

# Overview of Sending Emails

The smtplib library allows you to manually go through the steps of creating and sending an email in Python:

In [34]:
```
import smtplib
```

Create an SMTP object for a server. Here are the main Server Domain Name for the top email services. If you don't see your email server here, you may need to do a quick Google Search to see if there SMTP server domain name is available:

| Provider | SMTP server domain name |
|---|---|
| Gmail (will need App Password) | smtp.gmail.com |
| Yahoo Mail | smtp.mail.yahoo.com |
| Outlook.com/Hotmail.com | smtp-mail.outlook.com |
| AT&T | smpt.mail.att.net (Use port 465) |
| Verizon | smtp.verizon.net (Use port 465) |
| Comcast | smtp.comcast.net |

Next is to create an STMP object that can make the method calls to log you in to your email in order to send messages. Notice how also specify a port number. If the number 587 does not work on your computer, try using 465 instead. Keep in mind, a firewall or antivirus may prevent Python from opening up this port, so you may need to disable it on your computer.

In [35]:
```
smtp_object = smtplib.SMTP('smtp.gmail.com',587)
```

Next we run the ehlo() command which "greets" the server and establishes the connection. This method call should be done directly after creating the object. Calling it after other methods may result in errors in connecting later on. The first item in the tuple that is returned should be 250, indicating a successful connection.

In [36]:
```
smtp_object.ehlo()
```

Out[36]:
```
(250,
 b'smtp.gmail.com at your service, [47.143.81.4]\nSIZE 35882577\n8BITMIME\nSTARTTLS\nENHANCED
STATUSCODES\nPIPELINING\nCHUNKING\nSMTPUTF8')
```

When using the 587 port, this means you are using TLS encryption, which you need to initiate by running the starttls() command. If you are using port 465, this means you are using SSL and you can skip this step.

In [37]:
```
smtp_object.starttls()
```

Out[37]:
```
(220, b'2.0.0 Ready to start TLS')
```

Now its time to set up the email and the passwords. You should never save the raw string of your password or email in a script, because anyone that sees this script will then be able to see you email and password! Instead you should use input() to get that information. If you also don't want your password to be visible when typing it in, you can use the built-in **getpass** library that will hide your password as you type it in, either with asterisks or by just keeping it invisible.

In [38]:
```
# For hidden passwords
import getpass
```

In [39]:
```
result = getpass.getpass("Type something here and it will be hidden: ")
```

```
Type something here and it will be hidden: ········
```

In [40]:
```
# Just keep in mind that its still visible as an object internally:
result
```

Out[40]:
```
'a'
```

In [41]:
```
# Or just use input()
input("Enter your password")
```

```
Enter your passwords
```

Out[41]:
```
's'
```

---

**Note for Gmail Users, you need to generate an app password instead of your normal email password. This also requires enabling 2-step authentication. Follow the instructions here to set-up 2-Step Factor Authentication as well as App Password Generation:https://support.google.com/accounts/answer/185833?hl=en/. Set-up 2 Factor Authentication, then create the App Password, choose Mail as the App and give it any name you want. This will output a 16 letter password for you. Pass in this password as your login password for the smtp.**

---

In [ ]:
```
email = getpass.getpass("Enter your email: ")
password = getpass.getpass("Enter your password: ")
smtp_object.login(email,password)
```

Now we can send an email using the .sendmail() method.

In [46]:
```
from_address = getpass.getpass("Enter your email: ")
to_address = getpass.getpass("Enter the email of the recipient: ")
subject = input("Enter the subject line: ")
message = input("Type out the message you want to send: ")
msg = "Subject: " + subject + '\n' + message
smtp_object.sendmail(from_address,to_address,msg)
```

```
Enter your email: ••••••••
Enter the email of the recipient: ••••••••
Enter the subject line: This is a test
Type out the message you want to send: Here is the message.
```

Out[46]:
```
{}
```

If you get back an empty dictionary, then the sending was successful.

You can then close your session with the .quit() method.

In [47]:
```
smtp_object.quit()
```

Out[47]:
```
(221, b'2.0.0 closing connection j1sm22376227pgq.33 - gsmtp')
```

Now that we know how to send emails, its time to learn how to look through emails you've already recieved.