Mod 11 PowerShell Strings

SCRIPTING ESSENTIALS

DR. BURKMAN

Mod 10 PowerShell Strings

We already know to escape with the backtick `

- https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_special_characters?view=powershell-6

```
write-host "`"Escaping double quotes`""
```

Mod 10 PowerShell Strings

Putting a string in single quotes sets literal values as opposed to double quotes which will expand variables:

```
 cls
$myName = "Jim"

write-host("My name is $myName`r`n")

write-host('My name is $myName `r`n')
```

Mod 10 PowerShell Strings

For multiline strings just type your string how you want.

```
 cls
$myName = "Jim
Burkman"

write-host $myName
```

Replace

```
 $cat = "dog|mouse"
$rat = $cat -replace "\|", "`r`n"
#$rat = $cat.Replace("|", "`r`n")
write-host $rat

#You have to escape the pipe with -replace but not with .replace
```

Mod 10 PowerShell Strings

Just like in Python, except a slice is .. Instead of a comma. Strings are still immutable

```
 cls
$myName = "Jim
Burkman"

write-host $myname[0]
write-host $myName[0..2]
write-host $myName[-2]
write-host $myName[4,1,8]
write-host $myName[-2..-4] #Handy for reversing
```

Mod 10 PowerShell Strings

Methods and comparisons:

```
 cls
$myName = "Jim Burkman nita burkman TATIANA BURKMAN"

write-host $myName.ToUpper()
write-host $myName.ToLower()
write-host (Get-Culture).TextInfo.ToTitleCase($myName.ToLower())

write-host ($myName -ceq $myName.ToUpper())
write-host ($myName -ceq $myName.ToLower())
write-host ($myName -ceq (Get-
Culture).TextInfo.ToTitleCase($myName.ToLower()))
```

Mod 10 PowerShell Strings

Note that you have to assign the change back to the string if you want it to stay:

```
 cls
$myName = "Jim Burkman"

write-host $myname.ToUpper()

write-host $myname

$myName = $myName.ToUpper()

write-host $myname
```

Mod 10 PowerShell Strings

We can check to see if a value is in, starts with, or ends with a value:

```
 cls
$myName = "Jim Burkman nita burkman TATIANA BURKMAN"

write-host $myName.Contains("a")
write-host $myName.StartsWith("J")
write-host $myName.EndsWith("N")
```

Mod 10 PowerShell Strings

Join strings with +

```
cls

$first = "Jim"
$second = "Burkman"
$all = $first + " " + $second

write-host ($all)
```

Mod 10 PowerShell Strings

We still have the split method:

```
cls

$myString = "Jim Burkman plays a lot of video games"

$myArray = @()

$myArray += $myString.Split()

foreach($i in $myArray)
{
write-host ($i)
}
```

Mod 10 PowerShell Strings

We can trim:

```
 cls
$myString = "Jim Burkman plays a lot of video games"
write-host($myString.Trim("seJ"))
```

We can replace:

```
 cls
$myString = "Jim Burkman plays a lot of video games"
write-host($myString.Replace("Bu","La"))
```

A Note on Negating IF

The opposite of:

```
 if ($i.StartsWith($user_input)) {}
```

Is

```
If(-Not $dog.contains(“J”){}
```

-Not will test for the opposite of a condition. If the condition would normally return True, -Not makes it return False. If the condition would normally return False, -Not makes it return True.

Careful, though. If($dog -ne “Woofy”) is not the same as if(-Not $dog -eq “Woofy”). The latter will only activate if $dog first equals “Woofy”. Then it will negate it. The former will activate every time $dog is not equal to “Woofy”.