Demo 1 - The Basics

Part 1:

1. Cmdlets
2. Variables
3. String interpolation

```
dir
cls
Get-Date
$now = Get-Date
"Now is $now, aon"
"Now is $(Get-Date)"
```

Part 2:

1. .NET integration

```
$now.ToString("d")
$now.GetType()
$tomorrow = $now.AddDays(1)
"Now is $now, tomorrow will be $tomorrow"
```

Part 3:

1. .NET integration
2. Type discovery
3. Streaming pipeline

```
$desktop = New-Object System.IO.DirectoryInfo("C:\Users\Enrico\Desktop")
$desktop.Exists
$desktop.EnumerateFiles()
$desktop | Get-Members
$desktop.EnumerateFiles() | Get-Members

Add-Type -AssemblyName System.Windows.Forms
[System.Windows.Forms.MessageBox]::Show("Does this feel right?", "WinForms in PowerShell", 4, "Question")
```

Part 4:

1. Piping
2. Filtering and sorting
3. Conversion constants
4. Invoking static methods
5. Formatting output

```
Get-Process
Get-Process | where { $_.WorkingSet -gt 50000000 }
Get-Process | where { $_.WorkingSet -gt 50MB } | sort WorkingSet -Descending
Get-Process | where { $_.WorkingSet -gt 50MB } | sort WorkingSet -Descending | Format-Table
Get-Process | where { $_.WorkingSet -gt 50MB } | sort WorkingSet -Descending | Format-Table Name, WorkingSet
Get-Process | where { $_.WorkingSet -gt 50MB } | sort WorkingSet -Descending | Format-Table Name, { $_.Working
Get-Process | where { $_.WorkingSet -gt 50MB } | sort WorkingSet -Descending | `
Format-Table Name, @{Name="WorkingSet"; Expression={ $_.WorkingSet/1MB }}
Get-Process | where { $_.WorkingSet -gt 50MB } | sort WorkingSet -Descending | `
Format-Table Name, @{Name="WorkingSet"; Expression={"$($_.WorkingSet/1MB) MB" }}
Get-Process | where { $_.WorkingSet -gt 50MB } | sort WorkingSet -Descending | `
Format-Table Name, @{Name="WorkingSet"; Expression={"$([Math]::Floor($_.WorkingSet/1MB)) MB"}}
```

Part 5:

1. Output redirection
2. Aliases

```
Get-Service
Get-Service | where { $_.Status -eq "Stopped" }
Get-Service | where { $_.Status -eq "Stopped" } | Format-Table
Get-Service | where { $_.Status -eq "Stopped" } | Out-Host -Paging
Get-Service | where { $_.Status -eq "Stopped" } | Out-GridView

Get-Alias -Name where
Get-Alias sort
```

Part 6:

1. Dynamic typing
2. Here strings

```
$source = @"
namespace SweNug.Demo
{
    public static class BandwidthConverter
    {
        public static float ConvertMbpsToMBps(int value)
        {
            return value / 8f;
        }
    }
}
"@
Add-Type $source
[SweNug.Demo.BandwidthConverter]::ConvertMbpsToMBps(10)

$speed = 100
$speed | Get-Member
$speed = $speed | `
Add-Member -MemberType ScriptMethod -Name ToMBps `
-Value { [SweNug.Demo.BandwidthConverter]::ConvertMbpsToMBps($this) } -PassThru
$speed | Get-Member
$speed.ToMBps()
```

Demo 2 - Configuring IIS

> **Note:** Start PowerShell as Administrator and enable the execution of scripts:

```
Set-ExecutionPolicy -Scope Process Unrestricted
```

Part 1:

1. Modules
2. Discovery

```
Get-Module -ListAvailable
Import-Module WebAdministration
Get-Command -Module WebAdministration | Out-Host -Paging
Get-Command -Noun AppPool
```

Part 2:

1. Drives
2. IIS file system provider

```
Get-PSDrive
cd IIS:
ls Sites
ls AppPools
Test-Path C:\inetpub\wwwroot\Expresso
```

Part 3:

1. Creating an AppPool
2. Creating a Web Application
3. Setting properties through the IIS file system provider

```
New-WebAppPool Expresso
New-WebSite -Id 1 -Name Expresso -ApplicationPool Expresso -PhysicalPath C:\inetpub\wwwroot\Expresso
ls AppPools
Get-ItemProperty -Path AppPools\Expresso -Name managedRuntimeVersion
Set-ItemProperty -Path AppPools\Expresso -Name managedRuntimeVersion -Value v4.0
```

Part 4:

1. History
2. Redirect to file

```
Get-History | where { $_.CommandLine -ne "cls" } | Format-Table CommandLine -HideTableHeaders > DeployExpresso
```

Part 5:

1. Run the script `DeployExpresso.ps1` side to side with IIS manager to see the changes

Demo 3 - Using the Expresso Cmdlets

> **Note:** Configure the PowerShell console to run on .NET 4.0:

```
Copy-Item .\Expresso\Commands\powershell.exe.config "$env:SystemRoot\System32\WindowsPowerShell\v1.0"
```

then enable the execution of scripts:

```
Set-ExecutionPolicy -Scope Process Unrestricted
```

Part 1:

1. Modules

```
Copy-Item -Recurse -Force .\Sources\Expresso\Commands\bin\Debug\Expresso .\Documents\WindowsPowerShell\Modules
Get-Module -ListAvailable
Import-Module Expresso
Get-Command -Module Expresso
```

Part 2:

1. Retrieving posts
2. Creating posts

```
Get-Post
New-Post -Title "First Post" -Content "This is the first post." -Tags @("133t", "awesome")
Get-Post
```

Part 3:

1. Converting posts to CSV
2. Exporting and importing posts from file

```
Get-Post | Export-CSV .\Desktop\Posts.csv
Import-CSV .\Desktop\Posts.csv
Import-CSV .\Desktop\Posts.csv | New-Post -Tags @("133t", "awesome")
```