Advanced Scripting   
Authoring Modules

Last Updated: 3/23/2024 5:04 PM Version 1  
Document Prepared for: CYBER360 Student

# Name Click here to enter name ID Click here to enter id

# 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

Now you will create a module and add a few utility functions.

# Requirements

PowerShell (any platform)

# Setup

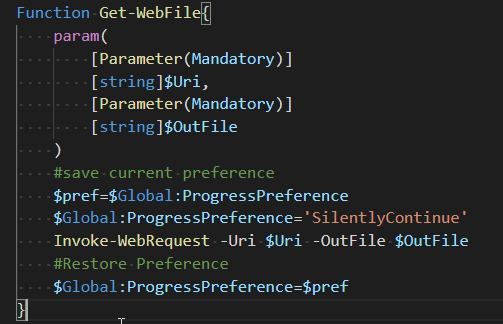
# Task 1—Create a Module

You may have noticed that PowerShell’s Invoke-Webrequest cmdlet is fairly slow when it downloads large files. As it turns out, this is because of the progress bar that it shows. Lets add a function that will turn off the progress bar before it downloads the file to speed things up, then return the progress bar to its previous state. The **$ProgressPreference** variable will control whether the progress bar is shown or not.

We’ll create a module for this utility.

## Steps

1. Create a file named MyUtilities.psm1
2. Add the following code to it:



1. Save your file and give it a test. But to test the module, you can’t just run it from VSCode. You need to import the module to the session to try it out. Use the **Import-Module** cmdlet:   
   Import-Module MyUtilities.psm1
2. View the commands in your module   
   Get-Command -Module MyUtilities
3. Try it out. It should download without the progress bar:

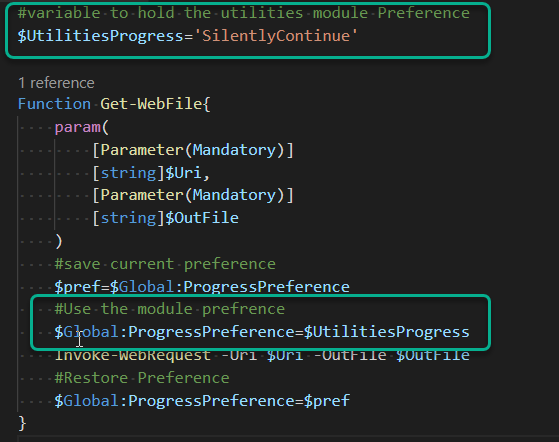
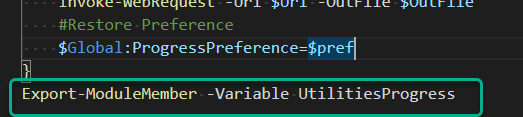
Get-WebFile cit2.cit.byui.edu/as/psfiles.zip junk.zip

1. If you get an error, fix you code and try again. To try again you must:
   1. Save your code.
   2. Load your module again. Since your module is already loaded you need to either
      1. Unload your module
      2. Reload your module
   3. Or
      1. If you can, use the **-force** switch parameter with **Import-Module**

# Task 2—Add a Script Level Variable

We want to add a method for the user to set a preference for the progress bar in the Utilities module. To do that you will need a variable to store the preference in.

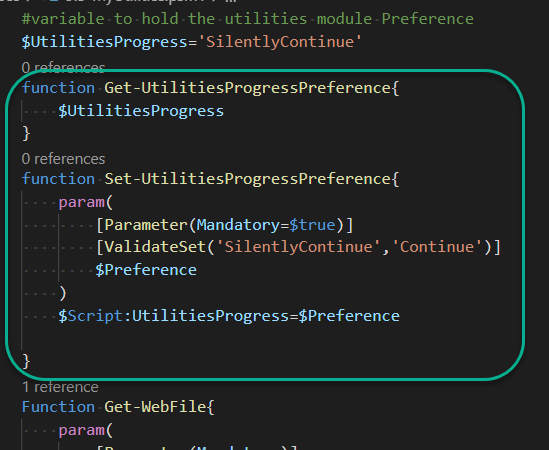
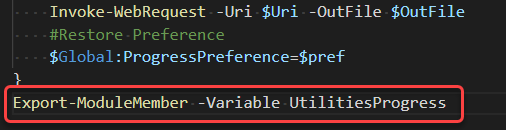
## Steps

1. Make the following changes to your module file.  
   
2. Save the file. Reload the module.
3. What happens if you enter   
   $UtilitiesProgress   
   at the CLI? Click or tap here to enter text.
4. Script level variables are not exposed by default. If you want to expose a script level variable, you can tell the module to do so with the **Export-ModuleMember** cmdlet. Add the following line of code at the bottom of the module:   
   
5. Save and re-import the module.
6. Enter:   
   Get-Module -Name MyUtilities
   1. What’s the value of the **ExportedCommands** property? Click or tap here to enter text.
   2. Again enter:  
      **$UtilitiesProgress**   
      at the CLI. Did you get a result this time? Click or tap here to enter text.
7. Get a list of the commands in the module. Is Get-WebFile listed? Click or tap here to enter text.
8. Once you use the **Export-ModuleMemeber** cmdlet, all members are hidden except those explicitly exported.

# Task 3—Provide Commands to Change the Module Preference

Allowing direct access to the **$UtilitiesProgress** variable is probably not a good idea. For one thing, the user could easily enter an invalid option. In this task you will add code to modify the **$UtilitiesProgress** variable and view the **$UtilitiesProgress** value.

## Steps

1. Add the functions **Get-UtilitiesProgressPreference** and **Set-UtilitiesProgressPreference**   
   
2. Remove the **Export-ModuleMember** line from the module.   
   
3. Save and re-import.
4. Test the new module using the new commands to set the module preferences.

# Wrap-up

Enter all of your module code here:

Click or tap here to enter text.

# Deliverable

Upload this document with completed answers to I-Learn Canvas.