# PowerShell Lab 3

In this lab you will create a simple PowerShell script and run it.

## Execution Policy

Create a simple script by putting a command (Get-Process, ls, whatever you like) into a text file. Give the file a name with the .ps1 extension.

Open PowerShell and navigate to the directory that holds your script. Try to run it by typing this (or whatever you called it)

.\script.ps1

Note that PowerShell follows the Linux convention that requires you to use the dot to explicitly say you want to run a command from the current directory. If you use Tab-complete, it will kindly supply the “.\” for you.

The command above should give you an error if your execution policy is set to the default. Check your policy with:

Get-ExecutionPolicy

Read the help for about\_Execution\_Policies and the slides in the third CyberAces Module, Syntax, Scripting, and Variables. Then change your policy (from an elevated prompt) to RemoteSigned and try to run your script again—it should work.

## Syntax, Variables, and Scripting Lesson

Reread the CyberAces Syntax, Variables, and Scripting module, and type some of the commands into PowerShell to see how they look.

## Exercise with data

Quite often in PowerShell, you are working with arrays of data output by commands. For example, you may have the users in a domain, members of groups, or email addresses and accounts from your mail system. We’ll work with some made up users, in the file users.csv from Canvas.

1. Use the cmdlet Import-Csv to load the data from users.csv into a variable (  
   $myvar = Import-Csv users.csv  
   note:  
   $myvar = Get-Content users.csv #will also get the data but will not put it into an object.  
   Import-Csv gives you a custom object  
   A screen shot of a computer

   Description automatically generated  
     
     
     
   Get-Content just gives you text  
   A screen shot of a computer

   Description automatically generated  
   )
2. Look at the structure of the variable by piping it into Get-Member. There should be three note properties if you used Import-Csv.
3. You can see all the data by typing the variable name ($myvar or whatever you called it.)
4. You can see one property of the data by typing a period after the variable name followed by the property ($myvar.property). Make a list of first names.
5. You can see the first user by using [0] ( $myvar[0] ), or the last name of the first user by typing $users[0].lastname. What is the last name of the fourth user? (Remember you start counting at zero)
6. You can see the last user by using [-1]. What is the last name of the last user?

Write a short script using either notepad, gedit, or VS Code that:

1. Imports the users.csv file into a variable
2. Sorts the variable by last name
3. Outputs the last name of the users

Hint: Pipe your variable into sort –property lastname, then pipe it into   
select -property lastname   
or  
ft –property lastname

Note: sort is an alias for Sort-Object, and select is an alias for Select-Object

# Hand in

Hand in your script and its output when users.csv is the input.

Read pages 94 – 97 (Sections 8.1 – 8.3) in the text.  Why does PowerShell use objects?