Advanced Scripting   
Arrays

Last Updated: 12/26/2022 4:00 PM 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 make extensive use of arrays. Whenever a cmdlet needs to return more than one item it creates an array and adds each item to the array. To make array processing easier whenever PowerShell needs an array if a scalar is passed PowerShell converts it to an array with a single element. Whenever a scalar is need you can pass an array with one element and PowerShell will unwrap the array into a scalar.

# Requirements

PowerShell

# Setup

# Task 1—Creating Arrays

## Steps

1. Create an empty array. Enter  
   $a=@()
   1. Use the GetType method to see what datatype $a contains. System.array
   2. The Count property tells you how many elements an array contains. What is the value of $a.count? 0
2. The comma operator ‘**,**’ is used to create arrays. Enter  
   $colors='red','blue','green'
3. Arrays can contain different data types. Enter  
   $mix=1,2,3,'a','b','c'
   1. How many elements are in the array? 6
4. You can turn any scalar into an array by using the unary ‘**,**’ operator in front of the value. Enter:  
   $one=,10
   1. How many elements are in the array? 1
5. Given the following PowerShell expression:  
   @(,@(1,2,3,4))
   1. How many elements are in the resultant array? 1
6. The **..** operator creates an array of integers.
   1. Enter  
      1..10
      1. What was returned? 1-10
   2. Enter  
      10..1
      1. What was returned? 10-1
   3. Enter   
      5..1
      1. What was returned? 5-1
   4. Enter   
      5..-5
      1. What was returned? 5 to -5
7. Enter the following command  
   $p=get-process
   1. What datatype does $p contain? System.array

# Task 2—Accessing Array Elements.

PowerShell arrays are 0 based, you can get a single element or an array of elements from an array. Use square bracketed index numbers such as **[**index**]** or **[**index1**,**index2**]** to access elements of an array.

## Steps

1. Accessing a single element of an array. You should have the array **$mix** from Task one. If not, recreate it: **$mix=1,2,3,'a','b','c'**. Retrieve the first element of the array:  
   $mix[0]
2. You can get multiple elements at once. Enter:  
   $mix[1,3]
   1. What was returned? 2,a
3. Write a statement that would get the 3rd 4th and 8th elements of an array name $array  
   $mix[4,5,9]
4. How many elements does the **$mix** array have?
5. What is the index of the last element?
6. What happens if you use an index that is out of bounds? Enter:  
   $mix[10]
   1. What was returned? Null is returned
7. Negative indexes work as well. Enter:  
   $mix[-1]
   1. What was returned? c
8. Negative indexes work as well. Enter:  
   $mix[-2]
   1. What was returned? b
9. Of course, the index can be a variable. Enter:  
   $i=1  
   $mix[$i]
   1. What was returned? 2
10. The index variable can be an array of index values.  
    $i=1,4,5  
    $mix[$i]
    1. What was returned? 2,b,c
11. Chang the value of an array item.
    1. Enter:  
       $mix[0]= 'do'
    2. View the results  
       $mix[0]

# Task 3— “Modifying” Arrays

Arrays can be modified—kindof… The **+** operator is defined for arrays as the concatenation operator. It is used to add elements to an array. Okay, so you can’t really modify the size of an array. When you add elements, a *new* array is created with the elements of the original array and the new element(s).

## Steps

1. Combine two arrays. Enter:  
   $a=1,2,3  
   $b= 'do','re','mi'  
   $a + $b
   1. Record the results: 123 do re me
2. Add elements to an array. Enter  
   $a=1,2,3  
   $a=$a+4  
   $a
3. Remove an item. There isn’t a real way to remove an item. But you can create a new array with a portion of the elements of the array. Try:  
   $new=$mix[0..2]
   1. What does $new contain? Do, 2,3

# Task 4—Reference Types

Arrays are reference types. When you assign an array to a variable that variable becomes a reference to the same array.

For the upcoming steps, **$mix** should contain **'do',2,3,'a','b','c'**. (If it doesn’t, reassign the variable so that it does.)

## Steps

1. Explore an array reference. Crate a variable and assign it to $mix:  
   $match=$mix
   1. What does $mix contain? Do,2,3,a,b,c
   2. What does $match contain? Do,2,3,a,b,c
2. Modify an element in $match.  
   $match[0]=100
   1. What does $mix[0] contain? 100
   2. What does that tell you? Arrays are updated when linked
3. If you add an element to an array it makes a new array. In that case the reference is broken. Add anelement to $mix:  
   $mix+=’d’
   1. What does $mix contain? 100,2,3,a,b,c,d
   2. What does $match contain? 100,2,3,a,b,c

# Deliverable

Upload this document with completed answers to i-learn.