Advanced Scripting   
Text and CSV data

Last Updated: 6/15/2020 4:26 PM Version 1  
Document Prepared for: CIT361 Student

# Name Zach Lucas ID 895649438

# 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

Working with text from various systems is a common task with PowerShell. This exercise will help you work with plain text and simple delimited data files.

# Requirements

PowerShell

# Setup

Make sure you have access to the psfiles example files. You can get a copy here <http://cf.esage.com/psfiles.zip>

# Task 1—Working with text.

## Steps

1. Make the psfiles/data folder your current directory.
2. The colors.txt file contains a list of color names. There is one name per line. Read the file into a variable with the Get-Content cmdlet.  
   $colors=Get-Content colors.txt
3. Use Measure-Object to see how many colors there are  
   $colors|measure
   1. How many colors are in the file? 1029
4. You can easily filter the colors by using the Select-String cmdlet. Select-String uses regular expression matching. It will return any string that matches the expression. Use Select-String to list all the colors that contain the word green  
   $colors|Select-String green
   1. How many colors have the word green in them? 91
5. Use the -like operator to perform basic pattern matching. To get all the colors that start with the letters ‘co’ enter  
   $colors|? {$\_ -like 'co\*'}
   1. How many colors start with ‘co’? 22
6. Get the first 10 colors in the colors.txt file  
   Get-Content .\colors.txt -head 10
   1. What is the 10th color in the file? Periwinkle
7. Get the last 5 colors in the file  
   Get-Content .\colors.txt -Tail 5
   1. What is the 5th from last color in the file? Pale Pink
8. List the contents of the colors file one page at a time  
   Get-Content .\colors.txt|Out-Host -Paging
9. Create a new text file with the first 10 colors and the last ten colors. First grab the first 10 items and store in the output file  
   Get-Content .\colors.txt -Head 10 |Set-Content firstlast.txt
10. Check to see if it worked  
    Get-Content .\firstlast.txt
11. Next grab the last 10 items and append it to the   
    Get-Content .\colors.txt -Tail 10 |Add-Content firstlast.txt
12. Check to see if the colors were added  
    Get-Content .\firstlast.txt
13. Grab the 11th – 20th colors from the file and store them in firstlast.txt  
    (Get-Content .\colors.txt)[10..19]|set-content .\firstlast.txt
    1. Describe what is in firstlast.txt. The 11th-20th colors of the colors.txt file
14. Finally Clear the firstlast.txtfile  
    Clear-Content .\firstlast.txt
    1. What is in firstlast.txt? Nothing

# Task 2—Working with CSV data

## Steps

1. Load the file gems.csv  
   $g=Import-Csv .\gems.csv
2. View the data  
   $g|ft
3. View the resulting data type  
   $g[0].Gettytpe()
   1. What is the datatype of each item? PSCustomObject
4. Save a list of all the minerals that have a hardness of 5 or more into a new csv file that only contains the mineral name and hardness in order of hardest to softest. On one line enter  
   Import-Csv gems.csv |Where {+($\_.hardness) -ge 5}|sort hardness -desc |select Mineral,Hardness|Export-Csv hardgems.csv
5. View the contents  
   cat hadgems.csv
   1. Did it work? Yes, but all of the hardness is 0
6. Diamond, with a hardness of 10 ended up at the bottom of the list, this is because the hardness was treated as a string rather than a number. We need to fix that.  
   Import-Csv gems.csv |Where {+($\_.hardness) -ge 5}|%{$\_.hardness = +$\_.hardness;$\_}|sort hardness -desc|select Mineral,Hardness|Export-Csv hardgems.csv
7. Did it work this time? No
8. Study the commands and describe what was done to make it work. It imported the csv file, found all the hardness values greater than 5, sorted them, selected the headers, and exported it as a new CSV file.

# Task 3—Tab Delimited Files

The “CSV” commands work with files that are not, separated as well. Now you will load a file that is separated with tabs.

## Steps

1. Import the tab separated file RushSongs.txt. Since the delimiter is a special character, we need to escape out the tab character when defining the delimiter. Enter (note the ` is the backtick not a single quote):  
   $s=Import-Csv .\RushSongs.txt -Delimiter "`t"
2. Take a look at the data  
   $s|ft

# Deliverable

Upload this document with completed answers to i-learn.