Example of PowerShell Arrays

To create an array that holds a given set of items, separate those items with commas:

```
     PS >$myArray = 1,2,"Hello World"
     PS >$myArray
     1
     2
     Hello World
```

To create an array of a specific size, use the `New-Object` cmdlet:

```
     PS >$myArray = New-Object string[] 10
     PS >$myArray[5] = "Hello"
     PS >$myArray[5]
     Hello
```

To store the output of a command that generates a list, use variable assignment:

```
     PS >$myArray = Get-Process
     PS >$myArray


     Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
     -------  ------    -----      ----- -----   ------     -- -----------
         274       6     1316       3908    33            3164 alg
         983       7     3636       7472    30             688 csrss
          69       4      924       3332    30     0.69   2232 ctfmon
         180       5     2220       6116    37            2816 dllhost
     (...)
```

To create an array that you plan to modify frequently, use an `ArrayList`, as shown by example below:

Using an ArrayList to manage a dynamic collection of items

```
PS >$myArray = New-Object System.Collections.ArrayList
PS >[void] $myArray.Add("Hello")
PS >[void] $myArray.AddRange( ("World","How","Are","You") )
PS >$myArray
Hello
World
How
Are
You
PS >$myArray.RemoveAt(1)
PS >$myArray
Hello
How
Are
You
```

```
# More arrays
#
<#
$os=@("linux", "windows")
$os+=@("mac")
Write-Host $os[1]     # print windows
Write-Host $os        # print array values
Write-Host $os.Count  # print length of array
Write-Host "My operating system is $($os[1])" # print the os type
#>

<# ----- OUPPUT -----
windows
linux windows mac
3
My operating system is windows
#>



# demonstrate hashing
$states = @{"Washington" = "Olympia"; "Oregon" = "Salem"; California =
"Sacramento"}

$states.Add("Illinois", "Springfield")

Write-Host @states       # print everything
Write-Host $states.Keys  # print all keys
Write-Host $states.Count # print length of hash

$states.GetEnumerator() | Sort-Object Name

<# ----- OUTPUT -----
California Sacramento Washington Olympia Illinois Springfield Oregon Salem
California Washington Illinois Oregon
4

Name                           Value
----                           -----
California                     Sacramento
Illinois                       Springfield
Oregon                         Salem
Washington                     Olympia
#>
```

```
<#
Note the difference between hashes and arrays
$a = @() # array
$a = @{} # hash
#>
```

# Example of PowerShell Functions

```
#

# func.ps1
function add($a, $b) {
  Write-Host "$a+$b is" ($a+$b)
}

# To run the function, do the following
add 7 10


function Add-Numbers
{
  Write-Host "$a+$b is" $args[0] + $args[1]
}

Add-Numbers 5 10



function display {
  Write-Host "Welcome to PowerShell"
}


function add($a, $b) {
  Write-Host "$a+$b is" ($a+$b)
}


function stringf([string]$a, [string]$b)
{
  Write-Host "a:", $a, " b:", $b
}
```

```
function wf1 {
   param ($a, $b, $c) # alternative way to pass arguments
   Write-Host "a:", $a
   Write-Host "b:", $b
   Write-Host "c:", $c
}


Function bar {
   $MyVariable = "Foo"
   Return $MyVariable
}
```