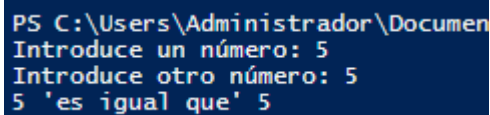


1. Realizar un script en PowerShell llamado "comparar.ps1" que se solicite al Usuario dos números enteros. El script debe mostrar por pantalla si un número es mayor, menor o igual que el otro número.

```
$numero1 = Read-Host "Introduce un número"
```

```
$numero2 = Read-Host "Introduce otro número"
```

```
if ($numero1 -gt $numero2){  
    echo "El $numero1 'es mayor que el' $numero2"  
}  
elseif ($numero1 -eq $numero2){  
    echo "$numero1 'es igual que' $numero2"  
}  
else{  
    echo "El '$numero1' es más pequeño que '$numero2'"  
}
```



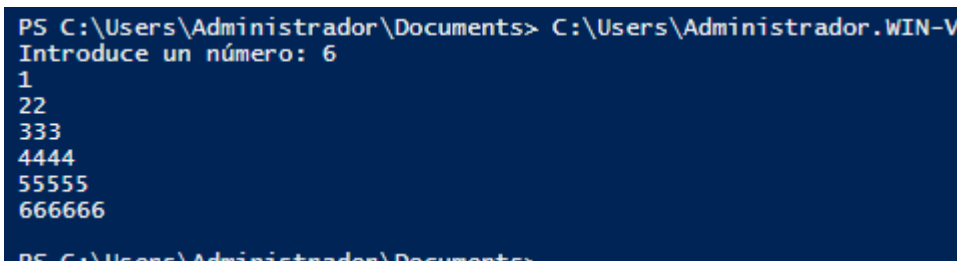
```
PS C:\Users\Administrador\Documents> .\comparar.ps1  
Introduce un número: 5  
Introduce otro número: 5  
5 'es igual que' 5
```

2. Realizar un script en PowerShell llamado "pirámide.ps1" que construya una pirámide de números hasta el límite que se le pida al usuario.

Por ejemplo, introduciendo el número 9 debería dar como salida:

```
$numero = Read-Host "Introduce un número"
```

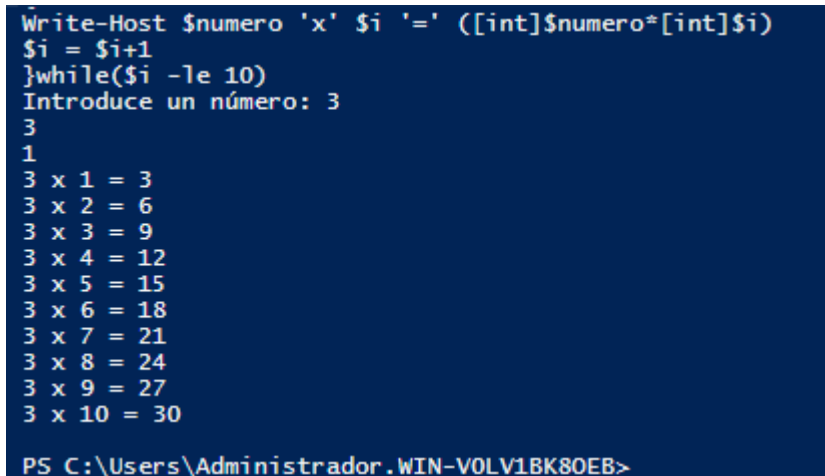
```
for ($i=1;$i -le $numero; $numero -1)  
{  
    for ($j=1; $j -le $i; $numero -1){  
        Write-Host -NoNewline $i  
        $j=$j+1  
    }  
    $i=$i+1  
    echo "  
}
```



```
PS C:\Users\Administrador\Documents> .\pirámide.ps1  
Introduce un número: 6  
1  
22  
333  
4444  
55555  
666666  
PS C:\Users\Administrador\Documents>
```

3. Realizar un script en PowerShell llamado "multiplicar.ps1" que solicite al usuario un número y muestre por pantalla la tabla de multiplicar de ese número.

```
$numero = Read-host "introduce un numero"
$i=1
$numero -as [int]
$i -as [int]
do
{
    Write-Host $numero 'x' $i '=' ([int]$numero*[int]$i)
    $i = $i+1
}while($i -le 10)
```



```
Write-Host $numero 'x' $i '=' ([int]$numero*[int]$i)
$i = $i+1
}while($i -le 10)
Introduce un número: 3
3
1
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
PS C:\Users\Administrador.WIN-VOLV1BK80EB>
```

4. Realizar un script en PowerShell llamado "help.ps1" que reciba un nombre de cmdlet y muestre todos los parámetros específicos de este cmdlet (Getcommand), ordenados en orden alfabético inverso y sin repetir.

Se deben excluir todos los parámetros comunes a todos los cmdlets que son:

"Verbose", "Debug", "WarningAction", "WarningVariable", "ErrorAction",
"ErrorVariable", "OutVariable", "OutBuffer"

```
$cmdlet= Read-Host "Introduce un cmdlet"
$excluir = "Verbose", "Debug", "WarningAction", "WarningVariable", "ErrorAction",
"ErrorVariable", "OutVariable", "OutBuffer"
$parametros = (Get-Command $cmdlet).Parameters.Keys | Sort-Object -Descending
-Unique
$parametros | Where-Object {$excluir -notcontains $_}
```

```
$cmdlet= Read-Host "Introduce un cmdlet"
$excluir = "Verbose", "Debug", "WarningAction", "WarningVariable", "ErrorAction",
"ErrorVariable", "OutVariable", "OutBuffer"
$parametros = (Get-Command $cmdlet).Parameters.Keys | Sort-Object -Descending -Unique
$parametros | Where-Object {$excluir -notcontains $_}
```

```
Introduce un cmdlet: get-process
PipelineVariable
Name
Module
InputObject
InformationVariable
InformationAction
IncludeUserName
Id
FileVersionInfo
ComputerName
```

5. Realizar un script en PowerShell llamado "usuario.ps1" que reciba un nombre de usuario y un nombre de un grupo de usuarios. La función solicitará una contraseña y creará el nuevo usuario haciéndole miembro del grupo proporcionado. Si el grupo no existe se creará.

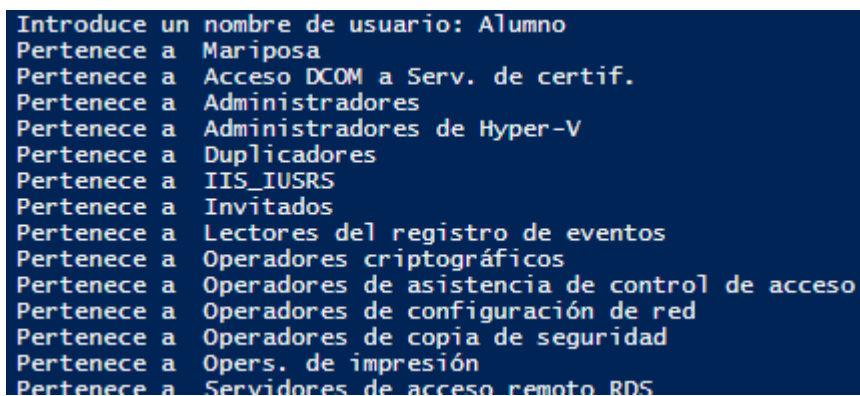
```
$usuario = Read-Host "Introduce un nombre de usuario"
$grupo = Read-Host "Introduce un nombre de grupo"
$contraseña = Read-Host "Introduce contraseña"
$contraseña = ConvertTo-SecureString -AsPlainText -Force String

Write-Host "Comprobando el grupo ingresado"
Get-LocalGroup $grupo
Write-Host "Si el grupo no se encuentra, lo creamos"
New-LocalGroup -Name $grupo -Description "Grupo creado desde Script"
Get-LocalGroup $grupo
Write-Host "Comprobando el usuario ingresado"
Get-LocalUser $usuario
Write-Host "Si el usuario ingresado no se encuentra, lo creamos"
New-LocalUser $usuario -Password $contraseña
Write-Host "Se añadirá el usuario al grupo"
Add-LocalGroupMember -Member $usuario -Group $grupo
write-host "Comprobamos que esté dentro"
Get-LocalGroupMember $grupo
```

```
Name           : WINSERVER2016\Clara
SID             : S-1-5-21-3081970298-2868366693-465237882-1001
PrincipalSource : Local
ObjectClass     : Usuario
```

6. Realizar un script en PowerShell llamado "grupos.ps1" que reciba un nombre de usuario y muestre para ese usuario todos los grupos a los que pertenece. Si el usuario no existe se informará.

```
$usuario = Read-Host "Introduce un nombre de usuario"  
foreach($grupo in Get-LocalGroup -ErrorAction SilentlyContinue){  
if ( $? -eq "True")  
{Write-Host "Pertenece a " $grupo}  
  
}  
  
$usuario = Read-Host "Introduce un nombre de usuario"  
foreach($grupo in Get-LocalGroup -ErrorAction SilentlyContinue){  
if ( $? -eq "True")  
{Write-Host "Pertenece a " $grupo}  
  
}
```



```
Introduce un nombre de usuario: Alumno  
Pertenece a Mariposa  
Pertenece a Acceso DCOM a Serv. de certif.  
Pertenece a Administradores  
Pertenece a Administradores de Hyper-V  
Pertenece a Duplicadores  
Pertenece a IIS_IUSRS  
Pertenece a Invitados  
Pertenece a Lectores del registro de eventos  
Pertenece a Operadores criptográficos  
Pertenece a Operadores de asistencia de control de acceso  
Pertenece a Operadores de configuración de red  
Pertenece a Operadores de copia de seguridad  
Pertenece a Opers. de impresión  
Pertenece a Servidores de acceso remoto RDS
```