Table of Contents

List of Figures

Chapter 1: Building of conformity program

1.1 Python: Conformity

1.1.1 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.

1.1.1.1 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.

1.1.2 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.

1.1.2.1 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

intranet.linkbynet.com/v7/api/1.1/Equipment.json/Search?Query=SWI-R6ATR01


```
[
  {
    "Id": 40363,
    "Name": "SWI-R6ATR01",
    "OutSourcingLevelName": "6 - Hébergement",
    "Type": "Switch Core\r\nN/A",
    "Company_Name": "SAGE SAS",
    "Ips": "",
    "Projects": "Conformity_FollowUp\r\nSAGE SI - RSX",
    "ProjectList": "31982,17256",
    "HostType": "Switch Core",
    "IsVirtualized": false,
    "IsDisabled": false,
    "CIType": "SWITCH_CORE",
    "OSType": "",
    "OSVersion": "",
    "Room": null,
    "Bay": null,
    "DNSBackup1": null,
    "DNSBackup2": null,
    "VCenterName": "",
    "Company_Id": 2537
  }
]
```


Figure 1: API that will query the Id

intranet.linkbynet.com/v7/api/1.1/Equipment.json/40363/General/Technical


```
{
  "Type": "Switch Core",
  "relationType": "",
  "CIClass": "",
  "CINumber": "",
  "Parent_Name": "Aucun",
  "ParentEquipments": "0 Equipment",
  "HostedEquipments": "0 Equipment",
  "VCenterName": null,
  "Cluster_Name": null,
  "VIPServer": "",
  "WebClient_Url": null,
  "Parent_Id": "",
  "Environment": "Undefined",
  "EnvironmentFunction": "N/A",
  "BackupFunction": "Aucun",
  "MonitoringFunction": "Aucun",
  "PrimaireOrSecondaire": "Primaire",
  "PrimaryServer_Name": null,
  "PrimaryServer_Id": null,
  "comment": "",
  "TeamInCharge": "Réseau et sécurité",
  "Firmware_Name": null,
  "id_cloud": null
}
```


Figure 2: API that contains technical details

intranet.linkbynet.com/v7/api/1.1/Equipment.json/40363/General/Support


```
{
  "ManufacturerName": "Cisco",
  "ProductName": null,
  "Model": "TBF",
  "Id": 40363,
  "Mid": "",
  "SerialNumber": "NOT SPECIFIED",
  "ProductNumber": "",
  "SupportContractNumber": "",
  "CustomerServicePhone": "",
  "UserFullName": "",
  "IsInStock": false,
  "StorageDate": 0,
  "MessageStorageSuppression": "<span style='color:red'>Please enter the date the equipment has been put in stock</span>",
  "IsDisabled": false,
  "IsBlacklisted": false,
  "ArchiveDate": 0,
  "MessageSuppression": "<span style='color:red'>Please enter the date the equipment has been archived</span>"
}
```


Figure 3: 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

1.1.3 Analysis and design

1.1.3.1 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.

2. 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".

3. 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.

4. 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/Authentif
ication.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.

5. 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.

6. 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/Equipm
ent.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/Equipmen
t.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.

7. 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.

8. Write the dataframe to excel using pandas

```
def writeToExcel(ticketNumber,IDList, NameList, HostTypeList, TeamInCh
argeList,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 + '.xl
sx', 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.

9. End.

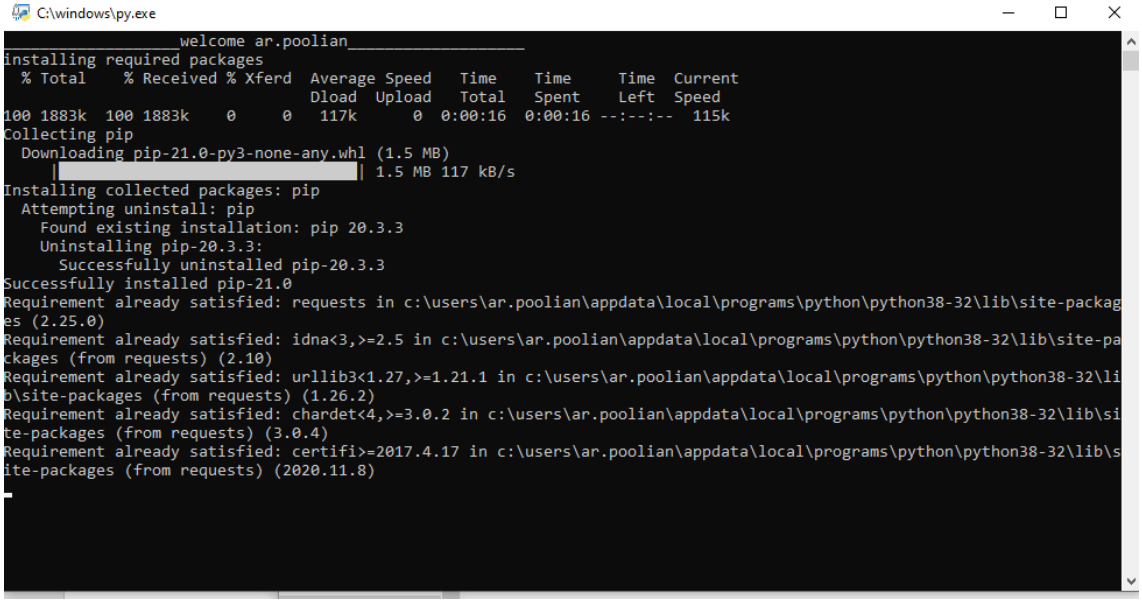
```
print("\n\nIf you are using this program you owe Arouven POOLIAN a lun
ch!")
input("press enter to exit ...")
#s.close()
exit(0)
```

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.

1.1.3.2 Testing

First, the testing without forcing to errors are made.

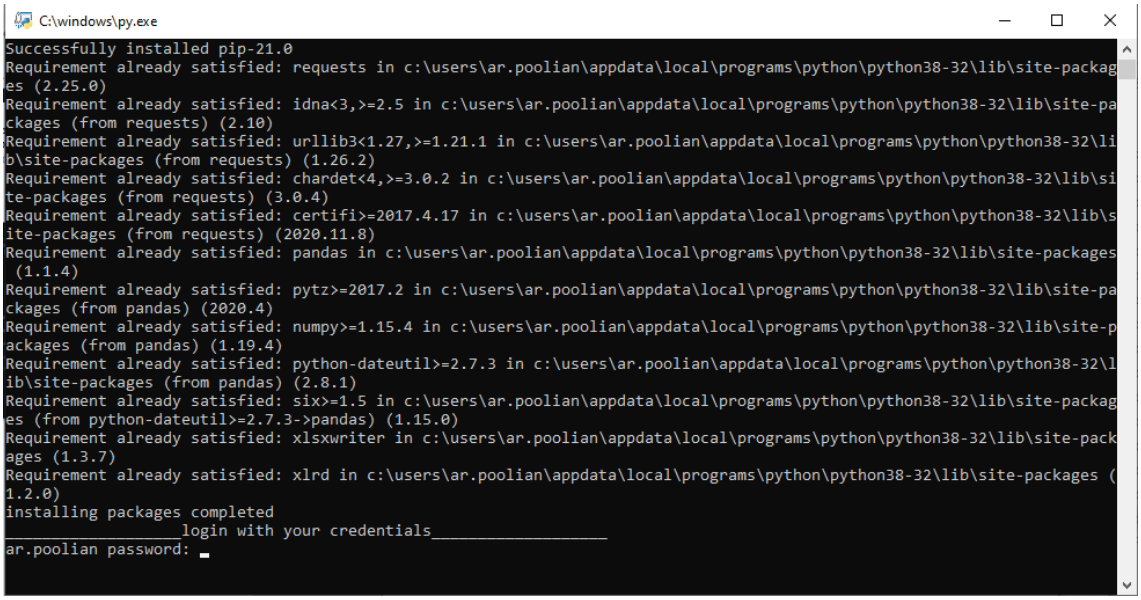
FIGURE 4 shows the testing of the welcome message and the enquiry of the username.



```
C:\windows\py.exe
_________________welcome ar.poolian_________________
installing required packages
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 1883k  100 1883k    0     0   117k      0  0:00:16  0:00:16 --:--:--  115k
Collecting pip
  Downloading pip-21.0-py3-none-any.whl (1.5 MB)
     |████████████████████████████████| 1.5 MB 117 kB/s
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.3.3
    Uninstalling pip-20.3.3:
      Successfully uninstalled pip-20.3.3
Successfully installed pip-21.0
Requirement already satisfied: requests in c:\users\ar.poolian\appdata\local\programs\python\python38-32\lib\site-packag
es (2.25.0)
Requirement already satisfied: idna<3,>=2.5 in c:\users\ar.poolian\appdata\local\programs\python\python38-32\lib\site-pa
ckages (from requests) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\ar.poolian\appdata\local\programs\python\python38-32\li
b\site-packages (from requests) (1.26.2)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\ar.poolian\appdata\local\programs\python\python38-32\lib\si
te-packages (from requests) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ar.poolian\appdata\local\programs\python\python38-32\lib\s
ite-packages (from requests) (2020.11.8)
```

Figure 4: Welcome user

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



```
C:\windows\py.exe
Successfully installed pip-21.0
Requirement already satisfied: requests in c:\users\ar.poolian\appdata\local\programs\python\python38-32\lib\site-packag
es (2.25.0)
Requirement already satisfied: idna<3,>=2.5 in c:\users\ar.poolian\appdata\local\programs\python\python38-32\lib\site-pa
ckages (from requests) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\ar.poolian\appdata\local\programs\python\python38-32\li
b\site-packages (from requests) (1.26.2)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\ar.poolian\appdata\local\programs\python\python38-32\lib\si
te-packages (from requests) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ar.poolian\appdata\local\programs\python\python38-32\lib\s
ite-packages (from requests) (2020.11.8)
Requirement already satisfied: pandas in c:\users\ar.poolian\appdata\local\programs\python\python38-32\lib\site-packages
 (1.1.4)
Requirement already satisfied: pytz>=2017.2 in c:\users\ar.poolian\appdata\local\programs\python\python38-32\lib\site-pa
ckages (from pandas) (2020.4)
Requirement already satisfied: numpy>=1.15.4 in c:\users\ar.poolian\appdata\local\programs\python\python38-32\lib\site-p
ackages (from pandas) (1.19.4)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\ar.poolian\appdata\local\programs\python\python38-32\l
ib\site-packages (from pandas) (2.8.1)
Requirement already satisfied: six>=1.5 in c:\users\ar.poolian\appdata\local\programs\python\python38-32\lib\site-packag
es (from python-dateutil>=2.7.3->pandas) (1.15.0)
Requirement already satisfied: xlsxwriter in c:\users\ar.poolian\appdata\local\programs\python\python38-32\lib\site-pack
ages (1.3.7)
Requirement already satisfied: xlrd in c:\users\ar.poolian\appdata\local\programs\python\python38-32\lib\site-packages (
1.2.0)
installing packages completed
_________________login with your credentials_________________
ar.poolian password:
```

Figure 5: 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.

```
installing packages completed
___________________login with your credentials___________________
ar.poolian password:
your are logged in
opening file 'Equipment_ID_Or_Name_list.txt'
please wait...
```

Figure 6: Reading the names and querying

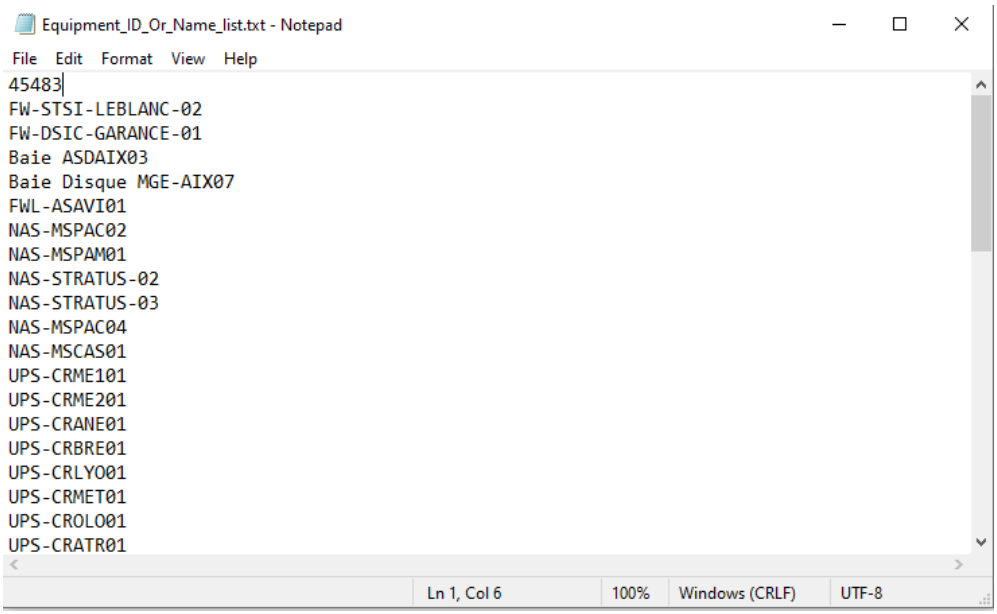


Figure 7: Filled Equipment_ID_Or_Name_list.txt file

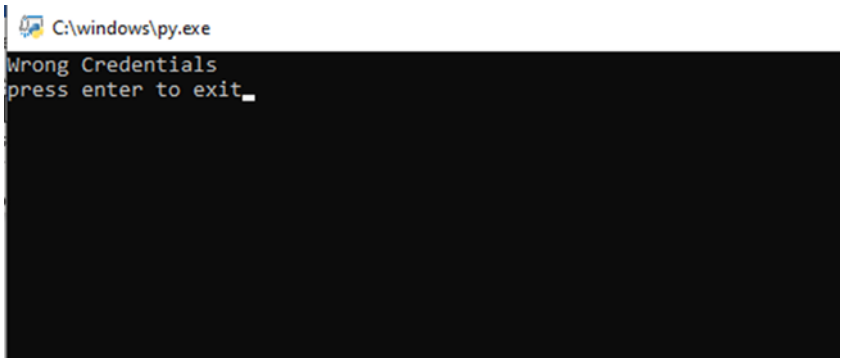


Figure 8: Wrong credentials

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.

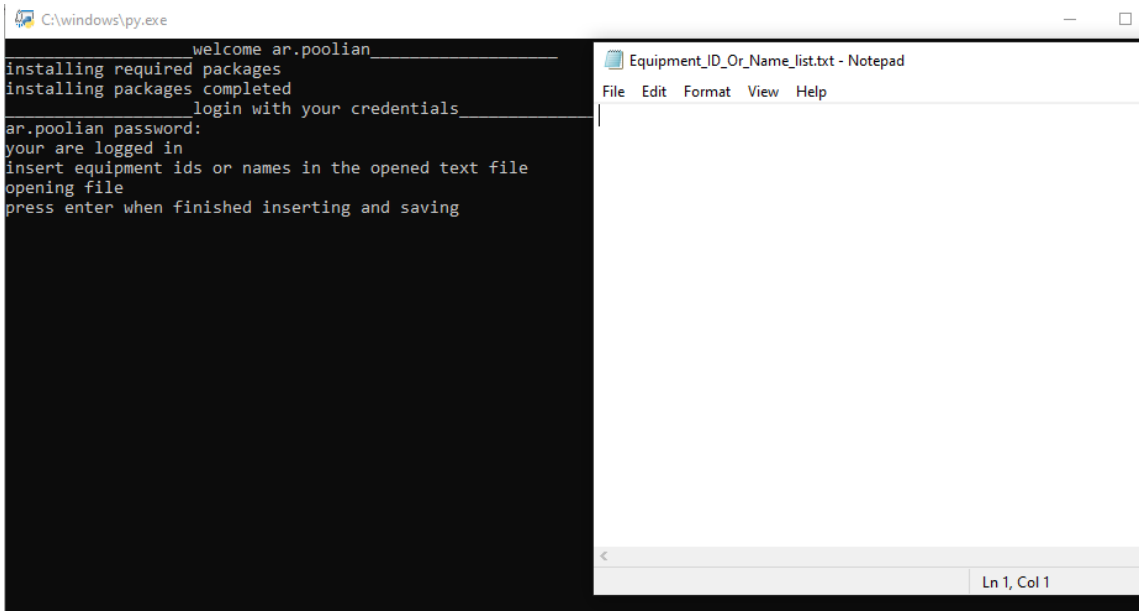


Figure 9: Prompt user to insert names or ids in the file

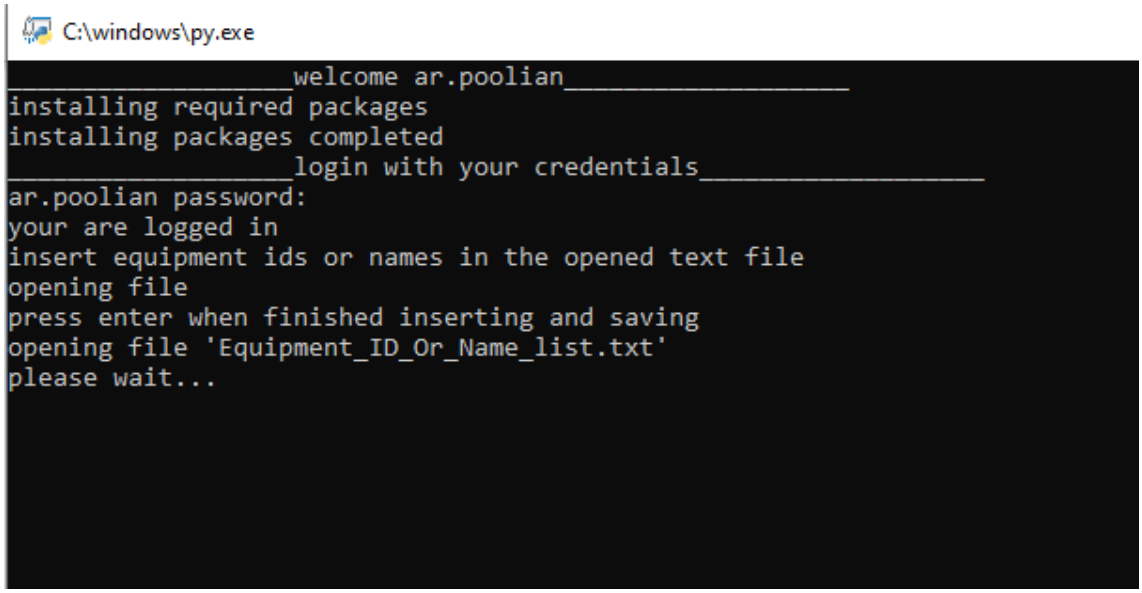


Figure 10: Querying idefix

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.

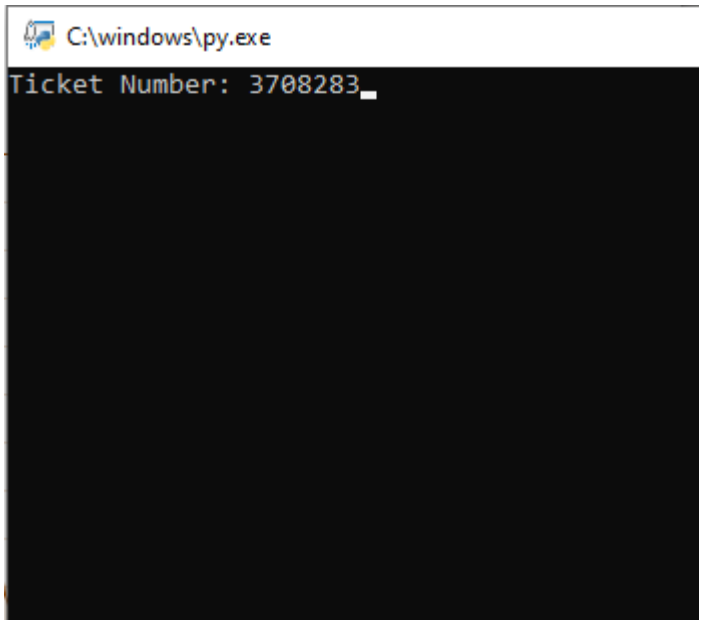


Figure 11: Request for the ticket number

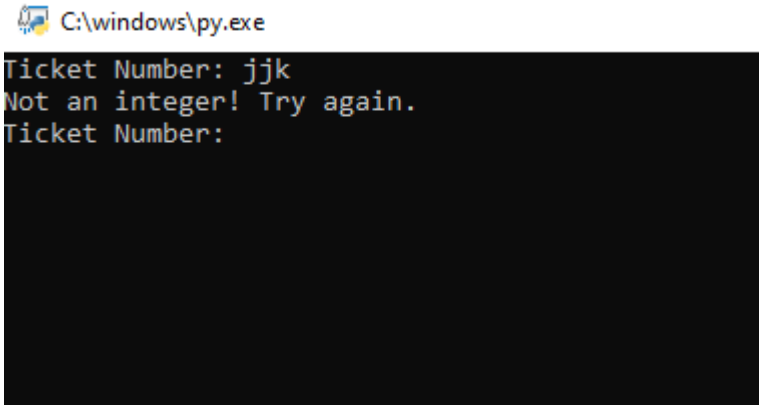
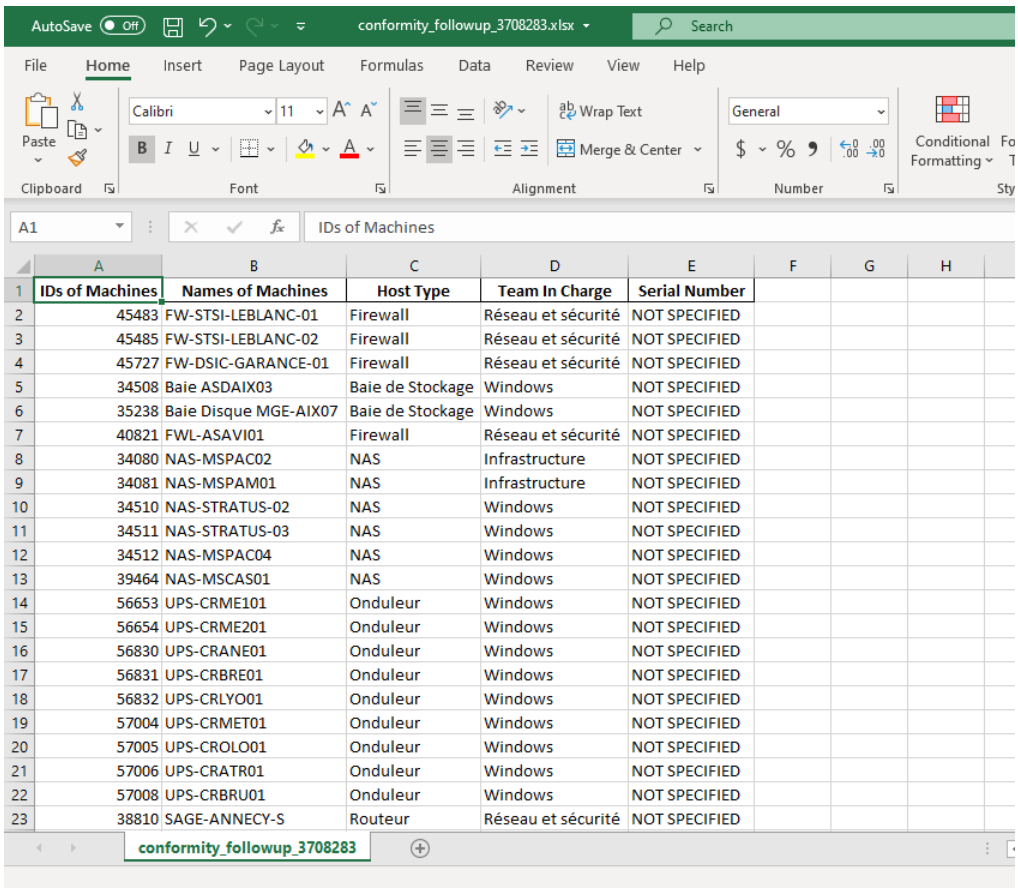


Figure 12: Not entering a number

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.



IDs of Machines	Names of Machines	Host Type	Team In Charge	Serial Number
45483	FW-STSI-LEBLANC-01	Firewall	Réseau et sécurité	NOT SPECIFIED
45485	FW-STSI-LEBLANC-02	Firewall	Réseau et sécurité	NOT SPECIFIED
45727	FW-DSIC-GARANCE-01	Firewall	Réseau et sécurité	NOT SPECIFIED
34508	Baie ASDAIX03	Baie de Stockage	Windows	NOT SPECIFIED
35238	Baie Disque MGE-AIX07	Baie de Stockage	Windows	NOT SPECIFIED
40821	FWL-ASAVI01	Firewall	Réseau et sécurité	NOT SPECIFIED
34080	NAS-MSPAC02	NAS	Infrastructure	NOT SPECIFIED
34081	NAS-MSPAM01	NAS	Infrastructure	NOT SPECIFIED
34510	NAS-STRATUS-02	NAS	Windows	NOT SPECIFIED
34511	NAS-STRATUS-03	NAS	Windows	NOT SPECIFIED
34512	NAS-MSPAC04	NAS	Windows	NOT SPECIFIED
39464	NAS-MSCAS01	NAS	Windows	NOT SPECIFIED
56653	UPS-CRME101	Onduleur	Windows	NOT SPECIFIED
56654	UPS-CRME201	Onduleur	Windows	NOT SPECIFIED
56830	UPS-CRANE01	Onduleur	Windows	NOT SPECIFIED
56831	UPS-CRBRE01	Onduleur	Windows	NOT SPECIFIED
56832	UPS-CRLYO01	Onduleur	Windows	NOT SPECIFIED
57004	UPS-CRMET01	Onduleur	Windows	NOT SPECIFIED
57005	UPS-CROLO01	Onduleur	Windows	NOT SPECIFIED
57006	UPS-CRATR01	Onduleur	Windows	NOT SPECIFIED
57008	UPS-CRBRU01	Onduleur	Windows	NOT SPECIFIED
38810	SAGE-ANNECY-S	Routeur	Réseau et sécurité	NOT SPECIFIED

Figure 13: Excel file filled with required data

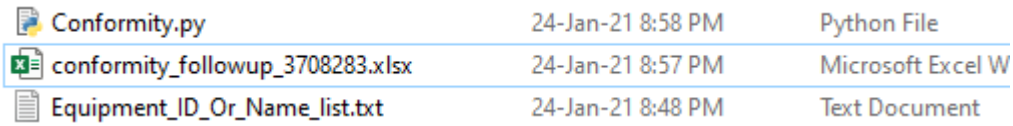


Figure 14: Created excel file

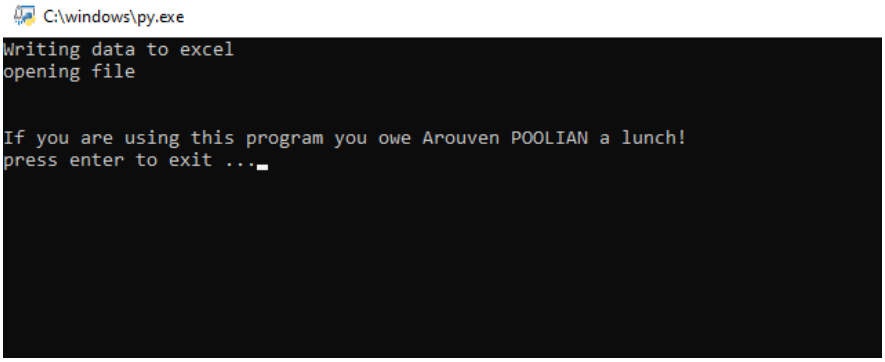


Figure 15: End of life of the script

1.1.4 Conclusion

1.1.4.1 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.

1.1.4.2 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.

1.1.4.3 Future work

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

Chapter 2: Appendix

2.1 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/Equ
ipment.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/Equip
ment.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, TeamInCharg
eList,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, i
ndex=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/Authentifica
tion.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)
```