#1 OPERACIONES BÁSICAS: SUMA, RESTA, PRODUCTO Y DIVISIÓN

**$a=**34

**$b=**25

**$suma= $a+$b**

**$resta= $a-$b**

**$producto= $a\*$b**

**$division= $a/$b**

write "$a + $b = $suma" write "$a - $b = $resta" write "$a \* $b = $producto" write "$a / $b = $division"

**34 + 25 = 59**

**write**

**34 - 25 = 9**

**write**

**34 \* 25 = 850**

**write**

**34 / 25 = 1.36**

**<#**

**SE HAN UTILIZADO DISTINTAS VARIABLES PARA EJECUTAR LAS DISTINTAS OPERACONES BÁSICAS (SUMA, RESTA, PRODUCTO Y DIVISIÓN)**

**#>**

#2 INTRODUCIR DATOS

**$nombre =** Read-Host "Tu nombre: "

**$password =** Read-Host **-**assecurestring "Tu clave: " #mostrar

write-host "Hola $nombre" Clear**-**Host

**$YearCalc =** Read-Host "¿cuando naciste?"

**$Now =** (Get-Date **-**uformat "%Y") **-$YearCalc**

**$Maybe = $Now -**1

Write-Host "tu tienes $Maybe o $Now años"

**Tu nombre: Jose Maria**

**Hola Jose Maria Clear-Host**

**¿cuando naciste?: 2000**

**tu tienes 20 o 21 años**

**<#**

**SE PUEDE APRECIAR LA ENTRADA DE DATOS PARA PODER CALCULAR FINALMENTE NUESTRA EDAD**

**#>**

#3 FUNCIÓN

**function** mensaje{

"hola desde una función"

}

mensaje

**function** ver{

"estas en: "

gl

}

ver

**<#**

**NO HE PODIDO EJECUTAR ESTE COMANDO, ME APARECEN ALGUNOS ERRORES Y NO HE SIDO CAPAZ DE SOLUCIONARLOS**

**#>**

#función con parámetros

**$g=**9.81

**Function** altura{

**param**(**$t**) **if**(**$t -eq** 0){

**return** 0

}**else**{

**return** (**$g\***[System.Math]**::**Pow(**$t**,2))**/**2

}

}

**$result=**altura(6.309) write-host "$result"

**hola desde una función**

**estas en:**

**Path**

**----**

**C:\Users\Viudez**

**<#**

**SE APRECIA UNA FUNCIÓN QUE CONDICIONA UN PARÁMETRO**

**#>**

#4 ESTRUCTURAS DE CONTROL

Write-Host "uso de do{}while(cond)"

**$i =** 1

**do** {

Write-Host **$i**

**$i++**

}**while** (**$i -le** 5)

Write-Host "uso de while(cond){}"

**$i =** 1

**while** (**$i -le** 5) {

Write-Host **$i**

**$i++**

}

Write-Host "uso de do{}until(cond)"

**$i =** 1

**do** {Write-Host **$i**; **$i++**}

**until** (**$i -gt** 5)

**$strResponse =** “Quit”

**do** {**$strResponse =** Read-Host “Are you sure you want to quit application**?** (Y**/**N)”}

**until** (**$strResponse -eq** “Y”)

Write-Host "uso de for"

**for** (**$i=**1; **$i -le** 5; **$i++**)

{Write-Host **$i**}

**$ints =** @( 1, 2, 3, 4, 5)

**for** (**$i=**0; **$i -le $ints**.Length – 1; **$i++**)

{Write-Host **$ints**[**$i**]}

Write-Host "uso de foreach"

**$ints =** @(1, 2, 3, 4, 5)

foreach (**$i in $ints**)

{Write-Host **$i**}

**uso de do{}while(cond)**

**1**

**2**

**3**

**4**

**5**

**uso de while(cond){}**

**1**

**2**

**3**

**4**

**5**

**uso de do{}until(cond)**

**1**

**2**

**3**

**4**

**5**

**Are you sure you want to quit application? (Y/N): y**

**uso de for**

**1**

**2**

**3**

**4**

**5**

**1**

**2**

**3**

**4**

**5**

**uso de foreach**

**1**

**2**

**3**

**4**

**5**

**<#**

**NO HE LOGRADO ENTENDER LO QUE PRETENDE HACER ESTE SCRIPT**

**#>**

#5 VECTORES

**$lista=**200,250,300,350,400

write-host "lista [$lista]" foreach(**$l in $lista**){

**if**(**$l -eq** 250){

"$l, valor encontrado"

}**else**{

**$result=$l\***2;

write-host "$l $l\*2: $result"

}

}

lista [200 250 300 350 400] foreach 200 250 300 350 400

if($l -eq 250){

"$l, valor encontrado"

}else{

$result=$l\*2;

write-host "$l $l\*2: $result"

}

**<#**

**NO HE PODIDO EJECUTAR ESTE COMANDO, ME APARECEN ALGUNOS ERRORES Y NO HE SIDO CAPAZ DE SOLUCIONARLOS**

**#>**

#CUENTA VOCALES

Clear**-**Host

**$cadena=**Read-Host "Introduce cadena" Write-Host "introduciste " **$cadena**

**$cont=**0

**$aux=**""

**for**(**$i=**0;**$i -le $cadena**.Length;**$i++**){

**$c=$cadena**[**$i**]

**if**(**$c -eq** "a" **-or $c -eq** "e" **-or $c -eq** "i" **-or $c -eq** "o" **-or $c -eq** "u"){

**$cont++**

**$aux+=**","**+$c**

}

}

Write-Host "no. de vocales: "**$cont**" ,vocales obtenidas: "**$aux**

**Introduce cadena Write-Host introduciste introduciste: prueba**

**no. de vocales: 3 ,vocales obtenidas: ,u,e,a**

**<#**

**ESTE SCRIPT PERMITE CONTAR LAS VOCALES DE LA CADENA QUE INTRODUZCAMOS Y, A SU VEZ, EXPRESA CUALES SON ESTAS**

**#>**

#CALCULAR VALOR FUTURO

clear**-**host

**$valor=**read-host "Valor: "

**$tasa=**read-host "Tasa:"

**$periodo=**read-host "Periodo:"

**$resultado=**0 #mostrar valores

write-host "valor : $valor" write-host "tasa : $tasa" write-host "periodo : $periodo" ""

""

"[Resultado]"

foreach (**$n in** 1..**$periodo**) {

**$resultado=**[math]**::**pow(1**+$tasa/**100,**$n**)

**$resultado=$resultado\*$valor**

write-host "$resultado periodo --> $n"

}

**Valor: 1**

**Tasa: 2**

**Periodo: 3**

**valor: 1 write-host tasa: 2 write-host periodo : 3**

**[Resultado]**

**1.02 periodo --> 1**

**1.0404 periodo --> 2**

**1.061208 periodo --> 3**

**<#**

**NO HE LOGRADO ENTENDER LO QUE PRETENDE HACER ESTE SCRIPT**

**#>**

#6 MENÚ

clear**-**host

write-host "#############################"

write-host ""

write-host " Menu "

write-host ""

write-host "1. Ver version"

write-host "2. Ver fecha"

write-host "3. Ver ayuda"

write-host "4. Abrir bloc de notas"

write-host "5. Abrir calculadora"

write-host "6. Salir"

write-host "#############################"

**$opc =** Read-Host "Tu opcion: "

write-host ""

write-host "introduciste [$opc]"

#if(opc != 0 || opc >= 6)

**if**(**$opc -ne** 0 **-or $opc -ge** 6){

**switch**(**$opc**){

1 {write-host "version" -ForegroundColor Cyan

get-host

}2

{write-host "fecha" -ForegroundColor Cyan

get-date

}3

{write-host "ayuda" -ForegroundColor Cyan

get-help

}4

{write-host "bloc de notas" -ForegroundColor Cyan

abreBloc

}

5 {write-host "calculadora" -ForegroundColor Cyan

abreCalc

}6

{write-host "fin" -ForegroundColor Red

**exit**

}

}#fin switch

}#############################

**Menu**

**1. Ver version**

**2. Ver fecha**

**3. Ver ayuda**

**4. Abrir bloc de notas**

**5. Abrir calculadora**

**6. Salir**

**#############################**

**Tu opcion: : 1**

**introduciste [1]**

**version**

**Name : Windows PowerShell ISE Host**

**Version : 5.1.19041.906**

**InstanceId : 8ca35501-98b4-4a4c-a57c-7dad32c8bcef**

**UI : System.Management.Automation.Internal.Host.InternalHostUserInterface**

**CurrentCulture : es-ES**

**CurrentUICulture : es-ES**

**PrivateData : Microsoft.PowerShell.Host.ISE.ISEOptions**

**DebuggerEnabled : True**

**IsRunspacePushed : False**

**Runspace : System.Management.Automation.Runspaces.LocalRunspace**

**<#**

**NO HE LOGRADO ENTENDER LO QUE PRETENDE HACER ESTE SCRIPT**

**#>**

#7 DATOS DEL SISTEMA

write-host "Datos del sistema:"

New-Object System.io.DriveInfo "C:" | Format-List **\***

**$drive =** New-Object System.io.DriveInfo "C:"

**$drive**.DriveFormat

**$drive**.VolumeLabel

#UBICACIÓN

**$variable=**gl "estas en: "

**$variable**

#FECHA

**$fecha=**Get-Date

write-host "hoy es "**$fecha**

#IMPRESORAS

write-host

write-host "ShowPrnH.ps1, Version 1.01"

write-host "Show available printers in HTML"

write-host "Written by Rob van der Woude"

write-host "http://www.robvanderwoude.com"

write-host

get-wmiobject **-**class Win32\_Printer | convertto-html

Name,**Default**,Network,PortName,DriverName,ServerName,ShareName **-**head "<title>All printers

available on $env:computername</title>`n<style type=`"text**/**css**`**">`nbody { padding: 8px;

line-height: 1.33 }`ntable { border-style: ridge }`ntd, th { padding: 10px; border-style:

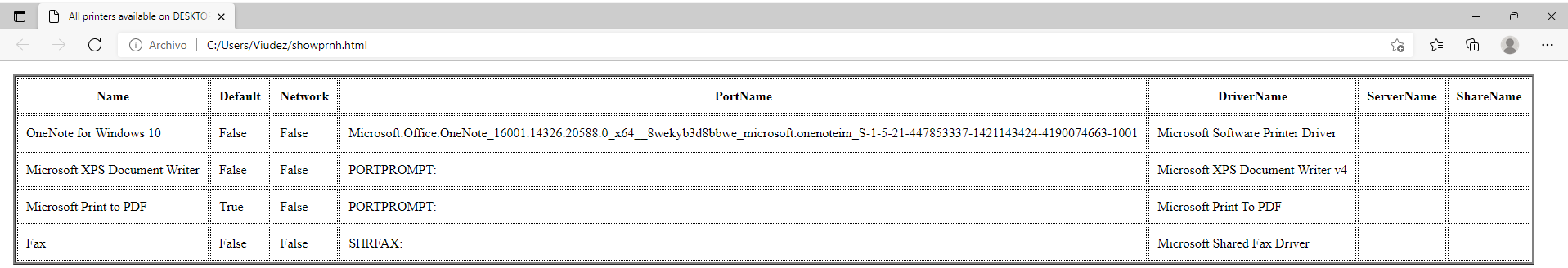
dotted; border-width: 1px }`nth { font-weight: bolder; text-align: center }`n</style>" |

out-file **-**FilePath "showprnh.html" **-**Encoding "ASCII"

invoke-item "showprnh.html"

**hoy es 07/12/2021 9:32:19**

**ShowPrnH.ps1, Version 1.01 write-host Show available printers in HTML write-host Written by Rob van der Woude write-host http://www.robvanderwoude.com write-host**



**<#**

**NO HE LOGRADO ENTENDER LO QUE PRETENDE HACER ESTE SCRIPT**

**#>**

#ARCHIVOS y guardarla en un archivo de texto \*.txt gci > listaArchivos.txt

#mostrar el nombre de los ARCHIVOS y el tamaño ordenados por tamaño gci |select name,length |sort length –desc

#Cómo mostrar el nombre de los ARCHIVOS y el tamaño cuyo tamaño sea 79 bytes en powershell ?

gci |select name,length | where {**$\_**.length **-eq** 76}

#8 ARCHIVOS txt

Clear**-**Host

**$Path =** "C:\Program Files\"

Get-Childitem **$Path -**recurse **-**force | Foreach {

**If** (**$\_**.extension **-eq** ".txt") {

Write-Host **$\_**.fullname

}

}

**$strResponse =** “salir”

**do** {**$strResponse =** Read-Host “Quiere salir de la aplicación**?** (Y**/**N)”}

**until** (**$strResponse -eq** “Y”) New-Item **-**Type f freespace.txt

**$date =** ( get**-** date ).ToString('yyyyMMdd')

**$file =** New-Item **-**type file "$date-

freespace.txt"

**$date =** ( get-date ).ToString('yyyyMMdd')

ForEach (**$system in** Get-Content "servicio.txt")

{Write-Host

**$system**}

**¿Quiere salir de la aplicación? (Y/N): y**

**Directorio: C: \Users\Administrador**

**Mode LastWriteTime Length Name**

**---- ------------ ------ -----**

**-a---- 5/12/2021 20:37 0 freespace.txt**

**<#**

**NO HE LOGRADO ENTENDER LO QUE PRETENDE HACER ESTE SCRIPT**

**#>**

#ATRIBUTOS DE ARCHIVOS

**$Path =** "C:\Program Files\" "{0,10} {1,-24} {2,-2}" **-**f **`**

" Size", "Last Accessed", "File Name "

Foreach (**$file in** Get-Childitem **$Path -**recurse **-**force)

{**If** (**$file**.extension **-eq** ".txt")

{

"{0,10} {1,-24} {2,-2}" **-**f **`**

**$file**.length, **$file**.LastAccessTime, **$file**.fullname

}

}

# CONTENIDO DE ARCHIVOS

clear

Get-Content C**:\**servicio.txt

**$a =** Get-Content C**:\**Users**\**fernando**\**Documents**\**Ejemplos**\**servicio.txt write-host "$a"

(Get-Content C**:\**Users**\**fernando**\**Documents**\**Ejemplos**\**servicio.txt)[0 .. 2]

**$arch=**get-content C**:\**Users**\**fernando**\**Documents**\**Ejemplos**\**servicio.txt ForEach-Object {Write-Host **$arch** -foregroundcolor red}

foreach (**$number in** 1..10 ) { **$number \*** 4}

**4**

**8**

**12**

**16**

**20**

**24**

**28**

**32**

**36**

**40**

**<#**

**NO HE LOGRADO ENTENDER LO QUE PRETENDE HACER ESTE SCRIPT**

**#>**

#9 ARCHIVOS Y MENÚ Y FUNCIONES

clear get-host

**$cadena=**"pelo"

**$archivo=**"C:\Users\fernando\Documents\Ejemplos\servicio.txt"

**$fecha=**(get-date)

**$fecha**

**$nombre =** Read-Host "Tu nombre es: "

**$password =** Read-Host **-**assecurestring "Tu password: " write-host "Bienvenido $nombre"

"Te encuentras en:" gl

"usuario:" whoami

**$cad=$cadena-replace**("l","rr")

**$cad**

"contenido del archivo:" get-content **$archivo**

**$compara=**whoami

**if**(**$nombre -eq $compara**){"iguales"}**else**{"no iguales"}

**function** abre{

**param**(**$nombre**)

**if**(**$nombre -eq** "fernando"){"start notepad"}**else**{"start calc, nada igual"}

}

#invocar a la función abre(**$nombre**)

**$numero=** read-host "número: "

**switch**(**$numero**){

1. {" abre"}
2. {" cierra"}
3. {" apaga"}

**default** {"inactivo"}

}

**try**{

"abriendo archivo, se ejecuto esto" abre(**$nombre**)

}**catch**{

"no se encuentra el archivo"

}

**martes, 5 de diciembre de 2021 20:48:04**

**Tu nombre es: : Jose Maria**

**Te encuentras en:**

**Drive : C**

**Provider : Microsoft.PowerShell.Core\FileSystem**

**ProviderPath : C:\Users**

**Path : C:\Users**

**usuario:**

**desktop-lsl2957\viudez**

**perro**

**contenido del archivo:**

**get-content : No se encuentra la ruta de acceso 'C:\Users\fernando\Documents\Ejemplos\servicio.txt' porque no existe.**

**no iguales**

**número: : 2**

**cierra**

**abriendo archivo, se ejecuto esto**

**start calc, nada igual**

**<#**

**NO HE LOGRADO ENTENDER LO QUE PRETENDE HACER ESTE SCRIPT**

**#>**

#10 PROCESOS

**$ubicacion=**gl

Write-Host "te encuentras en:" **$ubicacion**

Get-Help -Name Get-Process

#PROCESOS Y SERVICIOS

#############################################################################3

#Get-Process | ForEach-Object {Write-Host $\_.name -foregroundcolor cyan}

#write-host "Algo"

#############################################

#$a = (get-date).day

#$a = (get-date).dayofweek

#$a = (get-date).dayofyear

#$a = (get-date).hour

#$a = (get-date).millisecond

#$a = (get-date).minute

#$a = (get-date).month

#$a = (get-date).second

#$a = (get-date).timeofday

#$a = (get-date).year

#get-date -DisplayHint date

#$now=Get-Date -format "dd-MMM-yyyy HH:mm"

#get-date -format g

#(get-date).dayofyear

#$a = get-wmiobject win32\_bios -computer SERVER64

#$a | format-list -property Name, @{Label="BIOS Date "; `

#Expression={$\_.ConvertToDateTime($\_.ReleaseDate)}}

#######################################################

#dir <enter>

#ls <enter>

#gci <enter>

Get-ChildItem <enter>

#asignar un alias

Set-Alias gs Get-Service <enter>

#exportar contenido a un txt

Export-Alias **-**Path Aliases.txt <enter>

#INICIAR PROCESO CON FUNCIONES

**Function** abreBloc{ start notepad

}

**Function** abreCalc{ start calc

}

#EMULAR EL COMANDO TOP DE LINUX

**while** (1) { ps | sort **-**desc cpu | select **-**first 30; sleep -seconds 2; cls }

**1023 93 169220 172024 69,91 1524 1 powershell\_ise**

**1497 97 118836 8172 7,98 4868 1 SearchApp**

**1896 73 31420 58900 4,38 4076 1 explorer**

**470 23 8652 8532 2,64 2292 1 taskhostw**

**557 18 5872 13532 2,23 3416 1 sihost**

**445 17 6560 9148 1,64 3764 1 ctfmon**

**283 17 5964 7560 1,56 4720 1 RuntimeBroker**

**595 27 12196 10672 1,52 4984 1 RuntimeBroker**

**528 23 8388 21276 1,47 3476 1 svchost**

**616 28 19868 15832 1,41 4544 1 StartMenuExperienceHost**

**340 17 4024 5852 1,27 5572 1 RuntimeBroker**

**648 48 19472 8908 1,08 6820 1 OneDrive**

**720 33 17768 1068 0,47 6896 1 SystemSettings**

**1109 45 23240 81300 0,45 1384 1 msedge**

**434 24 5008 9472 0,45 7964 1 svchost**

**301 35 7252 6320 0,36 3584 1 taskhostw**

**596 27 11060 2580 0,31 6152 1 ShellExperienceHost**

**326 16 3972 12812 0,28 3984 1 svchost**

**535 22 8996 11008 0,25 3636 1 TextInputHost**

**533 38 23248 7096 0,23 4736 1 Microsoft.Photos**

**252 13 2364 3808 0,23 6624 1 VBoxTray**

**292 14 4424 7692 0,20 3432 1 svchost**

**400 19 106996 41164 0,14 4956 1 msedge**

**225 14 2844 88 0,14 5956 1 MusNotifyIcon**

**273 13 2864 3432 0,11 4684 1 RuntimeBroker**

**233 13 7236 9636 0,11 1120 1 conhost**

**219 16 3272 3704 0,09 2640 1 dllhost**

**215 11 2632 3292 0,08 6272 1 RuntimeBroker**

**322 19 6800 3444 0,08 6116 1 ApplicationFrameHost**

**320 18 8424 29476 0,06 3572 1 identity\_helper**

**<#**

**NO HE LOGRADO ENTENDER LO QUE PRETENDE HACER ESTE SCRIPT**

**#>**

#ENVIAR CORREO

**$filename =** “c**:\**scripts\_scott**\**test9999.xls”

**$smtpserver =** “smtp.gmail.com”

**$msg =** new-object Net.Mail.MailMessage

**$att =** new-object Net.Mail.Attachment(**$filename**)

**$smtp =** new-object Net.Mail.SmtpClient(**$smtpServer** )

**$smtp**.EnableSsl **= $True**

**$smtp**.Credentials **=** New-Object System.Net.NetworkCredential(“username”, “password\_here”); # Put username without the @GMAIL.com or – @gmail.com

**$msg**.From **=** “[username@gmail.com](mailto:username@gmail.com)”

**$msg**.To.Add(”[boss@job.com](mailto:boss@job.com)”)

**$msg**.Subject **=** “Monthly Report”

**$msg**.Body **=** “Good Morning, Last month’s LOGINS & GROUPCALLS **FOR** ALL GIDS **IN** SYSTEM IS ATTACHED”

**$msg**.Attachments.Add(**$att**)

**$smtp**.Send(**$msg**)

**<#**

**NO HE LOGRADO ENTENDER EJECUTAR ESTE SCRIPT**

**#>**