

# Administration et Supervision des réseaux

## RT3

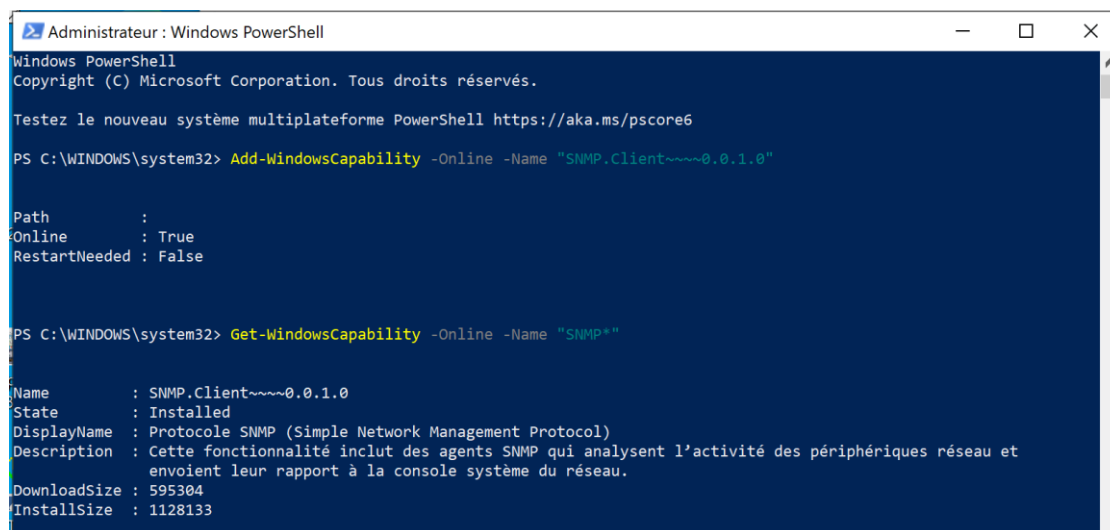
### TP : SNMP sur PC avec Python

#### Configurez SNMP pour l'hôte de PC.

Pour Windows 10 , il faut ajouter le service SNMP : (

<https://theitbros.com/snmp-service-on-windows-10/>

)



```
Administrateur : Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

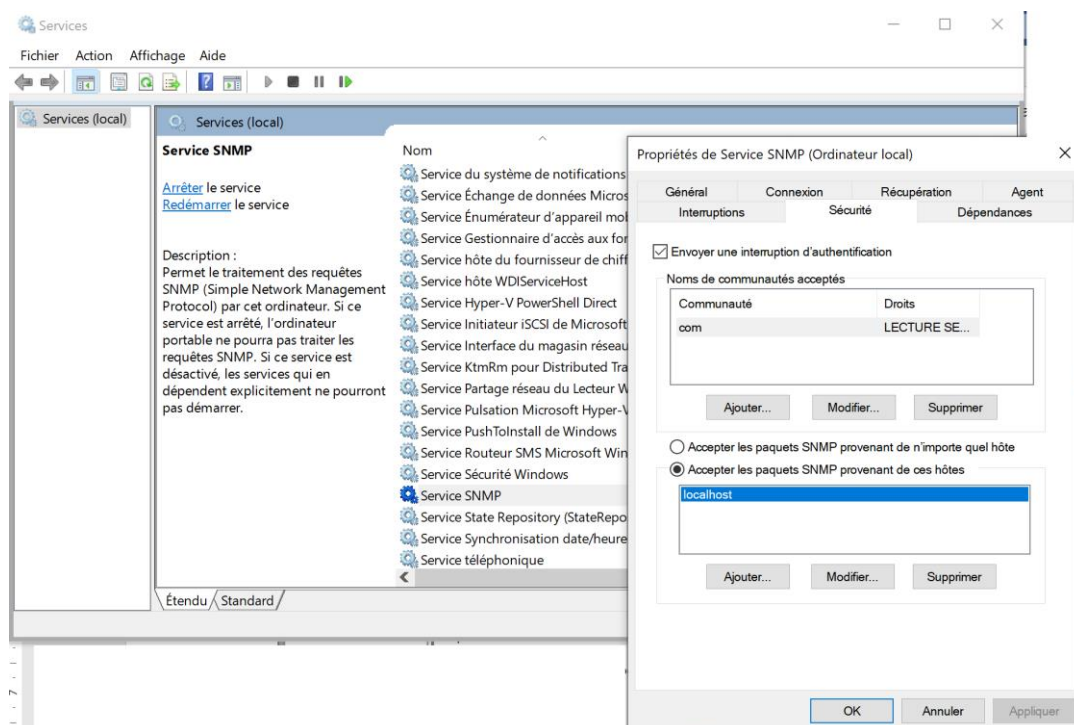
Testez le nouveau système multiplateforme PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> Add-WindowsCapability -Online -Name "SNMP.Client~~~~0.0.1.0"

Path      :
Online    : True
RestartNeeded : False

PS C:\WINDOWS\system32> Get-WindowsCapability -Online -Name "SNMP*"

Name      : SNMP.Client~~~~0.0.1.0
State     : Installed
DisplayName : Protocole SNMP (Simple Network Management Protocol)
Description : Cette fonctionnalité inclut des agents SNMP qui analysent l'activité des périphériques réseau et envoient leur rapport à la console système du réseau.
DownloadSize : 595304
InstallSize : 1128133
```



## Étape 1 : Installez un programme de gestion SNMP.

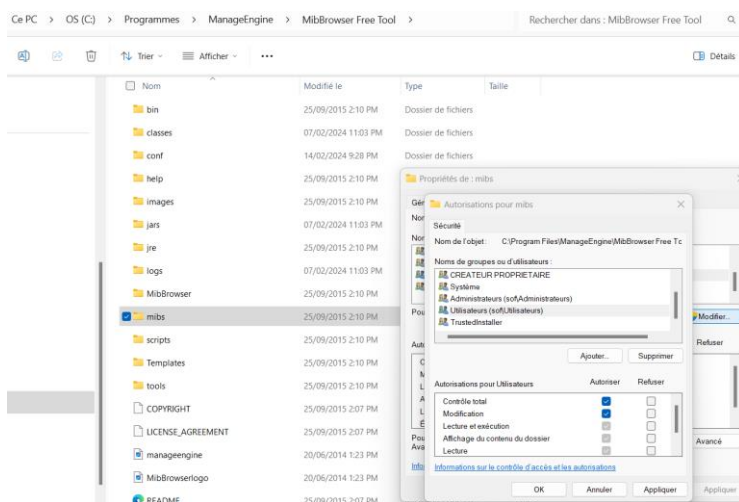
- Téléchargez et installez le logiciel **SNMP MIB Browser** de ManageEngine à partir de l'URL suivante : <https://www.manageengine.com/products/mibbrowser-free-tool/download.html>. Vous serez invité à saisir une adresse e-mail pour télécharger le logiciel.
- Exécutez le programme MibBrowser de ManageEngine.

- Si vous recevez un message d'erreur relatif à l'échec du chargement des bases de données MIB, procédez comme suit : Accédez au dossier MibBrowser Free Tool :

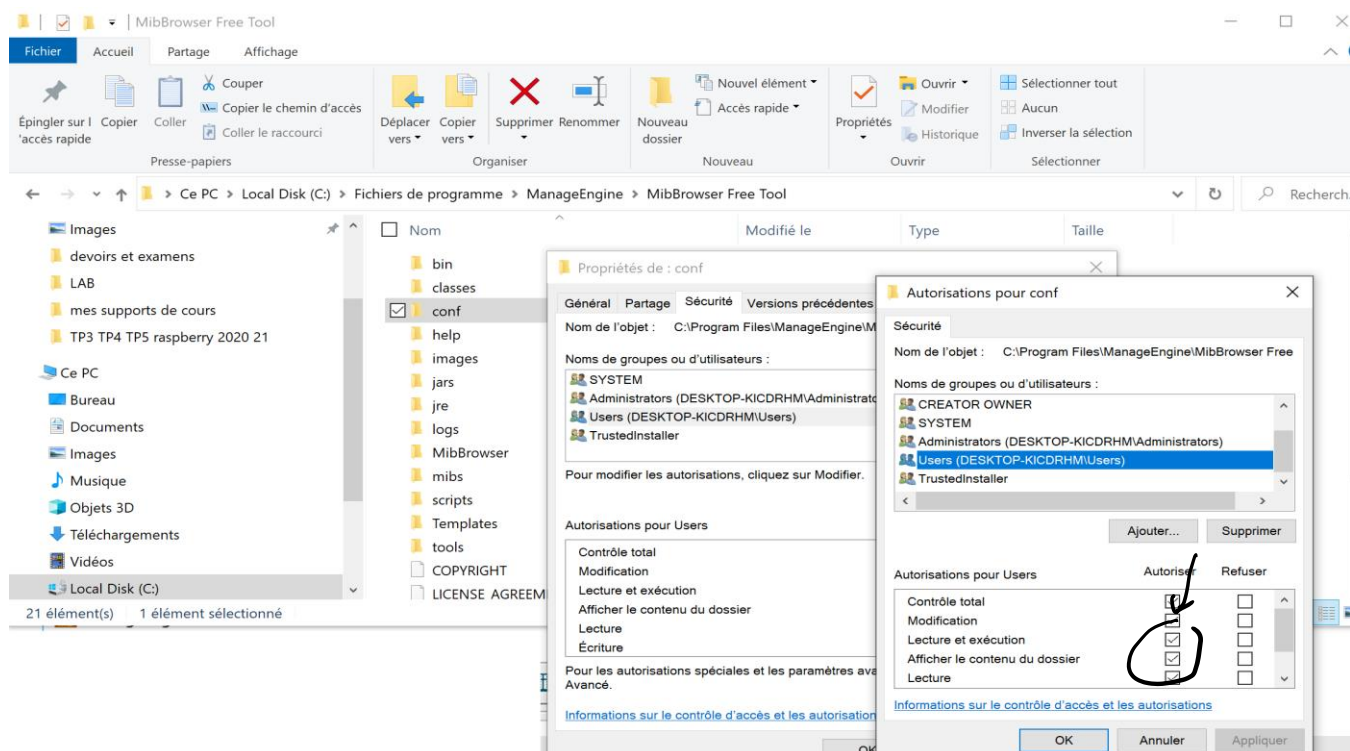
32 bits : C:\Program Files (x86)\ManageEngine\MibBrowser Free Tool

64 bits : C:\Program Files\ManageEngine\MibBrowser Free Tool

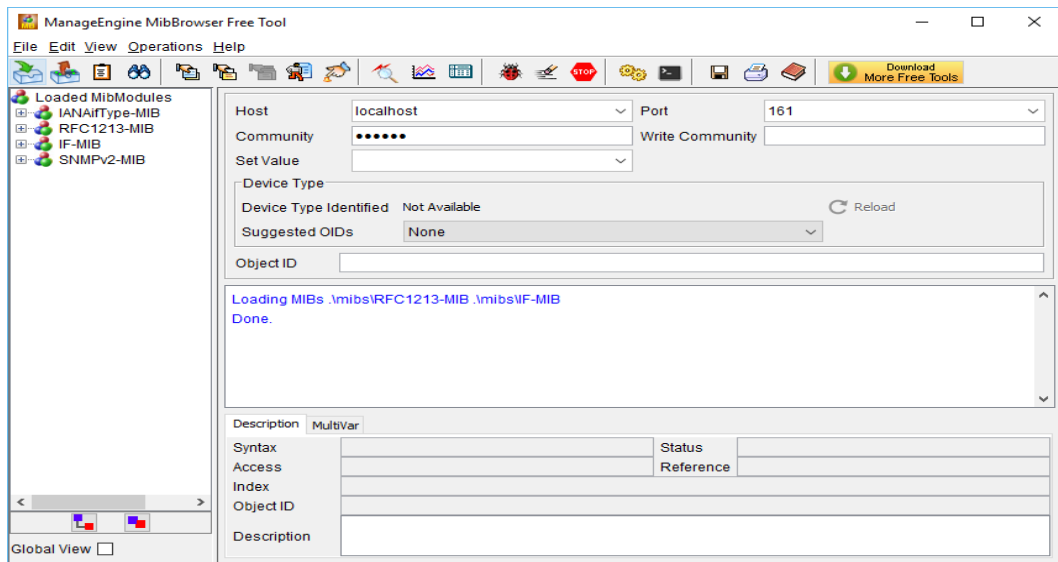
- Effectuez un clic droit sur le dossier **mibs**, Propriétés, puis sélectionnez l'onglet **Security**. Cliquez sur **Edit**. Sélectionnez **Users**. Cochez la case **Modifier** sous la colonne **Autoriser**. Cliquez sur **OK** pour modifier l'autorisation.



- Répétez l'étape précédente avec le dossier **conf**.



4) Exécutez à nouveau le programme MibBrowser de ManageEngine.



---

2<sup>ème</sup> alternative : module : netmiko

<https://github.com/ktbyers/netmiko/blob/develop/EXAMPLES.md#auto-detection-using-snmpv2c>

---

### Etape 3 : SNMP et Python ( Python 3.7.3)

Installer et tester ces codes et commandes pour être familiarisé avec package SNMP Python et ce pour récupérer les informations sur votre ordinateur local (pour lequel l'agent SNMP est activé).

Installer le package pySNMP

```
pip.exe install pysnmp
```

PySNMP dispose d'une librairie supplémentaire fournissant ces commandes: **pysnmp\_apps**, installez-la.

```
pip.exe install pysnmp_apps
```

Les commandes disponibles à présent sont les suivantes:

- snmpbulkwalk.py
- snmpget.py
- snmpset.py
- snmptranslate.py
- snmptrap.py
- snmpwalk.py

```
C:\MyProject>c:\Python35\Scripts\snmpget.py -v1 -c public demo.snmplabs.com  
sysLocation.0 sysDescr.0
```

Si cela ne fonctionne pas, essayez d'ajouter le nom de l'interpréteur python devant le nom du script:

```
C:\MyProject>c:\Python35\python.exe c:\Python35\Scripts\snmpget.py -v1 -c public demo.snmplabs.com sysLocation.0 sysDescr.0
```

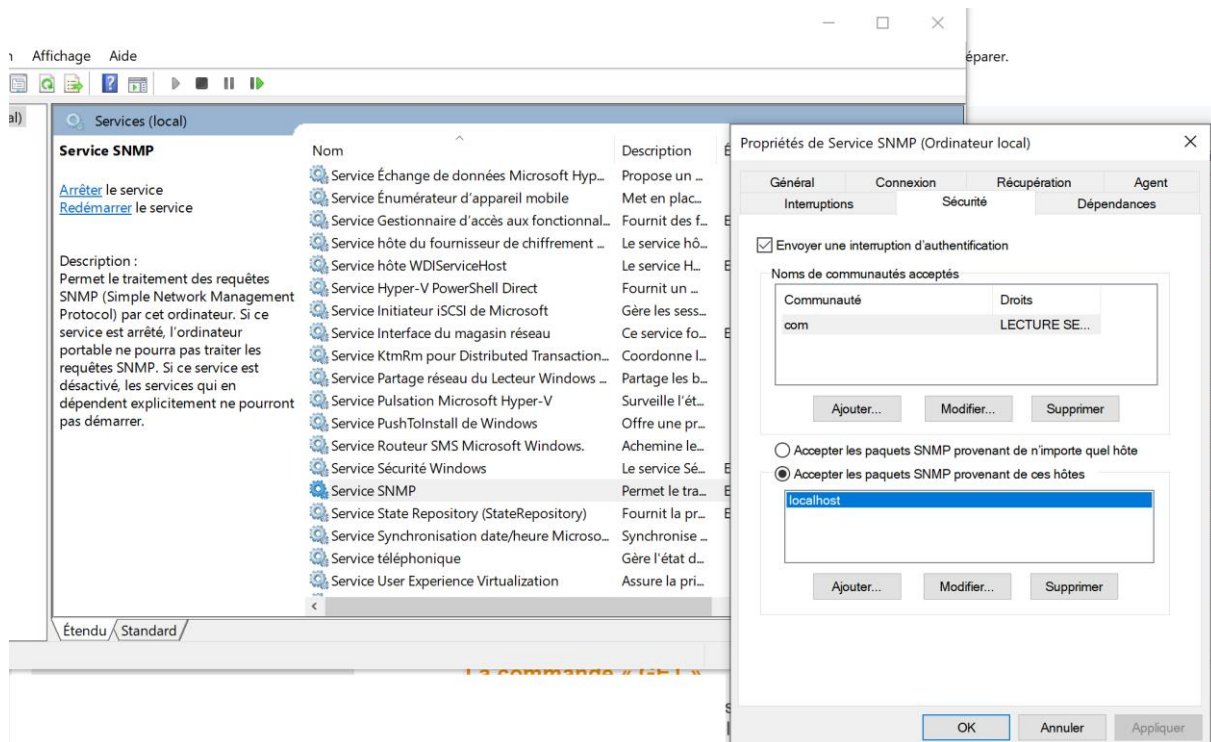
Mais bien sur, cela fonctionnera :

```
SNMPv2-MIB::sysLocation.0 = Moscow, Russia
```

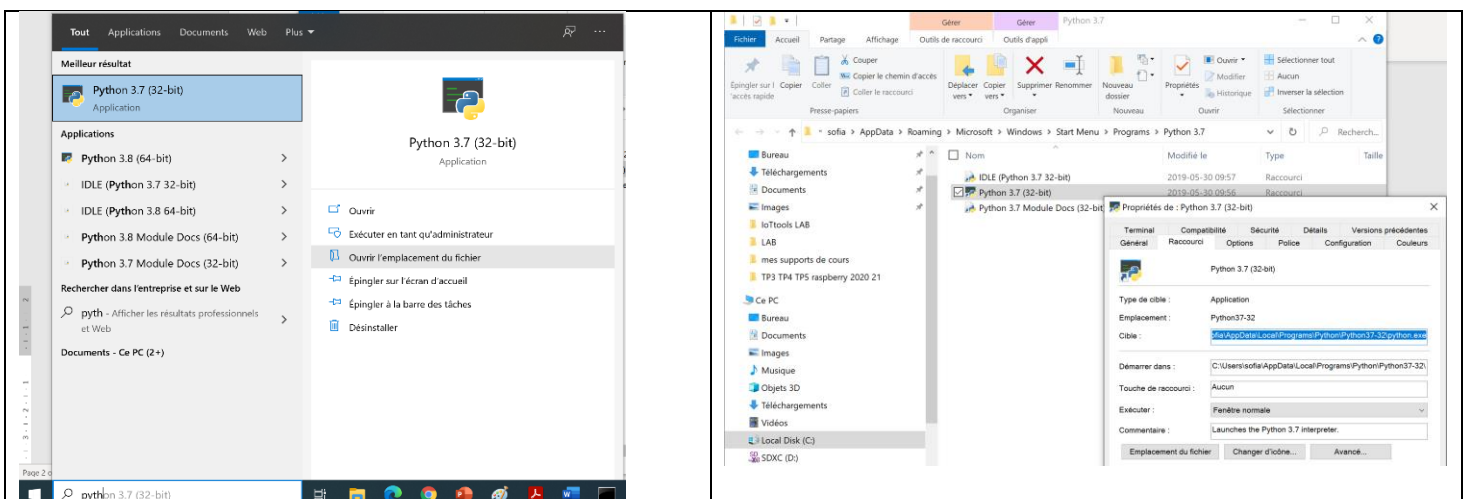
```
SNMPv2-MIB::sysDescr.0 = SunOS zeus.snmplabs.com 4.1.3_U1 1 sun4m
```

Il ne faut pas oublier d'activer le service SNMP dans votre ordinateur local.

Puis chercher



Puis chercher l'emplacement du sous répertoire : Python37-32\Scripts\ pour exécuter les commandes SNMP en ligne écris en python : snmpget.py, ...



```
Invite de commandes

C:\Users\sofia>python.exe C:\Users\sofia\AppData\Local\Programs\Python\Python37-32\Scripts\snmpget.py -v1 -c com 127.0.0.1 sysLocation.0 sysDescr.0
SNMPv2-MIB::sysLocation.0 = DisplayString:
SNMPv2-MIB::sysDescr.0 = DisplayString: Hardware: Intel64 Family 6 Model 142 Stepping 9 AT/AT COMPATIBLE - Software: Windows Version 6.3 (Build 18363 Mult
iprocessor Free)

C:\Users\sofia>
```

## La commande « GET »

Maintenant que nous avons passé en revue l'ensemble des éléments nécessaires à l'appel de commandes, nous allons ENFIN pouvoir écrire notre première commande GET. Pour envoyer un message « GET » il convient d'utiliser la fonction « getCmd » listée précédemment.

Enfin, lorsque vous exécutez cette commande vous pouvez demander plusieurs OID à la fois.

La syntaxe est la suivante :

```
pysnmp.hlapi.getCmd(snmEngine, authData, transportTarget, contextData,
*varBinds, **options)
```

Cette fonction retourne un tuple de 4 éléments :

- errorIndication : Une valeur considérée comme « True » si une erreur s'est produite dans l'engine
- errorStatus : Une valeur considérée comme « True » pour une erreur PDU
- errorIndex : L'index de la variable ayant provoqué l'erreur (commence à 0)
- varBinds : Une tuple contenant les valeurs retournées par la commande dans des instances de la classe « ObjectType »

Soit la commande suivante :

```
$ snmpget -v1 -c public demo.snmplabs.com sysLocation.0 sysDescr.0
```

Elle peut être exécutée via le code Python suivant :

```
from pysnmp.hlapi import *

data = (
    ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysLocation', 0)),
    ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysDescr', 0)),
    ObjectType(ObjectIdentity('.1.3.6.1.2.1.1.1.0')),
)

g = getCmd(SnmEngine(), CommunityData('public', mpModel=0),
            UdpTransportTarget(('demo.snmplabs.com', 161)),
            ContextData(), *data)

errorIndication, errorStatus, errorIndex, varBinds = next(g)

if errorIndication:
    print(errorIndication)
elif errorStatus:
    print('%s at %s' % (
        errorStatus.prettyPrint(),
        errorIndex and varBinds[int(errorIndex) - 1][0] or '?'
    ))
else:
    for varBind in varBinds:
```

```
print(' = '.join([x.prettyPrint() for x in varBind]))
```

- pour comprendre plus le type liste don't la syntaxe est entre accolades:  
<https://www.programiz.com/python-programming/list>

```
SNMPv2-MIB::sysLocation.0 = Moscow, Russia
SNMPv2-MIB::sysDescr.0 = SunOS zeus.snmplabs.com 4.1.3_U1 1 sun4m
SNMPv2-MIB::sysDescr.0 = SunOS zeus.snmplabs.com 4.1.3_U1 1 sun4m
```

Pour l'exécuter avec le protocole « v2c » changez simplement la valeur du paramètre « mpModel=1 » dans l'objet community.

Pour l'exécuter avec le protocole « v3 » remplacez l'instance « CommunityData » par une instance « UsmUserData ».

Exemple d'utilisateur (usr-sha-des/SHA/authkey1/DES/privkey1 ) :

```
UsmUserData("usr-sha-des"
            , authProtocol=usmHMACSHAAuthProtocol
            , authKey="authkey1"
            , privProtocol=usmDESPrivProtocol
            , privKey="privkey1" )
```

## La commande « GetNEXT »

La commande « GetNEXT » s'exécute exactement comme une commande « get », avec les mêmes paramètres.

Elle permet de récupérer l'élément suivant l'OID passé en argument.

```
from pysnmp.hlapi import *
g = nextCmd(SnmpEngine()
            , CommunityData('public', mpModel=1)
            , UdpTransportTarget(('demo.snmplabs.com', 161))
            , ContextData()
            , ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysObjectID', 0)))

errorIndication, errorStatus, errorIndex, varBinds = next(g)

if errorIndication:
    print(errorIndication)
elif errorStatus:
    print('%s at %s' % (errorStatus.prettyPrint()
                        , errorIndex and varBinds[int(errorIndex) - 1][0] or '?')
          )
else:
    for varBind in varBinds:
        print(' = '.join([x.prettyPrint() for x in varBind]))
```

```
SNMPv2-MIB::sysUpTime.0 = 280102869
```



Comme vous l'avez peut-être remarqué, la commande retourne un « générateur ». Vous pouvez donc la rappeler avec la fonction « next » pour avoir l'élément suivant. Ce qui vous permet de descendre toute une MIB en modifiant quelque peu le code ci-dessus, une fois la variable « g » instanciée.

```
from pysnmp.hlapi import *

g = nextCmd(SnmpEngine()
            , CommunityData('public', mpModel=1)
            , UdpTransportTarget(('demo.snmplabs.com', 161))
            , ContextData()
            , ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysObjectID', 0)))

for errorIndication, errorStatus, errorIndex, varBinds in g:

    if errorIndication:
        print(errorIndication)
    elif errorStatus:
        print('%s at %s' % (errorStatus.prettyPrint(),
                            errorIndex and varBinds[int(errorIndex) - 1][0] or '?')
              )
    else:
        for varBind in varBinds:
            print(' = '.join([x.prettyPrint() for x in varBind]))
```

```
SNMPv2-MIB::sysUpTime.0 = 280140241
SNMPv2-MIB::sysContact.0 = SNMP Laboratories, info@snmplabs.com
SNMPv2-MIB::sysName.0 = zeus.snmplabs.com
SNMPv2-MIB::sysLocation.0 = Moscow, Russia
SNMPv2-MIB::sysServices.0 = 72
SNMPv2-MIB::sysORLastChange.0 = 280140343
SNMPv2-MIB::sysORID.1 = PYSNMP-MIB::pysnmpObjects.1
SNMPv2-MIB::sysORDescr.1 = new comment
SNMPv2-MIB::sysORUpTime.1 = 123
```

## La commande « SET »

La commande «SET» s'exécute exactement comme une commande « get », avec les mêmes paramètres. Excepté que l'ObjectType contient un second argument : la nouvelle valeur.

```
from pysnmp.hlapi import *

def show_item( ):

    g = getCmd(SnmpEngine()
              , CommunityData('public', mpModel=1)
              , UdpTransportTarget(('demo.snmplabs.com', 161))
              , ContextData()
```

```

        , ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysORDescr', 1)))

errorIndication, errorStatus, errorIndex, varBinds = next(g)

for varBind in varBinds:
    print(' = '.join([x.prettyPrint() for x in varBind]))

# Show initial value
show_item()

# Setting new value
g = setCmd(SnmpEngine()
            , UsmUserData("usr-sha-des"
                           , authProtocol=usmHMACSHAAuthProtocol
                           , authKey="authkey1"
                           , privProtocol=usmDESPrivProtocol
                           , privKey="privkey1" )
            , UdpTransportTarget(('demo.snmplabs.com', 161))
            , ContextData()
            , ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysORDescr', 1), 'Hello
from Lannion using Kalray processor on Linux'))

errorIndication, errorStatus, errorIndex, varBinds = next(g)

print(errorIndication, varBinds)

show_item()

```

```

SNMPv2-MIB::sysORDescr.1 = Here is my new note
SNMPv2-MIB::sysORDescr.1 = Hello from Lannion using Kalray processor on Linux

```

Pour le SNMP version 3, voici un lien :

<https://man.archlinux.org/man/community/python-pysnmp/pysnmp.1.en>

## Etape 4 : programmation de SMNP avec Python

---

### I. Création d'application Python utilisant SMNP pour affichage d'information sur interface

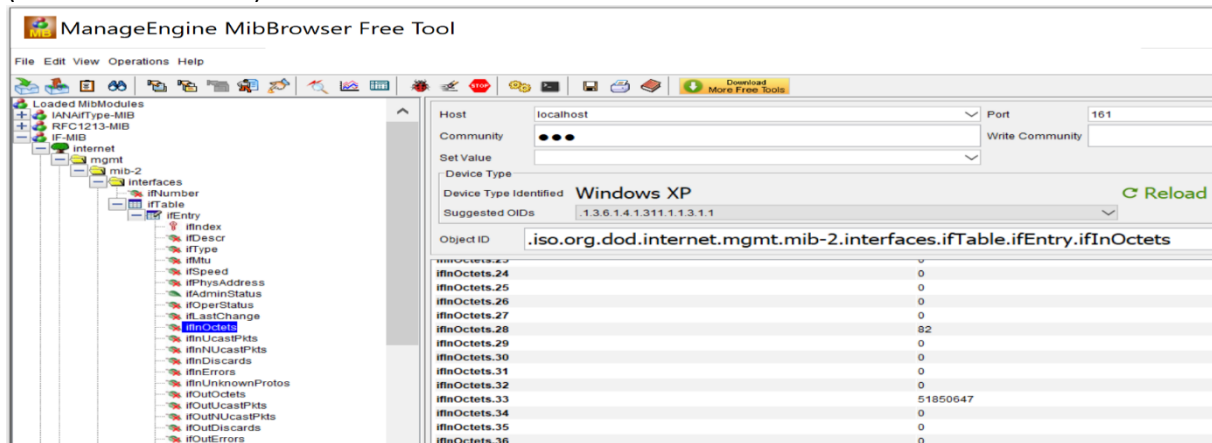
On voudrait maintenant créer une application Python qui affiche la description d'une interface active, l'adresse MAC et le volume (bits) de données reçus sur cette interface.

#### 1. Choix d'interface active

Affiche les interfaces actives où il y a des octets reçus (ifInOctets). Ici c'est l'interface de numéro 33.



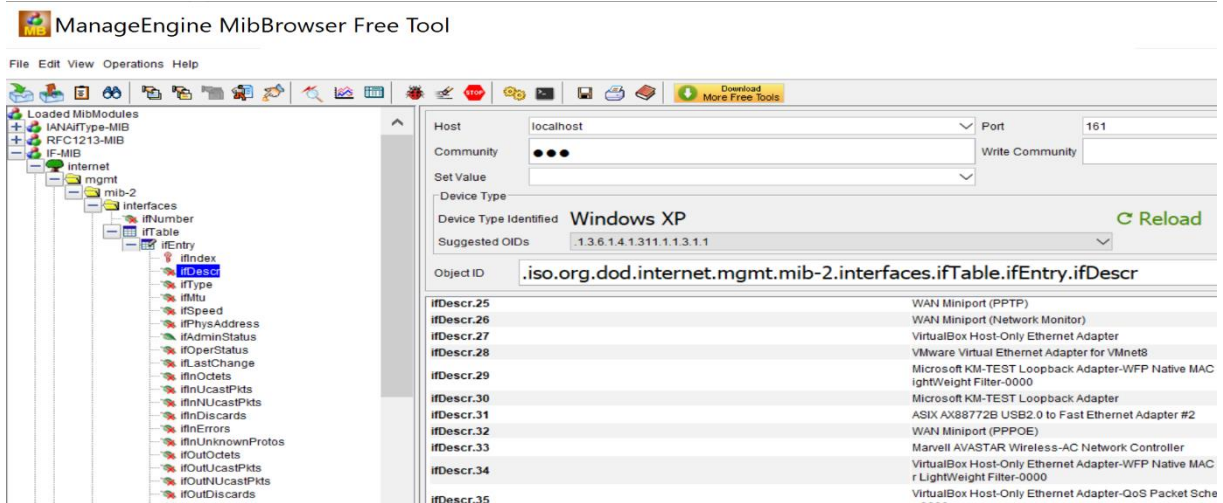
(.iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifInOctets)  
(.1.3.6.1.2.1.2.2.1.10)



Dans le TP précédent il y a comment installer ManageEngine Mibbrowser dont voici le lien pour télécharger :  
<https://www.manageengine.com/products/mibbrowser-free-tool/download.html>

Affichage du descripteur de l'interface (ifDescr) de numéro 33. Dans notre cas c'est la carte 'Marvell AVASTAR Wireless-AC Network Controller'

(.iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifDescr)  
(.1.3.6.1.2.1.2.2.1.2)



Affichage des adresses IP associée avec l'index en particulier l'index 33 pour l'interface Wifi (Object ID: .1.3.6.1.2.1.4.20.1.2)

(.iso.org.dod.internet.mgmt.mib-2.ip.ipAddrTable.ipAddrEntry.ipAdEntIfIndex)

Object ID	Value
ipAdEntIfIndex.192.168.1.1	28
ipAdEntIfIndex.192.168.217.1	11
Sent GET request to localhost: 161	
ipAdEntIfIndex.127.0.0.1	1
ipAdEntIfIndex.192.168.1.11	33
ipAdEntIfIndex.192.168.1.100	30
ipAdEntIfIndex.192.168.56.1	27
ipAdEntIfIndex.192.168.58.33	10
ipAdEntIfIndex.192.168.131.1	28
ipAdEntIfIndex.192.168.217.1	11

ipconfig /all

```

Invite de commandes
Serveurs DNS. . . . . : fec0:0:0:ffff::1%1
                  . . . . . : fec0:0:0:ffff::2%1
                  . . . . . : fec0:0:0:ffff::3%1
Serveur WINS principal . . . . . : 192.168.131.2
NetBIOS sur Tcpip. . . . . : Activé

Carte réseau sans fil WiFi :
Suffixe DNS propre à la connexion. . . . . :
Description. . . . . : Marvell AVASTAR Wireless-AC Network
Adresse physique. . . . . : C4-9D-ED-0D-71-B5
DHCP activé. . . . . : Oui
Configuration automatique activée. . . . . : Oui
Adresse IPv6 de liaison locale. . . . . : fe80::3932:4b88:de5e:dc59%33(préfé
Adresse IPv4. . . . . : 192.168.1.11(préfé
Masque de sous-réseau. . . . . : 255.255.255.0

```

2. Tester le programme suivant sur la lecture

SNMP :

```

AffichageTemp.py x snmpexp1.py x
1 from pysnmp.hlapi import *
2
3 data = (
4     ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysLocation', 0)),
5     ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysDescr', 0)),
6     ObjectType(ObjectIdentity('.1.3.6.1.2.1.2.2.1.10.33')),
7     ObjectType(ObjectIdentity('.1.3.6.1.2.1.2.2.1.2.33'))
8 )
9
10 g = getCmd(SnmpEngine(), CommunityData('com', mpModel=0)
11            , UdpTransportTarget(('127.0.0.1', 161))
12            , ContextData(), *data)
13
14 errorIndication, errorStatus, errorIndex, varBinds = next(g)
15
16 if errorIndication:
17     print(errorIndication)
18 elif errorStatus:
19     print('error status' )
20 else:
21     for varBind in varBinds:
22         l= [str(x) for x in varBind]
23         print (l[0], ' = ', l[1])
24
25
Shell x
>>> %Run snmpexp1.py
1.3.6.1.2.1.1.6.0

1.3.6.1.2.1.1.1.0
Hardware: Intel64 Family 6 Model 142 Stepping 9 AT/AT COMPATIBLE - Software: Windows Version 6.3 (Build 18363 Mult
iprocessor Free)
1.3.6.1.2.1.2.2.1.10.33
7025238
1.3.6.1.2.1.2.2.1.2.33
Marvell AVASTAR Wireless-AC Network Controller

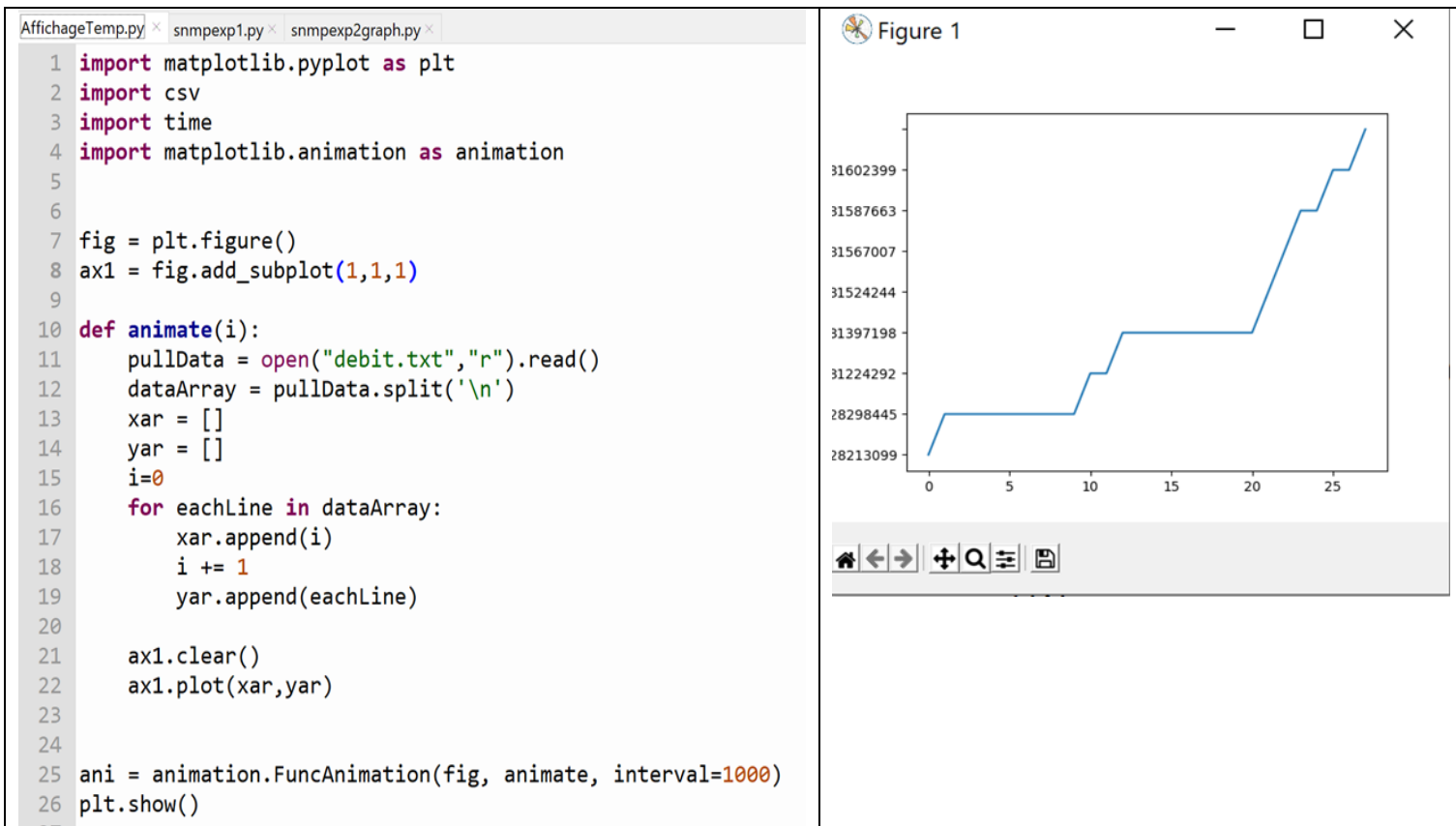
```

3. Soit le code suivant qui permet d'afficher le volume de données cerclées pour une interface et qui le sauvegarde dans un fichier texte ligne par ligne :

```
AffichageTemp.py x snmpexp1.py x snmpexp2graph.py x
1 from pysnmp.hlapi import *
2 import time
3
4
5 data = (
6     ObjectType(ObjectIdentity('.1.3.6.1.2.1.2.2.1.10.33'))
7 )
8
9
10
11 while True:
12     g = getCmd(SnmpEngine(), CommunityData('com', mpModel=0)
13               , UdpTransportTarget(('127.0.0.1', 161))
14               , ContextData() , data)
15     errorIndication, errorStatus, errorIndex, varBinds = next(g)
16
17     if errorIndication:
18         print(errorIndication)
19     elif errorStatus:
20         print('error status' )
21     else:
22         print (varBinds[0][1])
23         with open("debit.txt", "a+") as f:
24             f.write(str(varBinds[0][1]))
25             f.write("\n")
26         time.sleep(5)
27
28
Shell x
31602399
31602399
```

```
C:\ Invite de commandes
C:\Users\sofia>cd C:\cours\cours IoT\TP\tp raspberry\IoTtools LAB\TP3 TP4 TP5 raspberry 2020 21\
C:\cours\cours IoT\TP\tp raspberry\IoTtools LAB\TP3 TP4 TP5 raspberry 2020 21>python snmpexp2graph.py
32828146
32828146
32828146
32828146
32829496
32829496
```

4. Soit aussi le code python qui permet d'afficher graphiquement les données du fichier texte :



5. On veut changer le code python pour qu'il enregistre dans le fichier à la place du volume de données, le taux de transfert par seconde. Cela en calculant le volume de données ajouté pour une seconde.