```
 ____    __________________________________________________________________________________________________
//¯¯\\__//¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\\___
\\__//¯¯¯ Publishing [~] New-TranscriptionCollection                                              ___//¯¯\\
 ¯¯¯\\__________________________________________________________________________________________//¯¯\\__//
     ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯    ¯¯¯¯
```

```
\________________________________________________________________________________________________________________________/
  Introduction /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

So, in this document, I will cover a script called “New-TranscriptionCollection” which can be found here:
https://github.com/mcc85s/FightingEntropy/blob/main/Scripts/New-TranscriptionCollection.ps1

The purpose of this script, is to provide a transcription of an audio file.
This is NOT using an API to translate the words to text.

That is all done manually, as needed.
THOUGH TO BE PERFECTLY CLEAR...?
That API service COULD be used, to implement this script, audio to text.

Meaning, timing, positioning, phrasing, punctuation...

What I will do, is paste the function wrapper in the block below, and then I will cover the individual classes.

```
                                                                                                          ______________/
\________________________________________________________________________________________________________/ Introduction
  Script /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯
```

```powershell
<#
.SYNOPSIS
.DESCRIPTION
.LINK
.NOTES

 //==================================================================================================\\
//  Script                                                                                            \\
\\  Date       : 2023-03-01 09:12:48                                                                  //
 \\==================================================================================================//

    FileName   : New-TranscriptionCollection
    Solution   : [FightingEntropy()][2022.12.0]
    Purpose    : For categorizing a transcription of audio recordings
    Author     : Michael C. Cook Sr.
    Contact    : @mcc85s
    Primary    : @mcc85s
    Created    : 2023-03-01
    Modified   : 2023-03-01
    Demo       : N/A
    Version    : 0.0.0 - () - Finalized functional version 1
    TODO       : N/A

.Example
#>

Function New-TranscriptionCollection
{
    [CmdLetBinding()]Param(
    [Parameter(Mandatory)][String]$Name,
    [Parameter()][String]$Date)

    <# Classes go here #>

    If (!$Date)
    {
        $Date = [DateTime]::Now.ToString("MM/dd/yyyy")
    }

    [TranscriptionCollection]::New($Name,$Date)
}
```

```
                                                                                                                       ________/
\_____________________________________________________________________________________________________________________/ Script
  Class [TranscriptionDateTime] /--------------------------------------------------------------------------------------\
/------------------------------
```

This class is meant to turn a date/time string, into a [DateTime] object, and then to produce a [String] for the $Date and a [String] for the $Time, and then a [TimeSpan] object for $Position. This class ALSO allows for the position to be changed, and then tracked with the original object as the input object, in the second constructor.

```
Class TranscriptionDateTime
{
    [DateTime] $DateTime
    [String]       $Date
    [String]       $Time
    [TimeSpan] $Position
    TranscriptionDateTime([String]$Date,[String]$Time)
    {
        $This.Position = [TimeSpan]"00:00"
        $This.DateTime = [DateTime]"$Date $Time"
        $This.SetDateTime()
    }
    TranscriptionDateTime([Switch]$Flags,[Object]$Start,[String]$Position)
    {
        $This.Position = [TimeSpan]$Position
        $Current       = $Start.DateTime + $This.Position
        $This.DateTime = $Current
        $This.SetDateTime()
    }
    SetDateTime()
    {
        $This.Date     = $This.DateTime.ToString("MM/dd/yyyy")
        $This.Time     = $This.DateTime.ToString("HH:mm:ss")
    }
    [String] ToString()
    {
        Return $This.DateTime.ToString("MM/dd/yyyy HH:mm:ss")
    }
}
```

```
                                                                                              ______________________________/
\____________________________________________________________________________________________/ Class [TranscriptionDateTime]
  Class [TranscriptionParty] /-----------------------------------------------------------------------------------------\
/---------------------------
```

This is meant to represent a particular person within an audio transcription.

```
Class TranscriptionParty
{
    [Int32]    $Index
    [String]    $Name
    [String] $Initial
    TranscriptionParty([Int32]$Index,[String]$Name)
    {
        $This.Index   = $Index
        $This.Name    = $Name
        $This.Initial = ($Name -Split " " | % { $_[0] }) -join ''
    }
    [String] ToString()
    {
        Return $This.Initial
    }
}
```

```
                                                                                                 ___________________________/
\_______________________________________________________________________________________________/ Class [TranscriptionParty]
  Class [TranscriptionEntry] /-----------------------------------------------------------------------------------------\
/---------------------------
```

This is an extension of the actual string input of a particular party speaking, or annotation/context.

```
    Class TranscriptionEntry
    {
        [UInt32]         $Index
        [Object]         $Party
        [String]          $Date
        [String]          $Time
        [String]      $Position
        [String]      $Duration
        [String]          $Type
        [String]          $Note
        TranscriptionEntry([UInt32]$Index,[Object]$Party,[Object]$Position,[String]$End,[String]$Note)
        {
            $This.Index    = $Index
            $This.Party    = $Party
            $This.Date     = $Position.Date
            $This.Time     = $Position.Time
            $This.Position = $Position.Position
            $This.Duration = [TimeSpan]$End - $This.Position
            $This.Type     = Switch -Regex ($Note)
            {
                "^\*{1}" { "Action"    }
                "^\:{1}" { "Statement" }
                "^\#{1}" { "Context"   }
            }
            $This.Note     = $Note.Substring(1)
        }
        [String] ToString()
        {
            Return "[{0}] <{1}> {2}" -f $This.Time,$This.Party.Initial, $This.Note
        }
    }
```

```
                                                                                            ____________________________/
\___________________________________________________________________________________________/ Class [TranscriptionEntry]
  Class [TranscriptionFile] /----------------------------------------------------------------------------------------------\
/---------------------------
```

This object is meant to hold the metadata for a particular audio file, and allows time and tracking information to be input, as well as party and entry positioning.

```
    Class TranscriptionFile
    {
        [UInt32]       $Index
        [String]        $Name
        Hidden [Object] $Time
        [String]        $Date
        [String]       $Start
        [String]         $End
        [String]    $Duration
        [String]         $Url
        [Object]       $Party
        [Object]      $Output
        TranscriptionFile([UInt32]$Index,[String]$Name,[String]$Date,[String]$Start,[String]$D,[String]$Url)
        {
            $This.Index    = $Index
            $This.Name     = $Name
            $This.Time     = $This.DateTime($Date,$Start)
            $This.Date     = $This.Time.Date
            $This.Start    = $This.Time.Time
            $This.Duration = $This.Position($D)
            $This.End      = ([DateTime]"$Date $Start" + [TimeSpan]$D).ToString("HH:mm:dd")
            $This.Url      = $Url
            $This.Party    = @( )
            $This.Output   = @( )
        }
        [String] Position([String]$In)
        {
            $Out = Switch -Regex ($In)
            {
                "^\d{1}:\d{2}$" { "00:0$In"}
                "^\d{2}:\d{2}$" { "00:$In" }
                Default { $In }
            }

            Return $Out
        }
        [Object] DateTime([String]$Date,[String]$Start)
        {
            Return [TranscriptionDateTime]::New($Date,$Start)
        }
        [Object] GetPosition([String]$Position)
        {
            $Position = $This.Position($Position)
            Return [TranscriptionDateTime]::New([Switch]$True,$This.Time,$Position)
        }
        [Object] TranscriptionParty([Int32]$Index,[String]$Name)
        {
            Return [TranscriptionParty]::New($Index,$Name)
        }
        [Object] TranscriptionEntry([UInt32]$Index,[Object]$Party,[Object]$Position,[String]$End,[String]$Note)
        {
            $xEnd = $This.Position($End)
            Return [TranscriptionEntry]::New($Index,$Party,$Position,$xEnd,$Note)
        }
        Write([String]$String)
        {
            [Console]::WriteLine($String)
        }
        AddParty([String]$Name)
        {
            If ($Name -in $This.Party.Name)
            {
                Throw "Party [!] [$Name] already specified"
            }

            $This.Party += $This.TranscriptionParty($This.Party.Count,$Name)

            $This.Write("Party [+] [$Name] added.")
        }
        AddEntry([UInt32]$Index,[String]$Position,[String]$End,[String]$Note)
        {
            If ($Index -gt $This.Party.Count)
            {
                Throw "Party [!] [$Index] is out of bounds"
            }

            $This.Output += $This.TranscriptionEntry($This.Output.Count,
                                                     $This.Party[$Index],
                                                     $This.GetPosition($Position),
                                                     $End,
                                                     $Note)

            $This.Write("Entry [+] [$($This.Output[-1].Position)] added")
        }
        X([UInt32]$Index,[String]$Position,[String]$End,[String]$Note)
        {
            $This.AddEntry($Index,$Position,$End,$Note)
        }
        [String] ToString()
        {
            Return "({0}/{1})" -f $This.Date, $This.Name
        }
    }
```

```
                                                                                             ___________________________/
\___________________________________________________________________________________________/ Class [TranscriptionFile]
```

```
  Class [TranscriptionHistoryItem] /----------------------------------------------------------------------------------\
/----------------------------------
```

This is a single item that is meant to contain a chronologically inserted transcription entry.

```
    Class TranscriptionHistoryItem
    {
        [UInt32]      $Index
        [UInt32]       $File
        [UInt32]       $Rank
        [String]      $Party
        [String]   $Position
        [TimeSpan] $Duration
        [String]       $Note
        TranscriptionHistoryItem([UInt32]$Index,[UInt32]$File,[Object]$Item)
        {
            $This.Index    = $Index
            $This.File     = $File
            $This.Rank     = $Item.Index
            $This.Party    = $Item.Party
            $This.Position = $Item.Position
            $This.Duration = $Item.Duration
            $This.Note     = $Item.Note
        }
        [String] ToString()
        {
            Return "{0}/{1}/{2}" -f $This.Index, $This.File, $This.Party
        }
    }
```

```
                                                                                                  __________________________________/
\_________________________________________________________________________________/ Class [TranscriptionHistoryItem]
  Class [TranscriptionHistoryList] /----------------------------------------------------------------------------------\
/----------------------------------
```

This is an object that is kept totally separate from the rest of the controller class, and it is meant to keep entries in chronological order, and retain their information for tracking, and party allocation.

It also has a feature to where it will adjust the position of every entry if it was inserted in a different order.

```
    Class TranscriptionHistoryList
    {
        [Object] $Output
        TranscriptionHistoryList()
        {
            $This.Output = @( )
        }
        [Object] TranscriptionHistoryItem([UInt32]$Index,[UInt32]$File,[Object]$Item)
        {
            Return [TranscriptionHistoryItem]::New($Index,$File,$Item)
        }
        Add([UInt32]$File,[Object]$Current)
        {
            $This.Output += $This.TranscriptionHistoryItem($This.Output.Count,$File,$Current.Output[-1])
        }
        Finalize()
        {
            $Time = [TimeSpan]::FromSeconds(0)

            ForEach ($Item in $This.Output)
            {
                $Item.Position = $Time.ToString()
                $Time          = $Time + [TimeSpan]$Item.Duration
            }
        }
        [String] ToString()
        {
            Return "<TranscriptionHistory>"
        }
    }
```

```
                                                                                                  __________________________________/
\_________________________________________________________________________________________________/ Class [TranscriptionHistoryList]
  Class [TranscriptionSection] /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

This is an alternate option for tracking sections of the output. It is not currently being used, though, that does NOT mean, that it cannot be used, or that it will not be used...

It just means that it is there as a placeholder for future extensions.

```
    Class TranscriptionSection
    {
        [UInt32] $Index
        [String] $Name
        [String[]] $Content
        TranscriptionSection([UInt32]$Index,[String]$Name,[String[]]$Content)
        {
            $This.Index   = $Index
            $This.Name    = $Name
            $This.Content = $Content
        }
    }
```

```
                                                                                                      ______________________________/
\_____________________________________________________________________________________________________/ Class [TranscriptionSection]
  Class [TranscriptionCollection] /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

This is the controller class, which is actually a class factory.

This class allows all of the above classes, to be accessible in the same manner as using: “using namespace.whatever.thing.object.assembly.namespace2;” in C Sharp.

```
    Class TranscriptionCollection
    {
        [String]           $Name
        [String]           $Date
        [Object]           $File
        [Object]        $History
        Hidden [Int32] $Selected
        [Object]        $Section
        TranscriptionCollection([String]$Name,[String]$Date)
        {
            $This.Name     = $Name
            $This.Date     = ([DateTime]$Date).ToString("MM/dd/yyyy")
            $This.File     = @( )
            $This.History  = $This.TranscriptionHistoryList()
            $This.Selected = -1
            $This.Section  = @( )
        }
        [Object] TranscriptionFile([UInt32]$Index,[String]$Name,[String]$D,[String]$S,[String]$L,[String]$Url)
        {
            Return [TranscriptionFile]::New($Index,$Name,$D,$S,$L,$Url)
        }
        [Object] TranscriptionParty([Int32]$Index,[String]$Name)
        {
            Return [TranscriptionParty]::New($Index,$Name)
        }
        [Object] TranscriptionEntry([UInt32]$Index,[Object]$Party,[Object]$Position,[String]$End,[String]$Note)
        {
            Return [TranscriptionEntry]::New($Index,$Party,$Position,$End,$Note)
        }
        [Object] TranscriptionHistoryList()
        {
            Return [TranscriptionHistoryList]::New()
        }
        [Object] TranscriptionSection([UInt32]$Index,[String]$Name,[String[]]$Content)
        {
            Return [TranscriptionSection]::New($Index,$Name,$Content)
        }
        Write([String]$Line)
        {
            [Console]::WriteLine($Line)
        }
        Check([UInt32]$Index)
        {
            If ($Index -gt $This.File.Count)
            {
                Throw "Invalid file index"
            }
        }
        [Object] Get([UInt32]$Index)
        {
            $This.Check($Index)

            Return $This.File[$Index]
        }
        [Object] Current()
        {
            If ($This.Selected -eq -1)
            {
                Throw "File not selected"
            }

            Return $This.File[$This.Selected]
        }
        Select([UInt32]$Index)
        {
            $This.Check($Index)

            $This.Selected = $Index
        }
        AddFile([String]$Name,[String]$Date,[String]$Start,[String]$Duration,[String]$Url)
        {
            $Item = $This.TranscriptionFile($This.File.Count,$Name,$Date,$Start,$Duration,$Url)

            $Out  = @( )
            $Out += "Added [+] File     : [{0}]" -f $Item.Name
            $Out += "          Date     : [{0}]" -f $Item.Date
            $Out += "          Duration : [{0}]" -f $Item.Duration
            $Out += "          Url      : [{0}]" -f $Item.Url
            $Out += " "

            $Out | % { $This.Write($_) }

            $This.File += $Item
        }
        AddParty([String]$Name)
        {
            $Current = $This.Current()

            If ($Name -in $Current.Party.Name)
            {
                Throw "Party [!] [$Name] already specified"
            }

            $Current.Party += $This.TranscriptionParty($Current.Party.Count,$Name)

            $This.Write("Party [+] [$Name] added.")
        }
        AddEntry([UInt32]$Index,[String]$Position,[String]$End,[String]$Note)
        {
            $Current = $This.Current()

            If ($Index -gt $Current.Party.Count)
            {
                Throw "Party [!] [$Index] is out of bounds"
            }
            $Current.Output += $This.TranscriptionEntry($Current.Output.Count,
                                                        $Current.Party[$Index],
                                                        $Current.GetPosition($Position),
                                                        $Current.Position($End),
                                                        $Note)

            $This.History.Add($This.Selected,$Current)

            $This.Write("Entry [+] [$($Current.Output[-1].Position)] added")
        }
        AddContext([String]$Note)
        {
            $Current  = $This.Current()
            $Position = $Current.Output[-1].Position

            $Current.Output += $This.TranscriptionEntry($Current.Output.Count,
                                                        $This.TranscriptionParty(-1,"T X T"),
                                                        $Current.GetPosition($Position),
                                                        $Position,
                                                        $Note)

            $This.History.Add($This.Selected,$Current)

            $This.Write("Entry [+] [$($Current.Output[-1].Position)] added")
        }
        Finalize()
        {
            $This.History.Finalize()
        }
        X([UInt32]$Index,[String]$Position,[String]$End,[String]$Note)
        {
            $This.AddEntry($Index,$Position,$End,$Note)
        }
        C([String]$Note)
        {
            $This.AddContext($Note)
        }
        [String] Pad([String]$String,[UInt32]$Mode,[UInt32]$Length)
        {
            $Item = Switch ($Mode)
            {
                0 { $String.PadRight( $Length , " " ) }
                1 { $String.PadLeft(  $Length , " " ) }
            }

            Return $Item
        }
        [String] GetTitle()
        {
            Return "{0} [~] {1}" -f $This.Name, $This.Date
        }
        [String[]] GetSection()
        {
            $Out  = @( )
            $List = $This.History.Output | ? Party -eq Txt | ? Note -match "^\="

            ForEach ($Item in $List)
            {
                $Content = $Item.Note -Split "`n"
                $Out    += $Content[1].Substring(2,($Content[1].Length-4))
            }

            Return $Out
        }
        [String[]] GetFile()
        {
            $Out         = @{ }
            $Count       = $This.File.Count
            $Depth       = @{

                Index    = ([String]$Count).Length
                Name     = ($This.File.Name | Sort-Object Length)[-1].Length
            }

            If ($Depth.Index -lt 5)
            {
                $Depth.Index = 5
            }

            If ($Depth.Name -lt 4)
            {
                $Depth.Name = 4
            }

            ForEach ($Item in $This.File)
            {
                $Line = "{0} {1} {2} {3} {4} {5}" -f $This.Pad("Index",0,$Depth.Index),
                                                     $This.Pad("Name",0,$Depth.Name),
                                                     $This.Pad("Date",0,10),
                                                     $This.Pad("Start",0,8),
                                                     $This.Pad("End",0,8),
                                                     $This.Pad("Duration",0,8)
                $Out.Add($Out.Count,$Line)

                $Line = "{0} {1} {2} {3} {4} {5}" -f $This.Pad("-----",0,$Depth.Index),
                                                     $This.Pad("----",0,$Depth.Name),
                                                     $This.Pad("----",0,10),
                                                     $This.Pad("-----",0,8),
                                                     $This.Pad("---",0,8),
                                                     $This.Pad("--------",0,8)

                $Out.Add($Out.Count,$Line)

                $Line = "{0} {1} {2} {3} {4} {5}" -f $This.Pad($Item.Index,1,$Depth.Index),
                                                     $This.Pad($Item.Name,1,$Depth.Name),
                                                     $This.Pad($Item.Date,1,10),
                                                     $This.Pad($Item.Start,1,8),
                                                     $This.Pad($Item.End,1,8),
                                                     $This.Pad($Item.Duration,1,8)

                $Out.Add($Out.Count,$Line)
                $Out.Add($Out.Count," ".PadLeft(116," "))
                $Line = "[Url]: {0}" -f $Item.Url

                $Out.Add($Out.Count,$Line)
                $Out.Add($Out.Count," ".PadLeft(116," "))
            }

            Return $Out[0..($Out.Count-1)]
        }
        [String[]] GetOutput()
        {
            $Out      = @{ }
            $Count    = $This.History.Output.Count
            $Depth    = @{

                Index  = ([String]$Count).Length
                Party  = ($This.History.Output.Party | Sort Length | Select -Unique -Last 1).Length
                Buffer = 0
            }

            If ($Depth.Index -lt 5)
            {
                $Depth.Index = 5
            }

            If ($Depth.Party -lt 5)
            {
                $Depth.Party = 5
            }

            $Depth.Buffer = $Depth.Index + $Depth.Party + 10
```

```
# Header
$Line = "{0} {1} {2} {3}" -f $This.Pad("Index",0,$Depth.Index),
                             $This.Pad("Party",0,$Depth.Party),
                            "Position",
                            "Note"
$Out.Add(0,$Line)

$Line = "{0} {1} {2} {3}" -f $This.Pad("-----",0,$Depth.Index),
                             $This.Pad("-----",0,$Depth.Index),
                            "--------",
                            "----"

$Out.Add(1,$Line)

# Content
ForEach ($X in 0..($This.History.Output.Count-1))
{
    $Item    = $This.History.Output[$X]
    $Content = $Item.Note -Split "`n"

    If ($Item.Party -eq "TXT")
    {
        $Line = "{0} {1} {2} " -f $This.Pad($Item.Index,0,$Depth.Index),
                                  $This.Pad($Item.Party,0,$Depth.Party),
                                  $Item.Position

        $Line = $Line.PadRight(116,"=")

        $Out.Add($Out.Count,$Line)
        $Out.Add($Out.Count," ".PadRight(116," "))

        ForEach ($Slice in $Content)
        {
            $Out.Add($Out.Count,"    $Slice")
        }

        $Out.Add($Out.Count," ".PadRight(116," "))
        $Out.Add($Out.Count,"=".PadRight(116,"="))
    }

    Else
    {
        If ($Content.Count -eq 1)
        {
            $Line = "{0} {1} {2} $content" -f $This.Pad($Item.Index,0,$Depth.Index),
                                              $This.Pad($Item.Party,0,$Depth.Party),
                                              $Item.Position


            $Out.Add($Out.Count,$Line)
        }
        If ($Content.Count -gt 1)
        {
            $C = 0
            ForEach ($Slice in $Content)
            {
                If ($C -eq 0)
                {
                    $Line = "{0} {1} {2} {3}" -f $This.Pad($Item.Index,0,$Depth.Index),
                                                 $This.Pad($Item.Party,0,$Depth.Party),
                                                 $Item.Position,
                                                 $Slice
                }
                Else
                {
                    $Line = "{0} {1}" -f " ".PadRight($Depth.Buffer," "),
                                        $Slice
                }

                $C ++
                $Out.Add($Out.Count,$Line)
            }
```

```
                    }
                }

                $Out.Add($Out.Count," ".PadRight(116," "))
            }

            Return $Out[0..($Out.Count-1)]
        }
        [String] ToString()
        {
            Return "({0}) <TranscriptionCollection>" -f $This.File.Count
        }
    }
```

```
                                                                                        _________________________________/
\______________________________________________________________________________________/ Class [TranscriptionCollection]
  Output /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯
```

```
    Now, time to talk about the work above.
    I'll put together information below that wasn't featured in the video where I began to create this document.

    I have many things to add to this particular script (like the actual transcriptions and annotations).
```

```
$Ctrl = New-TranscriptionCollection -Name "Michael C. Cook Sr. vs. Saratoga County"

$Ctrl.AddFile("20210201 (Family Court).mp3","02/01/2021","10:39:19","00:14:41",
              "https://drive.google.com/file/d/12rvHS3-pZ1AB8wp0EpY4aP0cFh6TgNP_")
$Ctrl.AddFile("20210406 (Family Court).wav","04/06/2021","09:19:52","00:04:08",
              "https://drive.google.com/file/d/1J0CzI1nW5xwmWbwUVwOEMbhLUiZYEr4p")
$Ctrl.AddFile("SCSO-2022-013379.mp3","03/01/2022","16:30:43","00:21:10",
              "https://drive.google.com/file/d/1BNfF9vWjG4vBIO-8oXmIw6aLeNvFRjRL")
$Ctrl.AddFile("Mom Argument-(20220404).mp3","04/04/2022","12:13:55","00:55:10",
              "https://drive.google.com/file/d/1E5ERWMgj8GkznNZ_i0bAwjppkD_sWANd")
$Ctrl.AddFile("20220623.mp3","06/23/2022","19:55:34","00:24:11",
              "https://drive.google.com/file/d/1Q5JgJ_LLf4PYsil54_hHVo90kG7gViU6")
$Ctrl.AddFile("Mom Argument-(20220628).mp3","06/28/2022","11:05:44","02:01:31",
              "https://drive.google.com/file/d/1Z56uu5O52eAzJhUdiby_J8dQQXaOUENa")
$Ctrl.AddFile("Mark Market 32-(20230122).mp3","01/22/2023","16:43:58","00:04:35",
              "https://drive.google.com/file/d/1jTXlZ5oiuS3i0EWgmuoT2SMDzbr01aHc")
$Ctrl.AddFile("Mark Market 32-(20230205).mp3","02/05/2023","11:39:11","00:33:44",
              "https://drive.google.com/file/d/1m8WkmxdKA_jHXGjqhPvaTjIk7TqjNw_p")
$Ctrl.AddFile("SCSO-2023-013374.mp3","02/26/2023","19:17:16","00:07:28",
              "https://drive.google.com/file/d/1CvP8z-AsrOUFZTV4J5Yg2Y5afkMvEmZP")

$Ctrl.Select(0)
$Ctrl.AddParty("Michael C. Cook Sr.")
$Ctrl.AddParty("Family Court Receptionist")

$Ctrl.Select(1)
$Ctrl.AddParty("Michael C. Cook Sr.")
$Ctrl.AddParty("Paul Pelagalli")
$Ctrl.AddParty("Neil Weiner")
$Ctrl.AddParty("Lisa Mentes")
$Ctrl.AddParty("Heather Corey-Mongue")
$Ctrl.AddParty("Sarah Schellinger")

$Ctrl.Select(2)
$Ctrl.AddParty("Michael C. Cook Sr.")
$Ctrl.AddParty("Paul Pecor")
$Ctrl.AddParty("Daniel Nelson")
$Ctrl.AddParty("Jeffrey Margan")

$Ctrl.Select(3)
$Ctrl.AddParty("Michael C. Cook Sr.")
$Ctrl.AddParty("Fabienne S. K. Cook")

$Ctrl.Select(4)
$Ctrl.AddParty("Michael C. Cook Sr.")
$Ctrl.AddParty("SCSO Samuel Speziale")
$Ctrl.AddParty("SCSO Jared Gardner")
```

```
$Ctrl.Select(5)
$Ctrl.AddParty("Michael C. Cook Sr.")
$Ctrl.AddParty("Fabienne S. K. Cook")

$Ctrl.Select(6)
$Ctrl.AddParty("Michael C. Cook Sr.")
$Ctrl.AddParty("Mark <last name unknown>")
$Ctrl.AddParty("Manager <name unknown>")

$Ctrl.Select(7)
$Ctrl.AddParty("Michael C. Cook Sr.")
$Ctrl.AddParty("Mark <last name unknown>")
$Ctrl.AddParty("Girl 1")
$Ctrl.AddParty("Lady 1" )
$Ctrl.AddParty("Girl 2")
$Ctrl.AddParty("Lady 2")
$Ctrl.AddParty("Manager <name unknown>")

$Ctrl.Select(8)
$Ctrl.AddParty("Michael C. Cook Sr.")
$Ctrl.AddParty("K. Rossi")
```

So, right there, that will get us the following output.

```
Added [+] File     : [20210201 (Family Court).mp3]
          Date     : [02/01/2021]
          Duration : [00:14:41]
          Url      : [https://drive.google.com/file/d/12rvHS3-pZ1AB8wp0EpY4aP0cFh6TgNP_]

Added [+] File     : [20210406 (Family Court).wav]
          Date     : [04/06/2021]
          Duration : [00:04:08]
          Url      : [https://drive.google.com/file/d/1J0CzI1nW5xwmWbwUVwOEMbhLUiZYEr4p]

Added [+] File     : [SCSO-2022-013379.mp3]
          Date     : [03/01/2022]
          Duration : [00:21:10]
          Url      : [https://drive.google.com/file/d/1BNfF9vWjG4vBIO-8oXmIw6aLeNvFRjRL]

Added [+] File     : [Mom Argument-(20220404).mp3]
          Date     : [04/04/2022]
          Duration : [00:55:10]
          Url      : [https://drive.google.com/file/d/1E5ERWMgj8GkznNZ_i0bAwjppkD_sWANd]

Added [+] File     : [20220623.mp3]
          Date     : [06/23/2022]
          Duration : [00:24:11]
          Url      : [https://drive.google.com/file/d/1Q5JgJ_LLf4PYsil54_hHVo90kG7gViU6]

Added [+] File     : [Mom Argument-(20220628).mp3]
          Date     : [06/28/2022]
          Duration : [02:01:31]
          Url      : [https://drive.google.com/file/d/1Z56uu5O52eAzJhUdiby_J8dQQXaOUENa]

Added [+] File     : [Mark Market 32-(20230122).mp3]
          Date     : [01/22/2023]
          Duration : [00:04:35]
          Url      : [https://drive.google.com/file/d/1jTXlZ5oiuS3i0EWgmuoT2SMDzbr01aHc]

Added [+] File     : [Mark Market 32-(20230205).mp3]
          Date     : [02/05/2023]
          Duration : [00:33:44]
          Url      : [https://drive.google.com/file/d/1m8WkmxdKA_jHXGjqhPvaTjIk7TqjNw_p]

Added [+] File     : [SCSO-2023-013374.mp3]
          Date     : [02/26/2023]
          Duration : [00:07:28]
          Url      : [https://drive.google.com/file/d/1CvP8z-AsrOUFZTV4J5Yg2Y5afkMvEmZP]

Party [+] [Michael C. Cook Sr.] added.
Party [+] [Family Court Receptionist] added.
Party [+] [Michael C. Cook Sr.] added.
Party [+] [Paul Pelagalli] added.
Party [+] [Neil Weiner] added.
Party [+] [Lisa Mentes] added.
Party [+] [Heather Corey-Mongue] added.
Party [+] [Sarah Schellinger] added.
Party [+] [Michael C. Cook Sr.] added.
Party [+] [Paul Pecor] added.
Party [+] [Daniel Nelson] added.
Party [+] [Jeffrey Margan] added.
Party [+] [Michael C. Cook Sr.] added.
Party [+] [Fabienne S. K. Cook] added.
Party [+] [Michael C. Cook Sr.] added.
Party [+] [SCSO Samuel Speziale] added.
Party [+] [SCSO Jared Gardner] added.
Party [+] [Michael C. Cook Sr.] added.
Party [+] [Fabienne S. K. Cook] added.
Party [+] [Michael C. Cook Sr.] added.
Party [+] [Mark <last name unknown>] added.
Party [+] [Manager <name unknown>] added.
Party [+] [Michael C. Cook Sr.] added.
Party [+] [Mark <last name unknown>] added.
Party [+] [Girl 1] added.
Party [+] [Lady 1] added.
Party [+] [Girl 2] added.
Party [+] [Lady 2] added.
Party [+] [Manager <name unknown>] added.
Party [+] [Michael C. Cook Sr.] added.
Party [+] [K. Rossi] added.

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl

Name    : Michael C. Cook Sr. vs. Saratoga County
Date    : 03/01/2023
File    : {(02/01/2021/20210201 (Family Court).mp3),
           (04/06/2021/20210406 (Family Court).wav),
           (03/01/2022/SCSO-2022-013379.mp3),
           (04/04/2022/Mom Argument-(20220404).mp3)...}
History : <TranscriptionHistory>
Section : {}

PS Prompt:\>
```

```
PS Prompt:\> $Ctrl.File

Index    : 0
Name     : 20210201 (Family Court).mp3
Date     : 02/01/2021
Start    : 10:39:19
End      : 10:54:01
Duration : 00:14:41
Url      : https://drive.google.com/file/d/12rvHS3-pZ1AB8wp0EpY4aP0cFh6TgNP_
Party    : {MCCS, FCR}
Output   : {}

Index    : 1
Name     : 20210406 (Family Court).wav
Date     : 04/06/2021
Start    : 09:19:52
End      : 09:24:06
Duration : 00:04:08
Url      : https://drive.google.com/file/d/1J0CzI1nW5xwmWbwUVwOEMbhLUiZYEr4p
Party    : {MCCS, PP, NW, LM...}
Output   : {}

Index    : 2
Name     : SCSO-2022-013379.mp3
Date     : 03/01/2022
Start    : 16:30:43
End      : 16:51:01
```

```
Duration : 00:21:10
Url      : https://drive.google.com/file/d/1BNfF9vWjG4vBIO-8oXmIw6aLeNvFRjRL
Party    : {MCCS, PP, DN, JM}
Output   : {}

Index    : 3
Name     : Mom Argument-(20220404).mp3
Date     : 04/04/2022
Start    : 12:13:55
End      : 13:09:04
Duration : 00:55:10
Url      : https://drive.google.com/file/d/1E5ERWMgj8GkznNZ_i0bAwjppkD_sWANd
Party    : {MCCS, FSKC}
Output   : {}

Index    : 4
Name     : 20220623.mp3
Date     : 06/23/2022
Start    : 19:55:34
End      : 20:19:23
Duration : 00:24:11
Url      : https://drive.google.com/file/d/1Q5JgJ_LLf4PYsil54_hHVo90kG7gViU6
Party    : {MCCS, SSS, SJG}
Output   : {}

Index    : 5
Name     : Mom Argument-(20220628).mp3
Date     : 06/28/2022
Start    : 11:05:44
End      : 13:07:28
Duration : 02:01:31
Url      : https://drive.google.com/file/d/1Z56uu5O52eAzJhUdiby_J8dQQXaOUENa
Party    : {MCCS, FSKC}
Output   : {}

Index    : 6
Name     : Mark Market 32-(20230122).mp3
Date     : 01/22/2023
Start    : 16:43:58
End      : 16:48:22
Duration : 00:04:35
Url      : https://drive.google.com/file/d/1jTXlZ5oiuS3i0EWgmuoT2SMDzbr01aHc
Party    : {MCCS, M<nu, M<u}
Output   : {}

Index    : 7
Name     : Mark Market 32-(20230205).mp3
Date     : 02/05/2023
Start    : 11:39:11
End      : 12:12:05
Duration : 00:33:44
Url      : https://drive.google.com/file/d/1m8WkmxdKA_jHXGjqhPvaTjIk7TqjNw_p
Party    : {MCCS, M<nu, G1, L1...}
Output   : {}

Index    : 8
Name     : SCSO-2023-013374.mp3
Date     : 02/26/2023
Start    : 19:17:16
End      : 19:24:26
Duration : 00:07:28
Url      : https://drive.google.com/file/d/1CvP8z-AsrOUFZTV4J5Yg2Y5afkMvEmZP
Party    : {MCCS, KR}
Output   : {}
```

```
  So, right here, I'm collecting a number of audio files that I have stored on Google Drive.
  I have already transcribed a fair number of these files.

  However, I have to provide additional details, annotations, and get them formatted to fit within the script.
  I'll cover that at some point in the future, but let's be perfectly clear here...
                                                                                                              ________/
\____________________________________________________________________________________________________________/ Output
```

```
 Conclusion /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯

   The reason why I would have to go ahead and compile a list of these particular audio files...
   ...is mainly because [I constantly run into situations where someone is stupid].

   People I run into, they have a job to do.
   They're supposed to do it correctly, and follow these things called “laws”.

   Often times what happens, is that people don't do their jobs correctly...
   ...even if I'm recording how stupid they sound in an audio recording.

   I believe it is mainly because most people think that if they lie to someone else, there's no way that anyone
   could ever detect that they are blatantly lying. [That's because they're stupid].

   They get EXTREMELY cocky in some cases, where they don't realize that I'm building a very serious case against
   [Saratoga County].

   With each additional recording, I'm able to compile evidence that suggests that some people get paid to do
   their jobs, but- the job isn't getting done at all.
                                                                                                            ____________/
\_________________________________________________________________________________________________________/ Conclusion
 ________________________________________________________
 |¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|
 |                                    Michael C. Cook Sr. |
 |                                      Security Engineer |
 |                                   Secure Digits Plus LLC |
 |______________________________________________________|
 ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```