

Exercices : Accéder à une boîte mail avec POP et IMAP



Ilyas Bambrik

Table des matières



Introduction	3
I - Accéder à une boîte mail avec POP	4
II - Client IMAP	8
III - Envoie mail par SMTP	12
IV - Décoder le contenu avec l'encodage 64bits	15

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 :

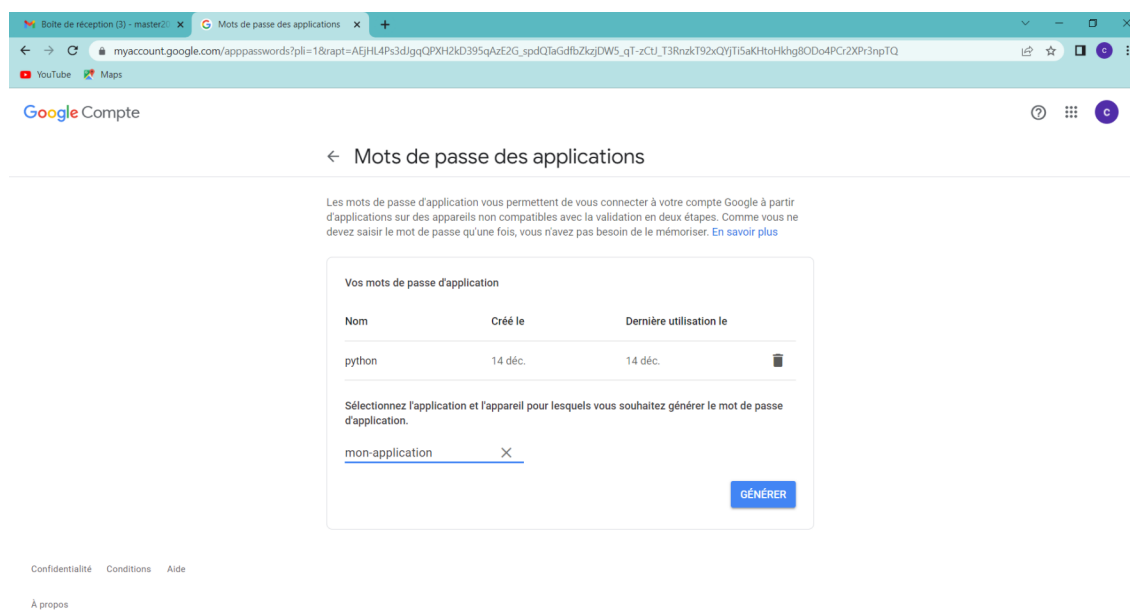
https://drive.google.com/open?id=1TO0TA1HqgrLZ8HYs-ZxqxKOW__m1qZBi

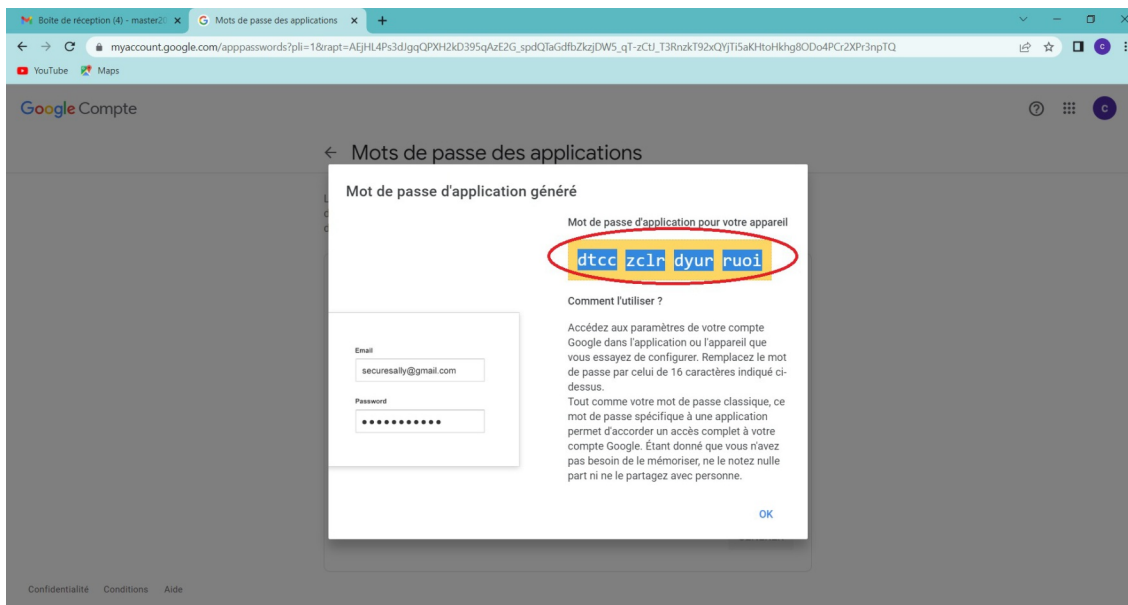


Accéder à une boîte mail avec POP



Avant de commencer de tester le code suivant vous devez créer une nouvelle boîte et activer l'authentification double facteur (Validation en deux étapes). Par la suite vous devez générer un mot de passe d'application qui sera utilisé pour accéder à votre compte. Les captures suivantes illustrent les étapes à suivre.





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

1. *USER*, *PASS* fonctionnent comme dans FTP pour transmettre le login (boîte mail) et mot de passe :
2. *STAT* : renvoie le nombre de mails dans la boîte 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 boîte ;
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*]):

```

1 # code python 2 client POP
2 import socket
3 import ssl
4
5 def RecevoirLaReponseServeur(ConnexionSecurise, Delimiteur_De_Fin_De_Reponse):
6     Reponse_recu= ConnexionSecurise.recv(1024)
7     if (" +OK" in Reponse_recu) and Delimiteur_De_Fin_De_Reponse=="":
8         return Reponse_recu
9     if ("-ERR" in Reponse_recu):
10        return Reponse_recu
11    while True:
12        # lire la reponse jusqu'a atteindre la sequence de caratere
13        Delimiteur_De_Fin_De_Reponse
14        if(Delimiteur_De_Fin_De_Reponse in Reponse_recu):
15            break
16        Reponse_recu+=ConnexionSecurise.recv(1024)
17    return Reponse_recu
18
19 NomDomaineDuServeurPOP="pop.gmail.com"
20 ConnexionAuServeur=socket.socket ()
21 ConnexionAuServeur.connect ( (NomDomaineDuServeurPOP,995) )
22 # creation d'une connexion securise
23 ConnexionSecurise=ssl.create_default_context ().wrap_socket (ConnexionAuServeur,
24     server_hostname=NomDomaineDuServeurPOP)
25 Commande=""
26 # liste de commande pour les quels la reponse setermine par "\r\n.\r\n" (un
27 point au debut de la ligne)
28 ListeCommande=["list", "top", "retr", "uidl"]
29 while True:
30     Delimiteur=""
31     for command_terminer_par_unpoint in ListeCommande:
32         # si la commande issue par le client appartient a ListeCommande
33         # la reponse de cette commande doit se terminee par "\r\n.\r\n"
34         if(Commande.strip().lower().startswith(command_terminer_par_unpoint)):
35             Delimiteur="\r\n.\r\n"
36         # les autres commande comme QUIT, PASS et USER auront une seule ligne
37         de reponse seulement(+OK ou -ERR )
38         print RecevoirLaReponseServeur(ConnexionSecurise,Delimiteur)
39         if(Commande.strip().lower().startswith("quit")):
40             break
41         Commande=raw_input ()
42         ConnexionSecurise.send(Commande+"\r\n")
43 ConnexionSecurise.close ()
44

```

Question

Testez les commandes suivantes et faites des captures d'écran des résultats (MasterInfo.tlemcen.MAIL@gmail.com et Master1SICMID2021 sont utilisés comme exemple seulement, vous devez utiliser le mot de passe d'application et le nom de votre boîte):

- USER MasterInfo.tlemcen.MAIL@gmail.com
- PASS Master1SICMID2021
- STAT
- UIDL
- LIST
- RETR 3
- QUIT

```

1 # code python 3 client POP
2 import socket
3 import ssl
4
5 def RecevoirLaReponseServeur(ConnexionSecurise, Delimiteur_De_Fin_De_Reponse):
6     Reponse_recu= ConnexionSecurise.recv(1024).decode("unicode_escape")
7     if ("+OK" in Reponse_recu) and Delimiteur_De_Fin_De_Reponse=="":
8         return Reponse_recu
9     if ("-ERR" in Reponse_recu):
10        return Reponse_recu
11    while True:
12        # lire la reponse jusqu'a atteindre la sequence de caratere
13        Delimiteur_De_Fin_De_Reponse
14        if(Delimiteur_De_Fin_De_Reponse in Reponse_recu):
15            break
16        Reponse_recu+=ConnexionSecurise.recv(1024).decode("unicode_escape")
17    return Reponse_recu
18
19 NomDomaineDuServeurPOP="pop.gmail.com"
20 ConnexionAuServeur=socket.socket()
21 ConnexionAuServeur.connect((NomDomaineDuServeurPOP,995))
22 # creation d'une connexion securise
23 ConnexionSecurise=ssl.create_default_context().wrap_socket(ConnexionAuServeur,
24     server_hostname=NomDomaineDuServeurPOP)
25 Commande=""
26 # liste de commande pour les quels la reponse setermine par "\r\n.\r\n" (un
27     point au debut de la ligne)
28 ListeCommande=["list","top","retr","uidl"]
29 while True:
30     Delimiteur=""
31     for command_terminer_par_unpoint in ListeCommande:
32         # si la commande issue par le client appartient a ListeCommande
33         # la reponse de cette commande doit se terminee par "\r\n.\r\n"
34         if(Commande.strip().lower().startswith(command_terminer_par_unpoint)):
35             Delimiteur="\r\n.\r\n"
36         # les autres commande comme QUIT, PASS et USER auront une seule ligne
37         de reponse seulement(+OK ou -ERR )
38         print(RecevoirLaReponseServeur(ConnexionSecurise,Delimiteur))
39         if(Commande.strip().lower().startswith("quit")):
40             break
41         Commande=input()
42         ConnexionSecurise.send((Commande+"\r\n").encode())
43 ConnexionSecurise.close()
44

```

Client IMAP



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 répondu] ;
- 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 supportées 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 récupérer 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*).


```

1 * OK Gimap ready for requests from 41.100.32.83 j4mb1024322636wrw
2
3 Prefixe_Commande_TAG001 Capability
4 * CAPABILITY IMAP4rev1 UNSELECT IDLE NAMESPACE QUOTA ID XLIST CHILDREN X-GM-EXT-
  1 XZZZY SASL-IR AUTH=XOAUTH2 AUTH=PLAIN AUTH=PLAIN-CLIENTTOKEN AUTH=OAUTHBEARER
  AUTH=XOAUTH
5
6 Prefixe_Commande_TAG001 OK Thats all she wrote! j4mb1024322636wrw
7
8 Prefixe_Commande_TAG002 LOGOUT
9 * BYE Logout Requested j4mb1024322636wrw
10
11 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]):

```

1 # code python 2 client IMAP
2 import socket
3 import ssl
4 import time
5 def RecevoirLaReponseServeur(ConnexionSecurise, Tag_Commande):
6     Reponse_recu=ConnexionSecurise.recv(1024)
7     Delimiteur_De_Fin_De_Reponse=Tag_Commande+" "
8     while True:
9         # lire la reponse jusqu'a atteindre la sequence de caractere
10        Delimiteur_De_Fin_De_Reponse (TAG de la commande)
11        if(Delimiteur_De_Fin_De_Reponse in Reponse_recu):
12            break
13        Reponse_recu+=ConnexionSecurise.recv(1024)
14    return Reponse_recu
15 NomDomaineDuServeurIMAP="imap.gmail.com"
16 ConnexionAuServeur=socket.socket()
17 ConnexionAuServeur.connect((NomDomaineDuServeurIMAP,993))
18 # creation d'une connexion securise
19 ConnexionSecurise=ssl.create_default_context().wrap_socket(ConnexionAuServeur,
20    server_hostname=NomDomaineDuServeurIMAP)
21 Commande=""
22 # recevoir le message de connexion
23 print ConnexionSecurise.recv(2048)
24 TAG_Commande=""
25 while True:
26     # lire la commande
27     Commande=raw_input()
28     # cree un tag a partir de l'horloge systeme (time.time())
29     TAG_Commande_Horloge=str(int(time.time()))
30     # envoie de la commande avec le TAG horloge
31     ConnexionSecurise.send(TAG_Commande_Horloge+" "+Commande+"\r\n")
32     print RecevoirLaReponseServeur(ConnexionSecurise,TAG_Commande_Horloge)
33     # si la commande tapee par le client == LOGOUT, sortir de la boucle de
34     lecture de commandes
35     if(Commande.strip().lower().startswith("logout")):
36         break
37 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

```

1 #code python 3 client IMAP
2 import socket
3 import ssl
4 import time
5 def RecevoirLaReponseServeur(ConnexionSecurise, Tag_Commande):
6     Reponse_recu=ConnexionSecurise.recv(1024).decode("unicode_escape")
7     Delimiteur_De_Fin_De_Reponse=Tag_Commande+" "
8     while True:
9         # lire la reponse jusqu'a atteindre la sequence de caractere
10        Delimiteur_De_Fin_De_Reponse (TAG de la commande)
11        if(Delimiteur_De_Fin_De_Reponse in Reponse_recu):
12            break
13        Reponse_recu+=ConnexionSecurise.recv(1024).decode("unicode_escape")
14    return Reponse_recu
15 NomDomaineDuServeurIMAP="imap.gmail.com"
16 ConnexionAuServeur=socket.socket()
17 ConnexionAuServeur.connect((NomDomaineDuServeurIMAP,993))
18 # creation d'une connexion securise
19 ConnexionSecurise=ssl.create_default_context().wrap_socket(ConnexionAuServeur,
20    server_hostname=NomDomaineDuServeurIMAP)
21 Commande=""
22 # recevoir le message de connexion
23 print( ConnexionSecurise.recv(2048).decode("unicode_escape"))
24 TAG_Commande=""
25 while True:
26     # lire la commande
27     Commande=input()
28     # cree un tag a partir de l'horloge systeme (time.time())
29     TAG_Commande_Horloge=str(int(time.time()))
30     # envoie de la commande avec le TAG horloge
31     ConnexionSecurise.send((TAG_Commande_Horloge+" "+Commande+"\r\n").encode())
32     print( RecevoirLaReponseServeur(ConnexionSecurise,TAG_Commande_Horloge))
33     # si la commande tapee par le client == LOGOUT, sortir de la boucle de
34     lecture de commandes
35     if(Commande.strip().lower().startswith("logout")):
36         break
37 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] :

```
1 # le meme programme fonctionne pour python 2 et 3
2 import smtplib
3 from email.mime.text import MIMEText
4 from email.header import Header
5 # nom domaine du serveur SMTP pour le quel ont souhaite transmettre un mail
6 smtp_host = 'smtp.gmail.com'
7
8 # donnees d'authentification du transmetteur (adresse mail et mot de passe)
9 AdresseMailExpediteur="MasterInfo.tlemcen.MAIL@gmail.com"
10 MotDePasse="CodingMonk?L7*5"
11
12 # adresse mail recepateur
13 AdresseMailRecepteur="VOTRE.ADRESSE.EMAIL.SVP@gmail.com"# ecrivez votre adresse
   mail ici
14
15 # Mail formate avec le format RFC822 ( voir le cours SMTP)
16 # contenu text
17 Mail_FormatRFC822 = MIMEText('Testing, Testing Hello! Are you there? k', 'plain'
   , 'utf-8')
18 # entete Subject:
19 Mail_FormatRFC822['Subject'] ='Sujet == THIS IS A MAIL'
20 # entete From:
21 Mail_FormatRFC822['From'] = AdresseMailExpediteur
22 # entete To:
23 Mail_FormatRFC822['To'] = AdresseMailRecepteur
24
25 # etablisement de la connexion avec le serveur SMTP smtp.gmail.com
26 Objet_de_connexion_avec_ServeurSMTP = smtplib.SMTP(smtp_host, 587, timeout=10)
27 # activation de l'affichage des commandes reponses echangees avec le serveur
28 Objet_de_connexion_avec_ServeurSMTP.set_debuglevel(1)
29 try:
30     # envoie de la commande STARTTLS (connection securisee)
31     Objet_de_connexion_avec_ServeurSMTP.starttls()
32     # envoie des informations d'authentification de l'expediteur
   (AdresseMailExpediteur,MotDePasse )
33     Objet_de_connexion_avec_ServeurSMTP.login(AdresseMailExpediteur,MotDePasse )
34     # envoie du mail
35     Objet_de_connexion_avec_ServeurSMTP.sendmail("MasterInfo.tlemcen.MAIL@gmail.
   com", AdresseMailRecepteur , Mail_FormatRFC822.as_string())
36 finally:
37     #envoie de la commande QUIT
38     Objet_de_connexion_avec_ServeurSMTP.quit ()
39
```

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 boîte mail (afin de s'authentifier) avant de transférer le mail au serveur où ça boîte mail est localisée. Ci dessous, la figure explique les commandes les plus importants dans ce processus :

```

Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help
>>> ===== KESTAKT =====
>>>
send: ehlo [169.254.4.120]\r\n' Envoie de la commande EHLO
reply: '250-smtp.gmail.com at your service, [41.100.32.83]\r\n'
reply: '250-SIZE 35882577\r\n'
reply: '250-8BITMIME\r\n\r\n'
reply: '250-STARTTLS\r\n\r\n'
reply: '250-ENHANCEDSTATUSCODES\r\n\r\n'
reply: '250-PIPELINING\r\n\r\n'
reply: '250 SMTPUTF8\r\n\r\n'
reply: retcode (250); Msg: smtp.gmail.com at your service, [41.100.32.83]
SIZE 35882577
8BITMIME
STARTTLS
ENHANCEDSTATUSCODES
PIPELINING
SMTPUTF8
send: 'STARTTLS\r\n' Envoie de startTLS (connexion sécurisée)
reply: '220 2.0.0 Ready to start TLS\r\n'
reply: retcode (220); Msg: 2.0.0 Ready to start TLS
send: ehlo [169.254.4.120]\r\n' Envoie de EHLO une nouvelle fois
reply: '250-smtp.gmail.com at your service, [41.100.32.83]\r\n'
reply: '250-SIZE 35882577\r\n'
reply: '250-8BITMIME\r\n\r\n'
reply: '250-AUTH LOGIN PLAIN XOAUTH2 PLAIN-CLIENTTOKEN OAUTHBEARER XOAUTH\r\n\r\n'
reply: '250-ENHANCEDSTATUSCODES\r\n\r\n'
reply: '250-PIPELINING\r\n\r\n'
reply: '250 SMTPUTF8\r\n\r\n'
reply: retcode (250); Msg: smtp.gmail.com at your service, [41.100.32.83]
SIZE 35882577
8BITMIME
AUTH LOGIN PLAIN XOAUTH2 PLAIN-CLIENTTOKEN OAUTHBEARER XOAUTH
ENHANCEDSTATUSCODES
PIPELINING
SMTPUTF8
send: 'AUTH PLAIN AE1hc3RlckluZm8udGxlbWV1bi5NQULMQGdtYWlsLmNvbQBDb2RpbmdNb25rP0
w3KjU=\r\n'
reply: '235 2.7.0 Accepted\r\n'
reply: retcode (235); Msg: 2.7.0 Accepted
send: 'mail FROM:<MasterInfo.tlemcen.MAIL@gmail.com> size=256\r\n'
reply: '250 2.1.0 OK 200sm22269406wmw.31 - gsmtpl\r\n'
reply: retcode (250); Msg: 2.1.0 OK 200sm22269406wmw.31 - gsmtpl
send: 'rcpt TO:<[redacted]@gmail.com>\r\n' Envoie de la commande RCPT
reply: '250 2.1.5 OK 200sm22269406wmw.31 - gsmtpl\r\n' avec TO
reply: retcode (250); Msg: 2.1.5 OK 200sm22269406wmw.31 - gsmtpl
send: 'data\r\n' Envoie de la CMD'DATA' pour commencer à transmettre le MAIL
reply: '354 Go ahead 200sm22269406wmw.31 - gsmtpl\r\n'
reply: retcode (354); Msg: Go ahead 200sm22269406wmw.31 - gsmtpl
data: (354, 'Go ahead 200sm22269406wmw.31 - gsmtpl') Contenu du mail
send: 'Content-Type: text/plain; charset="utf-8"\r\nMIME-Version: 1.0\r\nContent
-Transfer-Encoding: base64\r\nSubject: Sujet == THIS IS A MAIL\r\nFrom: MasterIn
fo.tlemcen.MAIL@gmail.com\r\nTo: [redacted]@gmail.com\r\n\r\nVGZzdGluZywgV
GVzdGluZyBIZWxsbyEgQXJlIHlvdSB0aGVyZT8gaw==\r\n.\r\n' Sequence fin Mail
reply: '250 2.0.0 OK 1547224626 200sm22269406wmw.31 - gsmtpl\r\n'
reply: retcode (250); Msg: 2.0.0 OK 1547224626 200sm22269406wmw.31 - gsmtpl
data: (250, '2.0.0 OK 1547224626 200sm22269406wmw.31 - gsmtpl')
send: 'quit\r\n' Envoie de la commande QUIT pour cloturer la connexion au serveur
reply: '221 2.0.0 closing connection 200sm22269406wmw.31 - gsmtpl\r\n'
reply: retcode (221); Msg: 2.0.0 closing connection 200sm22269406wmw.31 - gsmtpl
>>>

```

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

<https://myaccount.google.com/u/0/lesssecureapps?pli=1&pageld=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.

```

1 # code python 2
2 def Decode64bits(_4Octets_lu):
3     # prendre en parametre 4octets (4 caracteres)
4     global FichierDecode # variable globale
5
6     Sequence_4Octets=_4Octets_lu
7     ValeurBinaire=""
8     # si la sequence de caractere contient "=" alors celle-ci marque la fin du
    fichier
9     '''if("=" in Sequence_4Octets):
10         # elimine les caractere "=" a la fin de la sequence
11         Sequence_4Octets=Sequence_4Octets[:Sequence_4Octets.find("="):]'''
12
13     for octet_indx in range(len(Sequence_4Octets)):
14         # convertir le caractere a ca representation 64bits et ensuite vers ca
    representation binaire
15         binval=bin(ConvertirCaracter64bits(Sequence_4Octets[octet_indx]))[2::]
16         # complete la representation binaire par des 0s a gauche (pour avoir une
    representation 6bits)
17         binval="0"*(6-len(binval))+binval
18         ValeurBinaire+=binval
19
20     for Threebits_char in range(0,len(ValeurBinaire),8):
21         if(Threebits_char+8>len(ValeurBinaire)):
22             break
23         # pour toutes les sequence de 8 bits dans ValeurBinaire
24         # obtenir le caractere 8bits correspondant
25         CaractereASCII8bits=chr( int("0b"+ValeurBinaire[Threebits_char:
    Threebits_char+8:],0) )
26
27         # enregistrer le caractere sur fichier
28         FichierDecode.write(CaractereASCII8bits)
29     L=len(ValeurBinaire)
30     # s'il reste des bits supplementaires
31     if(L%8!=0):
32         # si la sequence de bits ne contient pas que des 0
33         if("0"*(L%8)!=ValeurBinaire[L-L%8::]):
34             # complete la representation par des 0 a droit
35             CaractereASCII8bits=chr( int("0b"+ValeurBinaire[L-L%8::]+(8-L%8)*"0",
    0) )
36

```

```

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
    caractere
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 ajoutees 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 61
84     if(caractere==" /"):
85         return 62
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

```



```

94 Contenu=""
95 Contenu_Restant=""
96 # lire le fichier. Pour chaque morceau de d'octets
97 for Octets in FichierEncode:
98
99     Contenu=Octets
100     Contenu=SupprimerCaracteresSupplementaires(Contenu)
101     Contenu=Contenu_Restant+Contenu
102
103     l=len(Contenu)
104     Contenu_Restant=""
105     # prendre une partie de la chaine de caractere avec une taille multiple de 4
106     # si des caracteres supplementaires restent, on les garde dans le tompon
107     "Contenu_Restant"
108     if (l%4!=0):
109         Contenu_Restant=Contenu[len(Contenu)-(l%4):len(Contenu):]
110     if (len(Contenu)<4):
111         continue
112     PartieContenu_deTailleMultiple_de_4=l-(l%4)
113     # appliquer le decodage sur chaque morceau de taille 4 caracteres
114     for i in range(0,PartieContenu_deTailleMultiple_de_4,4):
115         Decode64bits(Contenu[i:i+4:])
116 # fermeture du fichier de l'encodage
117 FichierDecode.close()
118

```

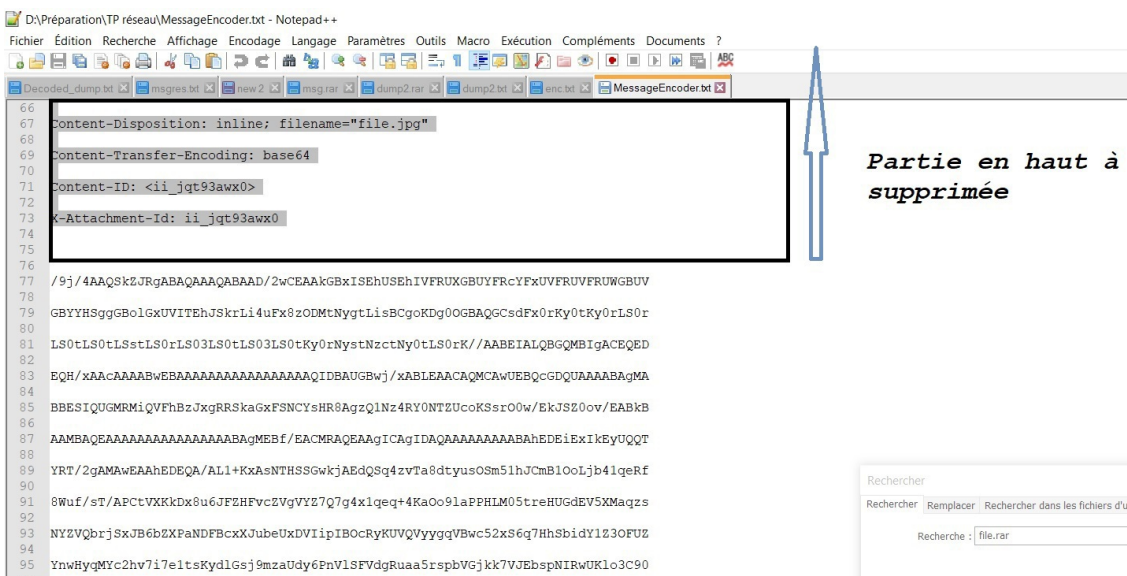
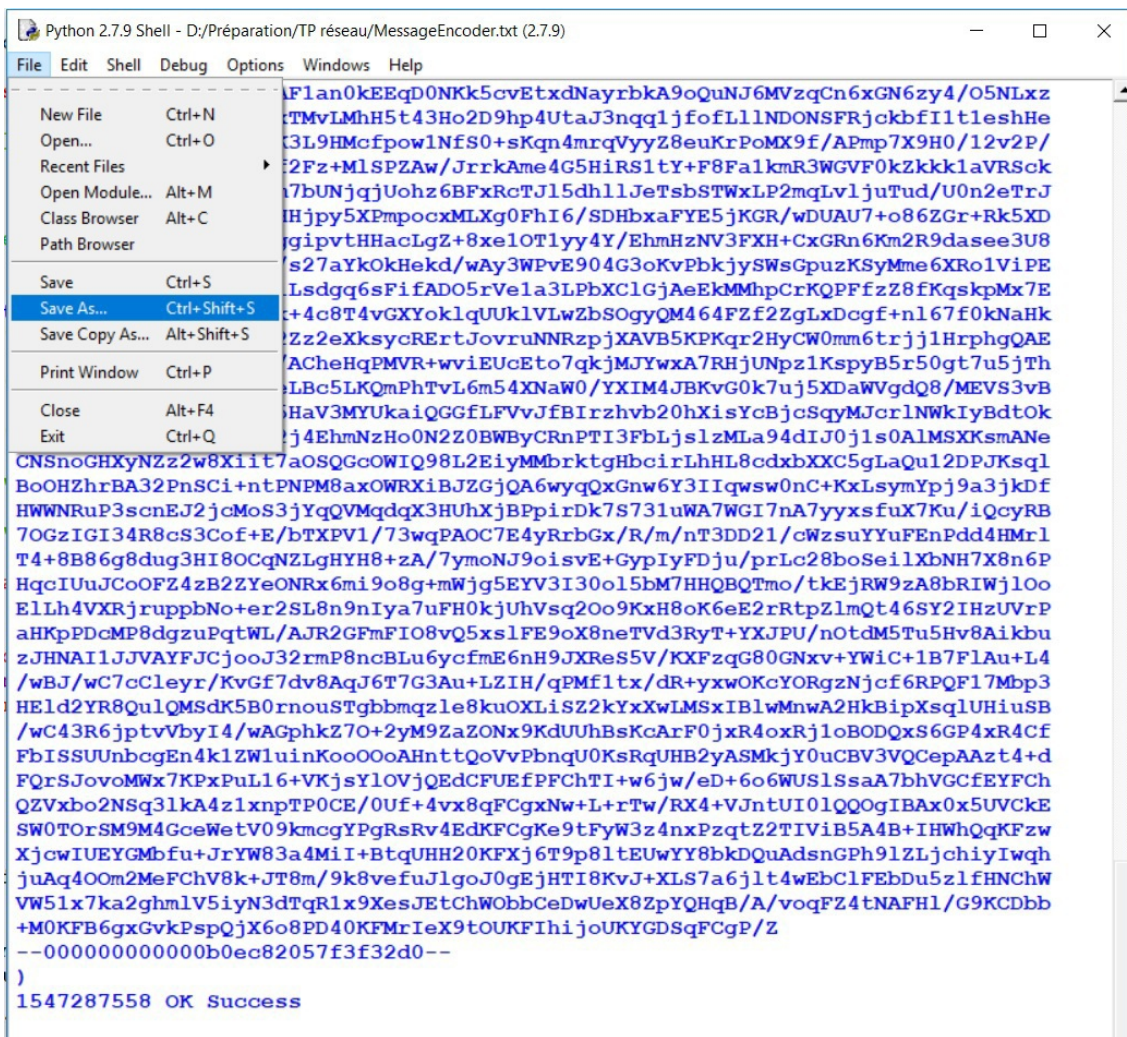
Pour tester ce code sur un contenu encodé par cette méthode, exécutez les étapes suivantes :

1. Lancez le client IMAP de l'exercice précédent
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
```

```

2 def Decode64bits(_4Octets_lu):
3     # prendre en parametre 4octets (4 caracteres)
4     global FichierDecode # variable globale
5
6     Sequence_4Octets=_4Octets_lu
7     ValeurBinaire=""
8     # si la sequence de caractere contient "=" alors celle-ci marque la fin du
    fichier
9     if("=" in Sequence_4Octets):
10        # elimine les caractere "=" a la fin de la sequence
11        Sequence_4Octets=Sequence_4Octets[:Sequence_4Octets.find("="):]
12
13    for octet_indx in range(len(Sequence_4Octets)):
14        # convertir le caractere a ca representation 64bits et ensuite vers ca
    representation binaire
15        binval=bin(ConvertirCaracter64bits(Sequence_4Octets[octet_indx]))[2::]
16        # complete la representation binaire par des 0s a gauche (pour avoir une
    representation 6bits)
17        binval="0"*(6-len(binval))+binval
18        ValeurBinaire+=binval
19
20    for Threebits_char in range(0,len(ValeurBinaire),8):
21        if(Threebits_char+8>len(ValeurBinaire)):
22            break
23        # pour toutes les sequence de 8 bits dans ValeurBinaire
24        # obtenir le caractere 8bits correspondant
25        CaractereASCII8bits=chr( int("0b"+ValeurBinaire[Threebits_char:
    Threebits_char+8:],0) )
26        #print (CaractereASCII8bits.encode(),ord(CaractereASCII8bits))
27        # enregistrer le caractere sur fichier
28        FichierDecode.write(bytes([ord(CaractereASCII8bits)]))
29    L=len(ValeurBinaire)
30    # s'il reste des bits supplementaires
31    if(L%8!=0):
32        # si la sequence de bits ne contient pas que des 0
33        if("0"*(L%8)!=ValeurBinaire[L-L%8::]):
34            # complete la representation par des 0 a droit
35            CaractereASCII8bits=chr( int("0b"+ValeurBinaire[L-L%8::]+(8-L%8)*"0",
    0) )
36
37        # enregistrer le caractere sur fichier
38        FichierDecode.write(bytes([ord(CaractereASCII8bits)]))
39
40    #fin de la procedure Decode64bits
41
42    # si le caractere n'appartient pas a [A..Z, a..z, 0..9, +, /, =], supprimer le
    caractere
43    def SupprimerCaracteresSupplementaires(Contenu):
44        Contenucleaned=""
45        for octet in Contenu :
46            caractere=(chr(octet))
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
65     return Contenucleaned # fin de la procedure
SupprimerCaracteresSupplementaires
66
67 # obtient la representation 64bits du caractere
68 ''' [A .. Z] = [0..25]
69 [a .. z] = [26..51]
70 [0..9]=[52..61]
71 "+" = 61
72 "/" = 62
73 "=" = 0 ( contrairement a A, les "=" sont ajoutees a la fin
74 pour que la taille de la chaine encdee soit un multiple de 4
75 '''
76 def ConvertirCaracter64bits(caractere):
77     if(caractere>='A' and caractere<='Z'):
78         return ord(caractere)-65
79     if(caractere>='a' and caractere<='z'):
80         return ord(caractere)-71
81     if(caractere>='0' and caractere<='9' ):
82         return ord(caractere)+4
83     if(caractere=="+"):
84         return 61
85     if(caractere==" /"):
86         return 62
87     # le dernier cas correspond a "=" (si caractere=="=" return 0)
88     return 0 # fin de la procedure ConvertirCaracter64bits
89
90 # lire le fichier qu'ont souhaite decoder
91 FichierEncode =open("MessageEncoder.txt", 'rb')
92 # ouvrir le fichier dans le quel ont souhaite enregistrer le fichier decode
93 FichierDecode=open("MessageDecoder.jpg", "wb")
94
95
96 Contenu=""
97 Contenu_Restant=""
98 # lire le fichier. Pour chaque morceau de d'octets
99 for Octets in FichierEncode:
100
101     Contenu=Octets
102     Contenu=SupprimerCaracteresSupplementaires(Contenu)
103     Contenu=Contenu_Restant+Contenu
104     l=len(Contenu)
105     Contenu_Restant=""
106     # prendre une partie de la chaine de caractere avec une taille multiple de 4
107     # si des caracteres supplementaires restent, on les garde dans le tompon
    "Contenu_Restant"
108     if(l%4!=0):
109         Contenu_Restant=Contenu[len(Contenu)-(l%4):len(Contenu):]
110     if(len(Contenu)<4):
111         continue
112

```

```
113     PartieContenu_deTailleMultiple_de_4=1-(1%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
```