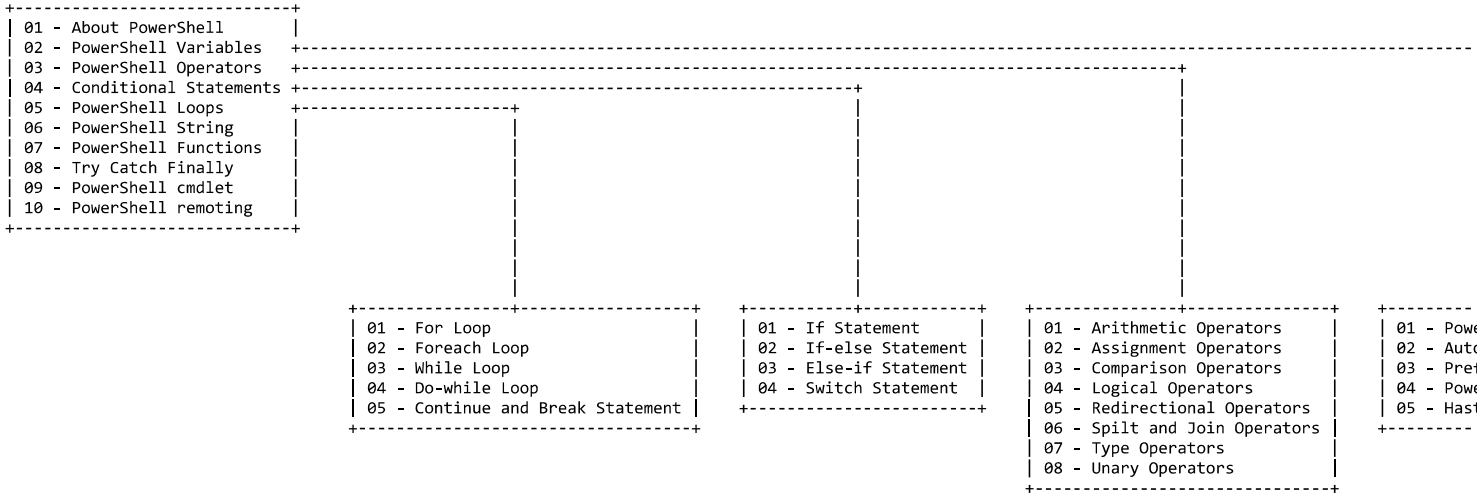
```
  //======================\\
 ||   PowerShell Tutorial   ||
  \\======================//
```



01 - About PowerShell

Three Core Cmdlets

01: Get-Command
02: Get-Help
03: Get-Member (click here)

Examples:

```
Get-Command -Noun Process
Get-Command -Noun *

Get-Command -Name service
Get-Command -Name *service*
```

A cmdlet is denoted as a verb-noun pair and has a .ps1 extension.
Every cmdlet has a help file and can be obtained by typing Get-Help -Detailed.

// Some of the popular cmdlets are: //

Get – To get something
Set – To define something
Start – To run something
Stop – To stop something that is running
Out – To output something
New – To create something

02 - PowerShell Variables

A variable is a unit of memory in which the data is stored.
A variable starts with the dollar ($) sign, such as $a, $ab.
The name of the variables are not case-sensitive, and they include spaces and special characters.
By default, the value of all the variables in a PowerShell is $null.

01 - PowerShell Variables	click here
02 - Automatic Variables	click here
03 - Preference Variables	click here
04 - PowerShell Array	click here
05 - Hast Table	click here

Examples
01 - PowerShell Variables: $a=1, $a="Hello World!"

>> To find the type of a variable, you can use the GetType() method.

Example 1:

```
PS C:\> $a = 1
```

```
PS C:\> $a.GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     Int32                                    System.ValueType

-------------------------------------------------------------------------------
Example 2:

$a = "Hello World!"
$a.GetType()

PS C:\> $a = "Hello World!"
PS C:\> $a.GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     String                                   System.Object
```

03 - PowerShell Operators

01 - Arithmetic Operators
02 - Assignment Operators
03 - Comparison Operators
04 - Logical Operators
05 - Redirectional Operators
06 - Spilt and Join Operators
07 - Type Operators
08 - Unary Operators

04 - Conditional Statements

01 - If Statement
02 - If-else Statement
03 - Else-if Statement
04 - Switch Statement

05 - PowerShell Loops

01 - For Loop
02 - Foreach Loop
03 - While Loop
04 - Do-while Loop
05 - Continue and Break Statement

06 - PowerShell String

A String can be defined in PowerShell by using the single or double-quotes.
It is a datatype that denotes the sequence of characters.
The PowerShell string is simply an object with a System.String type.

Examples 1: Single quotes in a string

```
PS C:\> $string_1='Hello World!'
PS C:\> $string_1
Hello World!
```

Examples 2: Double quotes in a string

```
PS C:\> $string_2="Hello World!"
PS C:\> $String_2
Hello World!
```

Examples 3: Concatenation the two string variables:

```
PS C:\> $string_1="Hello"
PS C:\> $string_2="World!"
PS C:\> $string_1+$string_2
HelloWorld!
PS C:\> $string_1 + $string_2
HelloWorld!
PS C:\> $string_1+" "+$string_2
```

```
Hello World!
```

Examples 3: Method concat() to concatenate the strings

```
PS C:\> $string_1="Hello"
PS C:\> $string_2="World!"
PS C:\> [System.String]::Concat($string_1,$string_2)
HelloWorld!

PS C:\> $string_1="Hello"
PS C:\> $string_2=" World!"
PS C:\> [System.String]::Concat($string_1,$string_2)
Hello World!
```

Examples 3: SubString()

```
The following example skips the first one character and returns the next three-character from the given string.
PS C:\> $string_1="Hello World!"
PS C:\> $string_1.SubString(1,3)
ell

PS C:\> $string_1.SubString(0,3)
Hel
PS C:\> $string_1.SubString(1,3)
ell
PS C:\> $string_1.SubString(2,3)
llo
```

Examples 4:String formatting is a process to insert some characters or string inside a string. We can format the string by using the -f operator.

```
PS C:\> $string_1="Hello Friend, I love"
PS C:\> $string_2="Windows PowerShell"
PS C:\> $string_3="Scripting"
PS C:\> $string_4 = "Hi, {0} {1}....{2}......" -f $string_1,$string_2,$string_3
PS C:\> $string_4
```

Examples 5: The replace() method accepts the two arguments and is used to replace the characters in a string.

```
PS C:\> $string_1="Hello World!"
PS C:\> $string_1.replace("o","AA")
HellAA WAArld!
```

07 - PowerShell Functions

08- Try Catch Finally

https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_try_catch_finally?view=powershell-7.3

```
# Save below code script.ps1

-------------------------------------------------------------------------
# Example 1
Write-Host "Example 1" -ForegroundColor Magenta
try { NonsenseString }
catch { "An error occurred - 1." }

# Example 2
Write-Host "Example 2" -ForegroundColor Magenta
try { $NonsenseString }
catch { "An error occurred - 2." }
Write-Host "No error found"

# Example 3
Write-Host "Example 3" -ForegroundColor Magenta
try { NonsenseString }
catch {
  Write-Host "An error occurred - 3."
  Write-Host $_
}

-------------------------------------------------------------------------

## OUTPUT ##

PS C:\roboticscript\examples> .\try_catch_finally.ps1
Example 1
An error occurred - 1.
Example 2
No error found
Example 3
An error occurred - 3.
The term 'NonsenseString' is not recognized as a name of a cmdlet, function, script file, or executable program.
Check the spelling of the name, or if a path was included, verify that the path is correct and try again.

-------------------------------------------------------------------------
```

```
| 09 - PowerShell cmdlet |
+------------------------+


PS C:\> Get-Date
26 October 2022 10:13:52 PM

PS C:\> Get-Date -Format dd-MM-yyyy_HH-mm-ssf
26-10-2022_22-13-566

PS C:\> Get-Date -Format dd-MM-yyyy_HH-mm-ssffff
26-10-2022_22-14-464717



+--------------------------+
| 10 - PowerShell remoting |
+--------------------------+


PowerShell remoting must be enabled on the remote computer. Use the Enable-PSRemoting cmdlet to enable PowerShell remoting.



PS C:\> Enable-PSRemoting

Example 1: One-To-One Remoting


PS C:\> $Cred = Get-Credential
PS C:\> Enter-PSSession -ComputerName dc01 -Credential $Cred
[dc01]: PS C:\Users\Administrator\Documents>
[dc01]: PS C:\Users\Administrator\Documents>cd \
[dc01]: PS C:\>
[dc01]: PS C:\> Get-Process | Get-Member
[dc01]: PS C:\> Exit-PSSession


Example 1: One-To-Many Remoting


PS C:\> Invoke-Command -ComputerName dc01, sql02, web01 {Get-Service -Name W32time} -Credential $Cred
```