Exercices : Accéder à une boite mail avec POP et IMAP

Table des matières

Introduction

Important :

Utilisez le code correspondant à votre version python (voir le nom du fichier entre crochets pour déterminer quel programme vous devez utiliser);

Les code des programmes Python 2 et Python 3 sont téléchargeable depuis le répertoire Google Drive suivant :

https://drive.google.com/open?id=1TO0TA1HqgrLZ8HYs-ZxqxKOw__m1qZBi

Accéder à une boite mail avec POP

Voici une boite mail pour tester des exemples POP et IMAP sur le serveur mail Gmail :

- *Nom utilisateur :* MasterInfo.tlemcen.MAIL@gmail.com
- *Mot de passe :* CodingMonk?L7*5

Important: Ne testez pas l'exemple avec votre boite mail avant de terminer les exercices.

POP (Post Office Protocol) est un protocole très simple permetant de connecter à une boite mail et télécharger les mails situés dedans. Voici les commandes les plus importantes :

1. *USER, PASS* fonctionnent comme dans FTP pour transmettre le login (boite mail) et mot de passe :
2. *STAT :* renvoie le nombre de mails dans la boite et la taille totale occupée par celle-ci ;
3. *LIST :* renvoie une liste contenant le numéro de chaque mail accompagné avec ça taille ;
4. *RETR :* récupère le mail désigné par le numéro en paramètre sous format RFC 822 (voir le cours SMTP).
5. *TOP :* renvoie les entête du message désigné et les N premières lignes du message ;
6. *UIDL :* retourne la liste des identificateurs uniques des messages dans la boite ;
7. *QUIT* : ferme la sessions avec le serveur POP

Exécutez le code client POP écrit en python 2 *[TP POP.py]* (voir le code python 3 à la fin de l'exercice si vous utilisez cette version [*TP POP 3.py*]):

```python
# code python 2 client POP
import socket
import ssl

def RecevoirLaReponseServeur(ConnexionSecurise, Delimiteur_De_Fin_De_Reponse):
    Reponse_recu= ConnexionSecurise.recv(1024)
    if (("+OK" in Reponse_recu))and Delimiteur_De_Fin_De_Reponse=="":
        return Reponse_recu
    if ("-ERR" in Reponse_recu):
        return Reponse_recu
    while True:
        # lire la reponse jusqu'a atteindre la sequence de caratere Delimiteur_De_Fin_De_Reponse
        if(Delimiteur_De_Fin_De_Reponse in Reponse_recu):
            break
        Reponse_recu+=ConnexionSecurise.recv(1024)
    return Reponse_recu


NomDomaineDuServeurPOP="pop.gmail.com"
ConnexionAuServeur=socket.socket()
ConnexionAuServeur.connect((NomDomaineDuServeurPOP,995))
# creation d'une connexion securise
ConnexionSecurise=ssl.create_default_context().wrap_socket(ConnexionAuServeur, server_hostname=NomDomaineDuServeurPOP)
Commande=""
# liste de commande pour les quels la reponse setermine par "\r\n.\r\n" (un point au debut de la ligne)
ListeCommande=["list","top","retr","uidl"]
while True:
    Delimiteur=""
    for command_terminer_par_unpoint in ListeCommande:
        # si la commande issue par le client appartient a ListeCommande
        # la reponse de cette commande doit se terminee par "\r\n.\r\n"
        if(Commande.strip().lower().startswith(command_terminer_par_unpoint)):
            Delimiteur="\r\n.\r\n"
        # les autres commande comme QUIT, PASS et  USER auront une seule ligne de reponse seulement(+OK ou -ERR )
    print RecevoirLaReponseServeur(ConnexionSecurise,Delimiteur)
    if(Commande.strip().lower().startswith("quit")):
        break
    Commande=raw_input()
    ConnexionSecurise.send(Commande+"\r\n")
ConnexionSecurise.close()

```

Question

Testez les commandes suivantes et faites des captures d'écran des résultats:

- USER MasterInfo.tlemcen.MAIL@gmail.com
- PASS CodingMonk?L7*5
- STAT
- UIDL
- LIST
- RETR 3
- QUIT

```python
# code python 3 client POP
import socket
import ssl

def RecevoirLaReponseServeur(ConnexionSecurise, Delimiteur_De_Fin_De_Reponse):
    Reponse_recu= ConnexionSecurise.recv(1024).decode("unicode_escape")
    if (("+OK" in Reponse_recu))and Delimiteur_De_Fin_De_Reponse=="":
        return Reponse_recu
    if ("-ERR" in Reponse_recu):
        return Reponse_recu
    while True:
        # lire la reponse jusqu'a atteindre la sequence de caratere Delimiteur_De_Fin_De_Reponse
        if(Delimiteur_De_Fin_De_Reponse in Reponse_recu):
            break
        Reponse_recu+=ConnexionSecurise.recv(1024).decode("unicode_escape")
    return Reponse_recu


NomDomaineDuServeurPOP="pop.gmail.com"
ConnexionAuServeur=socket.socket()
ConnexionAuServeur.connect((NomDomaineDuServeurPOP,995))
# creation d'une connexion securise
ConnexionSecurise=ssl.create_default_context().wrap_socket(ConnexionAuServeur, server_hostname=NomDomaineDuServeurPOP)
Commande=""
# liste de commande pour les quels la reponse setermine par "\r\n.\r\n" (un point au debut de la ligne)
ListeCommande=["list","top","retr","uidl"]
while True:
    Delimiteur=""
    for command_terminer_par_unpoint in ListeCommande:
        # si la commande issue par le client appartient a ListeCommande
        # la reponse de cette commande doit se terminee par "\r\n.\r\n"
        if(Commande.strip().lower().startswith(command_terminer_par_unpoint)):
            Delimiteur="\r\n.\r\n"
        # les autres commande comme QUIT, PASS et  USER auront une seule ligne de reponse seulement(+OK ou -ERR )
    print(RecevoirLaReponseServeur(ConnexionSecurise,Delimiteur))
    if(Commande.strip().lower().startswith("quit")):
        break
    Commande=input()
    ConnexionSecurise.send((Commande+"\r\n").encode())
ConnexionSecurise.close()

```

Client IMAP

II

IMAP (Internet Message Access Protocol) a le meme objectif que POP : permettre l'accès aux boites mails. Mais ce protocole offre, des fonctionnalités plus avancées :

- Création de catégories de mail ;
- Classification des mails par des flags [mail non lu, mail déjà lu, mail répondue] ;
- Recherche de mails par informations relatives (date réception, mail expéditeur, texte contenu dans le mail, etc)

Les commandes les plus importantes de IMAP :

- *CAPABILITY*: pour obtenir les fonctionnalités suppoirtées par le serveur
- *LOGIN* : pour connecter à un compte
- *LIST* : permet d'avoir les listes des dossiers (Inbox, Spam, Draft, etc) dans la boite
- *SELECT* : permet de selectionner un dossier
- *FETCH* : permet de recupérer le contenu entier ou partiel d'un mail
- *STORE* : permet de changer les flags d'un mail (“\Seen”, “\Answered”, “\Flagged”)
- *SEARCH* : permet d'avoir la liste des mails qui correspondent à une contrainte donnée
- *CLOSE* : permet de clôturer le dossier courant
- *Logout* : permet de se déconnecter

Un changement important par rapport à POP c'est que les commandes sont préfixés par un tag qui sert à faire correspondre la commande avec la réponse. Par exemple :

1. La commande `Capability` est préfixée par *Prefixe_Commande_TAG001* pour être envoyée au serveur IMAP de gmail (*Ligne 3*). La fin de la réponse du serveur sera préfixée par la même chaîne de caractères (*Ligne 6*).
2. La commande `Logout` est préfixée par *Prefixe_Commande_TAG002* pour être envoyée au serveur IMAP de gmail (*Ligne 8*). La fin de la réponse du serveur sera préfixée par la même chaîne de caractères (*Ligne 11*).

```
* OK Gimap ready for requests from 41.100.32.83 j4mb1024322636wrw

Prefixe_Commande_TAG001 Capability
* CAPABILITY IMAP4rev1 UNSELECT IDLE NAMESPACE QUOTA ID XLIST CHILDREN X-GM-EXT-1 XYZZY SASL-IR AUTH=XOAUTH2 AUTH=PLAIN AUTH=PLAIN-CLIENTTOKEN AUTH=OAUTHBEARER AUTH=XOAUTH

Prefixe_Commande_TAG001 OK Thats all she wrote! j4mb1024322636wrw

Prefixe_Commande_TAG002 LOGOUT
* BYE Logout Requested j4mb1024322636wrw

Prefixe_Commande_TAG002 OK Quoth the raven, nevermore... j4mb1024322636wrw
```

Les programmes client suivants assurent la génération des tags automatiquement, donc il suffit de saisir les commandes seulement. Exécutez le code python 2 *[TP IMAP.py]* client IMAP suivant (voir le code python 3 à la fin de l'exercice si vous utilisez cette version [*TP IMAP 3.py*]):

```python
# code python 2 client IMAP
import socket
import ssl
import time
def RecevoirLaReponseServeur(ConnexionSecurise, Tag_Commande):
    Reponse_recu=ConnexionSecurise.recv(1024)
    Delimiteur_De_Fin_De_Reponse=Tag_Commande+" "
    while True:
        # lire la reponse jusqu'a atteindre la sequence de caratere Delimiteur_De_Fin_De_Reponse (TAG de la commande)
        if(Delimiteur_De_Fin_De_Reponse in Reponse_recu):
            break
        Reponse_recu+=ConnexionSecurise.recv(1024)
    return Reponse_recu

NomDomaineDuServeurIMAP="imap.gmail.com"
ConnexionAuServeur=socket.socket()
ConnexionAuServeur.connect((NomDomaineDuServeurIMAP,993))
# creation d'une connexion securise
ConnexionSecurise=ssl.create_default_context().wrap_socket(ConnexionAuServeur, server_hostname=NomDomaineDuServeurIMAP)
Commande=""
# recevoir le message de connexion
print ConnexionSecurise.recv(2048)
TAG_Commande=""
while True:
    # lire la commande
    Commande=raw_input()
    # cree un tag apartir de l'horloge systeme (time.time())
    TAG_Commande_Horloge=str(int(time.time()))
    # envoie de la commande avec le TAG horloge
    ConnexionSecurise.send(TAG_Commande_Horloge+" "+Commande+"\r\n")
    print RecevoirLaReponseServeur(ConnexionSecurise,TAG_Commande_Horloge)
    # si la commande tapee par le client == LOGOUT, sortir de la boucle de lecture de commandes
    if(Commande.strip().lower().startswith("logout")):
        break
ConnexionSecurise.close()

```

Testez les commandes suivantes et enregistrez les captures d'écran :

(Bien lire les réponses serveur pour comprendre le traitement)

- `CAPABILITY`
- `LOGIN MasterInfo.tlemcen.MAIL CodingMonk?L7*5`
- `LIST "" "*"`
- `SELECT INBOX`
- `FETCH 2 FLAGS`
- `FETCH 2 BODY.PEEK[TEXT]`
- `STORE 2 +FLAGS (Answered)`
- `FETCH 2 FLAGS`
- `FETCH 2 BODY.PEEK[HEADER.FIELDS (To From Date Content-Type)]`
- `UID FETCH 4 BODY.PEEK[HEADER.FIELDS (To From Date Content-Type)]`
- `SEARCH SINCE 22-MAR-2018`
- `SEARCH BEFORE 31-DEC-2018`
- `SEARCH FROM ilyas9111@yahoo.fr`
- `SEARCH ALL`
- `UID SEARCH ALL`
- `FETCH 2 FLAGS`
- `CLOSE`

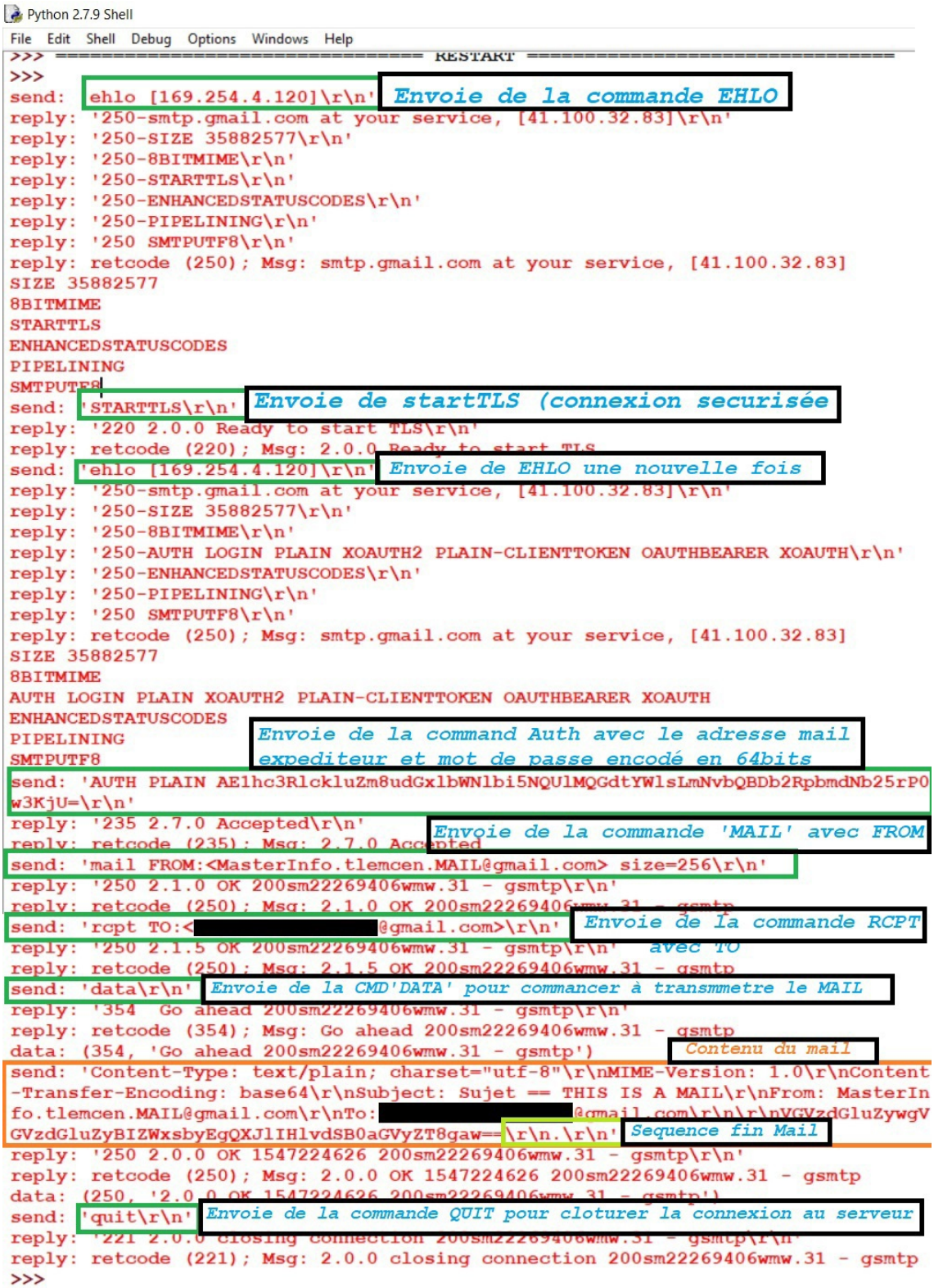
```python
#code python 3 client IMAP
import socket
import ssl
import time
def RecevoirLaReponseServeur(ConnexionSecurise, Tag_Commande):
    Reponse_recu=ConnexionSecurise.recv(1024).decode("unicode_escape")
    Delimiteur_De_Fin_De_Reponse=Tag_Commande+" "
    while True:
        # lire la reponse jusqu'a atteindre la sequence de caratere Delimiteur_De_Fin_De_Reponse (TAG de la commande)
        if(Delimiteur_De_Fin_De_Reponse in Reponse_recu):
            break
        Reponse_recu+=ConnexionSecurise.recv(1024).decode("unicode_escape")
    return Reponse_recu

NomDomaineDuServeurIMAP="imap.gmail.com"
ConnexionAuServeur=socket.socket()
ConnexionAuServeur.connect((NomDomaineDuServeurIMAP,993))
# creation d'une connexion securise
ConnexionSecurise=ssl.create_default_context().wrap_socket(ConnexionAuServeur, server_hostname=NomDomaineDuServeurIMAP)
Commande=""
# recevoir le message de connexion
print( ConnexionSecurise.recv(2048).decode("unicode_escape"))
TAG_Commande=""
while True:
    # lire la commande
    Commande=input()
    # cree un tag apartir de l'horloge systeme (time.time())
    TAG_Commande_Horloge=str(int(time.time()))
    # envoie de la commande avec le TAG horloge
    ConnexionSecurise.send((TAG_Commande_Horloge+" "+Commande+"\r\n").encode())
    print( RecevoirLaReponseServeur(ConnexionSecurise,TAG_Commande_Horloge))
    # si la commande tapee par le client == LOGOUT, sortir de la boucle de lecture de commandes
    if(Commande.strip().lower().startswith("logout")):
        break
ConnexionSecurise.close()

```

Envoie mail par SMTP

Exécutez le programme suivant pour voir les échanges des commandes / réponses avec le serveur SMTP nécessaires afin de transférer un mail [*TP1 SendMail_SMTP.py*] :

```
# le meme programme fonctionne pour python 2 et 3
import smtplib
from email.mime.text import MIMEText
from email.header    import Header
# nom domaine du serveur SMTP pour le quel ont souhaite transmettre un mail
smtp_host = 'smtp.gmail.com'

# donnees d'authentification du transmetteur (adresse mail et mot de passe)
AdresseMailExpediteur="MasterInfo.tlemcen.MAIL@gmail.com"
MotDePasse="CodingMonk?L7*5"

# adresse mail recepteur
AdresseMailRecepteur="VOTRE.ADRESSE.EMAIL.SVP@gmail.com"# ecrivez votre adresse mail ici

# Mail formate avec le format RFC822 ( voir le cours SMTP)
# contenu text
Mail_FormatRFC822 = MIMEText('Testing, Testing Hello! Are you there? k', 'plain', 'utf-8')
# entete Subject:
Mail_FormatRFC822['Subject'] ='Sujet == THIS IS A MAIL'
# entete From:
Mail_FormatRFC822['From'] = AdresseMailExpediteur
# entete To:
Mail_FormatRFC822['To'] = AdresseMailRecepteur

# etablissement de la connexion avec le serveur SMTP smtp.gmail.com
Objet_de_connexion_avec_ServeurSMTP = smtplib.SMTP(smtp_host, 587, timeout=10)
# activation de l'affichage des commandes reponses echangees avec le serveur
Objet_de_connexion_avec_ServeurSMTP.set_debuglevel(1)
try:
    # envoie de la commande STARTTLS (connection securisee)
    Objet_de_connexion_avec_ServeurSMTP.starttls()
    # envoie des informations d'authentification de l'expediteur (AdresseMailExpediteur,MotDePasse )
    Objet_de_connexion_avec_ServeurSMTP.login(AdresseMailExpediteur,MotDePasse )
    # envoie du mail
    Objet_de_connexion_avec_ServeurSMTP.sendmail("MasterInfo.tlemcen.MAIL@gmail.com", AdresseMailRecepteur , Mail_FormatRFC822.as_string())
finally:
    #envoie de la commande QUIT
    Objet_de_connexion_avec_ServeurSMTP.quit()

```

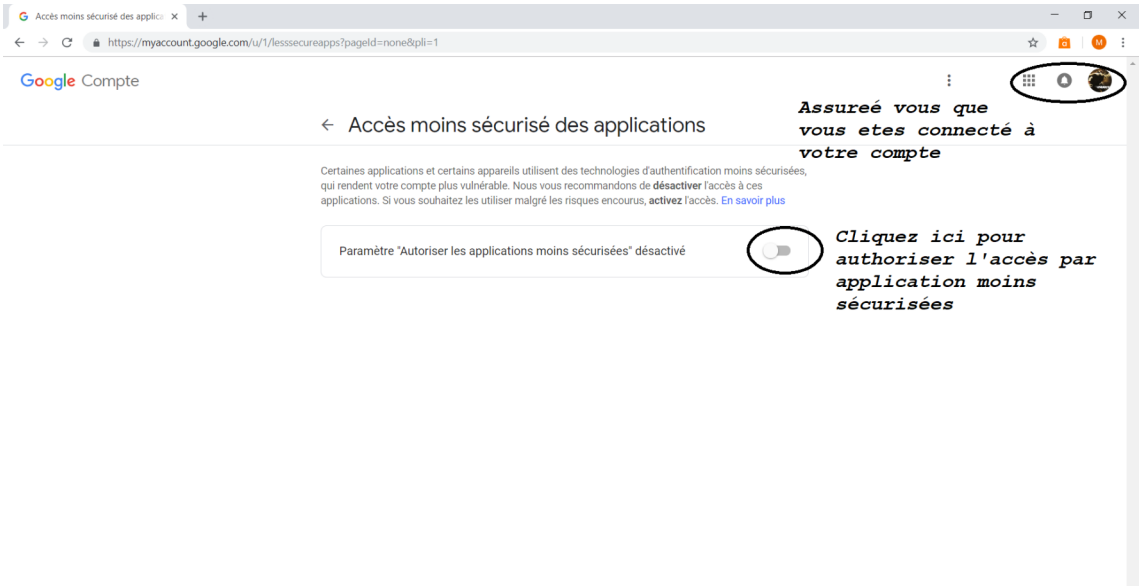
Comme expliqué dans le cours, les commandes SMTP seront échangées avec le serveur pour transmettre le message. Sauf que ici, le programme utilisateur connecte à ça boite mail (afin de s'authentifier) avant de transférer le mail au serveur où ça boite mail est localisée. Ci dessous, la figure explique les commandes les plus importants dans ce processus :



Si vous souhaitez transmettre un mail en utilisant votre propre adresse Gmail, activez l'accès à votre boite par des applications moins sécurisées à partir de ce lien :

https://myaccount.google.com/u/0/lesssecureapps?pli=1&pageId=none

Après le teste du programme, il est préférable de désactiver cette option.



Décoder le contenu avec l'encodage 64bits

IV

Le programme python2 [*TP Decode64bit.py*](voir le code python 3 à la fin de l'exercice [*TP Decode64bit py3.py*]) suivant permet de décoder un contenu transmis avec encodage 64bits. Cette encodage est largement utilisé pour transmettre les fichiers sur mails.

```
# code python 2
def Decode64bits(_4Octets_lu):
    # prendre en parametre 4octets (4 caracteres)
    global FichierDecode # variable globale

    Sequence_4Octets=_4Octets_lu
    ValeurBinaire=""
    # si la sequence de caractere contient "=" alors celle-ci marque la fin du fichier
    '''if("=" in Sequence_4Octets):
        # elimine les caractere "=" a la fin de la sequence
        Sequence_4Octets=Sequence_4Octets[:Sequence_4Octets.find("="):]'''

    for octet_indx in range(len(Sequence_4Octets)):
        # convertir le caractere a ca representation 64bits et ensuite vers ca representation binaire
        binval=bin(ConvertirCaracter64bits(Sequence_4Octets[octet_indx]))[2::]
        # complete la representation binaire par des 0s a gauche (pour avoir une representation 6bits)
        binval="0"*(6-len(binval))+binval
        ValeurBinaire+=binval

    for Threebits_char in range(0,len(ValeurBinaire),8):
        if(Threebits_char+8>len(ValeurBinaire)):
            break
        # pour toutes les sequence de 8 bits dans ValeurBinaire
        # obtenir le caractere 8bits correspandant
        CaractereASCII8bits=chr( int("0b"+ValeurBinaire[Threebits_char:Threebits_char+8:],0) )

        # enregistrer le caractere sur fichier
        FichierDecode.write(CaractereASCII8bits)
    L=len(ValeurBinaire)
    # s'il reste des bits supplementaires
    if(L%8!=0):
        # si la sequence de bits ne contient pas que des 0
        if("0"*(L%8)!=ValeurBinaire[L-L%8::]):
            # complete la representation par des 0 a droit
            CaractereASCII8bits=chr( int("0b"+ValeurBinaire[L-L%8::]+(8-L%8)*"0",0) )

```

```
37              # enregistrer le caractere sur fichier
38              FichierDecode.write(CaractereASCII8bits)
39      #fin de la procedure Decode64bits
40
41
42  # si le caractere n'appartient pas a [A..Z, a..z, 0..9, +, / , =], supprimer le
    caratere
43  def SupprimerCaracteresSupplementaires(Contenu):
44      Contenucleaned=""
45      for caractere in Contenu :
46          #print caractere
47          if(caractere>='A' and caractere<='Z'):
48              Contenucleaned+=caractere
49              continue
50          if(caractere>='a' and caractere<='z'):
51              Contenucleaned+=caractere
52              continue
53          if(caractere>='0' and caractere<='9' ):
54              Contenucleaned+=caractere
55              continue
56          if(caractere=="+"):
57              Contenucleaned+=caractere
58              continue
59          if(caractere=="/"):
60              Contenucleaned+=caractere
61              continue
62          if(caractere=="="):
63              Contenucleaned+=caractere
64              continue
65      return Contenucleaned # fin de la procedure
    SupprimerCaracteresSupplementaires
66  # obtient la representation 64bits du caractere
67  ''' [A .. Z] = [0..25]
68  [a .. z] = [26..51]
69  [0..9]=[52..61]
70  "+" = 61
71  "/" = 62
72  "=" = 0 ( contrairement a A, les "=" sont ajoutes a la fin
73  pour que la taille de la chaine encdee soit un multiple de 4
74  '''
75  def ConvertirCaracter64bits(caractere):
76      if(caractere>='A' and caractere<='Z'):
77          return ord(caractere)-65
78      if(caractere>='a' and caractere<='z'):
79          return ord(caractere)-71
80      if(caractere>='0' and caractere<='9' ):
81          return ord(caractere)+4
82      if(caractere=="+"):
83          return 62
84      if(caractere=="/"):
85          return 63
86      # le dernier cas correspond a "=" (si caractere=="=" return 0)
87      return 0 # fin de la procedure ConvertirCaracter64bits
88
89  # lire le fichier qu'ont souhaite decoder
90  FichierEncode =open("MessageEncoder.txt",'rb')
91  # ouvrir le fichier dans le quel ont souhaite enregistrer le fichier decode
92  FichierDecode=open("MessageDecoder.jpg","wb")
93
```

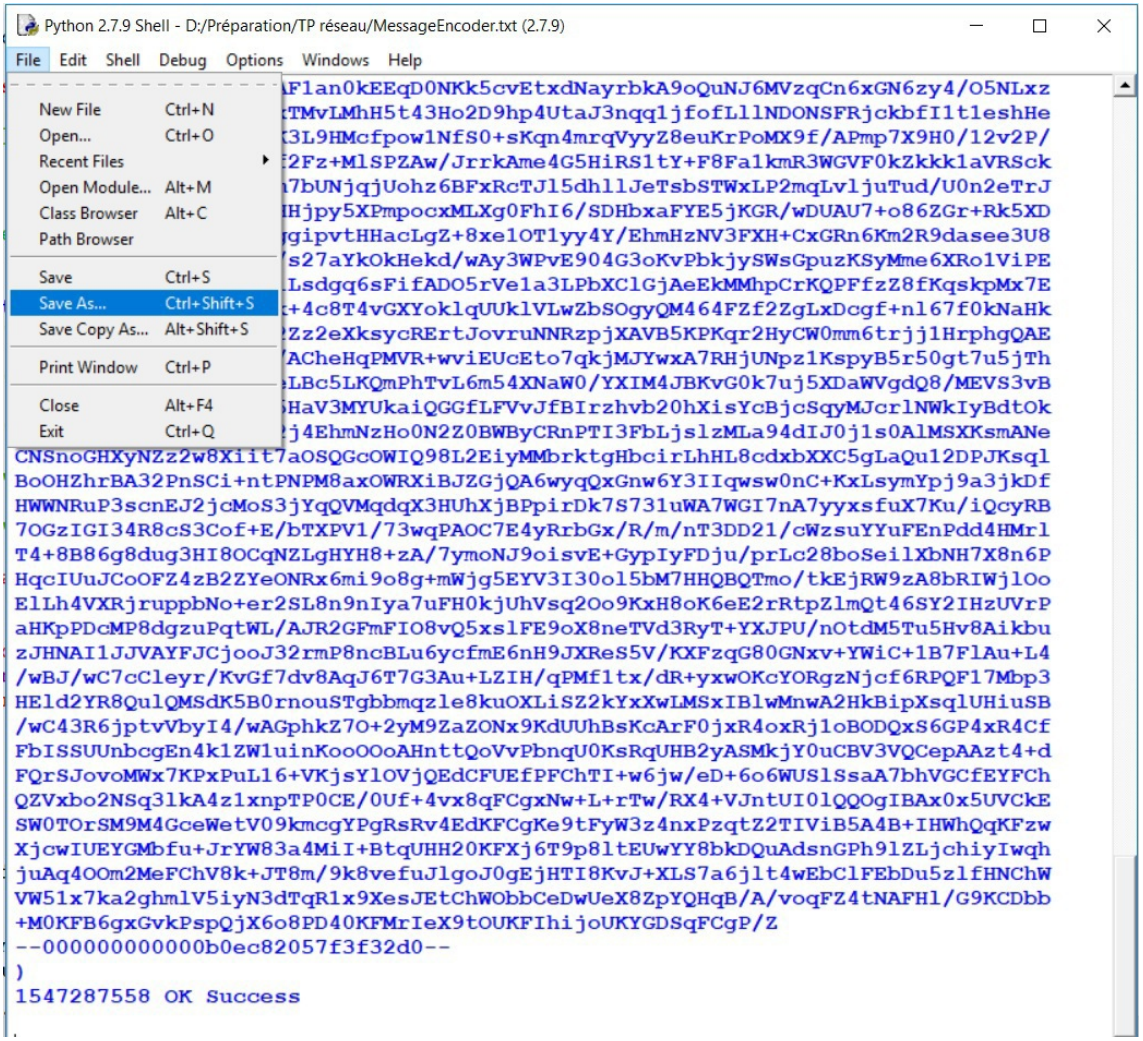
Décoder le contenu avec l'encodage 64bits

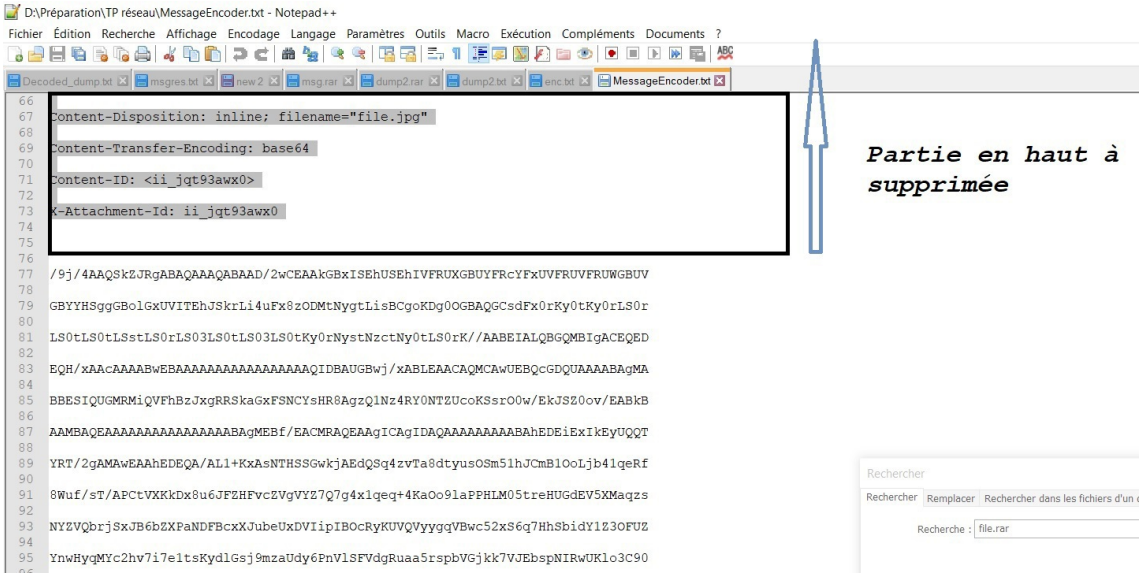
```
Contenu=""
Contenu_Restant=""
# lire le fichier. Pour chaque morceau de d'octets
for Octets in FichierEncode:

    Contenu=Octets
    Contenu=SupprimerCaracteresSupplementaires(Contenu)
    Contenu=Contenu_Restant+Contenu

    l=len(Contenu)
    Contenu_Restant=""
    # prendre une partie de la chaine de caratere avec une taille multiple de 4
    # si des caracteres supplementaires restent, on les garde dans le tompon "Contenu_Restant"
    if(l%4!=0):
        Contenu_Restant=Contenu[len(Contenu)-(l%4):len(Contenu):]
    if(len(Contenu)<4):
        continue
    PartieContenu_deTailleMultiple_de_4=l-(l%4)
    # appliquer le decodage sur chaque morceau de taille 4 caracteres
    for i in range(0,PartieContenu_deTailleMultiple_de_4,4):
        Decode64bits(Contenu[i:i+4:])
# fermeture du fichier de l'encodage
FichierDecode.close()

```

Pour tester ce code sur un contenu encodé par cette méthode, exécutez les étapes suivantes :

1. Lancez le client IMAP de l'exercice précédant
2. Exécutez les commandes IMAP suivantes :

   ```
   LOGIN MasterInfo.tlemcen.MAIL CodingMonk?L7*5
   SELECT INBOX
   UID FETCH 4 Body.Peek[Text]
   ```
3. Enregistrez le résultat à partir de IDLE sous le nom *MessageEncoder.txt* et dans le même répertoire que le programme décodage 64bits (le programme python en haut)
4. Enlevez les entêtes et les lignes de commandes pour garder seulement le contenu de *file.jpg* encodé (enlevez aussi la réponse du serveur à la fin du message)
5. Exécutez le programme de décodage (celui au début de cet exercice)
6. Ouvrez le fichier résultat (*MessageDecoder.jpg* sera une image lisible)





Voici le code python 3 du décodeur 64 bits [*TP Decode64bit py3.py*] présenté au début de cet exercice :

```
1 # code python 3
```

Décoder le contenu avec l'encodage 64bits

```
def Decode64bits(_4Octets_lu):
    # prendre en parametre 4octets (4 caracteres)
    global FichierDecode # variable globale

    Sequence_4Octets=_4Octets_lu
    ValeurBinaire=""
    # si la sequence de caractere contient "=" alors celle-ci marque la fin du fichier
    if("=" in Sequence_4Octets):
        # elimine les caractere "=" a la fin de la sequence
        Sequence_4Octets=Sequence_4Octets[:Sequence_4Octets.find("="):]

    for octet_indx in range(len(Sequence_4Octets)):
        # convertir le caractere a ca representation 64bits et ensuite vers ca representation binaire
        binval=bin(ConvertirCaracter64bits(Sequence_4Octets[octet_indx]))[2::]
        # complete la representation binaire par des 0s a gauche (pour avoir une representation 6bits)
        binval="0"*(6-len(binval))+binval
        ValeurBinaire+=binval

    for Threebits_char in range(0,len(ValeurBinaire),8):
        if(Threebits_char+8>len(ValeurBinaire)):
            break
        # pour toutes les sequence de 8 bits dans ValeurBinaire
        # obtenir le caractere 8bits correspandant
        CaractereASCII8bits=chr( int("0b"+ValeurBinaire[Threebits_char:Threebits_char+8:],0) )
        #print (CaractereASCII8bits.encode(),ord(CaractereASCII8bits))
        # enregistrer le caractere sur fichier
        FichierDecode.write(bytes([ord(CaractereASCII8bits)]))
    L=len(ValeurBinaire)
    # s'il reste des bits supplementaires
    if(L%8!=0):
        # si la sequence de bits ne contient pas que des 0
        if("0"*(L%8)!=ValeurBinaire[L-L%8::]):
            # complete la representation par des 0 a droit
            CaractereASCII8bits=chr( int("0b"+ValeurBinaire[L-L%8::]+(8-L%8)*"0",0) )

            # enregistrer le caractere sur fichier
            FichierDecode.write(bytes([ord(CaractereASCII8bits)]))

    #fin de la procedure Decode64bits

# si le caractere n'appartient pas a [A..Z, a..z, 0..9, +, / , =], supprimer le caratere
def SupprimerCaracteresSupplementaires(Contenu):
    Contenucleaned=""
    for octet in Contenu :
        caractere=(chr(octet))
        if(caractere>='A' and caractere<='Z'):
            Contenucleaned+=caractere
            continue
        if(caractere>='a' and caractere<='z'):
            Contenucleaned+=caractere
            continue
        if(caractere>='0' and caractere<='9' ):
            Contenucleaned+=caractere
            continue
```

```
        if(caractere=="+"):
            Contenucleaned+=caractere
            continue
        if(caractere=="/"):
            Contenucleaned+=caractere
            continue
        if(caractere=="="):
            Contenucleaned+=caractere

    return Contenucleaned # fin de la procedure SupprimerCaracteresSupplementaires

# obtient la representation 64bits du caractere
''' [A .. Z] = [0..25]
[a .. z] = [26..51]
[0..9]=[52..61]
"+" = 61
"/" = 62
"=" = 0 ( contrairement a A, les "=" sont ajoutes a la fin
pour que la taille de la chaine encdee soit un multiple de 4
'''
def ConvertirCaracter64bits(caractere):
    if(caractere>='A' and caractere<='Z'):
        return ord(caractere)-65
    if(caractere>='a' and caractere<='z'):
        return ord(caractere)-71
    if(caractere>='0' and caractere<='9' ):
        return ord(caractere)+4
    if(caractere=="+"):
        return 62
    if(caractere=="/"):
        return 63
    # le dernier cas correspond a "=" (si caractere=="=" return 0)
    return 0 # fin de la procedure ConvertirCaracter64bits

# lire le fichier qu'ont souhaite decoder
FichierEncode =open("MessageEncoder.txt",'rb')
# ouvrir le fichier dans le quel ont souhaite enregistrer le fichier decode
FichierDecode=open("MessageDecoder.jpg","wb")


Contenu=""
Contenu_Restant=""
# lire le fichier. Pour chaque morceau de d'octets
for Octets in FichierEncode:

    Contenu=Octets
    Contenu=SupprimerCaracteresSupplementaires(Contenu)
    Contenu=Contenu_Restant+Contenu
    l=len(Contenu)
    Contenu_Restant=""
    # prendre une partie de la chaine de caratere avec une taille multiple de 4
    # si des caracteres supplementaires restent, on les garde dans le tompon "Contenu_Restant"
    if(l%4!=0):
        Contenu_Restant=Contenu[len(Contenu)-(l%4):len(Contenu):]
    if(len(Contenu)<4):
        continue

```

Décoder le contenu avec l'encodage 64bits

```
113     PartieContenu_deTailleMultiple_de_4=l-(l%4)
114     # appliquer le decodage sur chaque morceau de taille 4 caracteres
115     for i in range(0,PartieContenu_deTailleMultiple_de_4,4):
116         Decode64bits(Contenu[i:i+4:])
117 # fermeture du fichier de l'encodage
118 FichierDecode.close()
119
```