CFGS ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED

IMPLANTACIÓN DE SISTEMAS OPERATIVOS

7.- Funciones

Índice

1.- Introducción

Una función es un **bloque de instrucciones a las que le damos un nombre**.

Se puede llamar tantas veces como lo necesitemos.

Nos ahorra volver a escribir el código.

La **definición o declaración más simple** sería:

```
Function <Nombre> {
        Bloque de instrucciones
}
```

2.- Funciones

Ejemplo:Funcion.ps1

```
#Descripción
#Nombre:
#Autor:
#Fecha:
#Versión:
#Declaración de parámetros
#Declaración de funciones
Function Get-Saludos {
Write-Host "Bienvenidos"
}
Function Get-Despedida {
Get-Date
}
#Bloque principal
Clear
Get-Saludos #Llamo a la función
Get-Despedida #LLamo a la función
```

```
Bienvenidos

viernes, 22 de noviembre de 2019 19:05:49
```

3.-Funciones con parámetros.

Funciones con parámetros: Nos permiten pasarles datos a una función.

Ejemplo:FuncionParametros1.ps1

1ª Forma: **Es más simple, pero también más limitada.**

```
function Get-Suma ([int]$x,[int]$y)
{
    $z=$x+$y
    Write-host "La suma es $z"
}
#Bloque principal
Clear
[int]$numero1=Read-host "Introduce un número entero"
[int]$numero2=Read-host "Introduce otro número entero"
#Llamo a la función
Get-Suma $numero1 $numero2
```

3.-Funciones con parámetros.

Funciones con parámetros: Nos permiten pasarles datos a una función.

Ejemplo:FuncionParametros2.ps1

```
#Declaración de funciones
Function Get-Suma {
    param ([int]$x,[int]$y)
    $z=$x+$y
    Write-host "La suma es $z"
}
#Bloque principal
Clear
[int]$numero1=Read-host "Introduce un número entero"
[int]$numero1=Read-host "Introduce otro número entero"
#Llamo a la función
Get-Suma $numero1 $numero2
```

2ª Forma:
Los **parámetros se definen en la 1ª línea**, dentro de la función.
Tiene más posibilidades. Por ejemplo, validar los datos.

3.-Funciones con parámetros.

Funciones con parámetros: Nos permiten pasarles datos a una función.

Ejemplo:FuncionParametros3.ps1→Estamos validando un rango de valores

```
Function Get-Suma {
    param ([ValidateRange(0,10)][int]$x,[int]$y)
    $z=$x+$y
    Write-host "La suma es $z"
}
#Bloque principal
Clear
[int]$numero1=Read-host "Introduce un número entero"
[int]$numero2=Read-host "Introduce otro número entero"
#Llamo a la función
Get-Suma $numero1 $numero2
```

4.- Parámetros al script

Script con parámetros: Cuando al ejecutar un script le pasamos datos.

Ejemplo:ScriptParametros1.ps1

1ª Forma: **Es más simple, pero también más limitada.**

```
#Declaración de parámetros
$parametros=$args.Count
$x=$args[0]
$y=$args[1]
#Declaración de funciones
#Bloque principal
Clear
#Comprobamos el número de parámetros introducidos
if ($parametros -ne 2)
{
    Write-host "No has introducido dos parámetros."
    Write-Host "Vuelve a intentarlo."
}else{
[int]$z=$x+$y
Write-host "La suma es $z"
}
```

$args.count → Número de parámetros introducidos.
$args[0] → Primer parámetro.
$args[1] → Segundo parámetro.

4.- Parámetros al script

Parámetros al script: Cuando al ejecutar un script le pasamos datos.

Ejemplo:ScriptParametros2.ps1

```
#Descripción
#Nombre:
#Autor:
#Fecha:
#Versión:
#Declaración de parámetros
param ([int]$x,[int]$y)
#Declaración de funciones
#Bloque principal
Clear
[int]$z=$x+$y
Write-host "La suma es $z"
```

2ª Forma:
Los **parámetros se definen en la 1ª línea**, dentro de la función.
Tiene más posibilidades.

```
PS C:\material> .\ScriptParametros2.ps1 3 4
```

4.- Parámetros al script

Parámetros al script: Podemos definir **los parámetros como obligatorios**:

[Paramater(Mandatory=$true)] [tipo] $Nombre_Parametro

Ejemplo:ScriptParametros3.ps1

```
#Declaración de parámetros
Param([Parameter(Mandatory=$true)][int]$x,
[Parameter(Mandatory=$true)][int]$y)
#Declaración de funciones
#Bloque principal
Clear
[int]$z=$x+$y
Write-host "La suma es $z"
```

4.- Parámetros al script

Parámetros al script: Podemos definir **los parámetros como obligatorios y en un rango de valores**:

Ejemplo:ScriptParametros4.ps1

```
#Declaración de parámetros
Param([ValidateRange (0,10)] [Parameter(Mandatory=$true)][int]$x,
[Parameter(Mandatory=$true)][int]$y)
#Declaración de funciones
#Bloque principal
Clear
[int]$z=$x+$y
Write-host "La suma es $z"
```

5.- Ámbito de las variables

Ámbito de las variables: es el lugar dónde la variable es conocida.

Ejemplo:FuncionParametros1.ps1

```
#Declaración de funciones
function Get-Suma ([int]$x,[int]$y)
{
    $z=$x+$y
    Write-host "La suma es $z"
}
#Bloque principal
Clear
[int]$numero1=Read-host "Introduce un número entero"
[int]$numero2=Read-host "Introduce otro número entero"
#Llamo a la función
Get-Suma $numero1 $numero2
```

Nota: Para que los ejercicios anteriores no influyan, Remove-variable z

La variable z ha sido definida dentro de la función. ¿Será conocida fuera de la función?

Añadimos al final del script: `Write-Host "La variable z es $z"`

5.- Ámbito de las variables

El ámbito es el lugar donde la variable es conocida.

Ámbito Global: Es conocida por toda la instancia de PowerShell.
Si la definimos en la consola, es conocida por un script que ejecutemos en esa consola.

Ámbito local:
La variable es conocida en el contenedor que se defina.
Si se define dentro de un script, es conocida por todo el script.
Si se define dentro de la función, es conocida sólo dentro de la función y no en el resto del script.

Ámbito script: La variable es conocida durante la ejecución del script.

5.- Ámbito de las variables

Ámbito script: Si escribimos **$script:Variable** → la variable será conocida en el script.

Ejemplo:FuncionParametros.ps1

```
function Get-Suma ([int]$x,[int]$y)
{
    $script:z=$x+$y
    Write-host "La suma es $z"
}
#Bloque principal
Clear
[int]$numero1=Read-host "Introduce un número entero"
[int]$numero2=Read-host "Introduce otro número entero"
#Llamo a la función
Get-Suma $numero1 $numero2
Write-Host "La variable z es $z"
```

Sugerencias/mejoras del tema

Sugerencias /mejoras del tema