

PowerShell: Parámetros y argumentos en scripts y funciones

En el siguiente ejemplo se tratan los argumentos pasados sin más.

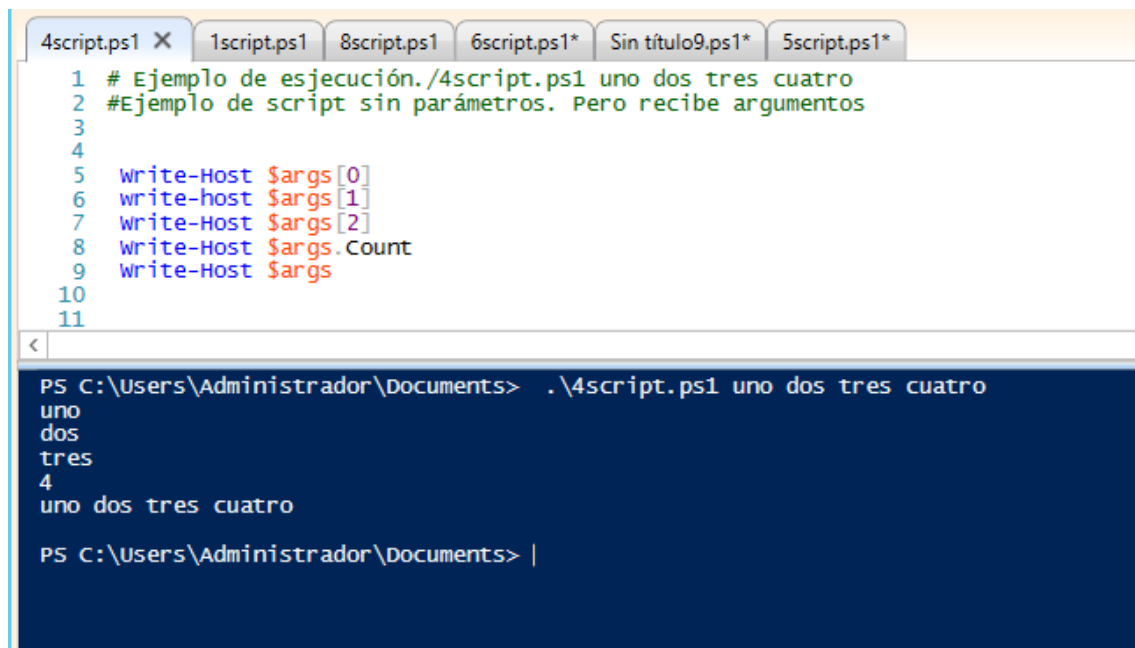
\$args[0]: es el primer argumento

\$args[1]: es el segundo argumento

\$args: es una cadena de texto que contiene todos los argumentos.

\$args.Count: Count es una propiedad de las clases de objetos, en este caso de la clase \$args que cuenta los objetos

cuatro: es el cuarto argumento



```
4script.ps1 X 1script.ps1 8script.ps1 6script.ps1* Sin título9.ps1* 5script.ps1*
1 # Ejemplo de ejecución./4script.ps1 uno dos tres cuatro
2 #Ejemplo de script sin parámetros. Pero recibe argumentos
3
4
5 Write-Host $args[0]
6 Write-Host $args[1]
7 Write-Host $args[2]
8 Write-Host $args.Count
9 Write-Host $args
10
11
```

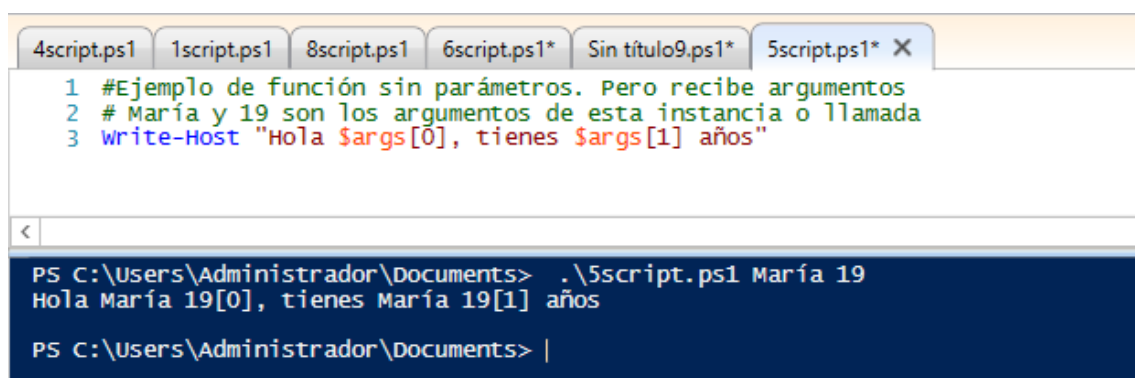
```
PS C:\Users\Administrador\Documents> .\4script.ps1 uno dos tres cuatro
uno
dos
tres
4
uno dos tres cuatro

PS C:\Users\Administrador\Documents> |
```

En el siguiente ejemplo se pretende utilizar los valores de los 2 argumentos enviados al script (María y 19)

Observa que falla, pues para **\$args[0]** se interpreta:

- \$args: como el string con todos los argumentos que se pasan al script
- [0]: como texto sin más.

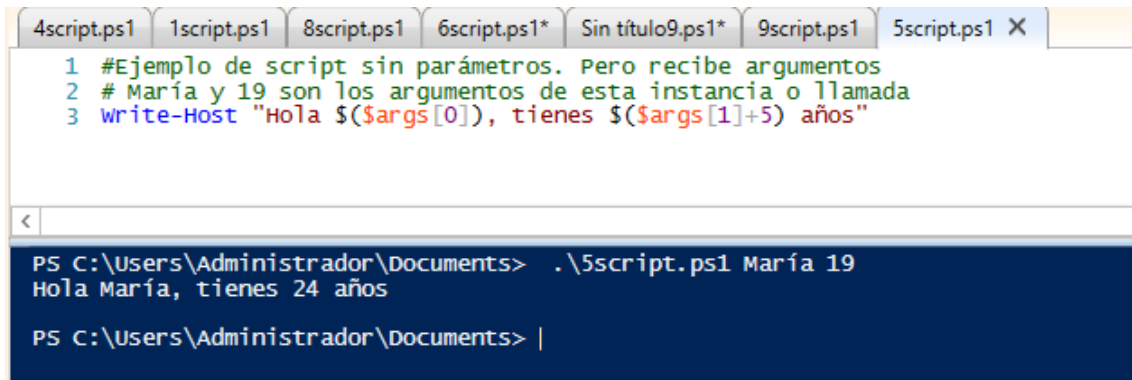


```
4script.ps1 1script.ps1 8script.ps1 6script.ps1* Sin título9.ps1* 5script.ps1* X
1 #Ejemplo de función sin parámetros. Pero recibe argumentos
2 # María y 19 son los argumentos de esta instancia o llamada
3 Write-Host "Hola $args[0], tienes $args[1] años"
```

```
PS C:\Users\Administrador\Documents> .\5script.ps1 María 19
Hola María 19[0], tienes María 19[1] años

PS C:\Users\Administrador\Documents> |
```

Para solucionarlo utilizamos: $\$(\text{aquí dentro la expresión})$ → Da como resultado la evaluación de la expresión.



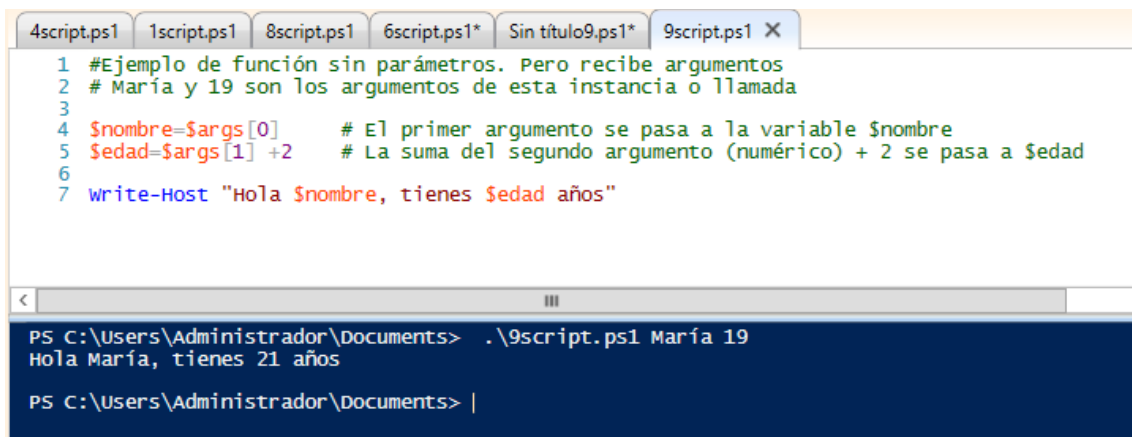
```
4script.ps1 1script.ps1 8script.ps1 6script.ps1* Sin título9.ps1* 9script.ps1 5script.ps1 X
1 #Ejemplo de script sin parámetros. Pero recibe argumentos
2 # María y 19 son los argumentos de esta instancia o llamada
3 Write-Host "Hola $($args[0]), tienes $($args[1]+5) años"

PS C:\Users\Administrador\Documents> .\5script.ps1 María 19
Hola María, tienes 24 años

PS C:\Users\Administrador\Documents> |
```

Ahora funciona como estaba previsto. Incluso en este caso aprovechamos a sumar 5 años a la edad de María.

Otra manera de solucionar sería almacenar en variables los argumentos pasados como parámetros. Así:

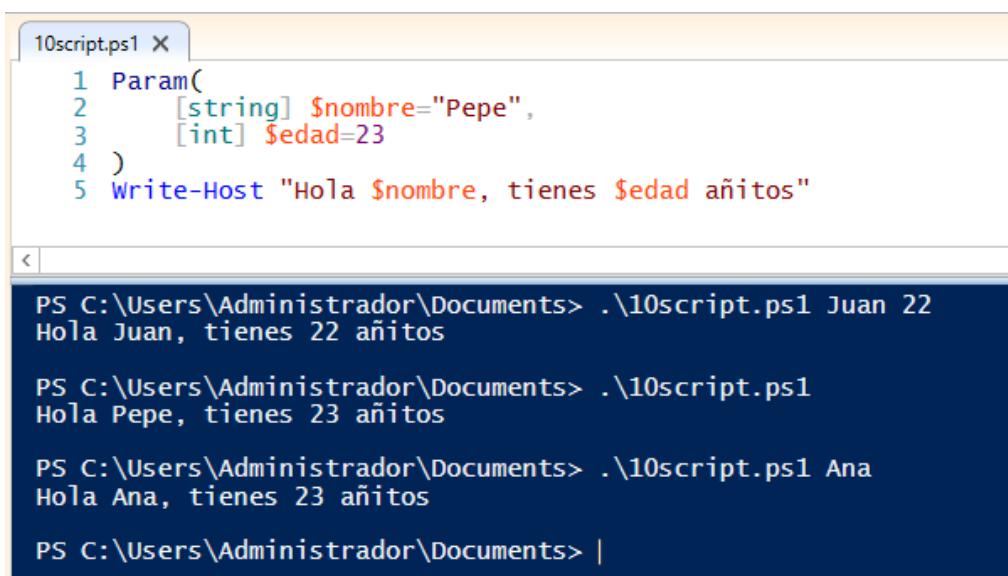


```
4script.ps1 1script.ps1 8script.ps1 6script.ps1* Sin título9.ps1* 9script.ps1 X
1 #Ejemplo de función sin parámetros. Pero recibe argumentos
2 # María y 19 son los argumentos de esta instancia o llamada
3
4 $nombre=$args[0] # El primer argumento se pasa a la variable $nombre
5 $edad=$args[1] +2 # La suma del segundo argumento (numérico) + 2 se pasa a $edad
6
7 Write-Host "Hola $nombre, tienes $edad años"

PS C:\Users\Administrador\Documents> .\9script.ps1 María 19
Hola María, tienes 21 años

PS C:\Users\Administrador\Documents> |
```

La forma más ortodoxa sería declarar los parámetros con unos argumentos por defecto:



```
10script.ps1 X
1 Param(
2     [string] $nombre="Pepe",
3     [int] $edad=23
4 )
5 Write-Host "Hola $nombre, tienes $edad añitos"

PS C:\Users\Administrador\Documents> .\10script.ps1 Juan 22
Hola Juan, tienes 22 añitos

PS C:\Users\Administrador\Documents> .\10script.ps1
Hola Pepe, tienes 23 añitos

PS C:\Users\Administrador\Documents> .\10script.ps1 Ana
Hola Ana, tienes 23 añitos

PS C:\Users\Administrador\Documents> |
```

En el ejemplo anterior, si se pasan argumentos los parámetros adquieren el nuevo valor, en caso contrario tienen los valores por defecto.

Si se pone un único argumento, corresponderá al primer parámetro.

Esta forma de declarar los parámetros permite especificar valores, para únicamente, algunos de ellos. Se pone "- nombre valor". Ejemplo:

```
10script.ps1 X listafecha.ps1
1 Param(
2     [string] $nombre="Pepe",
3     [int] $edad=23
4 )
5 Write-Host "Hola $nombre, tienes $edad añitos"

PS C:\Users\Administrador\Documents> .\10script.ps1 -edad 50
Hola Pepe, tienes 50 añitos

PS C:\Users\Administrador\Documents> .\10script.ps1 -edad 14 -nombre Clara
Hola Clara, tienes 14 añitos

PS C:\Users\Administrador\Documents>
```

Otro ejemplo más complejo:

```
10script.ps1 X listafecha.ps1 X
1 # Script que lista los archivos de una extensión pasada como parámetro
2 # si han sido modificados después de una fecha pasada por parámetro
3
4 Param (
5     [string] $ruta=".", #si no se especifica ruta, se toma el directorio actual
6     [string] $extension= "ps1", # si no se especifica extensión toma ps1
7     [datetime] $fecha
8 )
9
10 if (!$fecha) {
11     # si no se especifica una fecha de entrada, por defecto tomará las últimas 24 horas
12     $fecha = (Get-Date).AddDays(-1)
13 }
14
15 Get-ChildItem -Path $ruta -Include *.$extension -Recurse -File | Where-Object {$_.LastWriteTime -ge $fecha}
16

PS C:\Users\Administrador\Documents> .\listafecha.ps1 -fecha 05/14/21

Directorio: C:\Users\Administrador\Documents

Mode                LastWriteTime         Length Name
----                -
-a---         16/05/2021    23:17           111 10script.ps1
-a---         16/05/2021    23:15           287 1script.ps1
-a---         16/05/2021    20:53           498 2script.ps1
-a---         16/05/2021    20:52           585 3script.ps1
-a---         16/05/2021    18:03           238 4script.ps1
-a---         16/05/2021    18:22           182 5script.ps1
-a---         16/05/2021    23:15           241 6script.ps1
-a---         16/05/2021    23:15           170 7script.ps1
-a---         16/05/2021    18:00           523 8script.ps1
-a---         16/05/2021    18:16           336 9script.ps1
-a---         16/05/2021    18:28           127 ejemplo1.ps1
-a---         16/05/2021    23:32           635 listafecha.ps1
-a---         16/05/2021    20:55           471 test.ps1
```

Si queremos que un parámetro sea obligatorio, se indica con [Parameter(Mandatory=\$true)]

Para establecer un parámetro booleano, que tome el valor \$true cuando está presente, se hace poniendo [switch] \$parametro. Ejemplo:

```
10script.ps1  listafecha.ps1*  11script.ps1 X
1 #parámetro obligatorio
2 Param(
3     [Parameter(Mandatory=$true)] [int] $edad,
4     [string] $nombre,
5     [switch] $otro
6 )
7 Write-Host "Hola $nombre tienes $edad añitos y $otro ha habido un tercer parámetro"

PS C:\Users\Administrador\Documents> .\11script.ps1 32 Marta
Hola Marta tienes 32 añitos y False ha habido un tercer parámetro

PS C:\Users\Administrador\Documents> .\11script.ps1 32
Hola tienes 32 añitos y False ha habido un tercer parámetro

PS C:\Users\Administrador\Documents> .\11script.ps1
cmdlet 11script.ps1 en la posición 1 de la canalización de comandos
Proporcione valores para los parámetros siguientes:
edad: 45
Hola tienes 45 añitos y False ha habido un tercer parámetro

PS C:\Users\Administrador\Documents> .\11script.ps1 -nombre Vanessa
cmdlet 11script.ps1 en la posición 1 de la canalización de comandos
Proporcione valores para los parámetros siguientes:
edad: 34
Hola Vanessa tienes 34 añitos y False ha habido un tercer parámetro

PS C:\Users\Administrador\Documents>
```

Funciones

Del mismo modo que con los parámetros y argumentos de los scripts ocurre con el de las funciones.

Ejemplo de función que declara los parámetros que recogerán los argumentos:

```
4script.ps1  1script.ps1 X  8script.ps1  6script.ps1*  Sin título9.ps1*  9script.ps1  7script.ps1*
1 #Ejemplo de función con parámetros
2 Clear-Host
3 function ejecuta($nombre, $edad){ # $nombre y $edad son los parámetros
4     Write-Host "$nombre tiene $edad años"
5 }
6 ejecuta María 19 # María y 19 son los argumentos de esta instancia o llamada
7 ejecuta Pepe 20
8 ejecuta Ana 45
9

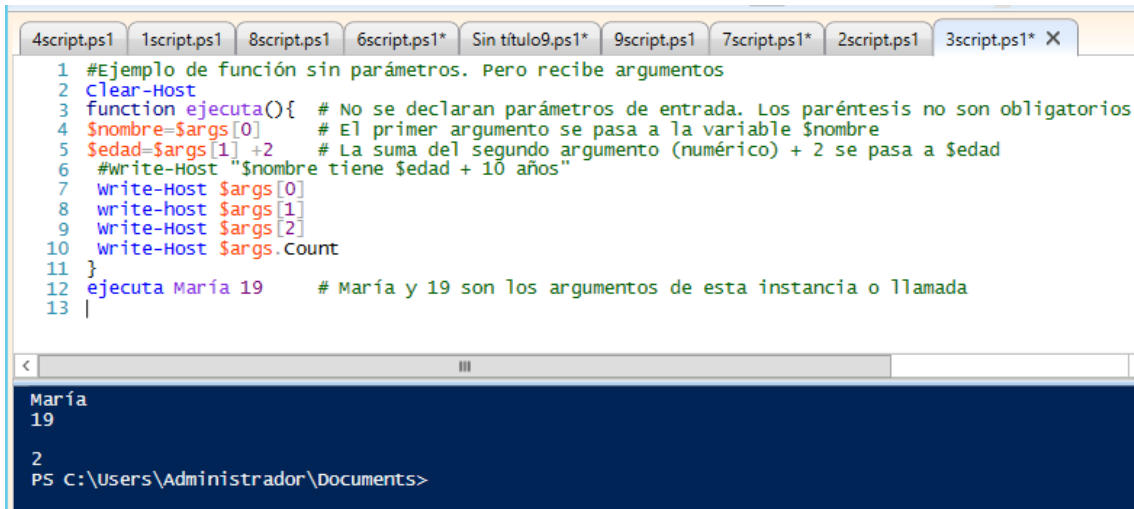
María tiene 19 años
Pepe tiene 20 años
Ana tiene 45 años
PS C:\Users\Administrador\Documents>
```

Ejemplo de función sin parámetros declarados:

```
4script.ps1  1script.ps1  8script.ps1  6script.ps1*  Sin título9.ps1*  9script.ps1  7script.ps1*  2script.ps1 X
1 #Ejemplo de función sin parámetros. Pero recibe argumentos
2 Clear-Host
3 function ejecuta(){ # No se declaran parámetros de entrada. Los paréntesis no son obligatorios
4     $nombre=$args[0] # El primer argumento se pasa a la variable $nombre
5     $edad=$args[1] +2 # La suma del segundo argumento (numérico) + 2 se pasa a $edad
6     Write-Host "$nombre tiene $edad años"
7 }
8 ejecuta María 19 # María y 19 son los argumentos de esta instancia o llamada
9 ejecuta Pepe 20
10 ejecuta Ana 45

María tiene 21 años
Pepe tiene 22 años
Ana tiene 47 años
PS C:\Users\Administrador\Documents>
```

Otro ejemplo:



```
1 #Ejemplo de función sin parámetros. Pero recibe argumentos
2 Clear-Host
3 function ejecuta(){ # No se declaran parámetros de entrada. Los paréntesis no son obligatorios
4 $nombre=$args[0]    # El primer argumento se pasa a la variable $nombre
5 $edad=$args[1]+2     # La suma del segundo argumento (numérico) + 2 se pasa a $edad
6 #Write-Host "$nombre tiene $edad + 10 años"
7 Write-Host $args[0]
8 Write-Host $args[1]
9 Write-Host $args[2]
10 Write-Host $args.Count
11 }
12 ejecuta María 19    # María y 19 son los argumentos de esta instancia o llamada
13 |
```

María
19

2
PS C:\Users\Administrador\Documents>