## Practice Exercise1: File Operations/Searching/Basic PowerShell Operators

1. Create a folder “TestingPurpose” then create 3 Subfolders inside this folder called Sub\_Folder\_1, Sub\_Folder\_2 and Sub\_Folder\_3
2. Create some empty test files inside these folders:

TypeA-Test1.txt, TypeA-Test2.txt … TypeA-Test50.txt into SubFolder1

TypeB-Test51.txt, TypeB-Test52.txt … TypeB-Test100 into SubFolder2

Needless to say that you have to use logic for creating these files. Do not create them by using 100 individual statements.

1. Move all files which have an odd number in their names to SubFolder2
2. Move all files which have even number in its name to SubFolder1
3. Rename folder SubFolder1 to “Even\_Files\_Container” and SubFolder2 to “Odd\_Files\_Container”
4. Prepare a list of all files currently existing inside folder TestingPurpose. Write this list to a text file called MasterFile.txt. List all the files inside “Even File Container” before you list the file in “Odd\_Files\_Container  
     
   Your output should look something like:

As of YYYYMMDD HH: MM, the files inside the parent folder “Testing Purpose” are:

C:\TestingPurpose\EvenFilesContainer\TypeB-Test2.txt

…

C:\TestingPurpose\OddFilesContainer\TypeA-Test99.txt

1. Delete all files which start with TypeA. At the conclusion of this exercise, the only folders remaining in both folders should be files that start with TypeB.
2. Repeat step 6 but this time, write you results to a text file called FinalResults.txt. List the files in the same order as you did in Step 6

## Practice Exercise 2:  Windows Service Related

Write PowerShell One-liners for each of the following. Your output should have a title explaining what the output shows.

1. Get all services which are stopped
2. Get all services whose name starts with letter "A"
3. Get all services which are set to start automatically (look for property StartType: Automatic)
4. Restart-Service Winmgmt
5. Export the service name and status into a csv text file called services.csv. The contents of the file should look like:

Example:

Service Name,Status

Service A,Running

Service B,Stopped

6.) Create an HTML file that is formatted in a table that shows the same information.

## Practice Exercise 3:  Windows Process Related

Write PowerShell One-liners for each of the following. Ensure that for each step, your output has a title explaining what the output shows:

1. Get a list of all Windows processes whose name starts with letter "A"
2. Get a list of processes whose name is svchost and whose PM is more than 100MB
3. Get the Process Name, Process ID and handleCount for all Windows processes whose PM is more then 100MB and CPU more than 1000s
4. Export the results of (3) to both html and CSV format

## Practice Exercise 4:  User Input

Write a simple calculator that asks for two pieces of data from the user:

1. Number of gallons used on a trip
2. Number of miles travelled on a trip

From this data, calculate the miles per gallon and display it to the user in well formatted output.

Your program should ask the user if they want to repeat the calculation and direct them to answer “yes” or “y” if they want to repeat the calculation or answer “n”

If they want to stop the calculations. You should allow the user to answer using capital letters or lower case letters and repeat the question if they supply any other answer than those that are valid.

## Practice Exercise 5:  Programming using PowerShell Loops

Write a PowerShell script using nested loops (for loop or while loop) to draw the below pattern of octothorpes:

#

##

###

####

#####

######

#######

########

#########

##########

########### <- 11 octothorpes

## Practice Exercise #6

Write a PowerShell script to do the following:

1. Create a directory called C:\DataFiles
2. Create 25 files of varying sizes between 1KB and 1MB with your own naming convention.  
   But don’t use something as simple as Test-1. Files should have a .dat extension.
3. Modify the creation and last write times to “age” the files between 10 and 100 days.

## Practice Exercise #7

Create a formatted HTML report that shows each module location from %PSModulePath%, the number of modules in each location and their total size, and then a listing of each module which shows the most recent version and the total size of all module files.

Extra bonus points if you can make it pretty with CSS formatting.

## Practice Exercise #8 - Inventory

Create a PS script that will take an inventory of your desktop system. The script should display information about:

* the CPU
* Memory (type and amount)
* Storage type and amount
* Version of Windows Installed
* Hot Fixes installed
* List of Locally defined users

The output should be in an organized format in an HTML file

## Practice Exercise #9 - User Maintenance

This exercise practices the following skills:

* Opening and reading a CSV file
* Working with strings
* Adding local user accounts
* Creating a file structure in the Windows file system
* Opening and writing an HTML file

At the beginning of every semester, you are emailed a CSV file of a list of students.

The first line of the file is three lines that are in the format.

-- Class list produced on <dd> <mm> <yyyy> --

<instructor name>

<class name>

From the fourth line onward, each line is as follows

<last\_name>, <first\_name>

After the first three lines, each line is one student that is enrolled in the class. There are a total of 16 students in the list (but your program needs to work with a file of any size).

Before you start this exercise, manually create this file and place it on your desktop

**Part 1: Create Local Accounts For The Users**

Each student:

* needs to be given an account as a local user on your system. The login name for the account is:

<fi><last\_name> where

* <fi> is the first letter of their first name
* <last\_name> is the users last name
* have a home folder created in a folder named "IT Class". The user's home\_folder is to be named the same as their user name

For example, John Smith will have a folder under "IT Class" called "JSmith" and his login name will be "JSmith". Passwords for each user will be initially set to

<last\_name>-<first-name>\_IT

You will also need to copy a file called "Read\_This\_First.txt" into each user's home folder. (You can manually create this file with whatever contents you think the user should be reading.

**Part 2: Creating a Class List Report**

When this task is complete, produce an HTML file to be used for attendance. This should look like the following:

**Class List for:** <class name>

**Instructor:** <instructor name>

**Total Enrolled:** <number>

|  |  |  |
| --- | --- | --- |
| First Name | Last Name | User Name |

Student's name should be listed in a three column table with borders and a table header.

## Practice Exercise #10 – Create Email Addresses from a CSV input file

You are given a CSV file where each line is of the form:

<lastName>,<firstName>.<4-digit graduation year>, <domain>

An example of a line is this file is

Gluck,Fredric,2023,sanford.org

Read each line of this file and create an email address with the form:

<Last Name>,<FirstName>, <gradYearLast2Digits><firstInitial><lastName>@sanford.org

An example of a correct email address is:

Gluck,Fredric,23fgluck@sanford.org

Write each new email address to a new text file.