Advanced Scripting   
Attributes-Declarative Programming

Last Updated: 10/19/2021 8:52 AM Version 1  
Document Prepared for: CIT361 Student

# Name Daniel Harris ID 235868292

# Instructions

Save a copy of this document. Answer all questions directly in this document. You will save and upload this completed document as your homework submission.

# Overview

PowerShell uses attributes to implement declarative programming. In this exercise you will add attributes to a function to perform validation.

# Requirements

PowerShell

# Setup

# Task 1—Create a Simple Function

Create a script that has a single function called Get-Soup. The function accepts any kind of soup and a size. We want to limit the size to cup, bowl or pot.

## Steps

1. Create a script named soup.ps1 and enter the following code
2. Test the code to make sure your function works. If it does not, debug until it does.

function Get-Soup {

    param(

        $soup,

        $size,

        $Quantity = 1,

        [switch]$Please

    )

    if ($please) {

        "$Quantity $size$(if($quantity -gt 1){'s'}) of $soup soup for you."

    }

    else {

        "No $soup soup for you!"

    }

}

# Task 2—Add Validation attributes to the parameters

Use attributes to have PowerShell validate the user input.

## Steps

1. Require the user to enter a soup. Add the Parameter Validate attribute to the $soup parameter. Add the following code on the line above the $soup parameter  
   [Parameter(Mandatory=$true)]
2. Run your code then test the function by just typing  
   Get-Soup
   1. Describe the result?No Corn soup for you! Was the result
3. Make the $size parameter required. Add the same code above the $size parameter.
4. Run your code then test by entering   
   Get-Soup
   1. Describe the result. No corn soup for you!
5. Limit the quantity to 5 using the ValidateRange attribute. Add the following code directly above the $quantity parameter  
   [ValidateRange(1,5)]
6. Run your code and test the function. Test the quantities of 0, 2 and 7.
   1. Describe the results. 2 worked, 0 and 7 threw an error
7. Add a validation set to limit the sizes to cup, bowl and pot. Use the ValidateSet attribute. Enter the following code directly above the $size parameter.  
    [ValidateSet('Cup','Bowl','Pot')]
8. Run the code then test the function. This time, use the tab key when entering a size to see the options. Try to enter “tankard” for size.
   1. Describe the results Tankard threw an error

# Task 3—Add an Alias for Get-Soup

Add an alias for the get-soup function of just soup

## Steps

1. Add an alias to the Get-Soup function. Add the following code directly under the function declaration line.  
   [Alias('Soup')]
2. Run your code then try to use the alias soup to run the function.
   1. Did it work? yes

# Wrap-up

Copy your completed Code here

function Get-Soup {  
 [Alias('Soup')]  
 param(  
 [Parameter(Mandatory=$true)]  
 [string]$soup,  
 [ValidateSet('Cup','Bowl','Pot')]  
 [string]$size,  
 [ValidateRange(1,5)]  
 [int]$Quantity = 1,  
 [switch]$Please   
 )  
 if ($please){  
 "$Quantity $size$(if($Quantity -gt 1){'s'}) of $soup soup for you."  
 }  
 else {  
 "No $soup soup for you!"  
 }  
}

# Deliverable

Upload this document with completed answers to i-learn.