar um dos métodos acima, você deverá ver a tela do terminal do PowerShell aberto conforme a imagem a seguir.



03

EXPLORANDO VARIÁVEIS

Variáveis são como pequenas caixas onde guardamos informações que podemos usar e manipular ao longo do código. No PowerShell, elas são indicadas pelo símbolo \$ seguido do nome da variável.

Explorando Variáveis

Tipos de Variáveis Booleanas, Numéricas e Texto

Variáveis Booleanas: armazenam valores True (verdadeiro) ou False (falso).

```
$verdadeiro = $true
$falso = $false
Write-Output $verdadeiro # Saída: True
Write-Output $falso # Saída: False
```

Variáveis Numéricas: armazenam números, podendo ser inteiros ou decimais.

```
$inteiro = 42
$decimal = 3.14
Write-Output $inteiro # Saída: 42
Write-Output $decimal # Saída: 3.14
```

Variáveis de Texto (Strings): armazenam cadeias de caracteres, que são sequências de letras, números e símbolos.

```
$saudacao = "Olá, mundo!"
Write-Output $saudacao # Saída: Olá, mundo!
```

Explorando Variáveis

Tipos de Variáveis Arrays e Objetos

Arrays: são coleções de valores que podem ser de qualquer tipo, acessados por índices.

```
$array = @(1, 2, 3, 4, 5)
Write-Output $array[0] # Saída: 1
Write-Output $array[4] # Saída: 5
```

Objetos: são estruturas mais complexas que podem conter múltiplas propriedades e métodos. No PowerShell, podemos criar objetos personalizados usando `PSCustomObject`.

```
$pessoa = [PSCustomObject]@{
    Nome = "Ana"
    Idade = 30
    Cidade = "São Paulo"
}
Write-Output $pessoa.Nome # Saída: Ana
Write-Output $pessoa.Idade # Saída: 30
Write-Output $pessoa.Cidade # Saída: São Paulo
```


Explorando Variáveis

Usando Variáveis em Cenários Reais

Exemplo 1: Calculando a Média de Números

Vamos calcular a média de um conjunto de números usando um array.

```
$numeros = @(10, 20, 30, 40, 50)
$soma = 0

foreach ($numero in $numeros) {
    $soma = $soma + $numero
}

$media = $soma / $numeros.Count
Write-Output "A média é: $media" # Saída: A média é: 30
```

Exemplo 2: Verificando a Conexão com um Site

Podemos usar variáveis booleanas para armazenar o status de uma verificação de conexão com um site.

```
$url = "http://example.com"
$response = Test-Connection -ComputerName $url -Quiet

if ($response) {
    Write-Output "O site está online."
} else {
    Write-Output "O site está offline."
}
```

04

USANDO CONSTANTES

Constantes são valores que, uma vez definidos, não podem ser alterados durante a execução do script. Elas são úteis quando você tem valores que precisam permanecer fixos, garantindo que seu código seja mais seguro e previsível.

Usando Constantes

Como Definir uma Constante

Para definir uma constante no PowerShell, usamos o cmdlet New-Variable com o parâmetro -Option definido como Constant.

```
New-Variable -Name "TAXA_JUROS" -Value 0.05 -Option Constant
```

Se tentarmos alterar o valor de uma constante, o PowerShell gera um erro, como demonstrado abaixo:

```
$TAXA_JUROS = 0.1  
# Erro: Cannot overwrite variable TAXA_JUROS because it is read-only or constant.
```

Exemplo: Conversão de Unidades

```
# Definindo a constante  
New-Variable -Name "CONVERSAO_POLEGADAS_CM" -Value 2.54 -Option Constant  
  
# Variável para valor em polegadas  
$polegadas = 10  
  
# Cálculo da conversão  
$centimetros = $polegadas * $CONVERSAO_POLEGADAS_CM  
  
Write-Output "$polegadas polegadas é igual a $centimetros centímetros"
```

05

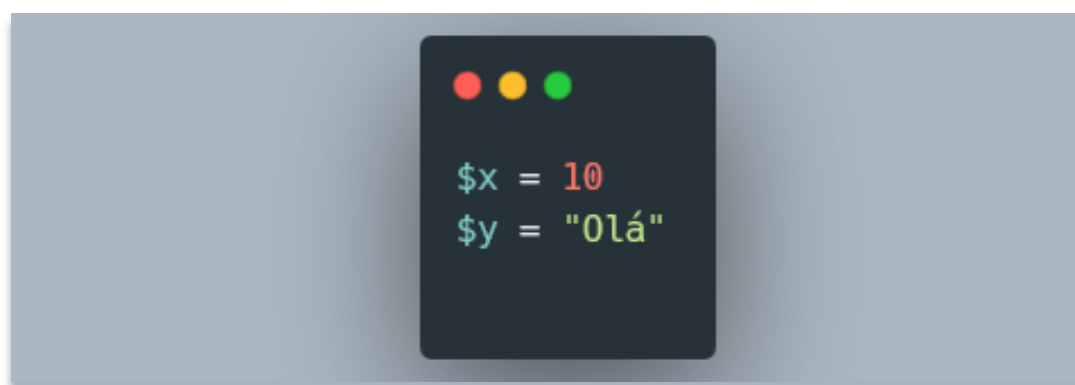
Explorando Operadores

Operadores são símbolos que informam ao PowerShell para realizar operações específicas em um ou mais operandos. Eles são fundamentais para manipular dados e controlar o fluxo de execução nos scripts.

Explorando Operadores

Operadores de Atribuição

Igual (=): os operadores de atribuição são usados para atribuir valores a variáveis.



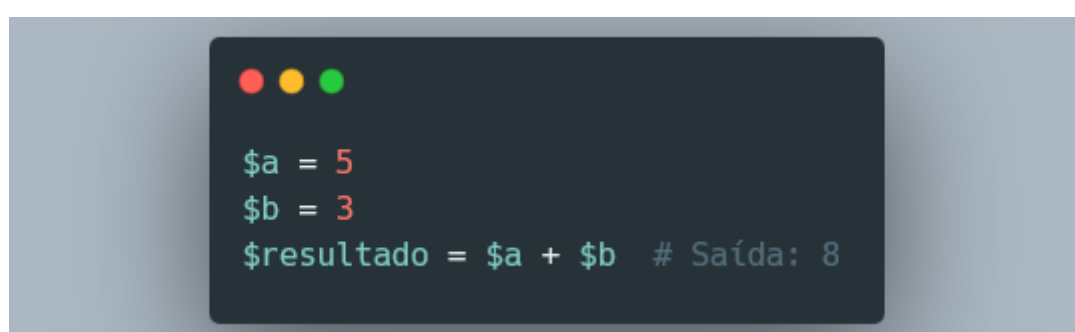
```
$x = 10
$y = "Olá"
```

Neste exemplo, \$x recebe o valor 10 e \$y recebe o valor "Olá".

Operadores Aritméticos

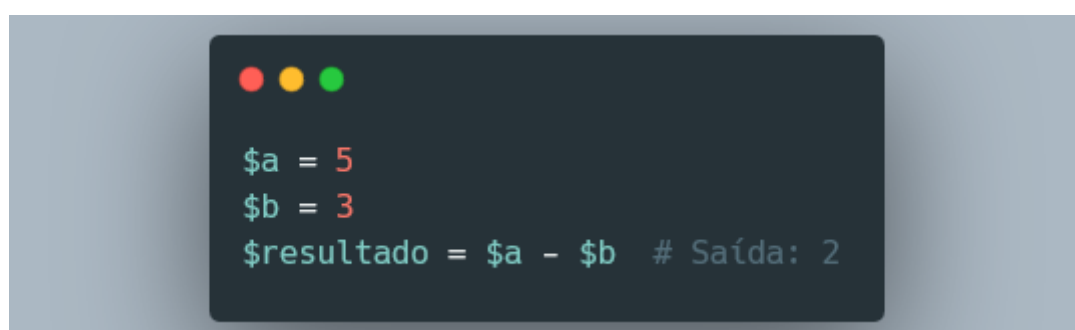
Os operadores aritméticos são usados para realizar operações matemáticas.

Soma (+):



```
$a = 5
$b = 3
$resultado = $a + $b # Saída: 8
```

Subtração (-):



```
$a = 5
$b = 3
$resultado = $a - $b # Saída: 2
```


Explorando Operadores

Operadores Aritméticos

Os operadores aritméticos são usados para realizar operações matemáticas.

Multiplicação (*):

```
$a = 5  
$b = 3  
$resultado = $a * $b # Saída: 15
```

Divisão (/):

```
$a = 6  
$b = 3  
$resultado = $a / $b # Saída: 2
```

Módulo (%): o operador de módulo retorna o resto da divisão.

```
$a = 5  
$b = 3  
$resultado = $a % $b # Saída: 2
```

Explorando Operadores

Operadores Relacionais

Os operadores relacionais são usados para comparar valores.

Igual a (-eq):

```
$a = 5  
$b = 5  
$comparacao = $a -eq $b # Saída: True
```

Diferente de (-ne):

```
$a = 5  
$b = 3  
$comparacao = $a -ne $b # Saída: True
```

Maior que (-gt)


```
$a = 5  
$b = 3  
$comparacao = $a -gt $b # Saída: True
```

Explorando Operadores

Operadores Relacionais

Os operadores relacionais são usados para comparar valores.

Menor que (-lt):



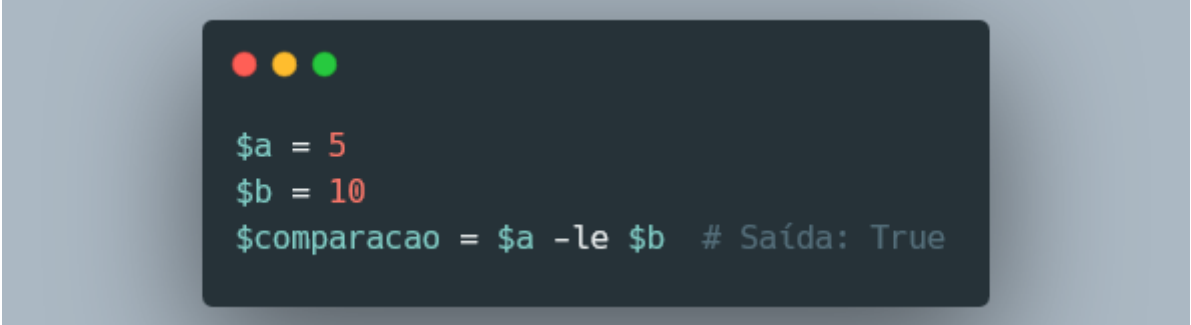
```
$a = 5  
$b = 10  
$comparacao = $a -lt $b # Saída: True
```

Maior ou igual a (-ge):



```
$a = 5  
$b = 5  
$comparacao = $a -ge $b # Saída: True
```

Menor ou igual a (-le)



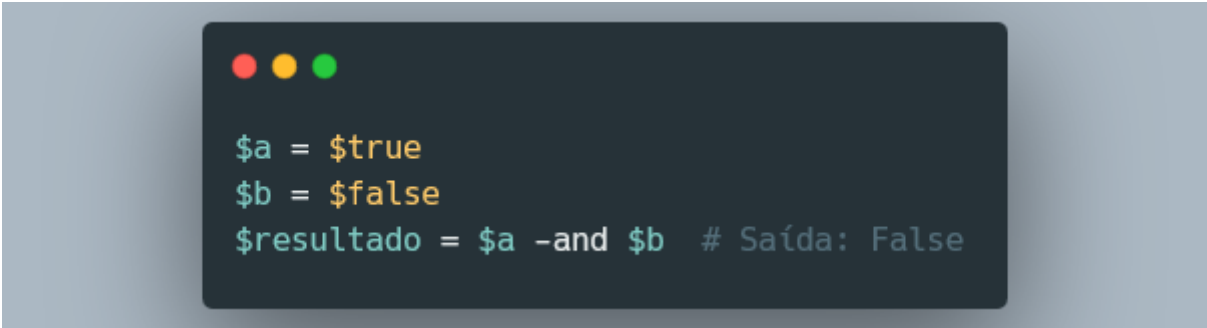
```
$a = 5  
$b = 10  
$comparacao = $a -le $b # Saída: True
```

Explorando Operadores

Operadores Lógicos

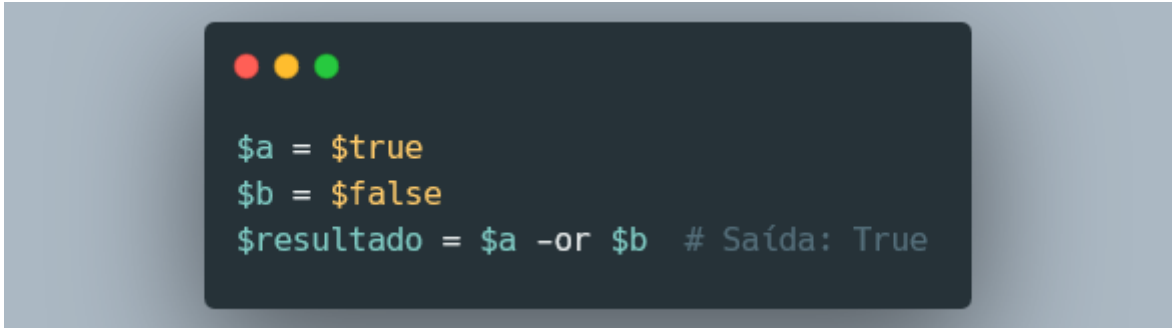
Os operadores lógicos são usados para combinar condições booleanas.

E Lógico (-and)



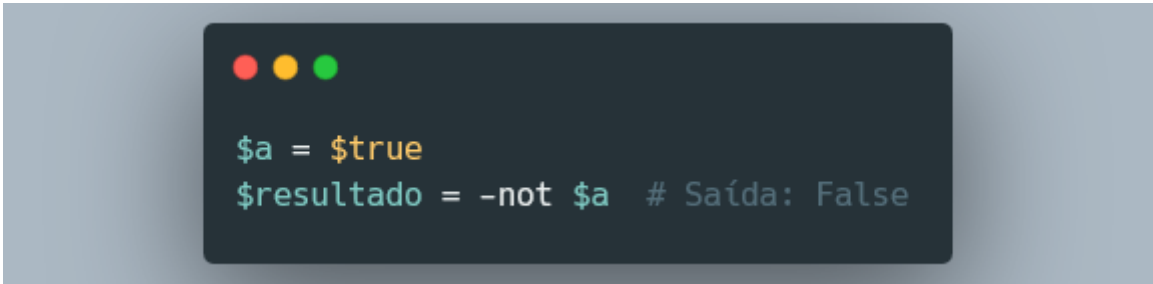
```
$a = $true  
$b = $false  
$resultado = $a -and $b # Saída: False
```

Ou Lógico (-or)



```
$a = $true  
$b = $false  
$resultado = $a -or $b # Saída: True
```

Não Lógico (-not)



```
$a = $true  
$resultado = -not $a # Saída: False
```

Explorando Operadores

Usando Operadores em Cenários Reais

Exemplo: Verificar Maioridade - vamos usar operadores relacionais e lógicos para verificar se uma pessoa é maior de idade.

```
$idade = 20
$maioridade = 18

if ($idade -ge $maioridade) {
    Write-Output "Você é maior de idade."
} else {
    Write-Output "Você é menor de idade."
}
```

Exemplo: Calculadora Simples - vamos criar uma calculadora simples usando operadores aritméticos.

```
$numero1 = 10
$numero2 = 5

$soma = $numero1 + $numero2
$subtracao = $numero1 - $numero2
$multiplicacao = $numero1 * $numero2
$divisao = $numero1 / $numero2

Write-Output "Soma: $soma"
Write-Output "Subtração: $subtracao"
Write-Output "Multiplicação: $multiplicacao"
Write-Output "Divisão: $divisao"
```


Agradecimientos



Obrigado por Ler Até Aqui

Esse Ebook foi gerado por IA, e diagramado por humano.
O passo a passo se encontra no meu Github

.

Esse conteúdo foi gerado com fins didáticos de construção, não foi realizado uma validação cuidadosa humana no conteúdo e pode conter erros gerados por uma IA.



<https://github.com/genivanss/prompts-recipe-to-create-a-ebook>