**Qué carajo es PowerShell**

¿Viste Bash? ¿Viste los archivos .bat? Son lenguajes de consola que interpretan órdenes y las hacen realidad para vos. PowerShell es la misma mierda. Pero, con un twist:

A diferencia de otros lenguajes de consola de comandos, que solo aceptan texto y devuelven texto, PowerShell se maneja puramente con objetos. Es decir, TODO en PowerShell gira entorno a un objeto.

**Banca. ¿Cómo es eso de que solo aceptan texto y devuelven texto?**

Cuando vos hacías un ls en Bash, lo que te devolvía eran cadenas de texto mostrándote los archivos. Si vos hacés un ls (Get-ChildItem) en PowerShell, lo que te devuelve es un objeto por cada uno de los archivos presentes, y te muestra un toString() de esos objetos.

**Qué carajo es un cmdlet**

Como siempre los informáticos son tan pelotudos que tienen que ponerle una palabra nueva a todo.

Un cmdlet es un script. Sí, como los scripts de bash, o de CMD, o lo que sea. Es un script y punto.

La única diferencia es que, según los que crearon PowerShell, los cmdlet son un script cuyos nombres son de la forma: *<verbo>-<nombre>*

El verbo indica la acción, y el nombre indica el objeto en el que se va a realizar esa acción. Por dar un ejemplo, *Get-Tuvieja* nos traería a tu javie para que nos haga algo en específico.

**Cómo carajo funciona el ForEach-Object**

La estructura de un foreach común y corriente se asemeja a algo así:

Foreach(Empleado emp in listaEmpleados) { … }

PowerShell tiene el foreach común y corriente, que te sirve para recorrer listas o arrays de objetos. Pero también, tiene un foreach diferente, el foreach-object.

El foreach-object está pensado para recorrer objetos que sean el resultado de la ejecución de un comando, sin importarnos de qué clase sean, ni en donde estén guardados.

Por ejemplo:

Get-ChildItem | ForEach-Object {

Write-Host $\_.name

}

Ahí lo que hacemos es recorrer todos los objetos que nos devuelva Get-ChildItem, y por cada uno de ellos estaríamos mostrando la propiedad “name”. Notar que utilizamos **$\_** para hacer referencia al objeto actual en el que estamos parados en cada ciclo del foreach. **$\_** es una variable especial de PowerShell.

**Cómo carajo envío parámetros a una función en PS**

Suponete que tenés una función que sea Sumar, la cual recibe dos enteros, porque vas a sumar dos números obviamente. Ahora querés llamarla, ¿cómo le envías los dos enteros)

De la misma forma que lo hacías en Bash:

Sumar $entero1 $entero2 # Bien! :)

Esa es la forma correcta. Si usás paréntesis para llamar a la función, lo estás haciendo mal, porque PowerShell interpreta lo que está dentro de los paréntesis como un array.

Sumar($entero1,$entero2) # MAL

**Cómo carajo declaro una función que reciba parámetros**

Podés hacerlo de la forma convencional, poniendo el nombre de los parámetros a recibir adentro de los paréntesis (incluso indicando el tipo de dato):

function foo([int]$a) { … }

O también podés hacerlo declarando un bloque de parámetros.

function foo() {

Param(

[int] $a

)

}

¿Cuál es la diferencia? No hay mucha, lo único bueno de hacerlo de esta última forma es que, si algún día te dan ganas, podés copiar el código de tu función y hacerlo funcionar como si fuera un script independiente. Se usa más esta última por comodidad.