# 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 page 7 of the 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.”
   3. hint: $\_.domain
   4. hint: comparison operators are on page 3 of the module

## 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 to create a new object (variable) and then add properties to it. <https://technet.microsoft.com/en-us/library/ff730946.aspx>

## Optional Exercise Creating New Objects

We will create a new object that has an email property in addition to the domain, firstname, and lastname properties. This is most easily done in a script, or in the ISE.

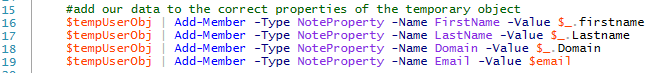
First we’ll need to create an empty object (variable) to save our new data. It is an array because it will be an array of objects—one object for each user. The “@” means array, and “()” says it’s empty, for now.  
$ $usermail = @()

Inside our Foreach-Object loop, we’ll use this to temporarily save the email address we generate.  
$email = $\_.firstname + “.” + $\_.lastname + “@” + $\_.domain + “.edu”

Then we’ll create a temporary object to hold the lastname, firstname, domain, and email, and assign the values we want. For example for the first user, domain is Starwars, email is [Luke.Skywalker@Starwars.com](mailto:Luke.Skywalker@Starwars.com), lastname is Skywalker, and firstname is Luke.

This creates the object.  
$tempUserObj = New-Object System.Object

These lines showing $tempUserObj | Add-Member … add the values to the new object. (It’s shown in a screenshot so you can’t copy and paste the entire script. After all, ISE has tab-complete so it’s not too bad.)

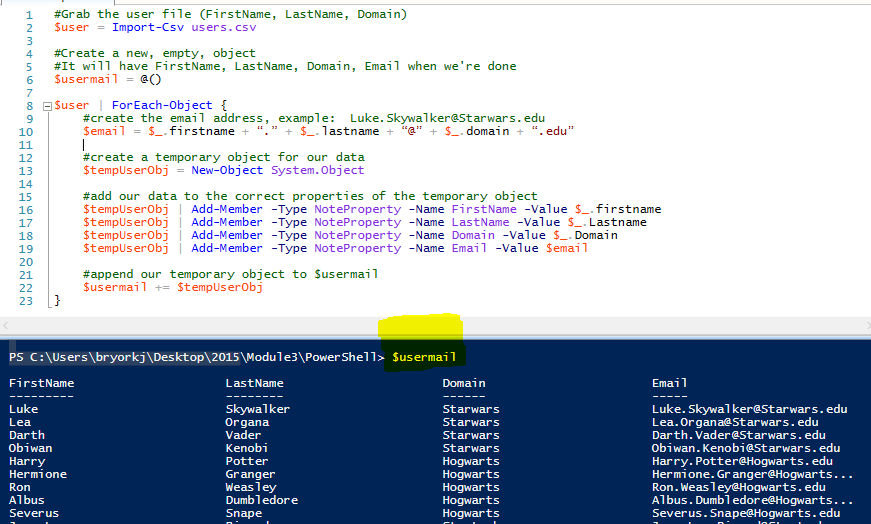


Finally we’ll append our temporary object to the $usermail array of objects.  
$usermail += $tempUserObj

When this is done, we have a new object (variable) that includes the properties we had before plus an email address. We can sort it, use Where-Object to select certain users, export it to a .csv file, and all will have our original properties plus the new email address we’ve created.

To see what $usermail holds, you can just type $usermail (after the script has run successfully.)

Here’s the entire script.



## Do this

Sort $usermail by LastName, and output the results. Export $usermail to a .csv file and open the result in a text editor and a spreadsheet.

## Another Way—using Foreach instead of Foreach-Object

We used the pipeline to stream our $user object (variable) into the code into our block of script that made a new object that included email addresses. We used the current pipeline object, $\_, to represent the user we were working on at the moment.

There is another Foreach that works more like what you have seen in other languages. The differences are explained in <https://blogs.technet.microsoft.com/heyscriptingguy/2014/07/08/getting-to-know-foreach-and-foreach-object/>. The syntax looks like this.

Foreach ($tempVariable in $dataArray) { script block}

To use foreach instead of Foreach-Object, we need to

1. not pipe $user and change Foreach-Object to foreach ($thisUser in $user) {
2. change current pipeline object $\_ to $thisUser

The main differences between foreach and Foreach-Object are that foreach uses more memory and runs faster.

