# PowerShell Lab 4

The cmdlet Where-Object is used to filter objects from the pipeline. You can think of it as selecting rows from a spreadsheet.

## Exercise with Where-Object

Print only the members of the Hogwarts domain

1. Load the file users.csv (same file as last exercise) into a variable using Import-Csv.
2. Pipe the variable into Get-Member to remind yourself of its structure.
3. Print the names of the users in the Hogwarts domain, using the method in slide 8 of the CyberAces module PowerShell—Flow Control and Output
   1. pipe the variable into Where-Object
   2. use a script block that compares the domain property of the current pipeline object to see if it is equal to the string “Hogwarts.”

## Exercise with ForEach-Object

Create email addresses for the users in the users.csv file in the form [firstname.lastname@domain.edu](mailto:firstname.lastname@domain.edu). For example, [Luke.Skywalker@starwars.edu](mailto:Luke.Skywalker@starwars.edu).

1. Load users.csv into a variable using Import-Csv
2. Pipe the variable into Foreach-Object as shown in slide 8
3. In the script block, create the email address with
   1. $\_.firstname + “.” + $\_.lastname + “@” + $\_.domain + “.edu”

This should give you a nice list of email addresses. It would be more useful if we could add an email property to our $users object. However, if you put something like this into your script block, PowerShell will complain that the property does not exist.  
$\_.email = $\_.firstname + “.” … (FAIL!)

There are several ways to get around this. One way would be to append the data to a file each time through the loop. File input and output is slow, though, so putting file output in the middle of a loop is usually a bad idea. The “PowerShell Way” is either to create a new “calculated property”, or to create a new object (variable) and then add properties to it. <https://technet.microsoft.com/en-us/library/ff730946.aspx>. We’ll leave new objects for an optional exercise.

## Exercise with Calculated Properties

A calculated property creates a name/value pair using the syntax  
@{ Name = ''; Expression = {}}

For the exercise on ForEach-Object, we iterated through the list of users and computed an email object for each one. We can get the same result without a loop using a calculated property. In this case, the Name in the calculated property could be “email” (or “emailAddress”, or whatever you want.) Inside the curly braces for Expression, you want to use the same code you used in step 3) above.

To do the same thing we did with ForEach-Object, but with a calculated property, do this.

1. Use Import-Csv to load the user data
2. Pipe the output of into Select-Object

Import-Csv users.csv | Select-Object -Property <<your calculated property goes here>>

Now modify your command so it outputs both the LastName and the calculated email address.

1. Pipe the output of your command into Get-Member and see how the properties look. It should be different when you piped the output of Import-Csv .\users.csv into Get-Member at the beginning of the lab. Why?

# Hand In

Hand in screenshots of your command and results for each exercise, Where-Object, ForEach-Object, and calculated properties.