Name: Zac Fawcett
Date: October 31st, 2021
Submission: PyWin Justification Document

CPRG 260 Project #2

Pywin32 Script

The problem:

Junior admins added recently have caused system downtime and damage to server data integrity. In order to avoid further problems, going forward the newest admins will have a probation period where they are severely restricted in their access to systems administration tasks. We are requesting an introductory level junior admin Python script for windows be created. This script shall be extensible as needed. Certain tasks and metrics that need to be accessed by the most junior admins will have functionality through this script's menu. We want to give them a fool proof way of doing basic admin tasks as avoiding further production downtime is of the upmost importance. The structure of this apparatus should be a simple command line menu.

The main things these junior's will be doing are troubleshooting printers, managing new users and groups, and general machine troubleshooting. The ability to delete users and groups should not be included. We request error checking on user entries to the program that detail what went wrong as well as an auditory signal when a mistake is made. Lastly a log of activity is requested. This log should be only for activity that actually made real changes to the system. There is no need to log the lookup of information or statistics. The logs should have what the activity was and the time and date it occurred. This log will be outputted upon termination of every session of the program and will be stored at C:\\ Users by default. This problem needs a solution without kneecapping the juniors too much, while maintaining the integrity of our company. We look forward to seeing what you come up with.

Management

Name: Zac Fawcett
Date: October 31st, 2021
Submission: PyWin Justification Document

Solution:

The basics:

In order to meet the demands of management I will create a looping python menu. The pywin32 api has various modules that will serve this project well. Firstly the basic concept of the script will be as follows; a command line menu that flashes a numbered options list. Entering one of the numbers on the menu will trigger the beginning of a task or will populate certain info on the screen. There will be error checking on the numbered options, wherein if the user enters an option not listed it will tell them that and reset the menu. Each of the options will have option specific error checking as well, always going back to the main menu if a problem arises so the user can try again if desired. There will also be a program terminating trigger from the main menu so the user can exit when they see fit. Certain actions on the menu will also trigger a logging feature. Upon termination of the program a log file will be written to that contains the actions and corresponding time to that action so management can always know the environment changing actions the junior administrators have taken.

The modules and functions:

Next I will go through each module and the functions contained in them. Each function will be justified as it pertains to its usefulness within the script program.

Name: Zac Fawcett
Date: October 31st, 2021
Submission: PyWin Justification Document

win32api

win32api.GetVersionEx()
This function returns the machines windows version as a tuple. I parsed the tuple into a string that read the major, and minor version with the build number. This is useful for the admins as they can use this info when troubleshooting things like updates or as a reference later if they need to downgrade because of version stability issues.

win32api.MessageBeep(win32con.MB_ICONWARNING)
This function simply uses the sound in the systems " MB_ICONWARNING" file slot to make a sound. This was used as a gauge of what the admin is doing. The only time the program makes a sound is when a mistake or incorrect input is made.

win32api.GetLocalTime()
This function returns the local time as a tuple. This was useful as I could call the local time whenever the user made a system change and add the time to the log file.

win32api.GetDomainName()
Returns the domain name the computer is in. Useful for troubleshooting active directory and domain name services as it verifies if you are in the domain you want to be or not.

win32api.GetTempPath()
This function returns the machines temp folder path. Useful to know as often programs save files here while running and sometimes keep them there afterwards. Knowing this path gives them the admin a good idea where a file they are looking for might be.

Name: Zac Fawcett
Date: October 31st, 2021
Submission: PyWin Justification Document

win32api.GetComputerNameEx()

This function returns the system machine name. This is useful as this info is often needed when troubleshooting domains and networks. It's important to know the name that other machines know your machine as.

win32api.GetWindowsDirectory()

This function returns the windows directory path. It is useful for troubleshooting operating system and kernel problems. Knowing where windows is saved on a computer allows the admin to quickly search for kernel files like the bootstrap, while diagnosing problems like POST or startup issues.

win32api.GlobalMemoryStatusEx()

This function returns the current system memory usage. Useful for troubleshooting performance issues, program specific memory leaks or over-utilization of RAM.

win32api.GetUserName()

Returns the name of currently logged in user. Useful for verification of current user directory and if the Admin is switching users through command line, this will quickly show them if that process was a success or not.

Overall The win32api module was used in this program for returning useful information that could be displayed on command by the admin. These functions were used in the program as a means of quickly delivering specific information that will help in the troubleshooting process of various problems as needed.

Name: Zac Fawcett
Date: October 31st, 2021
Submission: PyWin Justification Document

win32profile

win32profile.GetDefaultUserProfileDirectory()
This function returns a path to the default user directory. Often files are saved by default to this path for the logged in user. Useful for knowing where to look for user specific files.

win32profile.GetProfilesDirectory()
This function returns the path to the directory where user profiles are stored. If the admin wants to make changes to user attributes it is useful to know where this location is. They could navigate here and edit text files to specifically change individual user attributes or ever copy whole profiles as needed.

win32profile.GetEnvironmentStrings()
This function returns the system environment strings. Useful for seeing a list of system environment variables that are installed, could be used for trouble shooting memory usage or processes. In this program the list is printed and used as a menu for the next function.

win32profile.ExpandEnvironmentStringsForUser()

This function expands upon the chosen environment string to give extra information about that string. In this program, the user enters a listing from the previous function and is given the path to that variable on the system. Useful when the user needs the files associated with that system variable quickly.

Overall the win32profile module was used in this program to display paths to important directories that hold files vital to effective administration. The system environment variables are especially important as they give paths to important directories that might otherwise be troublesome to find.

Name: Zac Fawcett
Date: October 31st, 2021
Submission: PyWin Justification Document

win32file

win32file.GetDiskFreeSpace()

This function displays the free space on the disk that calls the function. Useful for a representation of space available for installations or partitioning and also to verify changes made by the admin (transfer of files, volume partition, installations etc).

win32file.FileEncryptionStatus()

This function returns whether or not a specified file is encrypted. The junior admins might come across a file that they cannot read and this function is useful in that, they will be able to tell if that file is corrupted or just encrypted.

Overall the win32file module was used in this program for displaying information useful in gauging available storage as well as verifying file transfers and installations. Also this module gives the user the ability to troubleshoot whether a files contents are corrupted or just encrypted.

Name: Zac Fawcett
Date: October 31st, 2021
Submission: PyWin Justification Document

win32print

win32print.GetDefaultPrinter()
This function returns the name of the device's default printer. Useful to know when troubleshooting print problems like print jobs going to wrong printer or needing the name of the printer for drivers

win32print.OpenPrinter()
This function opens a handle to the default printer. This was used to populate a variable to supply to the next function.

win32print.GetPrinter()
This function returns all the attributes of the printer associates with the handle supplied by the previous function. Useful when troubleshooting printer issues as it gives all the info on the default printer including IP address, model name, comments etc.

win32print.GetPrinterDriverDirectory()
This function provides the path to the directory that stores printer drivers. As the junior admins will largely be responsible for printers and their problems, this path is useful to quickly find copies of drivers and check if they are installed correctly and their versions.

The win32print module was used to supply all the required information for our junior admins to confidently and efficiently handle all printer related jobs and problems.

Name: Zac Fawcett
Date: October 31st, 2021
Submission: PyWin Justification Document

win32net

win32net.NetUserEnum()
Function that returns a list of dictionaries containing all usernames and attributes. This was used to list all users on the system as well as to compare against user entry for error checking in the add user to group, and list groups a user belongs to functions in the program. Useful to know all names of users for administration of groups and troubleshooting user specific issues.

win32net.NetUserGetLocalGroups()
Function that returns a list of local groups to which a specific user belongs. This was used in program to list the groups a user belongs to and also as error checking when using the add user to a local group function. This is useful when administering groups and troubleshooting permissions and group profiles.

win32net.NetLocalGroupEnum()
This function returns a list of groups on machine. In program it was used to create a list that was then cast to a string or tuple. These were then iterated through to print a list or for error checking against user input when executing the create a group or add user to group functions. Useful in when creating a group as a group might already exist with that name, or when administering groups for quick access to a list of available groups.

win32net.NetLocalGroupAddMembers()
Function to add users to a local group. Useful for administering groups which allows for easy control of file permissions and user capabilities. Used in program to add users to a group by name

Name: Zac Fawcett
Date: October 31st, 2021
Submission: PyWin Justification Document

win32net.NetUserGetInfo()

Function returns a dictionary containing user info from the user profile. Used in program to list info about users as requested by name by the user. Useful to see current user profile information.

The win32net module was used in this program for administering users and groups. The ability to create groups, add users to groups and then list local users and groups are vital to the spirit of this project. The junior admins can't be totally kneecapped but also can't be given too much power right away. This program provides an efficient balance between capabilities and restrictions.