POWERSHELL: SCRIPTS

IMPLANTACIÓN DE SISTEMAS
JUAN CARLOS NAVIDAD GARCÍA

1. ¿Qué es un script y cuál es su función principal?

Los scripts son fragmentos de código que tienen como objetivo realizar o añadir funciones a un sistema.
Su función principal es automatizar tareas.

2. ¿Qué significa ISE?

ISE → Integrated Scripting Environment. Básicamente es el entorno de desarrollo de scripts de Powershell.

3. Diferencias entre PowerShell y PowerShell ISE

Powershell a secas es la consola remodelada MS-DOS de Windows, simplemente se encarga de ejecutar funciones o devolver información, PowerShell ISE es el entorno de desarrollo de scripts, este desarrolla los scripts que después pueden ser ejecutados por PowerShell.

4. ¿Cuál puede ser la causa de no poder ejecutar un script en PowerShell?

Windows por defecto y seguridad inhabilita la ejecución de scripts con un nivel de seguridad restricted.

5. Comprobar los niveles de seguridad

Para comprobar el nivel de seguridad de nuestro equipo ejecutaremos el comando → get-ExecutionPolicy, este nos puede devolver las siguientes opciones:

1. Restricted: no permite ejecutar scripts.
2. Allsigned: todos los scripts tienen que estar autenticados.
3. Unrestricted: permite ejecutar cualquier script.
4. RemoteSigned: permite ejecutar nuestros scripts, pero no los remotos.

6. ¿Cómo se cambian los permisos?

El nivel de seguridad se cambia con el comando Set-ExecutionPolicy <<una de las opciones del ejercicio anterior>>.

7. ¿Cómo creamos los scripts en PowerShell?

Primero hay que crear un profile o un archivo en formato .ps1, dentro de este archivo, introduciremos el código que nosotros queramos realizar, posteriormente lo abrimos con PowerShell y se ejecutará:

Para crear un profile usaremos el siguiente comando:

New-Item -Path $profile -Type File -Force

Para comprobar que se ha creado:

test-path $profile

Para editarlo tendremos varias opciones, desde PowerShell podemos escribir “ise $profile” y lo podremos editar desde PowerShell ISE, aunque también, este archivo .ps1 de profile se nos crea por defecto en nuestra carpeta documentos y lo podremos editar desde cualquier editor de texto o incluso de programación como VisualStudio Code, Notepad ++, etcc.

8. Variables:

a. **Definición teórica y sintáctica:** las variables son espacios reservados en la memoria que, como su nombre indica, pueden cambiar de contenido a lo largo de la ejecución de un programa. Una variable corresponde a un área reservada en la memoria principal del ordenador.

b. **Acceso a la información:** para acceder a la información de una variable, basta con lanzar la variable como $variable, nos saldrá por pantalla su información.

c. **Asignación de información:** la información se le asigna a una variable en el siguiente formato: $variable= x.

d. **Declaración con un tipo de dato:**

e. **¿Distingue mayúsculas?:** PowerShell es keysensitive, quiere decir que no distingue entre mayúsculas ni minúsculas.

f. **¿Qué es el scope de una variable?:** El ámbito de una variable (llamado «scope» en inglés) es la zona del programa en la que se define la variable.

g. **Borrar el contenido de una variable y borrar variable**

Clear-variable -name nombreVariable

Remove-variable -name nombreVariable

h. Tipos de variables con ejemplos

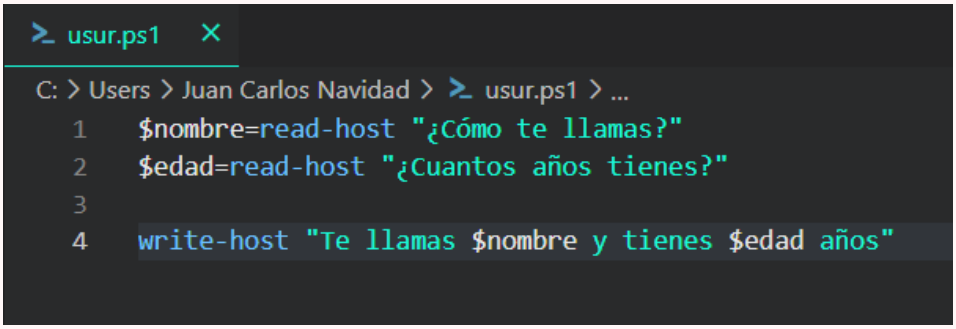
- Variables creadas por el usuario: $variable=2
- Variables predefinidas:
  - Automáticas: $$, $^, $null.
  - Personalización del comportamiento PS: $OutputEncoding.

i. **¿En qué se parece y en que se diferencia de las constantes?:** Son variables cuyo valor no cambia durante la ejecución de un programa. Únicamente puede ser leído. Las variables como tal, son modificables en comparación de las constantes.

9. ¿Cuál es la función de los comentarios en un script? Que tipos de comentarios existen y con qué símbolo se representan. Pon un ejemplo de cada

Los comentarios en los scripts de PowerShell son útiles para explicar los detalles de los procedimientos que sigue el script al ejecutarse.
Los comentarios se pueden dividir en comentarios de una línea (#) y varias líneas (<#Texto#>),

10. Cmdlet para interactuar con el usuario, pedirle informacion y mostrarle informacion.




```
[Juan Carlos Navidad Garcia]#.\usur.ps1
¿Como te llamas?: Juan Carlos Navidad
¿Cuantos años tienes?: 18
Te llamas Juan Carlos Navidad y tienes 18 años
```

11. Realiza un script documentado(comentarios) para Crear dos variables:

    - Asignándole el valor 20, e indicando el tipo entero
    - Asignándole el tipo carácter

    Muestre el contenido de las variables
    Muestra el tipo de datos de las variables
    Muestra su valor

usur.ps1

C: > Users > Juan Carlos Navidad > usur.ps1

```powershell
#Primera variable, especificaremos con [int] que es un valor entero
[int]$valor= 20

#Segunda variable, especificamos que es un valor de texto con [string]
[string]$valor2= "20"

#Por último, imprimimos en pantalla los dos valores
write-host $valor and $valor2
```

```
[Juan Carlos Navidad Garcia]#.\usur.ps1
20 and 20
```

12. Realiza un script documentado(comentarios) para pedirle al usuario su nombre y su edad mostrándole "tu nombre es … y tienes … años. Siendo los … el nombre y la edad del usuario.

```powershell
#Preguntamos su nombre y lo guardamos en una variable:

$nombre=read-host "¿Cómo te llamas?"

#Hacemos lo mismo pero con su edad

$edad=read-host "¿Cúantos años tienes?"

#Ahora mostramos en pantalla el valor de las variables

write-host "Te llamas $nombre y tienes $edad años"
```

```
[Juan Carlos Navidad Garcia]#.\usur.ps1
¿Como te llamas?: Juan Carlos Navidad
¿Cuantos años tienes?: 18
Te llamas Juan Carlos Navidad y tienes 18 años
```

13. Crear un script para saludar al usuario. Mostrara

    a. El Saludo: se proporciona con argumentos $args[0] y puede ser :

        i. Buenos días
        ii. Buenas tarde
        iii. Buenas noches

    b. El nombreUsuario: se proporciona con argumentos $args[1]

```
Write-Host $args[0]
Write-Host $args[1]
```

```
[Juan Carlos Navidad Garcia]#.\user.ps1 "Buenos días" "Juan Carlos"
Buenos días
Juan Carlos
[Juan Carlos Navidad Garcia]#.\user.ps1 "Buenos tardes" "Juan Carlos"
Buenos tardes
Juan Carlos
[Juan Carlos Navidad Garcia]#.\user.ps1 "Buenos noches" "Juan Carlos"
Buenos noches
Juan Carlos
```

14. Idem que el anterior pero con parámetros Param()

```
param($saludo, $nombre)
Write-Host "$saludo"
Write-Host "$nombre"
```

```
[Juan Carlos Navidad Garcia]#.\user.ps1 "Buenos días" "Juan Carlos"
Buenos días
Juan Carlos
[Juan Carlos Navidad Garcia]#.\user.ps1 "Buenos tardes" "Juan Carlos"
Buenos tardes
Juan Carlos
[Juan Carlos Navidad Garcia]#.\user.ps1 "Buenos noches" "Juan Carlos"
Buenos noches
Juan Carlos
```

15. Modifica el anterior para que, si no se introduce saludo, se muestre Buenos Días

```
param($saludo = "Buenos Días", $nombre)
Write-Host "$saludo"
Write-Host "$nombre"
```

```
[Juan Carlos Navidad Garcia]#.\user.ps1 -nombre "Juan Carlos"
Buenos Días
Juan Carlos
```

16. Idem que el anterior, pero pidiendo los datos al usuario en tiempo de ejecución

```
$saludo = read-host "¿Buenos días, Buenas tardes o Buenas noches?"
$nombre = read-host "¿Escribe tu nombre?"

write-host "$saludo, $nombre"
```

```
[Juan Carlos Navidad Garcia]#.\user.ps1
¿Buenos días, Buenas tardes o Buenas noches?: Buenos días
¿Escribe tu nombre?: Juan Carlos
Buenos días, Juan Carlos
```

17. Crea un diccionario llamado ordenador que guarde algunas características de los ordenadores:

    a. Marca
    b. So
    c. Procesador

    Después mostrara las marcas, so y el procesador de los ordenadores que tenga. Usa variable.elemento ($ordenador.marca)

```
[Juan Carlos Navidad Garcia]#$ordenadores = @{"Marca"="Asus"; "SO"="Windows
10"; "Procesador"="AMD Ryzen 9 5800X";}
[Juan Carlos Navidad Garcia]#$ordenadores

Name                           Value
----                           -----
SO                             Windows 10
Marca                          Asus
Procesador                     AMD Ryzen 9 5800X
```

18. Realiza un script documentado(comentarios) para mostrar el nivel de seguridad que tiene el sistema a fecha de hoy, es decir, tendría que mostrar por pantalla: Hoy día "fecha" el nivel de seguridad del sistema es NOTA: Recordad el uso de: $()

```
#Simplemente le pedimos que escriba la fecha y el nivel de seguridad con texto de por medio
#Para eso utilizamos $() para introducir comandos cmdlet dentro de un write-host

write-host

"Hoy día: $(Get-Date -format 'dd-MM-yy'), el nivel de seguridad del sistema es: $(Get-ExecutionPolicy)"
```

```
[Juan Carlos Navidad Garcia]#.\user.ps1

Hoy día: 25-10-22, el nivel de seguridad del sistema es: Unrestricted
```

19. Crea un script que muestre los números del 10 al 20

user.ps1

C: > Users > Juan Carlos Navidad > user.ps1 > $i

```
1   $i=9
2   while($i -ne 20)
3   {$i+=1
4   Write-Host $i
5   }
```

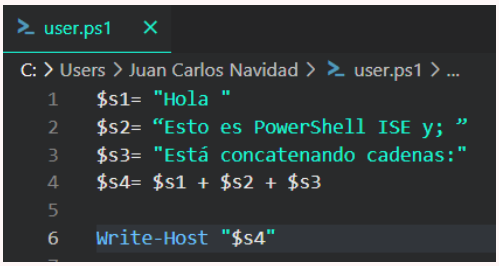
```
[Juan Carlos Navidad Garcia]#.\user.ps1
10
11
12
13
14
15
16
17
18
19
20
```

20. ¿Cuál es el operador de concatenación?

    +

21. En un script crea las siguientes variables y concaténalas en una sola llamada S4.

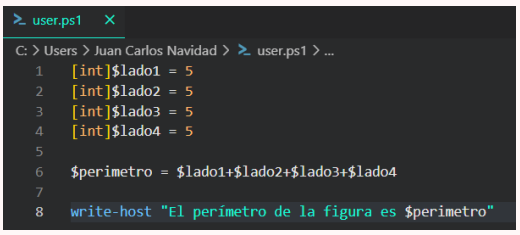
    $ s1= "Hola"
    $ s2= “Esto es PowerShell ISE y; ”
    $ s3= "Está concatenando cadenas:"
    Verifica que se ha realizado bien la concatenación S4




```
[Juan Carlos Navidad Garcia]#.\user.ps1
Hola Esto es PowerShell ISE y; Está concatenando cadenas:
```

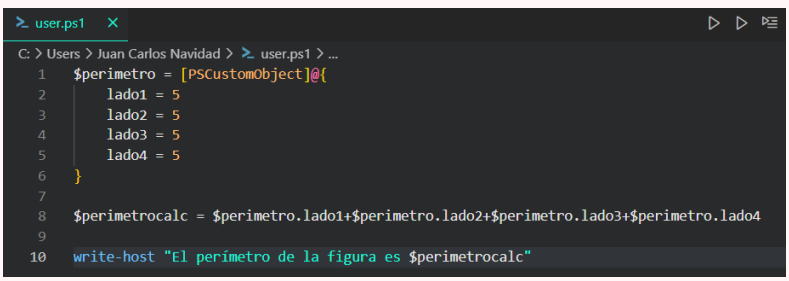
22. Pedir al usuario el lado de un cuadrado y calcula su perímetro. Almacena los datos en una estructura que te permita guardar la siguiente información:

    Lado1 = 5
    Lado2 = 5
    Lado3 = 5
    Lado4 = 5
    Perímetro = Resultado




```
[Juan Carlos Navidad Garcia]#.\user.ps1
El perímetro de la figura es 20
```

23. Modifica el script anterior para que muestre los datos de forma ordenada, para ello utiliza [PScustomObjet]




```
[Juan Carlos Navidad Garcia]#.\user.ps1
El perímetro de la figura es 20
```

24. Idem que el anterior, pero calculando el área de un rectángulo

```powershell
$area = [PSCustomObject]@{
    lado1 = 5
    lado2 = 5
    lado3 = 5
    lado4 = 5
}

$areacalc = $area.lado1*$area.lado2

write-host "El area de la figura es $areacalc"
```

```
[Juan Carlos Navidad Garcia]#.\user.ps1
El area de la figura es 25
```

25. Script que pida dos números y diga cual es el mayor

```powershell
$num1=Read-Host "Escriba el primer numero: "
$num2=Read-Host "Escriba el segundo numero: "

$mayor= if ($num1 -gt $num2)
{
    write-host "El número $num1 es mayor"
}
elseif ($num2 -gt $num1)
{
    Write-Host "El número $num2 es mayor"
}
```

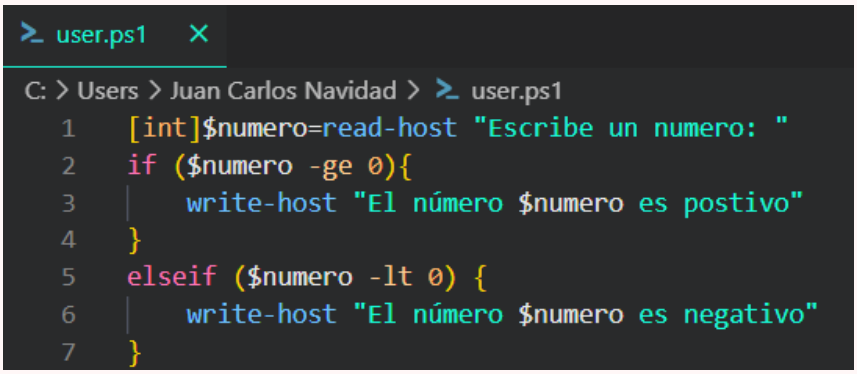
```
[Juan Carlos Navidad Garcia]#.\user.ps1
Escriba el primer numero: : 4
Escriba el segundo numero: : 2
El número 4 es mayor
```

26. Script que pida 3 números y diga cual es mayor

```powershell
$num1=Read-Host "Escriba el primer numero: "
$num2=Read-Host "Escriba el segundo numero: "
$num3=Read-Host "Escriba el tercer numero: "

$mayor= if ($num1 -gt $num2)
{
    write-host "El número $num1 es mayor"
}
elseif ($num1 -gt $num3)
{
    Write-Host "El número $num1 es mayor"
}
elseif ($num2 -gt $num1)
{
    Write-Host "El número $num2 es mayor"
}
elseif ($num2 -gt $num3)
{
    Write-Host "El número $num2 es mayor"
}
elseif ($num3 -gt $num2)
{
    Write-Host "El número $num3 es mayor"
}
elseif ($num3 -gt $num1)
{
    Write-Host "El número $num3 es mayor"
}
```

```
[Juan Carlos Navidad Garcia]#.\user.ps1
Escriba el primer numero: : 7
Escriba el segundo numero: : 4
Escriba el tercer numero: : 5
El número 7 es mayor
```

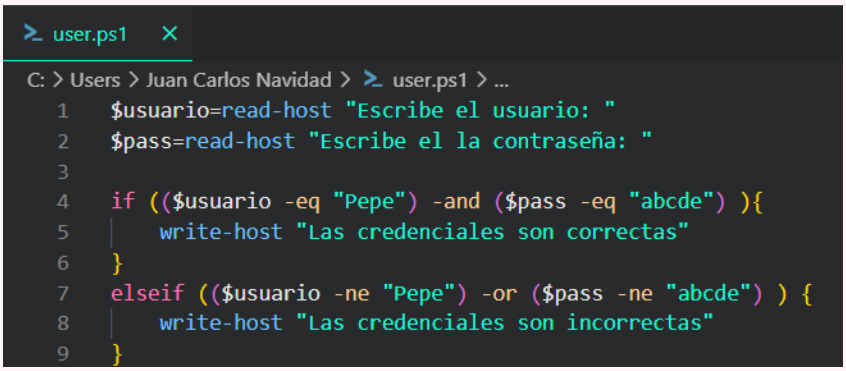
27. Script que pida un numero y diga si es Negativo o positivo.



```
user.ps1
C: > Users > Juan Carlos Navidad > user.ps1
1   [int]$numero=read-host "Escribe un numero: "
2   if ($numero -ge 0){
3       write-host "El número $numero es postivo"
4   }
5   elseif ($numero -lt 0) {
6       write-host "El número $numero es negativo"
7   }
```

```
[Juan Carlos Navidad Garcia]#.\user.ps1
Escribe un numero: : -4
El número -4 es negativo
[Juan Carlos Navidad Garcia]#.\user.ps1
Escribe un numero: : 6
El número 6 es postivo
```

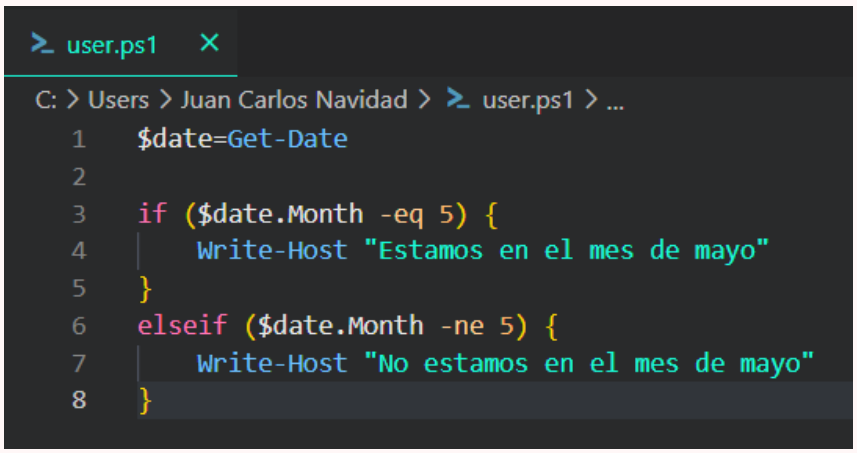
28. Script para pedir usuario y contraseña. Si coincide con “Pepe” y “abcde” indique has entrado en el sistema sino mostrará usuario no reconocido.



```
user.ps1
C: > Users > Juan Carlos Navidad > user.ps1 > ...
1   $usuario=read-host "Escribe el usuario: "
2   $pass=read-host "Escribe el la contraseña: "
3
4   if (($usuario -eq "Pepe") -and ($pass -eq "abcde") ){
5       write-host "Las credenciales son correctas"
6   }
7   elseif (($usuario -ne "Pepe") -or ($pass -ne "abcde") ) {
8       write-host "Las credenciales son incorrectas"
9   }
```

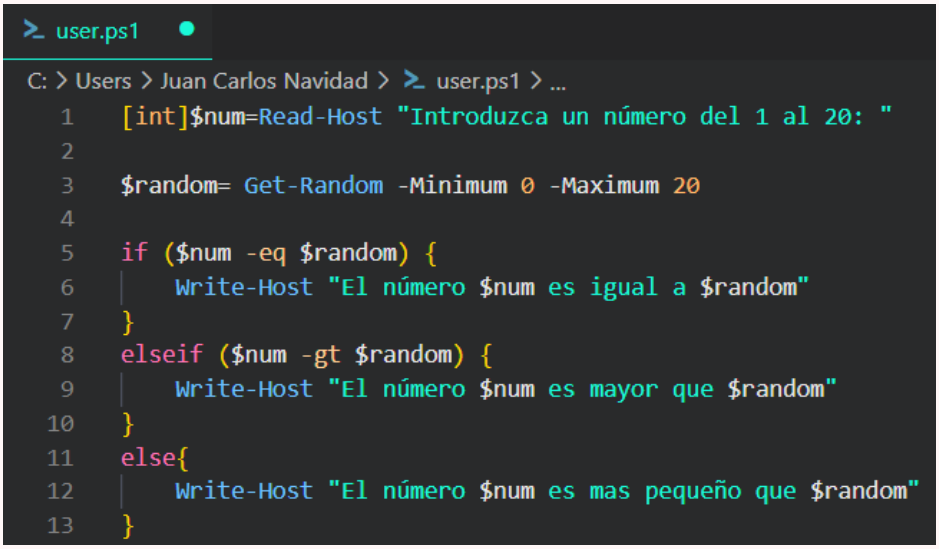
```
[Juan Carlos Navidad Garcia]#.\user.ps1
Escribe el usuario: : Pepe
Escribe el la contraseña: : abcde
Las credenciales son correctas
[Juan Carlos Navidad Garcia]#.\user.ps1
Escribe el usuario: : Pepe
Escribe el la contraseña: : fdfweff
Las credenciales son incorrectas
```

29. Script que compruebe si estamos en el mes de mayo. Informando Estamos en el mes de mayo o No estamos en el mes de mayo.




```
[Juan Carlos Navidad Garcia] -> ./user.ps1
No estamos en el mes de mayo
```

30. Realizar un script que escoja un número aleatorio entre 1 y 20, pregunte al usuario y le diga si es más pequeño o más grande.




```
[Juan Carlos Navidad Garcia] -> ./user.ps1
Introduzca un número del 1 al 20: : 14
El número 14 es igual a 14
[Juan Carlos Navidad Garcia] -> ./user.ps1
Introduzca un número del 1 al 20: : 0
El número 0 es mas pequeño que 4
[Juan Carlos Navidad Garcia] -> ./user.ps1
Introduzca un número del 1 al 20: : 20
El número 20 es mayor que 17
```

31. Realizar un script que escoja un número aleatorio entre 1y 20, pregunte al usuario y le diga si es más pequeño o más grande, que continué hasta que acierte. Haz que muestre el número de intentos hasta que lo acierta cuando finalice. Utiliza Get-Randorm(rango).

user.ps1

C: > Users > Juan Carlos Navidad > user.ps1

```powershell
while ($num -ne $random) {

    [int]$num=Read-Host "Introduzca un número del 1 al 20: "
    $random= Get-Random -Minimum 0 -Maximum 20

    if ($num -eq $random) {
    Write-Host "El número $num es igual a $random";
}
    elseif ($num -gt $random) {
    Write-Host "El número $num es mayor que $random";
}
    else{
    Write-Host "El número $num es mas pequeño que $random";
}
}
```

```
PS C:\Users\Juan Carlos Navidad> . "c:\Users\Juan Carlos Navidad\user.ps1"
Introduzca un número del 1 al 20: : 20
El número 20 es mayor que 18
Introduzca un número del 1 al 20: : 1
El número 1 es mas pequeño que 4
Introduzca un número del 1 al 20: : 2
El número 2 es mas pequeño que 13
Introduzca un número del 1 al 20: : 1
El número 1 es mas pequeño que 16
Introduzca un número del 1 al 20: : 2
El número 2 es mas pequeño que 3
Introduzca un número del 1 al 20: : 1
El número 1 es mas pequeño que 3
Introduzca un número del 1 al 20: : 4
El número 4 es igual a 4
PS C:\Users\Juan Carlos Navidad>
```

32. Realizar un script que pide un número al usuario y muestre su tabla de multiplicar tabla_multiplicar.ps1

user.ps1

C: > Users > Juan Carlos Navidad > user.ps1

```
1  [int]$numero=read-host "¿Escribe un numero: ?"
2  [int]$multiplicar=0
3  while($multiplicar -le 10)
4  {
5  Write-Host $numero "x" $multiplicar "=" ($numero*$multiplicar)
6  $multiplicar++
7  }
```

```
[Juan Carlos Navidad Garcia]#.\user.ps1
Escribe un numero: : 8
8 x 0 = 0
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
8 x 10 = 80
```

34. Crea un script con:

a. un array de 10 números inicializados a 0 (todos los numeros del array son 0.
b. Inicializalos con numeros aleatorios entre 10 y -10. Utiliza el cmdlet Get-Random para conseguir esos números. Usar lenght y bucle
c. Contabiliza cuántos números positivos, negativos e iguales a 0 hay en ese array. Usa bucle e If

usur.ps1

C: > Users > Juan Carlos Navidad > usur.ps1 > ...

```powershell
$array=@()
$array2=@()
[int]$pos=0
[int]$neg=0
[int]$eq=0

while ($array.length -lt 10){

$array+=get-random (-10..10)

continue

}

$array2+=$array

foreach ($array in $array2)
{

if ($array -gt 0) {

$pos+=1

}


elseif ($array -lt 0){

$neg+=1

}

elseif ($array -eq 0){

$eq+=1

}

}
write-host "`nLos números son: $array2`n"
write-host "Hay $pos números positivos, $neg negativos y $eq iguales`n"
```

```
[Juan Carlos Navidad Garcia] -> ./usur.ps1

Los números son: -7 -2 1 -7 0 10 6 -6 7 1

Hay 5 números positivos, 4 negativos y 1 iguales
```

35. Script que pida 10 números y los muestre en orden descendente

usur.ps1

C: > Users > Juan Carlos Navidad > usur.ps1 > ...

```powershell
Clear-Host
[int]$num = 0
$sum2 = @()
while ($num -lt 10) {
    $num++
    [int]$num2 = read-host "Escriba el $num número"
    write-host ""
    $sum2 = $sum2 + $num2
    continue
}
write-host ""
$sort = $sum2 | Sort-Object -Descending
write-host $sort
```

```
Escriba el 1 numero: 1
Escriba el 2 numero: 2
Escriba el 3 numero: 3
Escriba el 4 numero: 4
Escriba el 5 numero: 5
Escriba el 6 numero: 6
Escriba el 7 numero: 7
Escriba el 8 numero: 8
Escriba el 9 numero: 9
Escriba el 10 numero: 10

10 9 8 7 6 5 4 3 2 1
[Juan Carlos Navidad Garcia] ->
```

36. Utilizando arrays (para guardar los caracteres que pueden usar la contraseña) y números aleatorios, desarrolla un script de powershell que genere contraseñas. Max longitud 20.

usur.ps1 1

C: > Users > Juan Carlos Navidad > usur.ps1 > ...

```powershell
Clear-Host
$cont=0
$pass = @()
write-host "Contraseñas: `n"
while ($pass.Length -lt 20) {
    $cont+=1
    $random=((50..80) | Get-Random -count 10 |  % {[char]$_}) -join""
    $pass=$pass+"$random"
    Write-Host $random
    Write-Host ""
}
```

```
Contraseñas:

2O5G;6>879
E2;H9JCLN5
=G86@OL7BJ
:<HGEP376D
IHMD625EA8
KDLHBG5J@?
MBN5IA:=J7
6H87<NJ4L@
<?OGB9;K8>
KHFB>O;I3D
P45EHO=A3G
N@?2F<;O>3
IHP3:E=J2G
G?;92CLP7E
ANMP?;4OEC
A@OGE>L8B7
:O@682L?<9
<D;LA=G@N2
@G23B6HJ:C
HF<4KL7D6E
[Juan Carlos Navidad Garcia] ->
```

37. Crea un cmdlet de powershell que espere tres parámetros, un llamado $primero que espere un [int], otro llamado $segundo que espere un String que si no se introduce debe solicitarlo al usuario. Una vez hecho el programa debe hacer un bucle for que muestre el valor de $segundo tantas veces como indique $primero. Recuerda los parámetros se definen

    Param(
        [tipo]Parametro1,
        [tipo]Parametro2.
        ....
)

usur.ps1

```
C: > Users > Juan Carlos Navidad > usur.ps1
1   param (
2   [int]$num = $(Read-Host "`n Introduce el primer valor"),
3   [string]$cad = $(Read-Host "`n Introduce el segundo valor"
4   Write-Host "`n Se va a mostrar $num veces el numero $cad"
5   for ($sum = 0; $sum -lt $num; $sum+=1){
6       Write-Host "`n $cad `n"
7   }
```

```
[Juan Carlos Navidad Garcia] -> ./usur.ps1

 Introduce el primer valor: 3

 Introduce el segundo valor: 3

 Se va a mostrar 3 veces el numero 3

 3


 3


 3
```

38. Crear una calculadora muy básica con PowerShell la cual solo tendrá cuatro posibles operaciones: suma, resta, multiplicación y división. Usa una función llamada "operar" a la que se pasaremos dos números y una operación. Usa switch para realizar operación dependiendo de la operación pasada.

usur.ps1

```
C: > Users > Juan Carlos Navidad > usur.ps1 > ...
1   while (($opcion -lt 1) -or ($opcion -gt 4)){
        1 reference
2       function operar([int]$num1,[int]$num2,[int]$op){
3           switch ($op){
4             1 { $result = $num1 + $num2 }
5             2 { $result = $num1 - $num2 }
6             3 { $result = $num1 * $num2 }
7             4 { $result = $num1 / $num2 }
8           }
9           return $result
10        }
11
12          Clear-Host
13          Write-Host "1. Sumar "
14          Write-Host "2. Restar"
15          Write-Host "3. Multiplicar"
16          Write-Host "4. Dividir `n"
17          $opcion = Read-Host "¿Que desea hacer? "
18          Write-Host ""
19          $num3 = Read-Host "Introduce un número: "
20          $num4 = Read-Host "Introduce otro número: "
21          Write-Host ""
22
23          $res = operar $num3 $num4 $opcion
24          Write-Output "El resultado es: $res `n"
25
26      }
```

```
1. Sumar
2. Restar
3. Multiplicar
4. Dividir

¿Que desea hacer? : 1

Introduce un número: : 2
Introduce otro número: : 2

El resultado es: 4

[Juan Carlos Navidad Garcia] ->
```

```
1. Sumar
2. Restar
3. Multiplicar
4. Dividir

¿Que desea hacer? : 3

Introduce un número: : 3
Introduce otro número: : 3

El resultado es: 9

[Juan Carlos Navidad Garcia] ->
```

```
1. Sumar
2. Restar
3. Multiplicar
4. Dividir

¿Que desea hacer? : 2

Introduce un número: : 2
Introduce otro número: : 2

El resultado es: 0

[Juan Carlos Navidad Garcia] ->
```

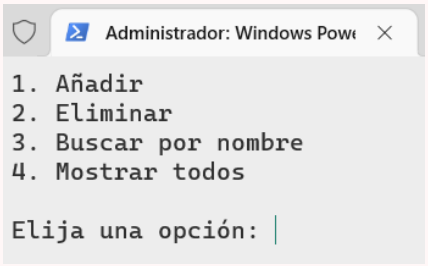
```
1. Sumar
2. Restar
3. Multiplicar
4. Dividir

¿Que desea hacer? : 4

Introduce un número: : 20
Introduce otro número: : 5

El resultado es: 4

[Juan Carlos Navidad Garcia] ->
```

39. Crea un cmdlet de PowerShell que defina un array relacional con usuarios. Los campos de cada elemento deben ser nombre_cuenta, nombre y password. El programa debe mantener ese array con un menú en el que se pueda:

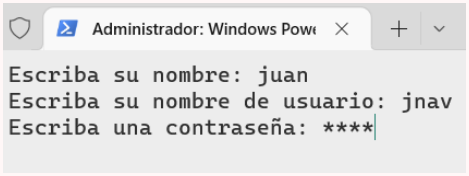
-añadir
-eliminar
-buscar por nombre
-mostrar todos

Al crear uno nuevo deben solicitarse datos y el campo password debe ocultarse al escribir.

```powershell
clear
$usuario=@{}
while (($opcion -lt 1) -or ($opcion -gt 4)){

    Write-Host "1. Añadir"
    Write-host "2. Eliminar"
    Write-Host "3. Buscar por nombre"
    Write-Host "4. Mostrar todos"

    [int]$opc=read-host "`nElija una opción"

    if ($opc -eq 1) {
        Clear-Host
        $nombre=read-host "Escriba su nombre"
        $usur=read-host "Escriba su nombre de usuario"
        $pass=read-host "Escriba una contraseña" -AsSecureString

        $usuario+=@{"nombre"=$nombre}
        $usuario+=@{"nombre_cuenta"=$usur}
        $usuario+=@{"password"=$pass}

        write-host

    }

    if($opc -eq 2) {

        $usuario.clear()

        write-host "El usuario ha sido borrado"

        clear

    }

        if($opc -eq 3) {

        clear

        $nombre1=read-host "`nEscriba su nombre"}

            if($usuario.nombre -eq $nombre1){

            write-host "`nEl nombre de usuario es:" $usuario.nombre_cuenta

            write-host "`nTu nombre es:" $usuario.nombre

            write-host "`nLa contraseña es:" $usuario.password

            write-host ""

            }


        if($opc -eq 4) {


        $usuario
        Write-Host ""

    }

}
```



Menú



Opción 1

```
Escriba su nombre: juan

El nombre de usuario es: jnav

Tu nombre es: juan

La contraseña es: System.Security.SecureString
```

Opción 3

```
1. Añadir
2. Eliminar
3. Buscar por nombre
4. Mostrar todos

Elija una opción: 4

Name                           Value
----                           -----
nombre                         juan
nombre_cuenta                  jnav
password                       System.Security.SecureString
```

Opción 4

42. Realizar un script que muestre 5 números aleatorios del 1 al 10 seguidos de el mismo número de asteriscos.

```powershell
$cont=0
[string]$ast="*"
while ($cont -lt 5) {

    $cont+=1

    [string]$random=Get-Random(1..5)

    $res=$random+$ast*$random

    Write-Host $res


}
```

```
[Juan Carlos Navidad Garcia] -> ./usur.ps1
5*****
4****
1*
3***
1*
```