

```
1  # Automate_Boring_Stuff
2
3  Sending Email and Text messages {
4
5      [Random Tasks EMailer]
6
7
8      < Mesrar Hamza >
9
10
11
12      Encadré par : < KHALFI Hamza >
13
14  }
```

Introduction du 'Problème';

Vérifier et répondre aux e-mails est une énorme perte de temps. Bien sûr, vous ne pouvez pas simplement écrire un programme pour gérer tous vos e-mails à votre place, car chaque message nécessite sa propre réponse. Mais vous pouvez toujours **automatiser** de nombreuses tâches liées aux e-mails par exemple :

- # Envoyer des e-mails au client.
- # Envoyer des Tâches à faire au membre de groupe.
- # Envoyer des e-mails de rappel de cotisations aux clients

Il est facile de le faire une fois que vous savez comment écrire des programmes qui peuvent envoyer et recevoir des e-mails.

Plan de 'Presentation' {

01 Présentation du problème

< Motivation
Et l'idée de solution >

02 SMTP Library

< Explication des "methodes"
utilisée dans SMTPlib >

03 Presentation du projet

< Code Source
Et exemple d'exécution >

}

01 {

[Présentation du problème]

< Motivation
Et l'idée de solution >

}

Problème; {

'Random Chore Assignment EMailer'

<p peut-être en classe ou à la maison et vous souhaitez répartir les tâches entre les collègues de travail (membres de la famille). et ces tâches doivent être envoyées dans un e-mail à chaque membre **au hasard**. le problème est que s'il y a 20 membres par exemple, il est difficile et ennuyeux de copier-coller et d'envoyer à chaque membre une tâche aléatoire. >

</p>

}

Motivation < /1 > {



< Mercury is the closest planet to the Sun and the smallest one in the Solar System—it's only a bit larger than the Moon >

}

Solution < /2 > {



< Ecrire un programme qui prend une liste d'emails et une liste des tâches et assigne à chaque email une tâche >

}

02 {

[SMTP Library]

< Comment envoyer un email
avec SMTP >

< Explication des “méthodes”
utilisée dans SMTPlib >

}

```
1 SMTP; {
2     'SMTP : Simple Mail Transfer Protocol'
3
4     <p est le protocole utilisé pour envoyer un e-mail. SMTP dicte
5     la façon dont les messages électroniques doivent être formatés,
6     chiffrés et relayés entre les serveurs de messagerie, ainsi que
7     tous les autres détails gérés par votre ordinateur une fois que
8     vous avez cliqué sur Envoyer. Vous n'avez pas besoin de
9     connaître ces détails techniques, car le module smtplib de
10    Python les simplifie en quelques fonctions. SMTP s'occupe
11    uniquement d'envoyer des e-mails à d'autres. Si vous souhaitez
12    récupérer et supprimer les e-mails qui vous sont envoyés.
13    Alors, vous devez utiliser un autre protocole appelé IMAP >
14 }
```


Etapes de Configuration 'SMTP' {

Etape 01 Connexion à un serveur SMTP
(création du SMTP objet)

Etape 02 Envoi du message SMTP 'Hello'
(Say hello to the server)

Etape 03 Démarrage du chiffrement TLS

Etape 04 Connexion au serveur SMTP
(Login with your account)

}

01 : Connexion à un serveur SMTP; {

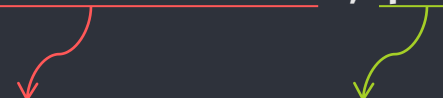
'création du SMTP objet'

<p Tout d'abord, vous devez savoir quel email '**provider**' vous utilisez et quel **port**, pour configurer votre serveur. Une simple recherche sur le Web pour *<"your provider"> smtp setting* devrait indiquer le serveur et le port à utiliser. >

Pour crée un objet SMPT :

```
>>> smtpObj = smtplib.SMTP("SMTP server domain", port)
```

Providers les plus connus :



Provider	SMTP server domain	port
Gmail	smtp.gmail.com	587
Outlook	smtp-mail.outlook.com	587
Yahoo	smtp.mail.yahoo.com	465 or 587

</p>

01 : Suite; {

<p Vous aurez besoin de cette création du objet (**smtpObj**). pour appeler les méthodes qui vous connectent et envoient des e-mails. Si l'appel **smtplib.SMTP()** échoue, votre serveur **SMTP** peut ne pas prendre en charge **TLS** sur le port **587**. Dans ce cas, vous devrez créer un objet **SMTP** en utilisant **smtplib.SMTP_SSL()** et le port **465** à la place de **587**. >

```
>>> smtpObj = smtplib.SMTP_SSL("SMTP server domain", port)
```

</p>

}

02 : SMTP "Hello" Message; {

'Ehlo Method'

<p pour établir une connexion au serveur et éviter les erreurs lors des appels des méthodes suivants. vous devez appeler la méthode **ehlo()** pour dire "Bonjour" au serveur **SMTP**. >

Pour faire ca :

```
>>> smtpObj.ehlo()
```

return value :

Si le premier élément du tuple renvoyé est l'entier 250
(le code pour "succès" dans SMTP)

</p>

}

03 : Démarrage du chiffrement TLS; {

'TLS Encryptions'

Si vous vous connectez au port **587** sur le serveur **SMTP** (c'est-à-dire que vous utilisez le cryptage **TLS**), vous devrez ensuite appeler la méthode **starttls()**. Cette étape obligatoire active le cryptage de votre connexion.

Si vous vous connectez au port **465** (à l'aide de **SSL**), le cryptage est déjà configuré et vous devez ignorer cette étape.

appeler starttls() :

```
>>> smtpObj.starttls() #(220, b'2.0.0 Ready to start TLS')
```

return value :

Le 220 dans la valeur de retour vous indique que le serveur est prêt.

</p>

04 : Connexion au serveur SMTP; {

'Login to your account'

<p Une fois votre connexion cryptée au serveur **SMTP** configurée, vous pouvez vous connecter avec votre nom d'utilisateur (généralement votre adresse e-mail) et votre mot de passe e-mail en appelant la méthode **login()**.

```
>>> smtpObj.login('my_email_address@gmail.com', 'MY_SECRET_PASSWORD')  
(235, b'2.7.0 Accepted')
```

Passez une *Str* de votre adresse e-mail comme premier argument et un *Str* de votre mot de passe comme deuxième argument. Le 235 dans la valeur de retour signifie que l'authentification a réussi.

Python lèvera une exception **smtplib.SMTP AuthenticationError** pour les mots de *pas*se incorrects.

}

</p>

Deux dernières étapes {

< Une fois que vous êtes connecté au serveur **SMTP** de votre email provider, il ne reste plus que ces deux étapes à terminer. >

< /1 > * Envoi d'un e-mail

< /2 > * Déconnexion du serveur **SMTP**

}

Envoi des emails; {

1
2 'sendmail() method'

3
4 >>> smtpObj.sendmail('sender_email', 'receiver_email', 'message')

5 La méthode **sendmail()** requiert trois arguments:

6
7 # Votre e-mail sous forme Str (pour email "from").

8
9 # L'e-mail du destinataire sous forme Str ou liste de Str pour
10 plusieurs destinataires (pour l'adresse "to").

11 # email message sous forme *Str* :

12 * Le début du mail message doit commencer par " **Sujet :** \n "
13 pour la ligne d'objet de l'email. Le " \n " sépare la
14 ligne d'objet du message principal de l'email.
15 }

Déconnexion du serveur SMTP; {

'quit() method'

<p Assurez-vous d'appeler la méthode **quit()** lorsque vous avez terminé d'envoyer des e-mails. Cela déconnectera votre programme du serveur **SMTP**. >

```
>>> smtpObj.quit()
```

```
(221, b'2.0.0 closing connection ko10sm23097611pbd.52 - gsmtplib')
```

<p Le 221 dans la valeur de retour signifie que la session se termine. >

}

03 {

[Présentation du projet]

< Code Source
Et exemple d'exécution >

}

Les outils utilisés 'In Code' {

Bibliothèque :

smtplib  50%


< **smtplib** utilisée pour
envoyer des emails à un
machine internet. >

Time & Os  05%

< 1er pour gestion du
temps.
2ème pour les commandes
système. >

fct  40%

< **fichie.py** contient les
fonction utilisée dans
le code main. >

random  05%

< Ce module implémente des
générateurs de valeurs
aléatoires pour diverses
distributions. >

}

Les fonctions utilisée; {

`def info_verf()`: fonction pour entrer votre *'login account'* informations et vérifier votre accès au programme. Prend aucune argument en entrer et return l'email et mot de pass.

`def ver_server_statu(a)`: fonction vous informer que le serveur est prêt. Prend en entrer un entier est ne return rien.

`def email_list()`: cette fonction return une dictionnaire des emails, déjà stocké.

`def tasks_list()`: cette fonction return la liste des Taches, déjà stocké.

```
1 Suite 01; {
2
3     def assign_task() : # fonction pour donner chaque member une tache aléatoire.
4
5     # Explication du fonction :
6
7     * Tasks = tasks_list() : # pour extraire la liste des taches.
8
9     * for email in emails.keys(): # emails est un dictionnaire contenant
10                                     # des informations sur les membres.
11         random_task = random.choice(Tasks) ← # Tirer une Tâche au hasard avec
12                                             random.choice()
13         email_dict[email] = random_task ← # stocker Cette tâche dans un
14                                             dictionnaire avec l'email de member
15                                             comme key.
16     Tasks.remove(random_task) # supprimer la tâche pour éviter la redondance.
17 }
```

```
1 Suite 02; {
2     for email in email_dict:
3         message = str(
4             "Subject: Classe Tache\n"
5             + "Hi {} {},\n".format(emails[email][0], emails[email][1])
6             + str_msg
7             + email_dict[email]
8         )
9         print(
10             emails[email][0],
11             emails[email][1] + " \t\tVotre tache est : \t\t "
12             + email_dict[email],
13         )
14         smtpObj.sendmail(admin_user, email, message)
15     } # Voir le code source pour bien comprendre
```

```
1 Suite 03; {
2
3
4     # planifier le programme pour envoyer les taches aux membres chaque
5     semaine
6
7     for i in range(52):    # 52 pour nombre de semaine dans l'année
8         assign_task()
9         time.sleep(7 * 86400) # 86400s dans 1 jour
10
11     # Déconnexion du serveur SMTP
12     smtpObj.quit()
13
14 }
```

Recommendations; {

SMTP



< install full library to avoid any rise of error. "Type **pip install smtplib** in your code editor terminal" >

Account info



< Use your correct email and password to *login*. if still rise error enable "*less app secure*" in your account >

Same Directory



< when you download the file of programme make sure tha all file is in the *same directory* >

}


```
1 Thanks; {
```

```
2  
3 'Do you have any questions?'
```

```
4 mesrarhamza48@gmail.com
```

```
5 +212 633-099608
```

```
6 ez7mz
```

```
7  
8  
9 Find me in :
```



```
10  
11 CREDITS: This presentation was created by  
12 MESRAR HAMZA, including icons and  
13 infographics & images by Freepik & Slidesgo  
14
```

```
}  
14
```