Writing a Script

* Scripts are PowerShell commands stored in a file
  + .ps1 file extension
* Start with Get-Process script

Create new script

Get-Process

Save script

Check execution policy

* You can run PowerShell scripts by
  + Entering the entire path
  + Change to directory and use .\
    - Tab autofill will put .\
  + Right-click on script in file explorer -> Run with PowerShell

Double-click on script in explorer [fail]

test.ps1 from console [fail]

.\test.ps1 [success]

* Sort results descending by CPU

Get-Process | Sort-Object -Descending -Property CPU

* Filter results to top 10 and limit fields

Get-Process |

Sort-Object -Descending -Property cpu |

select -first 10 processname, id, cpu

* Add ability to choose how many

param(

$top

)

Get-Process |

Sort-Object -Descending -Property cpu |

select -first $top processname, id, cpu

* Show properties available for Get-Process

Get-Process | Get-Member

* Add ability to choose property

param(

$category,

$top

)

Get-Process |

Sort-Object -Descending -Property $category |

select -first $top processname, id, $category

* Modify parameter properties

param(

[parameter(mandatory=$true)][string]$category,

[int]$top = 10

)

* Format output

$format = @{label="Memory(Mb)";expression={[int]($\_.ws/1mb)}}

* Use IF statement to fix formatting

if ($category -eq "ws"){

$format = @{label="Memory(Mb)";expression={[int]($\_.ws/1mb)}}

} else {

$format = $category

}

* Probably should use SWITCH but for now fix IF

if ($category -eq "ws") {

$format = @{label="Memory(Mb)";expression={[int]($\_.ws/1mb)}}

} elseif ($category -eq "pm") {

$format = @{label="Memory(Mb)";expression={[int]($\_.pm/1mb)}}

} else {

$format = @{label=$category;expression={[int]($\_.$category)}}

}