UNIT 2 LAB 3: CREATING SELECTION STATEMENTS

OVERVIEW

Most scripts contain at least one selection statement. These statements are often used to determine whether certain lines of code are executed; they are also used to choose between multiple blocks of code to execute. In this lab, you will practice constructing selection statements using the if, elseif, and else statements and the switch statement.

OBJECTIVES

5.1. Use input and output commands.
5.3. Create selection statements.
5.4. Initialize data structures.
5.8. Use arithmetic, string, and logical operators

PREREQUISITES

Lab - Creating a Basic Script is complete

TASKS

CREATING AN INVENTORY SCRIPT

In this step, you will create an inventory script. The pseudo code for this task is shown below.

- Display a menu with options for creating an inventory.
- Prompt the user to choose an option.
- Execute code based on the option the user chose.

At this point, we are just investigating how the logic will work and will implement the actual code for each option we create later.

CREATING A SCRIPT FROM A TEMPLATE

Use the following steps to create a new script from the previously created template:

1. Login to the DC-1 Virtual Machine.
2. Open PowerShell ISE.
3. Open the template.ps1
4. Rename the script Get-Inventory.ps1
5. In the **SYNOPSIS** section, **change** the **text** to read "**This is a script to inventory computer system hardware and software**"
6. In the **DESCRIPTION** section, **change** the **text** to read "**This script provides a menu with a list of options for inventorying computer system hardware and software**"

7. In the first **EXAMPLE** section, **change** the **text** to read

   Get-Inventory.ps1
   This example starts the program and displays a menu with options for inventorying computer systems.

8. **Remove** the **second EXAMPLE** section.
9. **Save** the **file.**

CREATING A MENU

In this step, we will implement the first part of our pseudo code:

- Display a menu with options for creating an inventory.

To create the menu you will create and initialize a variable to hold the menu text and the menu choice. To do this, perform the following:

1. **In** the **initialization section** of your script **add** the following:

```
$menu_choice = ""
$menu = @"
                    Inventory System Menu
================================================
        A - Installed Software Inventory
        B - Basic Hardware Inventory
                X - Exit
================================================
"@
```

Note: If you copy and paste the text into the script console, you may need to add tabs to get the correct alignment.

2. **In** the **main body** section, **add** the following:

```
$menu
```

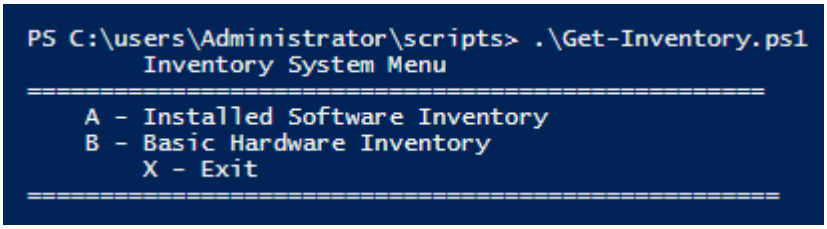
3. **Correct** any spelling and syntax **errors**; these are indicated by red squiggles (just like misspelled words in Word).

4. **Save** the **script**.

Notes:
The @" Text "@ section is known as a here string. This is a shortened way of adding a lot of text and formatting at once. PowerShell preserves the line feeds, carriage returns, and text as typed. The other way to do this would require a number of escape characters and the Write-Host cmdlet.

The only code in the main section is the $menu statement. The $menu on a line by itself is the easiest way to display the contents to the screen. We could also use the Write-Host cmdlet here.

5. Run the script. The output should be like the screen shown below.



GETTING KEYBOARD INPUT

For users that are not experienced with the command line, the easiest way to get input is to provide a menu and then allow them to select an option. In order to implement this, we need some way to read what the user types. PowerShell provides this feature in the ***Read-Host*** cmdlet.

In this step, we will implement the second part of our pseudo code:

- Prompt the user to choose an option.

To prompt the user to choose an option and store that option in the $menu_choice variable, perform the following:

1. Add the following line of code to the Main Body section on the line after the $menu statement:

   $menu_choice = Read-Host -Prompt "Choose a menu option"

2. Correct any errors and save the script.

3. Run the script. Your output should be similar to the screen below.

```
PS C:\users\Administrator\scripts> .\Get-Inventory.ps1
        Inventory System Menu
================================================
    A - Installed Software Inventory
    B - Basic Hardware Inventory
        X - Exit
================================================
Choose a menu option: A

PS C:\users\Administrator\scripts> |
```

4. What value is stored in $menu_choice? In the example above, it should be "A", but you don't know for sure until you look at the contents of the variable. While you are still testing a script, it is helpful to display the contents of the variables to the console so that you are sure what their values are.
5. **Add** the following **statement** in the **Main body** section of the script **after** the **line** with the ***Read-Host*** cmdlet.

   $menu_choice

6. **Save** and **run** the **script** again. Your screen should be similar to the one below.

```
PS C:\users\Administrator\scripts> .\Get-Inventory.ps1
        Inventory System Menu
================================================
    A - Installed Software Inventory
    B - Basic Hardware Inventory
        X - Exit
================================================
Choose a menu option: B
B
```

7. Notice the last line. This line shows you the contents of the $menu_choice variable. This confirms what we think it should be.

ADDING AN IF STATEMENT

In this step, you will implement the logic for your menu. If the user presses one of the keys, we want the script to perform a specific action. At this point in the script, we will just display a message to verify the correct code is being executed. Later, we will implement the code that will be contained in each section.

To add an If statement to your script, perform the following:

1. Remove the line with the lone $menu_choice variable. You no longer need this since your test was successful.

2. Add the following code below the line with the Read-Host statement:

```
if ( $menu_choice -eq "A")
        {
                Write-Host "Inventorying Installed Software…"
                Write-Host "This code is not yet implemented."
        }
ElseIf ( $menu_choice -eq "B")
        {
                Write-Host "Inventorying Installed Hardware…"
                Write-Host "This code is not yet implemented."
        }
ElseIf ( $menu_choice -eq "X" )
        {
                Write-Host "This I can do."
                Write-Host "Exiting…"
        }
```

3. Correct any errors and save the script.
4. Run and test the script using each of the options. Your screen should be similar to the one below.

```
PS C:\users\Administrator\scripts> .\Get-Inventory.ps1
        Inventory System Menu
==================================================
    A - Installed Software Inventory
    B - Basic Hardware Inventory
        X - Exit
==================================================
Choose a menu option: A
Inventorying Installed Software…
This code is not yet implemented.

PS C:\users\Administrator\scripts> .\Get-Inventory.ps1
        Inventory System Menu
==================================================
    A - Installed Software Inventory
    B - Basic Hardware Inventory
        X - Exit
==================================================
Choose a menu option: B
Inventorying Installed Hardware…
This code is not yet implemented.

PS C:\users\Administrator\scripts> .\Get-Inventory.ps1
        Inventory System Menu
==================================================
    A - Installed Software Inventory
    B - Basic Hardware Inventory
        X - Exit
==================================================
Choose a menu option: X
This I can do.
Exiting…
```

UNIT 2 LAB 3: CREATING SELECTION STATEMENTS

NESTING AN IF STATEMENT

Often you will need to place an If statement within an If statement; this is known as nesting, and you can do it with looping statements as well.

To nest an If statement, perform the following:

1. In the last if statement script block replace the existing code with the following code:

```
$sure = Read-Host -Prompt “Are you sure you want to exit? (Y)”
if ($sure -eq “Y”)
        {
                Write-Host “This I can do.”
                Write-Host “Exiting…”
        }
Else
        {
               Write-Host "Not Exiting :)"
        }
```

2. The last part of your script should look like this:

```
ElseIf ( $menu_choice -eq “X” )
    {
        $sure = Read-Host -Prompt "Are you sure you want to exit? (Y)"
        if ($sure -eq "Y")
        {
            Write-Host “This I can do.”
            Write-Host “Exiting…”
        }
        Else
        {
            Write-Host "Not Exiting :)"
        }
    }
```

3. Correct any errors and save your script.

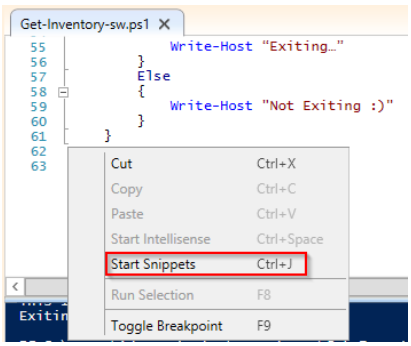
4. Run and test your script. You should have output similar to the screen below.

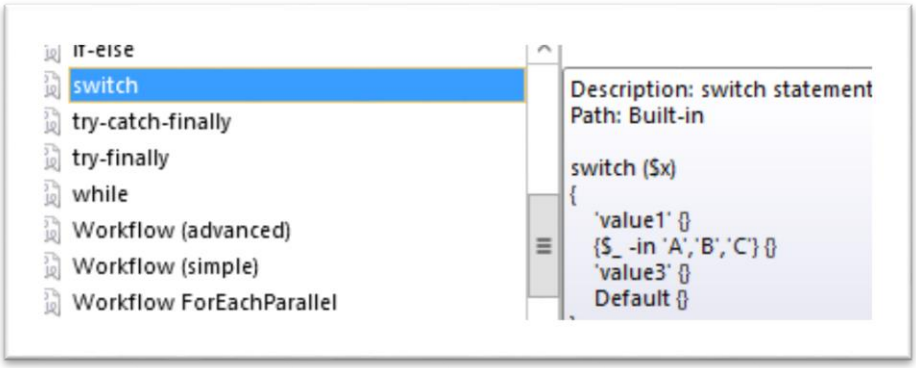
```
PS C:\users\Administrator\scripts> .\Get-Inventory.ps1
        Inventory System Menu
=================================================
    A - Installed Software Inventory
    B - Basic Hardware Inventory
        X - Exit
=================================================
Choose a menu option: X
Are you sure you want to exit? (Y): N
Not Exiting :)

PS C:\users\Administrator\scripts> .\Get-Inventory.ps1
        Inventory System Menu
=================================================
    A - Installed Software Inventory
    B - Basic Hardware Inventory
        X - Exit
=================================================
Choose a menu option: X
Are you sure you want to exit? (Y): y
This I can do.
Exiting…
```

CREATING A SWITCH STATEMENT

We can create menu logic in our script more cleanly using a switch statement. In this step, we will modify our previous script to use a switch statement.

1. **Save** the **Get-Inventory.ps1** script as **Get-Inventory-sw.ps1**. This will create a new script without overwriting the previous script.
2. **In** the **Get-Inventory-sw.ps1** script, **right-click** an **empty line** at the bottom of the script and **choose Start Snippets** from the context menu.
3. This will give you the list of options shown below.
4. **Choose** the **switch** option in the drop down menu and **press enter**. This will insert the code template shown on the right.



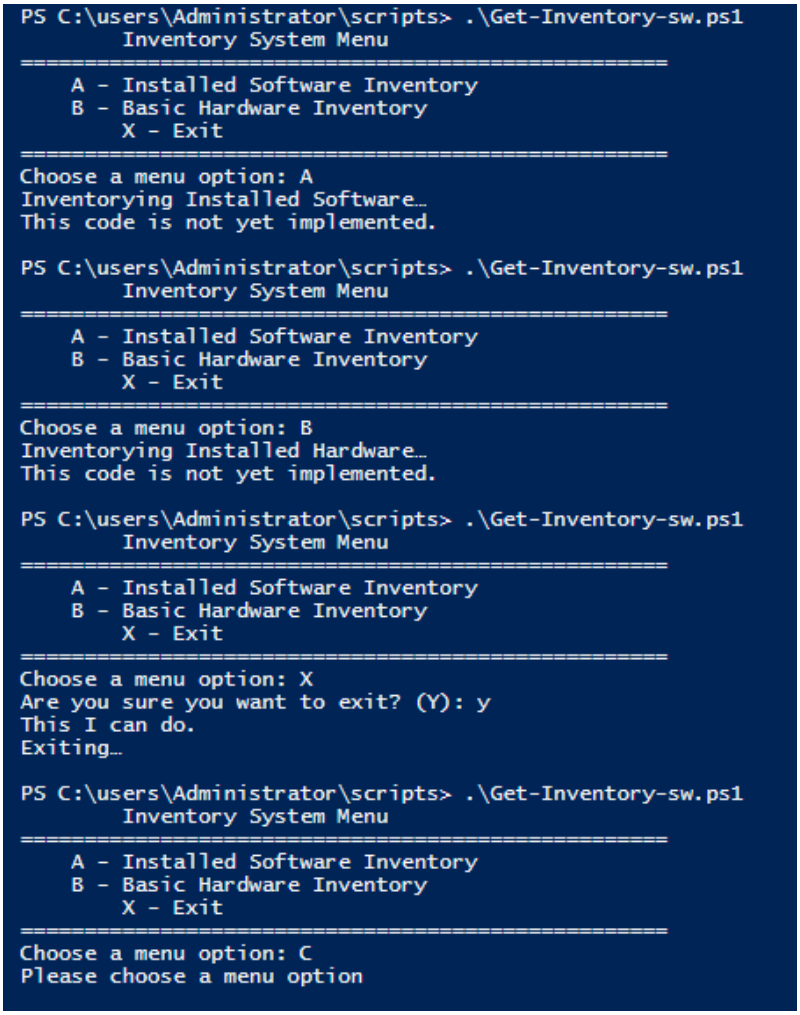


5. In the template perform the following:
    a. **Change** the **$x** to **$menu_choice**
    b. **Change value1** to **A**
    c. **Delete** the **next line**.
    d. **Change value3** to **B**
    e. On the **next line add 'X' { }**
    f. In the **Default** script block **add Write-Host "Please choose a menu option"**
    g. **Move** the **code from** the corresponding **if** or **Elseif block to** the corresponding **script block in** the **switch statement**.
    h. **Delete** the entire **if and elseif statements**.
6. Your final code should look like the screen below.

```
switch ($menu_choice)
{
    'A' {
            Write-Host "Inventorying Installed Software…"
            Write-Host "This code is not yet implemented."
        }
    'B' {
            Write-Host "Inventorying Installed Hardware…"
            Write-Host "This code is not yet implemented."
        }
    'X' {
            $sure = Read-Host -Prompt "Are you sure you want to exit? (Y)"
            if ($sure -eq "Y")
            {
                Write-Host "This I can do."
                Write-Host "Exiting…"
            }
            Else
            {
                Write-Host "Not Exiting :)"
            }
        }
    Default {Write-Host "Please choose a menu option"}
}
```

7. **Correct** any **errors** and **save** your **script**.

8. Run and test your script. Your output should be similar to the screen below.



LAB EVALUATION

Upload your completed scripts in the drop box for this assignment. You will be graded for this assignment as follows:

1. 50% - Get-Inventory.ps1 script is complete and executes correctly.
2. 50% - Get-Inventory-sw.ps1 script is complete and executes correctly.