Table of Contents

[Chapter 1: Building of conformity program 3](#_Toc62825933)

[1.1 Python: Conformity 3](#_Toc62825934)

[1.1.1 Introduction 3](#_Toc62825935)

[1.1.1.1 Problem Statement 3](#_Toc62825936)

[1.1.2 Existing work 3](#_Toc62825937)

[1.1.2.1 Requirements 3](#_Toc62825938)

[1.1.3 Analysis and design 6](#_Toc62825939)

[1.1.3.1 Implementation 6](#_Toc62825940)

[1.1.3.2 Testing 13](#_Toc62825941)

[1.1.4 Conclusion 18](#_Toc62825942)

[1.1.4.1 Summary 18](#_Toc62825943)

[1.1.4.2 Critical appraisal 18](#_Toc62825944)

[1.1.4.3 Future work 18](#_Toc62825945)

[Chapter 2: Appendix 19](#_Toc62825946)

[2.1 Annexe: Full conformity.py code 19](#_Toc62825947)

List of Figures

[Figure 1: API that will query the Id 4](#_Toc62825948)

[Figure 2: API that contains technical details 4](#_Toc62825949)

[Figure 3: API of the section support 5](#_Toc62825950)

[Figure 4: Welcome user 13](#_Toc62825951)

[Figure 5: Installing Packages and prompt for the password 13](#_Toc62825952)

[Figure 6: Reading the names and querying 14](#_Toc62825953)

[Figure 7: Filled Equipment\_ID\_Or\_Name\_list.txt file 14](#_Toc62825954)

[Figure 8: Wrong credentials 14](#_Toc62825955)

[Figure 9: Prompt user to insert names or ids in the file 15](#_Toc62825956)

[Figure 10: Querying idefix 15](#_Toc62825957)

[Figure 11: Request for the ticket number 16](#_Toc62825958)

[Figure 12: Not entering a number 16](#_Toc62825959)

[Figure 13: Excel file filled with required data 17](#_Toc62825960)

[Figure 14: Created excel file 17](#_Toc62825961)

[Figure 15: End of life of the script 17](#_Toc62825962)

# Chapter 1: Building of conformity program

## Python: Conformity

### Introduction

Helping an admin on his/her task about conformity. Conformity is the verification of equipment details on the intranet (called idefix). If they match the client details about that equipment, it is conformed, else we had to make it conform.

#### Problem Statement

After helping him/her a possibility of automation was found. After a talk with the manager, the manager has approved the project. The intranet already has some APIs, the use of those would ease the task of an admin in a way that he/she has no more repetitive task to perform.

### Existing work

There was no existing work done, my work will be novel for this company. There is only the APIs of the intranet. As from this, the program will be built. The coming chapter will talk about the selection of the methods that will be used to bring the program to life.

#### Requirements

The selected programming language will be python.

For this project, the requirements were:

* Get the id of the equipment
* Get the name
* Get the team in charge
* Get Host Type
* Get the serial number



Figure : API that will query the Id



Figure : API that contains technical details



Figure : API of the section support

After searching among lots of APIs, the needed APIs are the 3 below.

Figure 1 shows the API query that contains the information about:

* id of the equipment
* name

Figure 2 shows the API Technical that contains the:

* Team in charge
* Host Type

Figure 3 shows API support that contains the:

* serial number

### Analysis and design

#### Implementation

Step to build the software:

1. Creating a list that will store the data that will be queried.

s =requests.Session()

IDList = []

NameList = []

HostTypeList = []

TeamInChargeList = []

SerialNumberList =[]

These are global variables that will be filled throughout the running of other methods/functions in the program.

1. Welcome the user

def welcome():

  print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_welcome " + getpass.getuser() + "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

This is a welcome message that will get the username of the current user and output “welcome John.Doe”.

1. Installing python packages

def installPackages():

  print("installing required packages")

  os.system("curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py")

  os.system("python get-pip.py")

  os.system("pip install requests")

  os.system("pip install pandas")

  os.system("pip install xlsxwriter")

  os.system("pip install xlrd")

The program itself will install the required python dependencies and installing pip, the package installer for Python.

1. Get the user to login and save to session

login=getpass.getuser()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_login with your credentials\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

password = getpass.getpass(prompt=login + ' password: ')

credentials = {'login': login, 'password': password}

loggedIn = s.post("https://intranet.linkbynet.com/v7/api/1.1/Authentification.json/LogIn", credentials)

if loggedIn.ok:

  print("your are logged in")

  readFile("Equipment\_ID\_Or\_Name\_list.txt")

This part will prompt the user to enter his/her password. If “loggedIn.ok” means that HTTP code return was 200, therefore you are logged in to the intranet. Session in python was used to keep the user logged in and make his/her query to idefix without having to log in again. This will decrease the querying speed. Then the file “Equipment\_ID\_Or\_Name\_list.txt” will be read.

1. Read the equipment list from a text file

def readFile(filepath):

  if (pathlib.Path(filepath).exists()) and (os.stat(filepath).st\_size > 0):

    populateLists()

  else:

    open(filepath, "w")

    print("insert equipment ids or names in the opened text file")

    openFile(filepath)

    input('press enter when finished inserting and saving')

    readFile(filepath)

def openFile(filepath):

  print("opening file")

  if platform.system() == 'Darwin':       # macOS

    subprocess.call(('open', filepath))

  elif platform.system() == 'Windows':    # Windows

    os.startfile(filepath)

  else:                                   # linux variants

    subprocess.call(('xdg-open', filepath))

The “readFile” method will verify if the file exists and of its size is greater than 0 (that means that it contains something or is not empty) then it will populate the python variables. If the previous condition is not met, it will create and open the file “Equipment\_ID\_Or\_Name\_list.txt” and pause itself until you fill the opened file, saved and press enter. This is called a recursive method.

1. Read the id or name of the equipment pass them to the technical API and the support API. While querying these APIs, the required fields are stored into their respective lists.

def populateLists():

print("opening file 'Equipment\_ID\_Or\_Name\_list.txt'")

print("please wait...")

fe = open("Equipment\_ID\_Or\_Name\_list.txt", "r")

Lines = fe.readlines()

for line in Lines:

equipmentIDOrName = str(line.strip())

reqQuery = s.get("https://intranet.linkbynet.com/v7/api/1.1/Equipment.json/Search?Query=" + equipmentIDOrName)

# "Id"

# "Name"

# "OutSourcingLevelName"

# "Type"

# "Company\_Name"

# "Ips"

# "Projects"

# "ProjectList"

# "HostType"

# "IsVirtualized"

# "IsDisabled"

# "CIType"

# "OSType"

# "OSVersion"

# "Room"

# "Bay"

# "DNSBackup1"

# "DNSBackup2"

# "VCenterName"

# "Company\_Id"

equipmentID = ''

Name = ''

TeamInCharge = ''

HostType = ''

SerialNumber=''

if reqQuery.ok:

jsonQueryAPI = json.loads(json.dumps(reqQuery.json()[0]))

equipmentID = jsonQueryAPI['Id']

Name = jsonQueryAPI['Name']

HostType = jsonQueryAPI['HostType']

reqTechnical = s.get("https://intranet.linkbynet.com/v7/api/1.1/Equipment.json/" + str(equipmentID) + "/General/Technical")

# "Type"

# "relationType"

# "CIClass"

# "CINumber"

# "Parent\_Name"

# "ParentEquipments"

# "HostedEquipments"

# "VCenterName"

# "Cluster\_Name"

# "VIPServer"

# "WebClient\_Url"

# "Parent\_Id"

# "Environment"

# "EnvironmentFunction"

# "BackupFunction"

# "MonitoringFunction"

# "PrimaireOrSecondaire"

# "PrimaryServer\_Name"

# "PrimaryServer\_Id"

# "comment"

# "TeamInCharge"

# "Firmware\_Name"

# "id\_cloud"

if reqTechnical.ok:

jsonTechAPI = reqTechnical.json()

TeamInCharge = jsonTechAPI['TeamInCharge']

else:

TeamInCharge = "fail to get data"

reqSupport = s.get("https://intranet.linkbynet.com/v7/api/1.1/Equipment.json/" + str(equipmentID) + "/General/Support")

# "ManufacturerName"

# "ProductName"

# "Model"

# "Id"

# "Mid"

# "SerialNumber"

# "ProductNumber"

# "SupportContractNumber"

# "CustomerServicePhone"

# "UserFullName"

# "IsInStock"

# "StorageDate"

# "MessageStorageSuppression"

# "IsDisabled"

# "IsBlacklisted"

# "ArchiveDate"

# "MessageSuppression"

if reqSupport.ok:

jsonSupportAPI = reqSupport.json()

SerialNumber = jsonSupportAPI['SerialNumber']

else:

SerialNumber = "fail to get data"

else:

equipmentID = "fail to get ID"

Name = "fail to get ID"

HostType = "fail to get ID"

IDList.append(equipmentID)

NameList.append(Name)

HostTypeList.append(HostType)

TeamInChargeList.append(TeamInCharge)

SerialNumberList.append(SerialNumber)

fe.close()

The snippet above is where the magic occurs. While reading the file “Equipment\_ID\_Or\_Name\_list.txt”, each line is stored in a variable named “equipmentIDOrName”. This is then pass in the idefix API "https://intranet.linkbynet.com/v7/api/1.1/Equipment.json/Search?Query=" + equipmentIDOrName. From the returned values below:

"Id", "Name", "OutSourcingLevelName", "Type", "Company\_Name", "Ips", "Projects", "ProjectList", "HostType", "IsVirtualized", "IsDisabled", "CIType", "OSType", "OSVersion", "Room", "Bay", "DNSBackup1", "DNSBackup2", "VCenterName", "Company\_Id"

The id, Name and HostType are stored in their respective lists.

equipmentID = jsonQueryAPI['Id']

Name = jsonQueryAPI['Name']

HostType = jsonQueryAPI['HostType']

Then another 2 APIs are being queried to retrieve the serial number and team in charge. Why querying 2 times? Because the APIs that contain the serial number and team in charge require the id as input which won’t be declared if the file “Equipment\_ID\_Or\_Name\_list.txt” contains the name rather than the ids.

1. Request the ticket number

def inputNumber(message):

  while True:

    try:

       userInput = int(input(message))

    except ValueError:

       print("Not an integer! Try again.")

       continue

    else:

       return userInput

       break

Then the program will request for the ticket number which will be inserted in an excel sheet later. It will check if really a number was inserted. And will continue to prompt until it is the case.

1. Write the dataframe to excel using pandas

def writeToExcel(ticketNumber,IDList, NameList, HostTypeList, TeamInChargeList,SerialNumberList):

  print("Writing data to excel")

  df = pd.DataFrame({

    "IDs of Machines": IDList,

    "Names of Machines": NameList,

    "Host Type": HostTypeList,

    "Team In Charge": TeamInChargeList,

    "Serial Number": SerialNumberList,

  })

  writer = pd.ExcelWriter('conformity\_followup\_' + ticketNumber + '.xlsx', engine='xlsxwriter')# pylint: disable=abstract-class-instantiated

  df.to\_excel(writer, sheet\_name='conformity\_followup\_' + ticketNumber, index=False)

  writer.save()

  openFile(str('conformity\_followup\_' + ticketNumber + '.xlsx'))

This piece of code will take the lists populated previously and output them in an excel file and sheet with the name “conformity\_followup\_” + ticketNumber + “.xlsx” and 'conformity\_followup\_' + ticketNumber respectively. After that, the excel file will open automatically.

1. End.

print("\n\nIf you are using this program you owe Arouven POOLIAN a lunch!")

input("press enter to exit ...")

#s.close()

exit(0)

That will be great to have lunch from the person’s that uses the script. From here everything is done, the program exits with code 0 which means successful execution. The full code is attached in Annexe: Full conformity.py code.

#### Testing

First, the testing without forcing to errors are made.

Figure 4 shows the testing of the welcome message and the enquiry of the username.

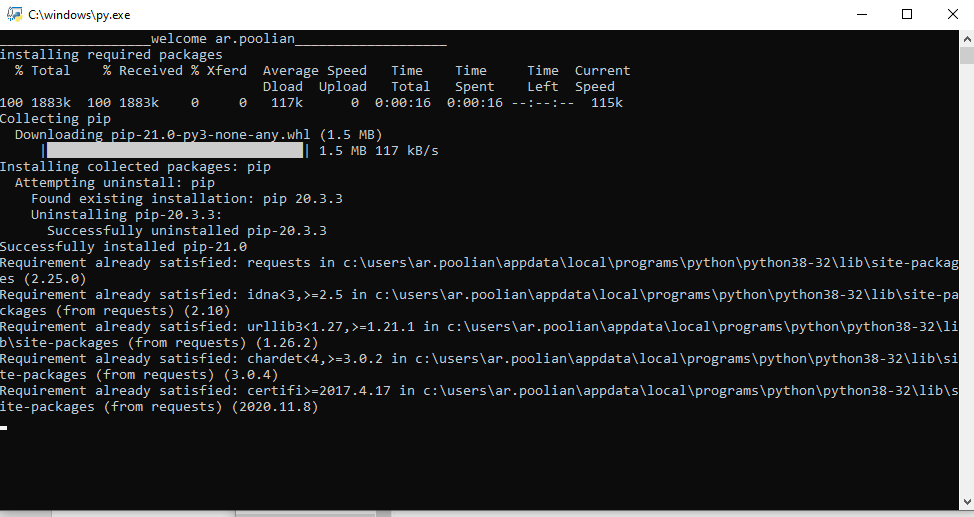


Figure : Welcome user

Figure 5 shows how the program installs the required pip packages and the prompting of the password which is invisible while inserting.

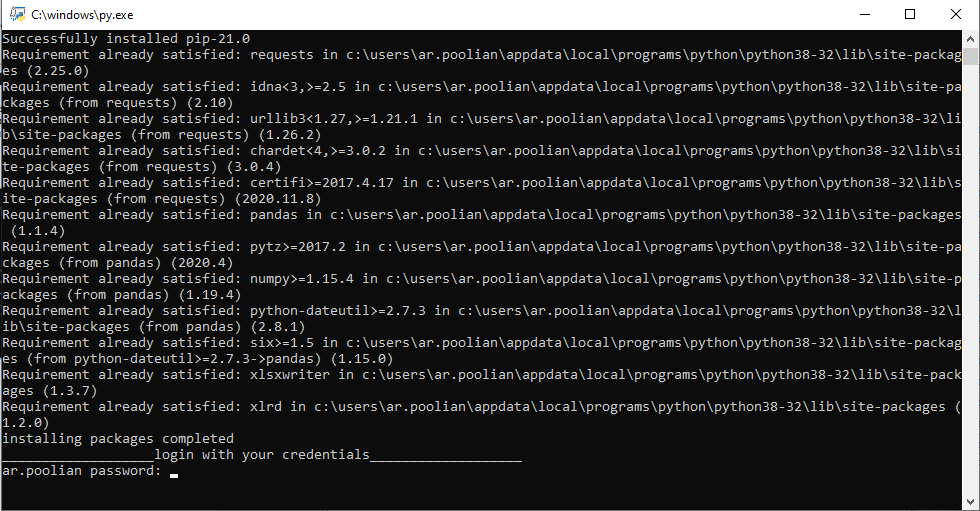


Figure : Installing Packages and prompt for the password

Figure 6 shows how the script is reading from an equipment list (Equipment\_ID\_Or\_Name\_list.txt) that contain something first and then later the test will be made on no file and no name in the file. The data in the file is shown in Figure 7.

Figure 8 shows what happens if the credential is wrongly inserted. It will through an error due to not receiving a 200 HTTP code and will exit.

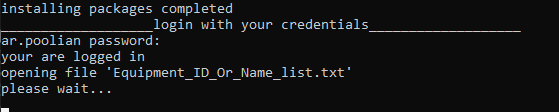


Figure : Reading the names and querying

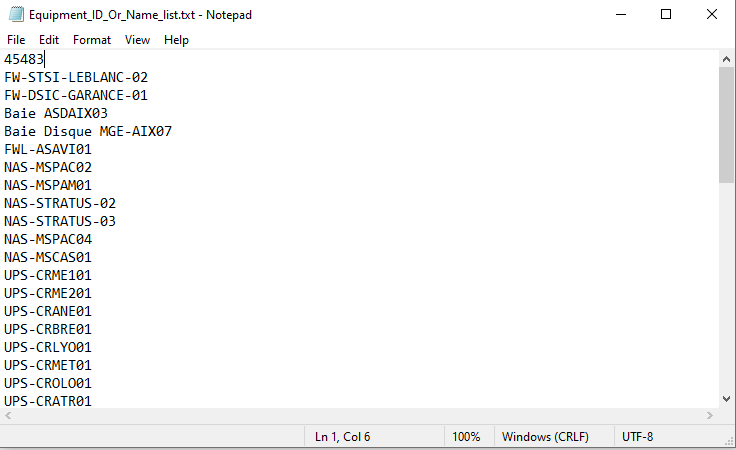


Figure : Filled Equipment\_ID\_Or\_Name\_list.txt file



Figure : Wrong credentials

Figure 9 shows if there is nothing in the file or the file does not exist, the script will create it an opens it and ask the user to insert the names or ids of the equipment and save the file. Once saved, the program will continue normally as shown in Figure 10.

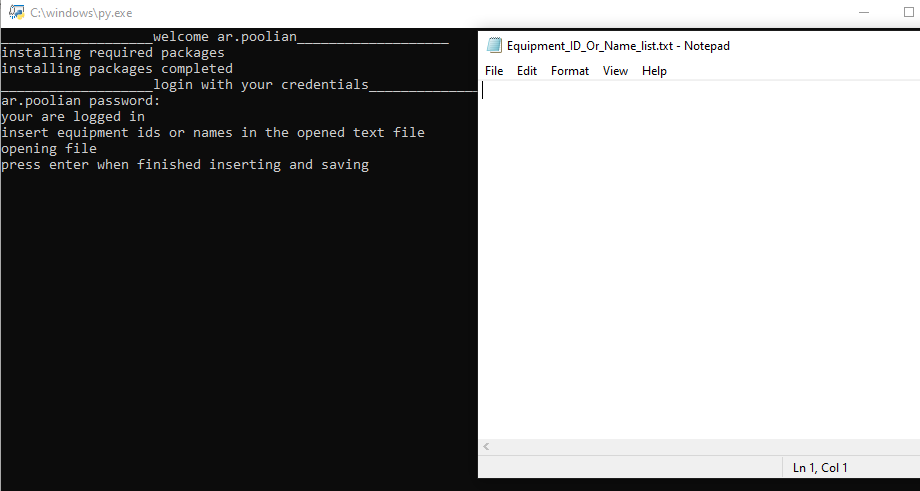


Figure : Prompt user to insert names or ids in the file

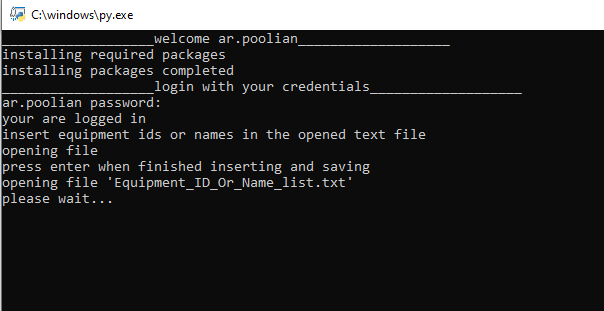


Figure : Querying idefix

Figure 11 shows how the program prompt to insert the ticket number (uniquely identified by the intranet(idefix)). This was made to separate the job of each conformity ticket. So as the admin did not mix with other conformity tickets (conformity job). The script will check if the input was a number if not it will continue to prompt for a number as shown in Figure 12.

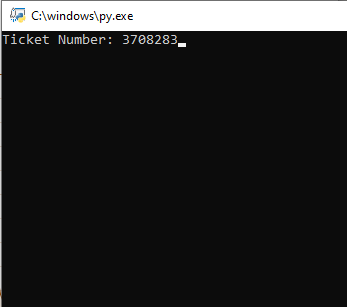


Figure : Request for the ticket number

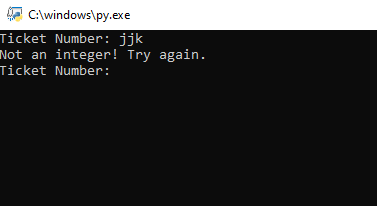


Figure : Not entering a number

The outputs of the full program are shown below. Figure 13 shows the output excel sheet fill with the required data and the name as per the ticket number. Figure 14 shows the created excel file in the windows file explorer and of course, the ending of the program is shown in Figure 15.

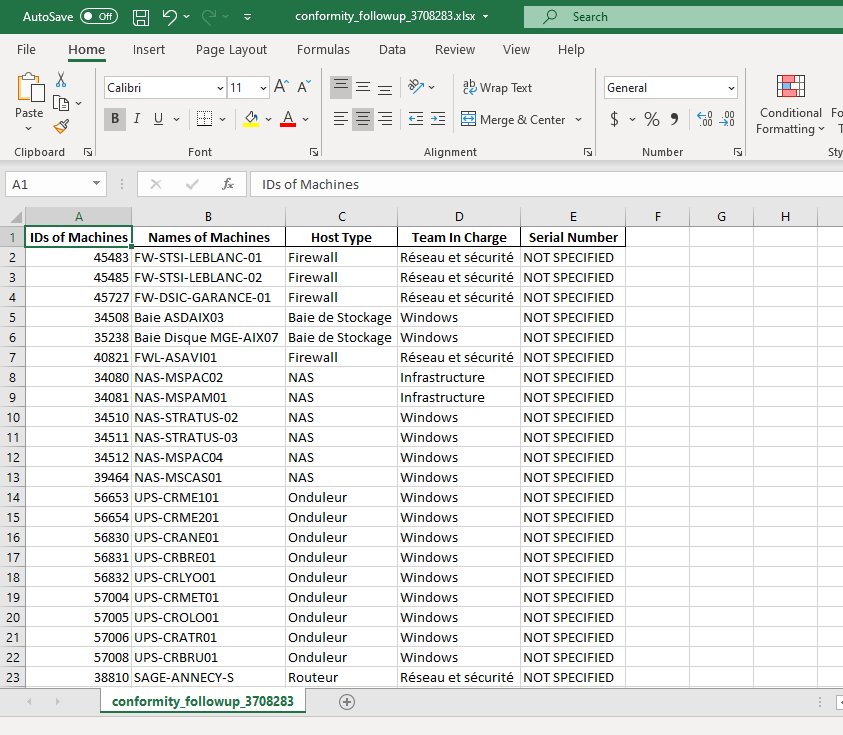


Figure : Excel file filled with required data



Figure : Created excel file

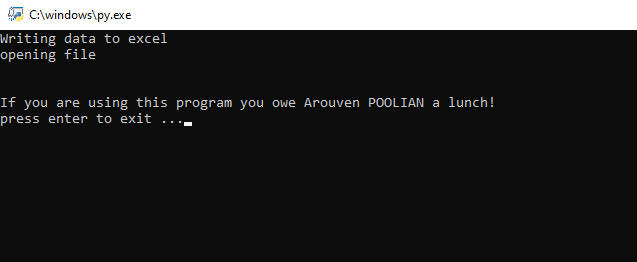


Figure : End of life of the script

### Conclusion

#### Summary

The program certainly took time to be build but in the long term, the time will be diminished as the program works a lot faster than a human. 2-hours work is being done in less than 5 minutes. The experiences gained are the use of APIs, JSON manipulation, API to query other APIs, Sessions in Python... But my main experience gained stays do not work hard, work smart.

#### Critical appraisal

The query part was a bit longer just a please wait is not enough, a percentage bar would be great and more appealing. Testing on other platforms such as Linux and Mac should be taken into consideration.

#### Future work

Make use of GUI that have checkboxes to select the desired keys and values.

# Appendix

## Annexe: Full conformity.py code

import json

import getpass

import os

import pathlib

import subprocess

import platform

import requests# pip install requests

import pandas as pd  # pip install pandas

def welcome():

  print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_welcome " + getpass.getuser() + "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

def installPackages():

  print("installing required packages")

  os.system("curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py")

  os.system("python get-pip.py")

  os.system("pip install requests")

  os.system("pip install pandas")

  os.system("pip install xlsxwriter")

  os.system("pip install xlrd")

def populateLists():

  print("opening file 'Equipment\_ID\_Or\_Name\_list.txt'")

  print("please wait...")

  fe = open("Equipment\_ID\_Or\_Name\_list.txt", "r")

  Lines = fe.readlines()

  for line in Lines:

    equipmentIDOrName = str(line.strip())

    reqQuery = s.get("https://intranet.linkbynet.com/v7/api/1.1/Equipment.json/Search?Query=" + equipmentIDOrName)

    # "Id"

    # "Name"

    # "OutSourcingLevelName"

    # "Type"

    # "Company\_Name"

    # "Ips"

    # "Projects"

    # "ProjectList"

    # "HostType"

    # "IsVirtualized"

    # "IsDisabled"

    # "CIType"

    # "OSType"

    # "OSVersion"

    # "Room"

    # "Bay"

    # "DNSBackup1"

    # "DNSBackup2"

    # "VCenterName"

    # "Company\_Id"

    equipmentID = ''

    Name = ''

    TeamInCharge = ''

    HostType = ''

    SerialNumber=''

    if reqQuery.ok:

      jsonQueryAPI = json.loads(json.dumps(reqQuery.json()[0]))

      equipmentID = jsonQueryAPI['Id']

      Name = jsonQueryAPI['Name']

      HostType = jsonQueryAPI['HostType']

      reqTechnical = s.get("https://intranet.linkbynet.com/v7/api/1.1/Equipment.json/" + str(equipmentID) + "/General/Technical")

      # "Type"

      # "relationType"

      # "CIClass"

      # "CINumber"

      # "Parent\_Name"

      # "ParentEquipments"

      # "HostedEquipments"

      # "VCenterName"

      # "Cluster\_Name"

      # "VIPServer"

      # "WebClient\_Url"

      # "Parent\_Id"

      # "Environment"

      # "EnvironmentFunction"

      # "BackupFunction"

      # "MonitoringFunction"

      # "PrimaireOrSecondaire"

      # "PrimaryServer\_Name"

      # "PrimaryServer\_Id"

      # "comment"

      # "TeamInCharge"

      # "Firmware\_Name"

      # "id\_cloud"

      if reqTechnical.ok:

        jsonTechAPI = reqTechnical.json()

        TeamInCharge = jsonTechAPI['TeamInCharge']

      else:

        TeamInCharge = "fail to get data"

      reqSupport = s.get("https://intranet.linkbynet.com/v7/api/1.1/Equipment.json/" + str(equipmentID) + "/General/Support")

      # "ManufacturerName"

      # "ProductName"

      # "Model"

      # "Id"

      # "Mid"

      # "SerialNumber"

      # "ProductNumber"

      # "SupportContractNumber"

      # "CustomerServicePhone"

      # "UserFullName"

      # "IsInStock"

      # "StorageDate"

      # "MessageStorageSuppression"

      # "IsDisabled"

      # "IsBlacklisted"

      # "ArchiveDate"

      # "MessageSuppression"

      if reqSupport.ok:

        jsonSupportAPI = reqSupport.json()

        SerialNumber = jsonSupportAPI['SerialNumber']

      else:

        SerialNumber = "fail to get data"

    else:

      equipmentID = "fail to get ID"

      Name = "fail to get ID"

      HostType = "fail to get ID"

    IDList.append(equipmentID)

    NameList.append(Name)

    HostTypeList.append(HostType)

    TeamInChargeList.append(TeamInCharge)

    SerialNumberList.append(SerialNumber)

  fe.close()

def readFile(filepath):

  if (pathlib.Path(filepath).exists()) and (os.stat(filepath).st\_size > 0):

    populateLists()

  else:

    open(filepath, "w")

    print("insert equipment ids or names in the opened text file")

    openFile(filepath)

    input('press enter when finished inserting and saving')

    readFile(filepath)

def openFile(filepath):

  print("opening file")

  if platform.system() == 'Darwin':       # macOS

    subprocess.call(('open', filepath))

  elif platform.system() == 'Windows':    # Windows

    os.startfile(filepath)

  else:                                   # linux variants

    subprocess.call(('xdg-open', filepath))

def writeToExcel(ticketNumber,IDList, NameList, HostTypeList, TeamInChargeList,SerialNumberList):

  print("Writing data to excel")

  df = pd.DataFrame({

    "IDs of Machines": IDList,

    "Names of Machines": NameList,

    "Host Type": HostTypeList,

    "Team In Charge": TeamInChargeList,

    "Serial Number": SerialNumberList,

  })

  writer = pd.ExcelWriter('conformity\_followup\_' + ticketNumber + '.xlsx', engine='xlsxwriter')# pylint: disable=abstract-class-instantiated

  df.to\_excel(writer, sheet\_name='conformity\_followup\_' + ticketNumber, index=False)

  writer.save()

  openFile(str('conformity\_followup\_' + ticketNumber + '.xlsx'))

def inputNumber(message):

  while True:

    try:

       userInput = int(input(message))

    except ValueError:

       print("Not an integer! Try again.")

       continue

    else:

       return userInput

       break

##########################################

# for lis in jsn\_list:

#     for key,val in lis.items():

#         print(key, val)

s =requests.Session()

IDList = []

NameList = []

HostTypeList = []

TeamInChargeList = []

SerialNumberList =[]

os.system('cls' if os.name == 'nt' else 'clear')

welcome()

installPackages()

print("installing packages completed")

login=getpass.getuser()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_login with your credentials\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

password = getpass.getpass(prompt=login + ' password: ')

credentials = {'login': login, 'password': password}

loggedIn = s.post("https://intranet.linkbynet.com/v7/api/1.1/Authentification.json/LogIn", credentials)

if loggedIn.ok:

  print("your are logged in")

  readFile("Equipment\_ID\_Or\_Name\_list.txt")

else :

  os.system('cls' if os.name == 'nt' else 'clear')

  print("Wrong Credentials")

  input("press enter to exit")

  exit(0)

os.system('cls' if os.name == 'nt' else 'clear')

ticketNum = str(inputNumber('Ticket Number: '))

os.system('cls' if os.name == 'nt' else 'clear')

writeToExcel(ticketNum, IDList, NameList, HostTypeList, TeamInChargeList,SerialNumberList)

print("\n\nIf you are using this program you owe Arouven POOLIAN a lunch!")

input("press enter to exit ...")

exit(0)