Project #2 – Serial Number Sorter

Objectives

- Code a short Python program that works with device Serial numbers and information.

Your mission...

You goal is to create a Python script (Program) that will open a file of comma delimited information about devices and sort that into 2 nested dictionaries. The main dictionary will use the manufacturer (Cisco, Meraki, Juniper, Aruba & Dell) as the Keys. The Values for each Key will be another dictionary. This dictionary will use the device Serial Number as the Key, with the Values being device information stored as a List.

For ease, a file containing sample data will be provided – this will present the data in a uniform manner (one entry per row) for working with:

Device Manufacturer,Device Serial Number,Device Type,Device Model,Port Density,Management IP Address.

The program will sort devices according to Manufacturer and Serial Number, perform validation, ask the user if they wish to create a new entry, and write the output to a new file.

Requirements

Create 2 files:

- The main program – ***SerialSorter.py*** which will open the file provided and pass as a variable to a function in your module. The main program will then get formatted Dictionary data back from a function within the module. The main program will ask the User if they wish to add any new serial numbers and device information, adding this to the dictionary using a second function in the imported module. The main program will then write this data to a new file, in the same location as the original ("C:\SerialNumbers\").
- A Module you will import – ***SerialModule.py*** which will contain 2 functions to be imported:
  - One which will sort the given values to a dictionary entry as described below; and
  - One which will add new values to the dictionary as described below.

Include a documentation header (as well as comments in your code as well).

<u>Write a short program that meets all of the following requirements.</u>

<u>Module SerialModule.py:</u>

i) First function – serialSorter:

(1) Take 1 argument – the variable containing the text file data opened in the main file;

(2) Assign each line entry into 2 nested dictionaries:

(a) The first dictionary should use the Manufacturers as the Keys:

(i) Cisco

NOTE: This is **<u>NOT</u>** a group assignment and must be completed individually.

(ii) Meraki

(iii) Juniper

(iv) Aruba

(v) Dell

HINT: You need to define the Keys in the outer Dictionary (Cisco, Meraki etc.) but the keys for the inner dictionary and the assignment (Serial Numbers) can be allocated automatically using a for loop.

(b) The Values will be the entries of the second, nested dictionary for each value from the file that contains that manufacturer:

(i) The serial number will be the Key

(ii) The values will be Device Type, Model, Port Density, stored as a list

(c) The program should check that the Serial Number is valid – only Serial Numbers with TEN (10) alphanumeric digits is valid. If a serial number is invalid, the value should NOT be written to the dictionary (the program should NOT crash or stop) – that value will be skipped;

(d) The program should also check if the Serial Number is a duplicate. If it is, this value should also be skipped. Again, this should NOT cause the program to stop or crash;

(e) The function should return the new nested dictionaries as a single variable back to the main program.

ii) Second Function – serialAdder:

(1) Take 2 arguments – the sorted dictionary from the above, and a list of values input by the user in the main program;

(2) Loop through each of the 5 values provided as input by the User (Manufacturer, Serial Number, Device Type, Model and Port Density) and add the entry to the appropriate dictionary key (see example). i.e. if I add a new Cisco router, this should be added to the "Cisco" key in the outer dictionary, using the Serial Number I input as the key for the inner dictionary;

(3) Only the 5 Manufacturers specified – Cisco, Meraki, Juniper, Aruba and Dell should be permitted, otherwise an error message is displayed (program does not crash);

(4) The input values should be checked for duplicate Serial numbers and an error message displayed if a duplicate is found (does not cause the program to crash);

(5) The Serial Number Input should also be checked to ensure it is 10 characters long and only alphanumeric – again an error message is returned without crashing the program;

(6) Other validation will be performed by the main program.

2. <u>SerialSorter.py:</u>

i) When the main program loads, it sorts the Serial File from "C:\SerialNumbers\". The program will read the contents of the file line by line and then store the contents of this file as a variable. This will be passed as an argument to the serialSorter function you have imported from your Module (ERRORS SHOULD BE CAPTURED AS DETAILED IN THE MARKING RUBRIC);

ii) A sorted, nested dictionary will be returned and printed to the screen in a clean format;

iii) The main program will then ask the User if they wish to add new devices to the sorted dictionary:

NOTE: This is **<u>NOT</u>** a group assignment and must be completed individually.

(1) If NO – the sorted information is ***written to a new file*** in the "C:\SerialNumbers\" directory

(2) If YES – the User is then prompted to enter the new information, ***which will be stored as a list variable***:

  (a) Index [0] = Manufacturer. For the purposes of this program ONLY the following are valid, any other input should prompt the user to correct their entry:

    (i) "Cisco"

    (ii) "Meraki"

    (iii) "Juniper"

    (iv) "Aruba"

    (v) "Dell"

  (b) Index [1] = Serial Number

    (i) The Serial Number MUST be 10 alphanumeric digits only – input that is more or less than 10 digits, or that contain special characters such as @#$% should prompt the user to correct their entry (duplicates will be checked in the function)

  (c) Index [2] = Device Type

  (d) Index [3] = Model

  (e) Index [4] = Port Density

  (f) Blank entries for indexes 2 - 4 are not permitted and should force the user to correct their input

  (g) The program should capture errors and continue to loop until the user terminates;

  (h) The main program should then pass this list as an argument to the SerialAdder function (along with the sorted dictionary already generated). This function will perform a validation check as indicated above and add the new entry at the correct location within the dictionary. The modified dictionary will then be returned to the main program;

  (i) The main program will then write this information ***to a new file*** in the "C:\SerialNumbers\" directory

BONUS:

(1) Ask the user to provide a directory and file for sorting rather than having this hard coded (0.5)

(2) Ask the user to provide a directory and file for writing the contents of the sorted Serial Numbers rather than having this hard coded (0.5)

(3) Perform validation that the write path exists and that the file is a .txt format (1)

(4) Automatically correct if the write file extension is not .txt (1)

N.B. ALL basic functionality must be complete and the program fully functional to be eligible for Bonus Marks.

When the **program ends** output to the screen: "This program is courtesy of: ***YourName*** "

TIPS: This might feel a bit over whelming. If so, I recommend the following approach.

Step 1: Create the Function that will be needed for sorting Serial Numbers

Just call and pass it test rows of data to ensure it is accurately assessing the information passed

Step 2: Create the Function that will be needed for adding Serial Numbers

Call the function and send a test, presorted dictionary and new values to ensure it is vetting the input correctly and adding the new entries at the correct location (according to manufacturer)

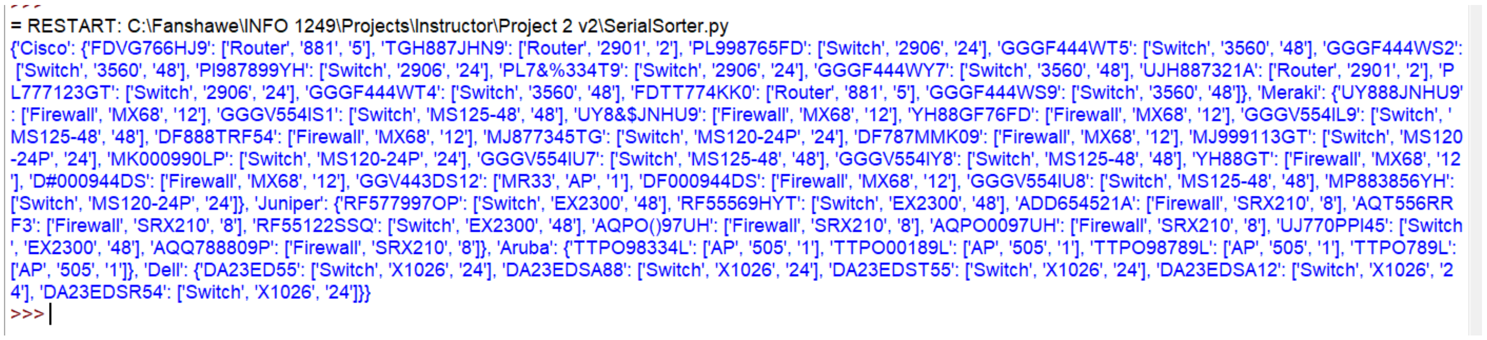
Step 3: Create a simple program the reads the sample input file. Breaks each line into the information you want to capture and ensure each part can be used individually for entry into the dictionary – HINT, preformatting the information from the text file into a list first will help you sort and add into the dictionaries, as each element will be at the same index

Step 4: Create a simple program that asks for input and correctly opens the file; create a simple program that correctly asks for, validates and enters user input into a list variable as described

Step 5: Combine all these steps into one program that meets the above program requirements.

START by ensuring you are able to:

1. Read from the file correctly
2. Pass a stream of text to an imported module function correctly
3. Have that module assign keys and values and return to the main program correctly:

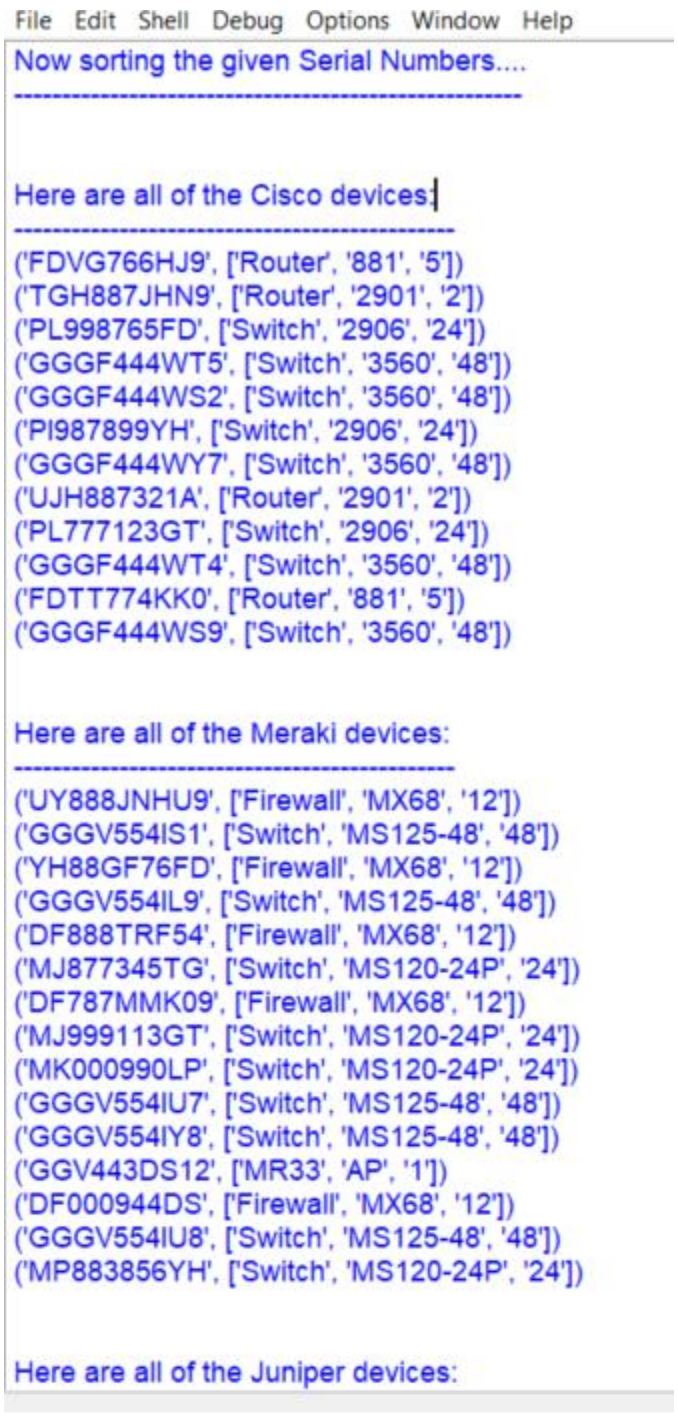


```
= RESTART: C:\Fanshawe\INFO 1249\Projects\Instructor\Project 2 v2\SerialSorter.py
{'Cisco': {'FDVG766HJ9': ['Router', '881', '5'], 'TGH887JHN9': ['Router', '2901', '2'], 'PL998765FD': ['Switch', '2906', '24'], 'GGGF444WT5': ['Switch', '3560', '48'], 'GGGF444WS2':
 ['Switch', '3560', '48'], 'PI987899YH': ['Switch', '2906', '24'], 'PL7&%334T9': ['Switch', '2906', '24'], 'GGGF444WY7': ['Switch', '3560', '48'], 'UJH887321A': ['Router', '2901', '2'], 'P
L777123GT': ['Switch', '2906', '24'], 'GGGF444WT4': ['Switch', '3560', '48'], 'FDTT774KK0': ['Router', '881', '5'], 'GGGF444WS9': ['Switch', '3560', '48']}, 'Meraki': {'UY888JNHU9'
: ['Firewall', 'MX68', '12'], 'GGGV554IS1': ['Switch', 'MS125-48', '48'], 'UY8&$JNHU9': ['Firewall', 'MX68', '12'], 'YH88GF76FD': ['Firewall', 'MX68', '12'], 'GGGV554IL9': ['Switch', '
MS125-48', '48'], 'DF888TRF54': ['Firewall', 'MX68', '12'], 'MJ877345TG': ['Switch', 'MS120-24P', '24'], 'DF787MMK09': ['Firewall', 'MX68', '12'], 'MJ999113GT': ['Switch', 'MS120
-24P', '24'], 'MK000990LP': ['Switch', 'MS120-24P', '24'], 'GGGV554IU7': ['Switch', 'MS125-48', '48'], 'GGGV554IY8': ['Switch', 'MS125-48', '48'], 'YH88GT': ['Firewall', 'MX68', '12
'], 'D#000944DS': ['Firewall', 'MX68', '12'], 'GGV443DS12': ['MR33', 'AP', '1'], 'DF000944DS': ['Firewall', 'MX68', '12'], 'GGGV554IU8': ['Switch', 'MS125-48', '48'], 'MP883856YH':
['Switch', 'MS120-24P', '24']}, 'Juniper': {'RF577997OP': ['Switch', 'EX2300', '48'], 'RF55569HYT': ['Switch', 'EX2300', '48'], 'ADD654521A': ['Firewall', 'SRX210', '8'], 'AQT556RR
F3': ['Firewall', 'SRX210', '8'], 'RF55122SSQ': ['Switch', 'EX2300', '48'], 'AQPO()97UH': ['Firewall', 'SRX210', '8'], 'AQPO0097UH': ['Firewall', 'SRX210', '8'], 'UJ770PPI45': ['Switch
', 'EX2300', '48'], 'AQQ788809P': ['Firewall', 'SRX210', '8']}, 'Aruba': {'TTPO98334L': ['AP', '505', '1'], 'TTPO00189L': ['AP', '505', '1'], 'TTPO98789L': ['AP', '505', '1'], 'TTPO789L':
['AP', '505', '1']}, 'Dell': {'DA23ED55': ['Switch', 'X1026', '24'], 'DA23EDSA88': ['Switch', 'X1026', '24'], 'DA23EDST55': ['Switch', 'X1026', '24'], 'DA23EDSA12': ['Switch', 'X1026', '2
4'], 'DA23EDSR54': ['Switch', 'X1026', '24']}}
>>>
```

This is the most difficult part of the program. Ensure this works before doing any validation.

From there, start to test the values of the file entries (length, alphanumeric content and duplicates).

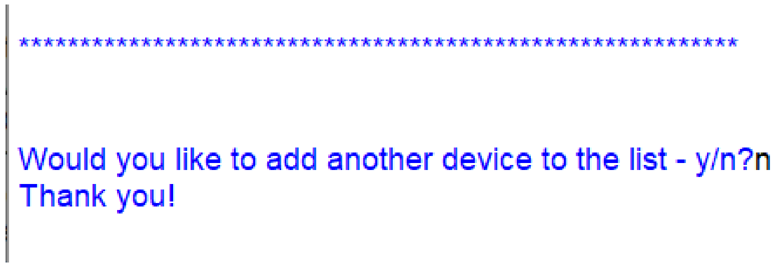
NOTE: This is **NOT** a group assignment and must be completed individually.



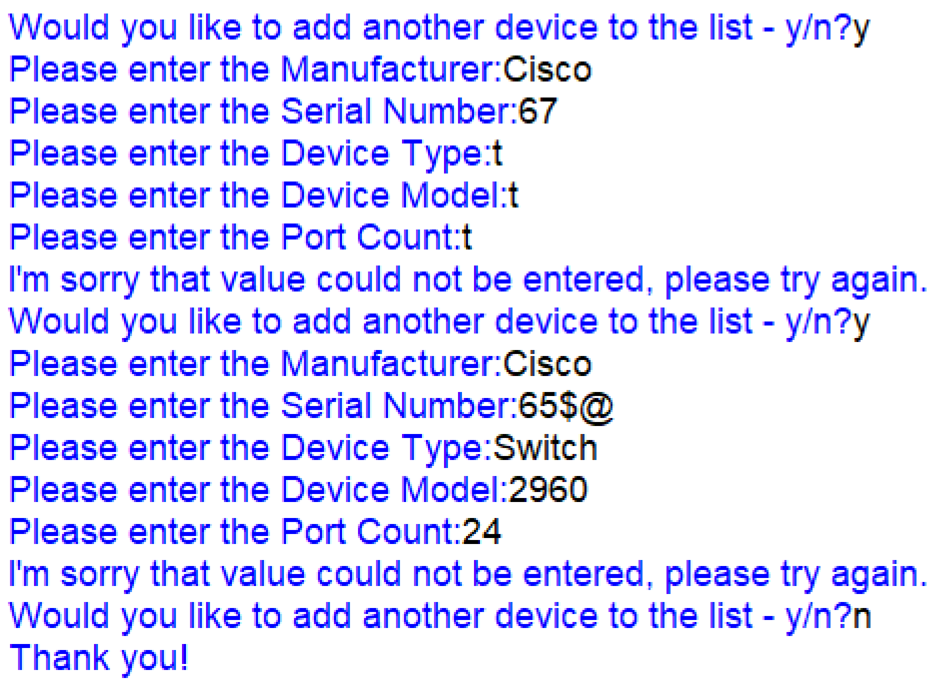
```
Now sorting the given Serial Numbers....
-------------------------------------------------------


Here are all of the Cisco devices:
-----------------------------------------------
('FDVG766HJ9', ['Router', '881', '5'])
('TGH887JHN9', ['Router', '2901', '2'])
('PL998765FD', ['Switch', '2906', '24'])
('GGGF444WT5', ['Switch', '3560', '48'])
('GGGF444WS2', ['Switch', '3560', '48'])
('PI987899YH', ['Switch', '2906', '24'])
('GGGF444WY7', ['Switch', '3560', '48'])
('UJH887321A', ['Router', '2901', '2'])
('PL777123GT', ['Switch', '2906', '24'])
('GGGF444WT4', ['Switch', '3560', '48'])
('FDTT774KK0', ['Router', '881', '5'])
('GGGF444WS9', ['Switch', '3560', '48'])


Here are all of the Meraki devices:
-----------------------------------------------
('UY888JNHU9', ['Firewall', 'MX68', '12'])
('GGGV554IS1', ['Switch', 'MS125-48', '48'])
('YH88GF76FD', ['Firewall', 'MX68', '12'])
('GGGV554IL9', ['Switch', 'MS125-48', '48'])
('DF888TRF54', ['Firewall', 'MX68', '12'])
('MJ877345TG', ['Switch', 'MS120-24P', '24'])
('DF787MMK09', ['Firewall', 'MX68', '12'])
('MJ999113GT', ['Switch', 'MS120-24P', '24'])
('MK000990LP', ['Switch', 'MS120-24P', '24'])
('GGGV554IU7', ['Switch', 'MS125-48', '48'])
('GGGV554IY8', ['Switch', 'MS125-48', '48'])
('GGV443DS12', ['MR33', 'AP', '1'])
('DF000944DS', ['Firewall', 'MX68', '12'])
('GGGV554IU8', ['Switch', 'MS125-48', '48'])
('MP883856YH', ['Switch', 'MS120-24P', '24'])


Here are all of the Juniper devices:
```

NOTE: This is **NOT** a group assignment and must be completed individually.

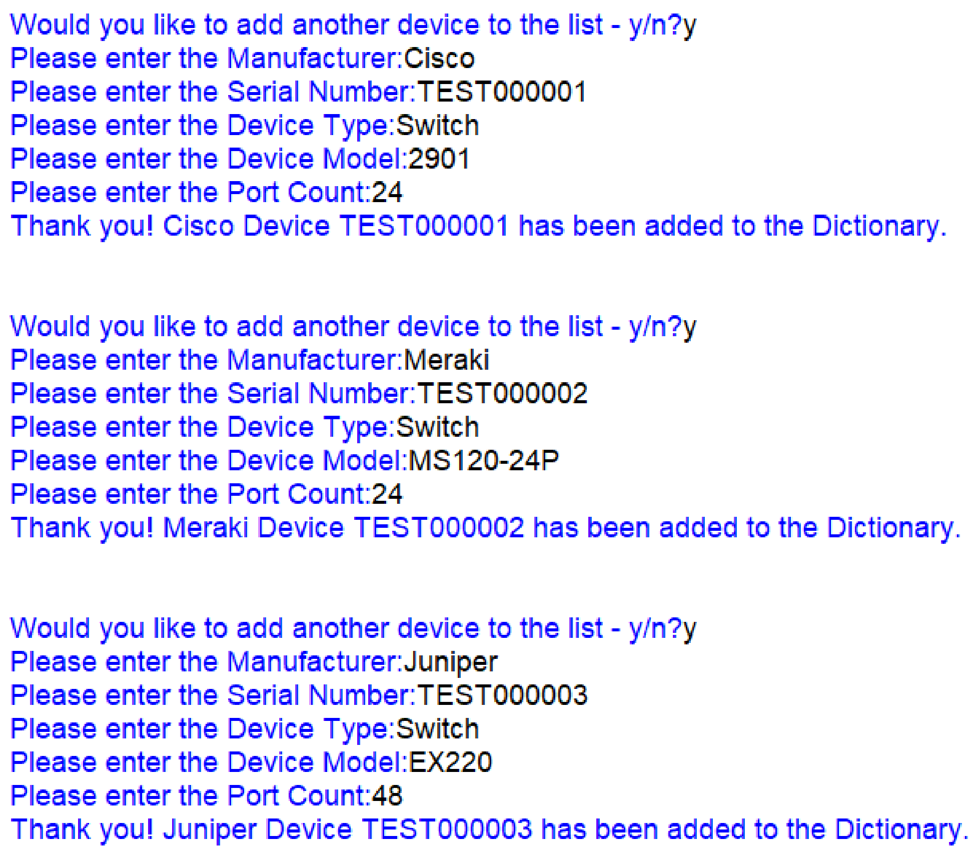
We can then use the second function to ask the user if they wish to add a device and write the final information to a file.



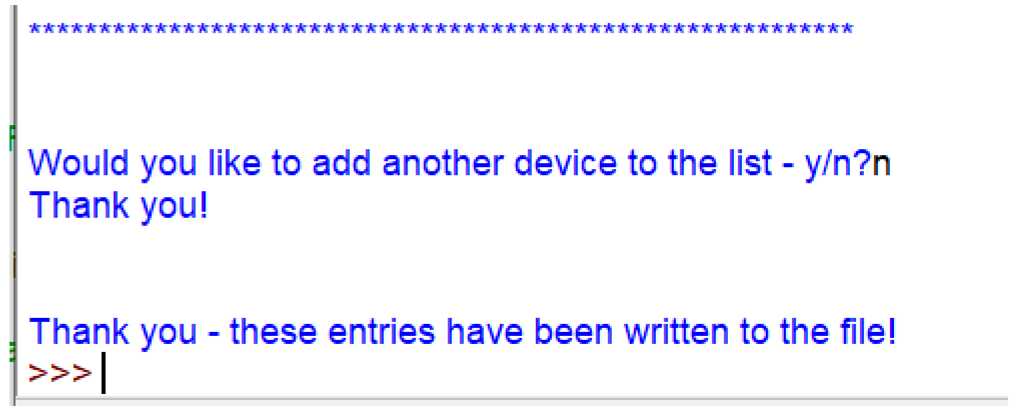
```
**************************************************************



Would you like to add another device to the list - y/n?n
Thank you!
```



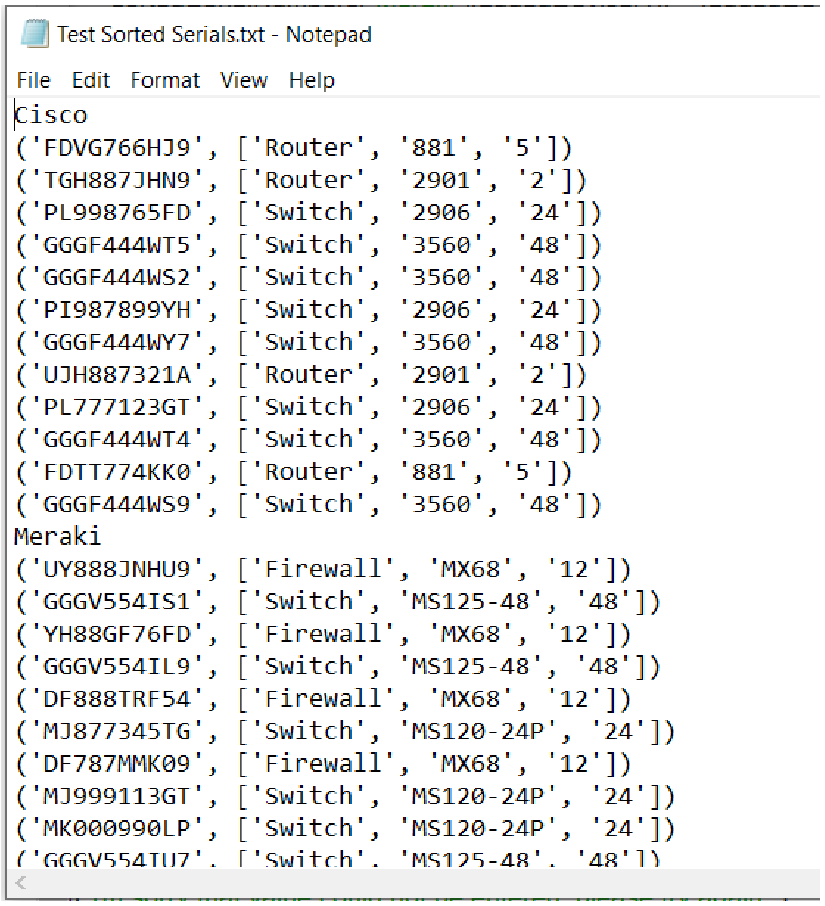
```
Would you like to add another device to the list - y/n?y
Please enter the Manufacturer:Cisco
Please enter the Serial Number:67
Please enter the Device Type:t
Please enter the Device Model:t
Please enter the Port Count:t
I'm sorry that value could not be entered, please try again.
Would you like to add another device to the list - y/n?y
Please enter the Manufacturer:Cisco
Please enter the Serial Number:65$@
Please enter the Device Type:Switch
Please enter the Device Model:2960
Please enter the Port Count:24
I'm sorry that value could not be entered, please try again.
Would you like to add another device to the list - y/n?n
Thank you!
```



```
Would you like to add another device to the list - y/n?y
Please enter the Manufacturer:Cisco
Please enter the Serial Number:TEST000001
Please enter the Device Type:Switch
Please enter the Device Model:2901
Please enter the Port Count:24
Thank you! Cisco Device TEST000001 has been added to the Dictionary.

Would you like to add another device to the list - y/n?y
Please enter the Manufacturer:Meraki
Please enter the Serial Number:TEST000002
Please enter the Device Type:Switch
Please enter the Device Model:MS120-24P
Please enter the Port Count:24
Thank you! Meraki Device TEST000002 has been added to the Dictionary.

Would you like to add another device to the list - y/n?y
Please enter the Manufacturer:Juniper
Please enter the Serial Number:TEST000003
Please enter the Device Type:Switch
Please enter the Device Model:EX220
Please enter the Port Count:48
Thank you! Juniper Device TEST000003 has been added to the Dictionary.
```



```
**************************************************************


Would you like to add another device to the list - y/n?n
Thank you!


Thank you - these entries have been written to the file!
>>>
```



Test Sorted Serials.txt - Notepad

File Edit Format View Help

```
Cisco
('FDVG766HJ9', ['Router', '881', '5'])
('TGH887JHN9', ['Router', '2901', '2'])
('PL998765FD', ['Switch', '2906', '24'])
('GGGF444WT5', ['Switch', '3560', '48'])
('GGGF444WS2', ['Switch', '3560', '48'])
('PI987899YH', ['Switch', '2906', '24'])
('GGGF444WY7', ['Switch', '3560', '48'])
('UJH887321A', ['Router', '2901', '2'])
('PL777123GT', ['Switch', '2906', '24'])
('GGGF444WT4', ['Switch', '3560', '48'])
('FDTT774KK0', ['Router', '881', '5'])
('GGGF444WS9', ['Switch', '3560', '48'])
Meraki
('UY888JNHU9', ['Firewall', 'MX68', '12'])
('GGGV554IS1', ['Switch', 'MS125-48', '48'])
('YH88GF76FD', ['Firewall', 'MX68', '12'])
('GGGV554IL9', ['Switch', 'MS125-48', '48'])
('DF888TRF54', ['Firewall', 'MX68', '12'])
('MJ877345TG', ['Switch', 'MS120-24P', '24'])
('DF787MMK09', ['Firewall', 'MX68', '12'])
('MJ999113GT', ['Switch', 'MS120-24P', '24'])
('MK000990LP', ['Switch', 'MS120-24P', '24'])
('GGGV554IU7', ['Switch', 'MS125-48', '48'])
```

NOTE: This is **NOT** a group assignment and must be completed individually.

MARKING RUBRIC:

Marking Scheme	*Marks*	*Description*
	2	Code runs correctly (Meets the requirements) without crashing (All or nothing).
	2	Has a comment header block listing program name, purpose, coder, and date. Comments are used throughout to explain what is happening (All or nothing).
	2	Creating needed variables and sequences (Lists and Dictionaries) needed.
	2	Module is imported and used correctly.
	6	Function for sorting a file of Serial numbers is correctly called and passed correct information. The function accurately reads the information and validates information passed to it. Function correctly identifies and flags duplicated serial numbers. Function correctly and accurately generates a nested dictionary based on manufacturer and serial number with the correct information stored as values.
	4	Main program asks for, validates and stores information for new devices. This is correctly passed to the second function, which properly validates the information and correctly adds it to the dictionary. Invalid and duplicate entries are captured and flagged without the program crashing.
	4	Properly opens a File for READ to extract Serial information from the text file without error. This is stored and passed as a variable correctly. IO errors are captured in a TRY and EXCEPT block (program does not crash).
	4	Properly opens (creates) a file for WRITE for newly arranged information to be written to at the correct location. IO errors are captured in a TRY and EXCEPT block (program does not crash).
	2	Output shows and writes the correct information.
	2	Screen input / output is in a nice format. (Your Design).
	3 (B)	**BONUS**: Ask the user for the path to the input/output files and performs validation/auto correction. IMPORTANT: Main program must not crash and must perform basic functionality to be eligible for bonus marks.
	30	**TOTAL Marks** (Worth 20% of your final grade in INFO-1249)

NOTE: This is **NOT** a group assignment and must be completed individually.