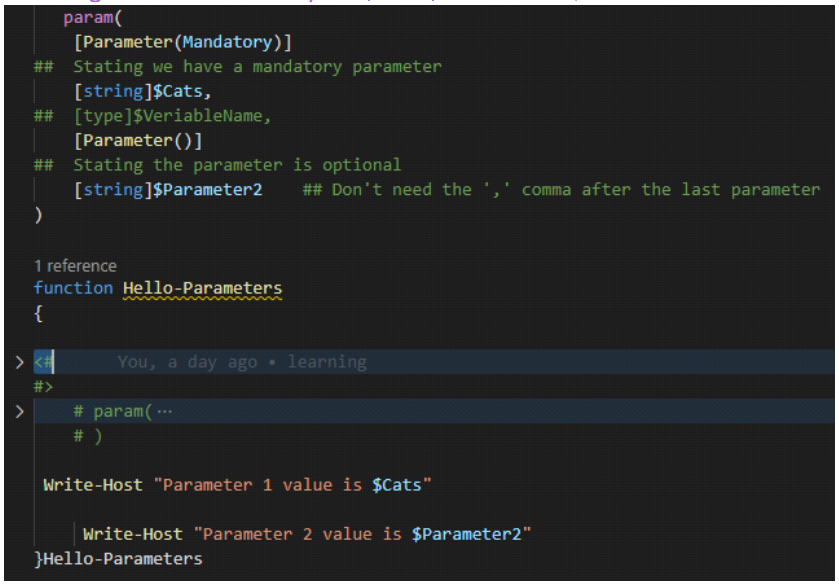
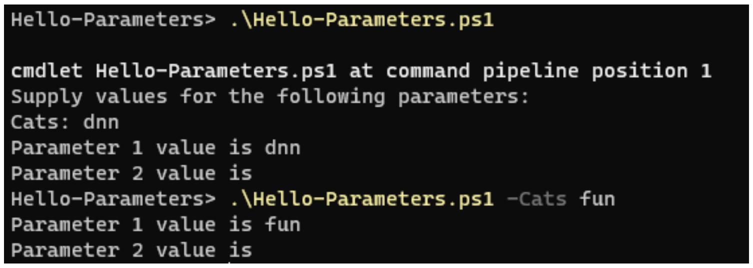
Parameters

Tuesday, 11 August, 2020 8:17 PM

How Parameters in scripts works, in my mind.

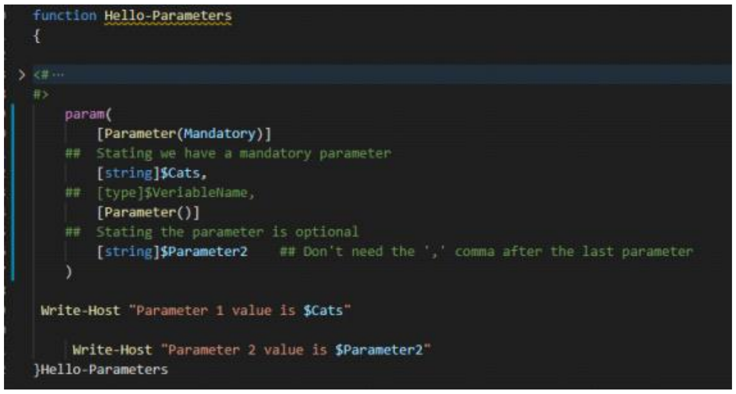
- Functions need to load before they can be used. This needs to be understood.
- When you add the parameters before the function block then you can call parameters with the script because the system will match the parameter while it loads the function. Then when running the function it says oh, wait, I have this $Parameter over here
    - 

```
  param(
    [Parameter(Mandatory)]
##  Stating we have a mandatory parameter
    [string]$Cats,
##  [type]$VeriableName,
    [Parameter()]
##  Stating the parameter is optional
    [string]$Parameter2    ## Don't need the ',' comma after the last parameter
)

1 reference
function Hello-Parameters
{

<#        You, a day ago • learning
#>
    # param(⋯
    # )

 Write-Host "Parameter 1 value is $Cats"

     Write-Host "Parameter 2 value is $Parameter2"
}Hello-Parameters
```

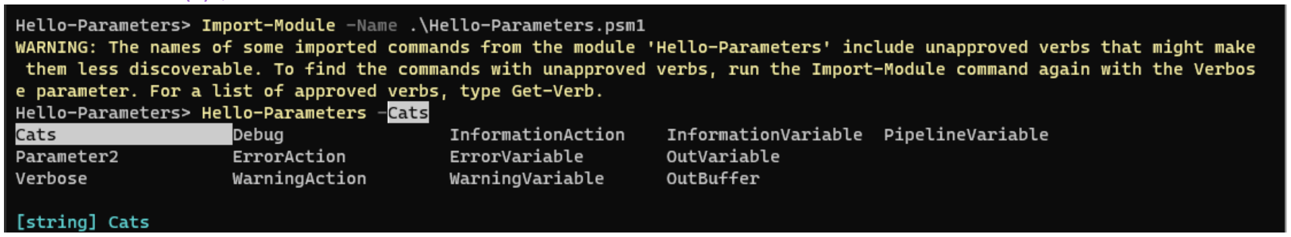
    - This above script works both when assigning -Parameter(s) $Value when calling the script
    - Or
    - By filling in the answers when the script calls for the information. A bit more inter active.
        - 

```
Hello-Parameters> .\Hello-Parameters.ps1

cmdlet Hello-Parameters.ps1 at command pipeline position 1
Supply values for the following parameters:
Cats: dnn
Parameter 1 value is dnn
Parameter 2 value is
Hello-Parameters> .\Hello-Parameters.ps1 -Cats fun
Parameter 1 value is fun
Parameter 2 value is
```

- When you add the parameter with-in the function the script/shell needs to load the function and cannot yet read the parameters with-in it. Thus you cannot run and call the parameter at the same time. Here you are loading and calling the function in the same step.

- 

```
function Hello-Parameters
{

> <#···
#>
    param(
        [Parameter(Mandatory)]
    ##  Stating we have a mandatory parameter
        [string]$Cats,
    ##  [type]$VeriableName,
        [Parameter()]
    ##  Stating the parameter is optional
        [string]$Parameter2    ## Don't need the ',' comma after the last parameter
    )

 Write-Host "Parameter 1 value is $Cats"

     Write-Host "Parameter 2 value is $Parameter2"
}Hello-Parameters
```

- However, when it comes to module's you add the parameters in the function.
    - With modules you either Import-Module -Name $Name Or you add them to the model folder for your version of PowerShell.
    - In this case the function is loaded when you start your PowerShell session. So the system knows that it has a function that goes by this-name and needs/options are -Parameter(s) $Value
    - 

```
Hello-Parameters> Import-Module -Name .\Hello-Parameters.psm1
WARNING: The names of some imported commands from the module 'Hello-Parameters' include unapproved verbs that might make
 them less discoverable. To find the commands with unapproved verbs, run the Import-Module command again with the Verbos
e parameter. For a list of approved verbs, type Get-Verb.
Hello-Parameters> Hello-Parameters -Cats
Cats                Debug               InformationAction   InformationVariable  PipelineVariable
Parameter2          ErrorAction         ErrorVariable       OutVariable
Verbose             WarningAction       WarningVariable     OutBuffer

[string] Cats
```

    - What is really nice about adding the parameters to your script is how they make it easier to use as you can see the options and, if done correctly, accept values from the pipe '|'. (I'm looking into this next.)
    - When you look into automate the process the function is already there ready for use. You are now looking for an automatized process for