Advanced Scripting   
Loop Flow Control Structures

Last Updated: 5/17/2024 3:53 PM Version 1  
Document Prepared for: CYBER360 Student

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

# Instructions

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

# Overview

For this exercise, you will write a few short scripts to practice using PowerShell loop control structures.

# Setup

## Requirements

* Visual Studio Code, PowerShell

# Task 1 —Foreach-Object loop cmdlet, .foreach() object method

## Steps

1. At a PowerShell prompt, Enter:  
   **1..4 | Foreach-Object { "Progress: $\_ of 4" }**
   1. Do it again, but this time use the shorter alias **%** instead of **Foreach-Object**:  
      **1..4 | % {"Progress: $\_ of 4"}**
   2. Modify the string in that pipeline’s script block, so that it produces output that looks like this:  
        
      **Progress: 25%  
      Progress: 50%  
      Progress: 75%  
      Progress: 100%**   
        
      Enter your modified string here:  
      **1..4 | % {**Click or tap here to enter text.**}**
2. At a PowerShell prompt, Enter:  
   **(1..4).foreach({ "Progress: $\_ of 4" })**
   1. Modify the string in that method’s script block argument, so that its output looks like this:  
        
      **Remaining: 3 of 4  
      Remaining: 2 of 4  
      Remaining: 1 of 4  
      Remaining: 0 of 4**
      1. Enter your modified string here:  
         **(1..4).foreach({**Click or tap here to enter text.**})**

# Task 2—Loop control statements: while, do, foreach, for

## Steps

1. In VS Code, copy the following code skeleton outline into a new script:  
   **$i = \_  
   while ( \_ ) {  
    \_  
   }**
   1. Everywhere you see a single underscore \_, replace with suitable code to make the script produce this output:  
      **Number 5  
      Number 6  
      Number 7  
      Number 8  
      Number 9  
      Number 10**
   2. Enter your working code snippets here:  
      **$i =** Click or tap here to enter text. **while (** Click or tap here to enter text. **) {** Click or tap here to enter text. **}**
2. Add the following code skeleton outline:  
   **$i = \_  
   do {  
    \_  
   } while ( \_ )**
   1. Everywhere you see a single underscore \_, replace it with suitable code to produce this output:  
      **Number 100  
      Number 110  
      Number 120  
      Number 130  
      Number 140  
      Number 150**
   2. Enter your working code snippets here:  
      **$i =** Click or tap here to enter text. **do {** Click or tap here to enter text. **} while (** Click or tap here to enter text. **)**
3. Add the following code skeleton outline:  
   **$i = \_  
   do {  
    \_  
   } until ( \_ )**
   1. Everywhere you see a single underscore \_, replace it with suitable code to produce this output:  
      **Number 128  
      Number 256  
      Number 384  
      Number 512  
      Number 640  
      Number 768**
   2. Enter your working code snippets here:  
      **$i =** Click or tap here to enter text. **do {** Click or tap here to enter text. **} until (** Click or tap here to enter text. **)**
4. Add the following code skeleton outline:  
   **for ($k=-9; \_; \_) {  
    \_  
   }**
   1. Everywhere you see a single underscore \_, replace it with suitable code to produce this output:  
      **Number -9  
      Number -12  
      Number -15  
      Number -18  
      Number -21  
      Number -24**
   2. Enter your working code snippets here:  
      **for ($k=-9;** Click or tap here to enter text.**;** Click or tap here to enter text.**) {** Click or tap here to enter text. **}**
5. Add the following code skeleton outline:  
   **$listing = Get-ChildItem  
   foreach ( \_ in $listing ) {  
    \_  
   }**
   1. Everywhere you see a single underscore \_, replace it with suitable code that outputs the name and the size (in kilobytes) of the files (but not the subdirectories) in the current directory.
   2. Enter your working code snippets here:  
      **$listing = Get-ChildItem  
      foreach (** Click or tap here to enter text. **in $listing ) {** Click or tap here to enter text. **}**
6. Use any loop statement of your choice to write code that outputs and “confirms” the numbers 3 through 12, but *skips* confirming every third number, like this:  
   **Number 3  
   Confirm: 3  
   Number 4  
   Confirm: 4  
   Number 5  
   Number 6  
   Confirm: 6  
   Number 7  
   Confirm: 7  
   Number 8  
   Number 9  
   Confirm: 9  
   Number 10  
   Confirm: 10  
   Number 11  
   Number 12  
   Confirm: 12**   
     
   Your code must use the keyword for skipping a loop iteration; see examples in textbook chapter 2.
   1. Enter your working looping code here:  
       Click or tap here to enter text.

# Deliverable

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