Scripts

First example

```
get-process | where { $_.Handles -gt 750 } | Sort CPU -Descending | select -First 10

get-process | where { $_.ProcessName -eq "chrome" } | select -First 10
```

Powershell basics

```
#POWERSHELL BASICS


<# a multiline comment

- Single line comments start with a pound sign(#)

- Not case sensitive, like vbscript

#>

<#
VARIABLES

Variables must start with a dollar sign($) and are loosely typed.  They can
store data types, such as integers, strings, bools, arrays, objects, etc...
You can reassign a variable at will, without getting an error.

Automatic variables are premade, such as $true, $false, $null

#>


# Examples of Variables
$a = 12                          # integer
$a = "Word"                      # string
$a = @(12, "Word")               # array
$a = Get-ChildItem C:\Windows    # object
$a = Get-Date                    # datetime

<#
OBJECTS

Everything in Powershell is an object, so they have properties, methods, and
events.  These depend on what type of object is used.  You will seldom use
events, so we will skip them.
#>

#Example of an object
$processName = "chrome"; # string object
$processName.length; # length is a method
#returns 6

$processName.Replace("c", "C") # replace is a method
#returns "Chrome"

<#
LOGIC STATEMENTS

Allow you to control the flow of your program.  You use operators
you might not be familiar with, such as = and <>.  Powershell uses
operators that start with a dash(-), such as -eq, -ne, -gt, -ge, -lt, -lte, -contains
#>

# if then statment
if ($color -eq "blue") {
    "Match"
}
else {
```

```
    "No Match"
}

#prints Match

# There is also a foreach and loop statements
foreach ($item in $collection) {
    #do something with $item
}

# For loop, often used with arrays
for ($i = 0; $i -lt $array.Count; $i++) {
    #do something
}

# While loop, runs until the condition is no longer true
while (condition) {

}

<#
DATA STRUCTURES

Allow you to store and organize data.

#>

# Array

$colors = @('red', 'blue', 'green');

# Arrays use zero-based index.  So red index is 0, blue = 1, and green = 2
$colors[0]
# returns red, the first element

# There is also an Hashtable (or dictionary)

#Hashtable
$hash = @{ Number = 1; Shape = "Square"; Color = "Blue" }

# Access the elements by name
$hash["Number"]
#returns 1

<#
COMMANDS

Two flavors functions and cmdlets.  The main difference is that functions are written
in
Powershell and cmdlets are built in another language, but are accessible in
Powershell.
#>

#Command
Get-ChildItem

#Command w/ parameters
Get-ChildItem -Exclude *.txt

# Aliases are short cut names or characters.  Dir is the alias for get-childitem Use
get-alias to see them all
dir -Exclude *.txt

# Discovering commands
get-command

get-command *item*

# Getting help
get-help get-item

# Examples of commands you might use
```

```powershell
# Get free space using WMI
Get-CimInstance -ClassName Win32_LogicalDisk

# Get a list of app pools in IIS
Get-IISAppPool
```

Piping VBScript

```
set fso = CreateObject("scripting.filesystemobject")
set folder = fso.getfolder("d:\syslog")
for each file in folder.files
    on error resume next
    if lcase(left(file.name,10)) = "currentlog" then
        file.name = "ArchiveLog.txt"
    end if
next

set folder = nothing
set fso = nothing
```

Piping Powershell Scripts

```powershell
# Rename a log file
get-childitem -Path "d:\syslog" | Where-Object { $_.Name.StartsWith("CurrentLog") } |
Rename-Item -NewName "ArchiveLog"

# Finding users who haven't logged on in 90 days
search-adaccount -accountinactive -datetime "1/1/2020" -usersonly | disable-adaccount

# Listing IP Addresses for a Computer
Get-CimInstance -Class Win32_NetworkAdapterConfiguration -Filter IPEnabled=$true |
  Select-Object -ExpandProperty IPAddress
```

Sample Log

```
66.249.65.107 - - [08/Oct/2007:04:54:20 -0400] "GET /support.html HTTP/1.1" 200 11179
"-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
111.111.111.111 - - [08/Oct/2007:11:17:55 -0400] "GET / HTTP/1.1" 200 10801
"http://www.google.com/search?q=log+analyzer&ie=utf-8&oe=utf-8
&aq=t&rls=org.mozilla:en-US:official&client=firefox-a" "Mozilla/5.0 (Windows; U;
Windows NT 5.2; en-US; rv:1.8.1.7) Gecko/20070914 Firefox/2.0.0.7"
111.111.111.111 - - [08/Oct/2007:11:17:55 -0400] "GET /style.css HTTP/1.1" 200 3225
"http://www.loganalyzer.net/" "Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US;
rv:1.8.1.7) Gecko/20070914 Firefox/2.0.0.7"
```

Regular Expressions

```powershell
# Find the ip address and when it was used from our log file
$filename = "d:\Projects\MTUG Talk\samplelog.txt";
$regex = '^(\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b) - - \[(.*)\]'
select-string -Path $filename -Pattern $regex -AllMatches | foreach-object {
$_.Matches } | foreach-object { $_.Value }
```

Error Handling

```powershell
# Handle errors gracefully
try {
    $result = 10 / 0;
    $result
}
catch {
```

```
    "Oops there was a error"
}
finally {
    "Hey I run regardless"
}
```

Remote Scripts

```
# Run a script remotely to disable hearts on pcs
invoke-command -computerName PC1, PC5, PC12 -ScriptBlock {
    Disable-WindowsOptionalFeature -FeatureName "Hearts"
}

# Microsoft Article on Security Considerations with Remote Scripting -
https://docs.microsoft.com/en-
us/powershell/scripting/learn/remoting/winrmsecurity?view=powershell-7
```

Complex Script

```
#More complex script

# Creating a job in task scheduler
$action = New-ScheduledTaskAction -Execute 'Powershell.exe' `
    -Argument '-NoProfile -WindowStyle Hidden -command "& {get-eventlog -logname
Application -After ((get-date).AddDays(-1)) | Export-Csv -Path c:\fso\applog.csv -
Force -NoTypeInformation}"'

$trigger =  New-ScheduledTaskTrigger -Daily -At 9am

Register-ScheduledTask -Action $action -Trigger $trigger -TaskName "AppLog" -
Description "Daily dump of Applog"

#Example from https://devblogs.microsoft.com/scripting/use-powershell-to-create-
scheduled-tasks/

# Exporting Service Data to Excel
get-service | select-object -Property Name, Status, @{ Name = 'Timestamp'; Expression
= { Get-Date -Format 'MM-dd-yy hh:mm:ss' }} |
Export-Excel 'd:\projects\mtug talk\ServiceStates.xlsx' -WorksheetName 'Services'
```