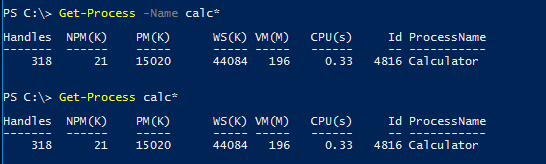
# PowerShell Lab 2

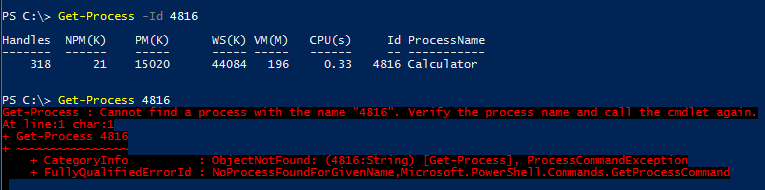
## More with help and positional parameters

It is good to be able to read the help that shows possible entries for a command, no matter what operating system you use. Here we’ll use PowerShell help as an example.

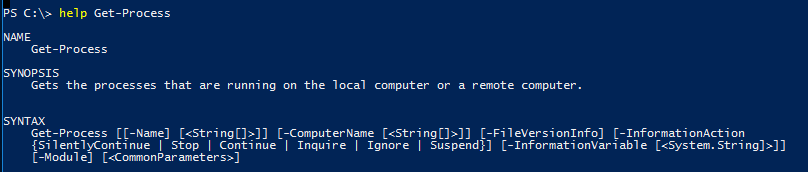
Sometimes you have to enter the parameter name with a cmdlet, and sometimes you don’t. For example, both of these commands work.



However, this command only works when the parameter name is in the command.



So, why does Get-Process calc\* work, but Get-Process 4816 doesn’t, and how do you tell the difference? We can find out using help.



Terms not enclosed in brackets are mandatory. Anything enclosed in square brackets [ ] is optional. For example, you can enter this parameter entirely,

[-ComputerName [<String[]>]]

or not at all. If you do enter -ComputerName, you can enter a string with a computer name, or not.

If you look closely, the parameter “Name” itself is also enclosed in square brackets. It is optional.

Get-Process [[-Name] [<String[]>]]

If PowerShell sees Get-Process calc\*, it knows calc\* is not a parameter name because it does not have a dash in front of it. Since the -Name is optional, it decides the string calc\* must be a name. This only works when it is the first parameter, hence “positional parameter.” Notice that the next parameter down the line only works if you enter -ComputerName; it is not enclosed in square brackets the way [-Name] is. There are cmdlets that have more than one optional parameter. In that case the values have to be entered in the same position, or order, that they are shown in help

For Get-Process, none of the other parameters have the parameter name (as opposed to the parameter value) in square brackets, so “Name” is the only positional parameter for this command.

## Exercise with positional parameters

Look at the help for Get-ChildItem, also known by its aliases, gci, dir and ls. Change directory (or Set-Location) to the root of the C:\ drive. Use Get-ChildItem (or an alias) to find all the files and directories in C:\Windows sub-directory that start with the letter “e”, using the parameters “Path” and “Filter”. Execute the command once with parameter names and once without. Note: you can do the same thing without “Filter” if you put the wildcard in the path. 1) Hand in a screenshot of your results.

## Exercise with Pipes

Analyze the following command (it is just one line, too long to fit on the page), and say what the command does before you run it. Note: parameters without values, like CaseSensitive, are called “switches.” They cause something to happen or not, depending on whether or not they are present.

PS C:\> Get-ChildItem c:\windows\system32\\*.txt | Select-String -Pattern "

Microsoft" -CaseSensitive

## Exercise with regular expressions

Like most modern shells and languages, PowerShell can use regular expressions.

Examine the regex "\w+@[a-zA-Z\_]+\.[a-zA-Z]{2,6}" to see if you can determine what it matches. Then test your guess by running it against the file sample-addresses.txt (from Canvas) with Select-String and the regex as the value for the parameter –Pattern. Note: Select-String is not a direct replacement for Linux grep, but it has similar capabilities.

Now, do the same with these patterns:

(\d{1,3}\.){3}\d{1,3}  
([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2}  
([0-9a-fA-F]{2}[:-]){5}[0-9a-fA-F]{2}

Make a regex that will grab the URL from sample-addresses.txt. There are lots of ways to do it, you don’t need to get fancy.

Note: \w means any word character, normally alphanumeric. This includes digits, as well as many math symbols and various other symbols.

Note: \d is the same as [0-9] and \s is any whitespace character. \D means not(\d), ie not a digit, and \S means not(\s) or not whitespace.

This site is handy for regular expression information in PowerShell:  
<http://www.powershellcookbook.com/recipe/qAxK/appendix-b-regular-expression-reference>