Universidad Gerardo Barrios

# Bloque 2- Gestión de librerías y entornos virtuales

## Gestión de correos.

**Ingeniería en Sistemas y Redes Informáticas**

Ing. Willian Alexis Montes Girón - Semana 9

# Competencia de la asignatura

Construye aplicaciones utilizando el lenguaje de programación Python que brinden soluciones en las áreas de escritorio, web y machine learning, para dar respuesta a las necesidades de las empresas y sociedad en general, utilizando herramientas vigentes, uso de estandares, buenas practicas, tecnologías como Git, Devops, Containers y Orquestadores, trabajando de forma individual y colaborativa.

# Criterio de evaluación del Bloque

Construye aplicaciones con Python haciendo uso de diferentes librerías de machine learning necesaria para dar solución a las necesidades empresariales trabajando de forma individual o colaborativa

# Contenidos

1 SMTP

2 SMTPLIB.

Contenido
uno

# SMTP

# Protocolo SMTP y SMTPS

SMTP significa protocolo simple de transferencia de correo. Se trata de un protocolo de comunicación que se utiliza para enviar y recibir mensajes de correo electrónico a través de Internet. Los servidores de correo y otros agentes de transferencia de mensajes (MTA) utilizan SMTP para enviar, recibir y reenviar mensajes de correo. El protocolo simple de transferencia de correo seguro (SMTPS) es un método para proteger el SMTP mediante la seguridad de la capa de transporte

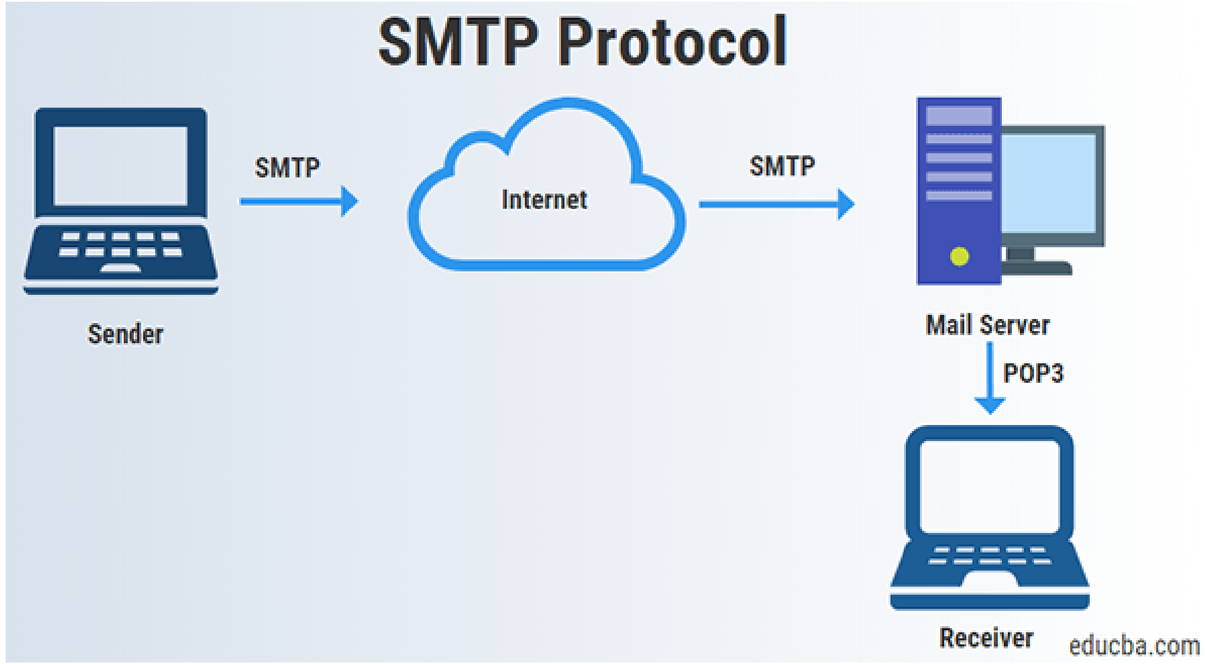

# Servidor smtp

Un servidor SMTP, también conocido como servidor de correo saliente, es una computadora o un software que administra los mensajes del correo electrónico salientes.

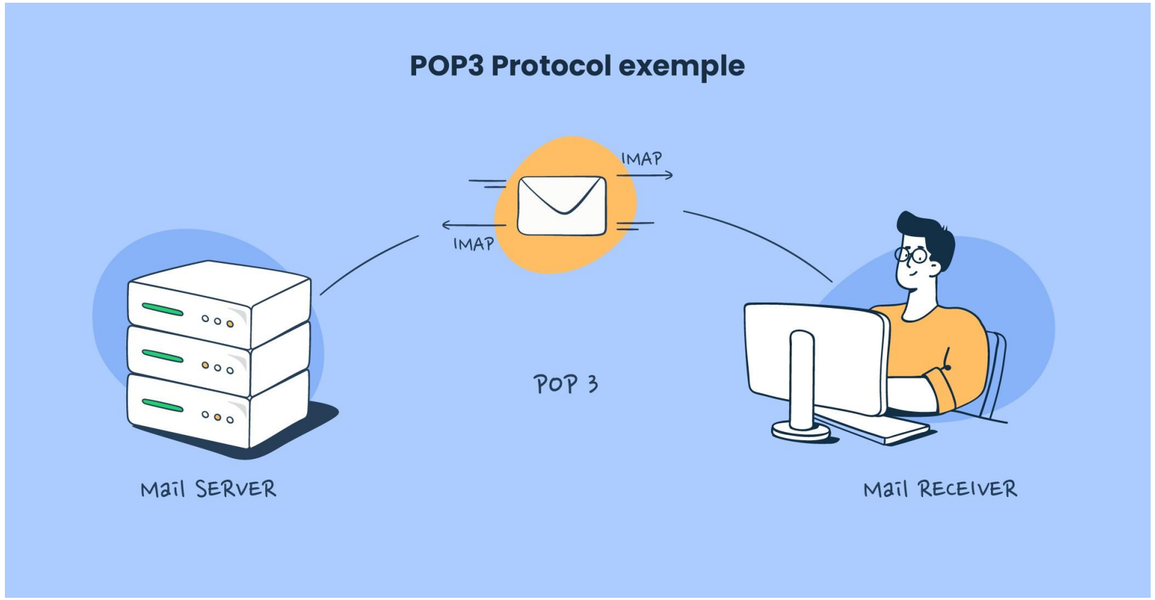
Un servidor SMTP se refiere específicamente al componente del servidor de correo que utiliza el protocolo simple de transferencia de correo (SMTP) para enviar correos salientes. Mientras que el servidor de correo administra los correos electrónicos entrantes y salientes, el servidor SMTP se ocupa solo de enviar y retransmitir los correos electrónicos salientes a los destinos apropiados.

# ¿Cómo funciona SMTP?

En el modelo del protocolo simple de transferencia de correo (SMTP), el cliente o el servidor de correo electrónico del remitente actúa como cliente SMTP, y el servidor de correo electrónico del remitente actúa como servidor SMTP. Este cliente inicia una conexión con el servidor y transmite el correo electrónico completo con los detalles del destinatario, el asunto y el cuerpo. El servidor procesa este correo electrónico y determina el próximo servidor adecuado en función de la dirección del destinatario



# Protocolo POP3



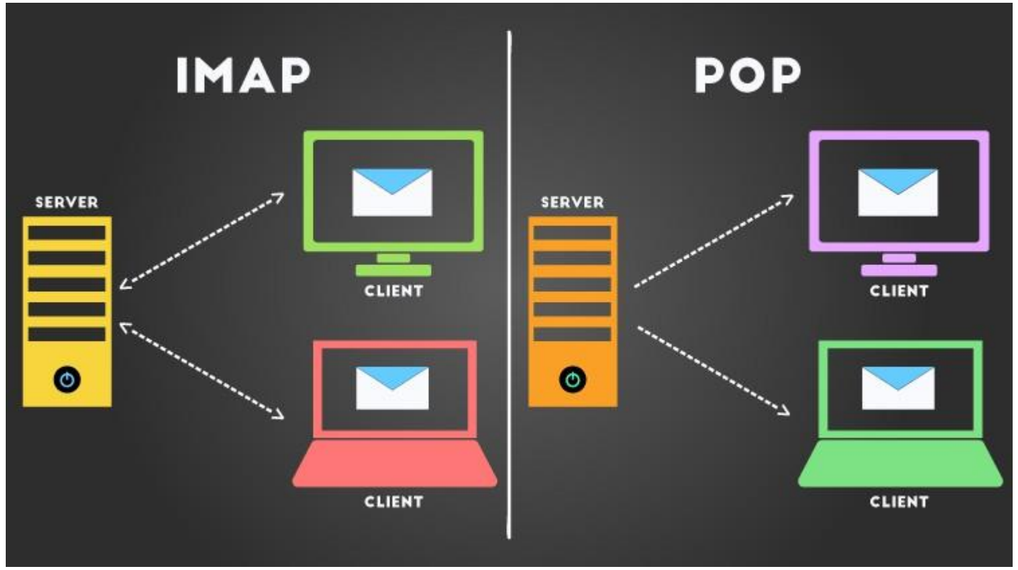

POP3 (Post Office Protocol versión 3) es un protocolo de correo entrante unidireccional que descarga una copia de los mensajes desde un servidor de correo electrónico a una máquina local. Una vez que el protocolo de correos completa el proceso, elimina los datos originales de la bandeja de entrada del servidor.

Sin embargo, hoy en día muchos proveedores dan la opción de mantener las copias originales intactas, lo que permite a los usuarios ver el mismo contenido cuando acceden a los mensajes desde una plataforma diferente

# Protocolo IMAP

A diferencia de POP3, IMAP (Internet Message Access Protocol) es un protocolo de correo entrante bidireccional que sólo descarga los encabezados del correo electrónico en lugar de su contenido completo.

Como resultado, los mensajes de correo electrónico reales se mantienen en el servidor después de ser recuperados para su visualización, haciéndolos accesibles desde otra plataforma. Este protocolo también sincroniza los cambios realizados en el cliente de correo electrónico con el servidor, de ahí la comunicación bidireccional.

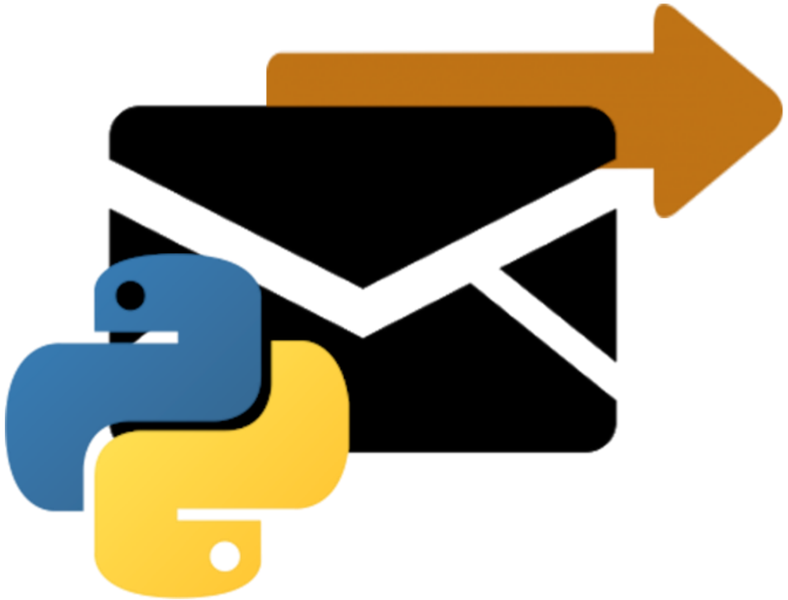


Contenido dos

SMTPLIB

# SMPTLIB

smtplib es un módulo incluido en la biblioteca estándar de Python que permite a los desarrolladores enviar correos electrónicos utilizando el protocolo SMTP (Simple Mail Transfer Protocol), el cual es un estándar para la transmisión de correos electrónicos a través de Internet. Este módulo proporciona las herramientas necesarias para conectar un programa escrito en Python a un servidor SMTP y, a través de ese servidor, enviar correos electrónicos a otros servidores de correo o cuentas de correo electrónico..



# Protocolo TLS

Transport Layer Security, o TLS, es un protocolo de seguridad ampliamente adoptado, diseñado para facilitar la privacidad y la seguridad de los datos en las comunicaciones por Internet. Un caso de uso primario de TLS es la encriptación de las comunicaciones entre aplicaciones web y servidores, como los navegadores que cargan un sitio web. TLS también puede usarse para encriptar otras comunicaciones como el correo electrónico, los mensajes y la voz sobre IP (VoIP).

# Ejemplo de código

Python, a diferencia de programas como Outlook o Thunderbird que cuentan con interfaces gráficas amigables, requiere que interactuemos directamente con el protocolo SMTP mediante funciones específicas, como se puede ver en el siguiente ejemplo interactivo de un shell Python.

```
>>> import smtplib
>>> smtpObj = smtplib.SMTP('smtp.example.com', 587)
>>> smtpObj.ehlo()
(250, b'mx.example.com at your service, [216.172.148.131]\nSIZE 35882577\
n8BITMIME\nSTARTTLS\nENHANCEDSTATUSCODES\nCHUNKING')
>>> smtpObj.starttls()
(220, b'2.0.0 Ready to start TLS')
>>> smtpObj.login('bob@example.com', 'MY_SECRET_PASSWORD')
(235, b'2.7.0 Accepted')
>>> smtpObj.sendmail('bob@example.com', 'alice@example.com', 'Subject: So
long.\nDear Alice, so long and thanks for all the fish. Sincerely, Bob')
{}
>>> smtpObj.quit()
(221, b'2.0.0 closing connection ko10sm23097611pbd.52 - gsmtp')
```

# Seleccionar el servidor

Cada proveedor de correo electrónico tiene sus propias configuraciones para poder hacer uso de sus servicios, se encuentran fácilmente en la web pero también está la siguiente tabla de referencia:

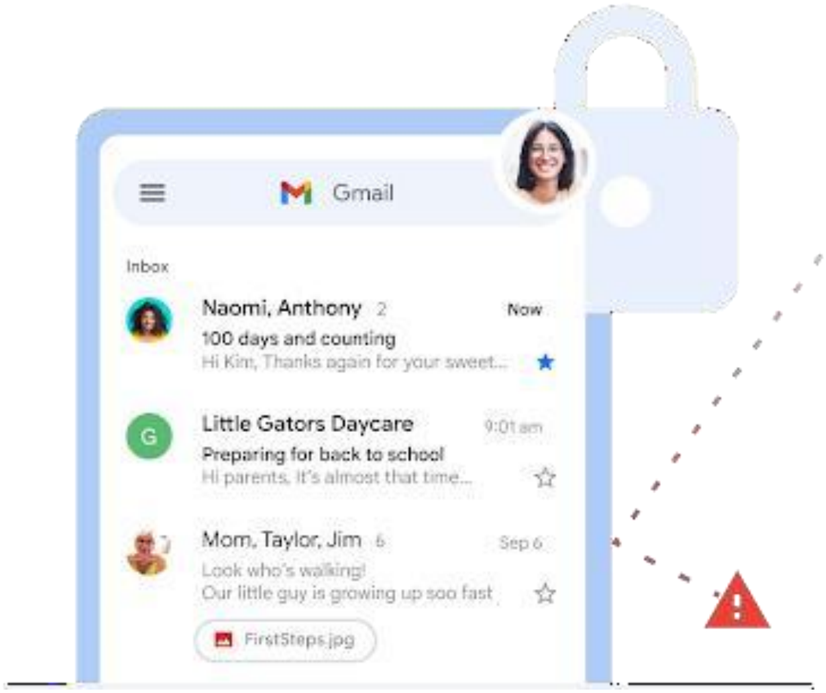
| Provider | SMTP Server | SMTP Port | SMTP Host Name | Use TLS | Use Start TLS |
|---|---|---|---|---|---|
| Plasec Brinkster | mymail.brinkster.com | 2525 | leave blank/empty | no | no |
| Yahoo Mail | smtp.mail.yahoo.com | 587 | leave blank/empty | no | no |
| Hotmail | smtp.live.com | 587 | leave blank/empty | yes | yes |
| Google Gmail | smtp.gmail.com [or smtp-relay.gmail.com if enabled within client's Gmail account] | 587 | leave blank/empty | no | yes |
| Microsoft Office 365 | smtp.office365.com | 587 | leave blank/empty | no | yes |

# Creando y configurando la cuenta de correo

Puedes utilizar la cuenta de correo de tu preferencia. Por ejemplo para configurar una cuenta de Gmail puedes crear una exclusivamente para enviar correos.

Para configurar una dirección de Gmail para probar tu código, realiza lo siguiente:

- Crea una nueva cuenta de Google.
- Activa la opción "Permitir aplicaciones menos seguras". Ten en cuenta que esto facilita que otros puedan acceder a tu cuenta.
- En el caso que uses la autenticación en dos pasos deberás configurar una contraseña para aplicaciones



# Creando el objeto smtp

Una vez que tengas el nombre de dominio y la información del puerto de tu proveedor de correo electrónico, crea un objeto SMTP llamando a smtplib.SMTP(), pasando el nombre de dominio como argumento de tipo cadena, y el puerto como argumento de tipo entero. El objeto SMTP representa una conexión a un servidor de correo SMTP y tiene métodos para enviar correos electrónicos


```
>>> smtpObj = smtplib.SMTP('smtp.gmail.com', 587)
>>> type(smtpObj)
<class 'smtplib.SMTP'>
```

# “Hello” al servidor

Una vez que tengas el objeto SMTP, llama a su método, curiosamente llamado ehlo(), para "saludar" al servidor de correo SMTP. Este saludo es el primer paso en el protocolo SMTP y es importante para establecer una conexión con el servidor. No necesitas conocer los detalles específicos de estos protocolos. Solo asegúrate de llamar al método ehlo() como el primer paso después de obtener el objeto SMTP, de lo contrario, las llamadas a los métodos posteriores resultarán en errores.

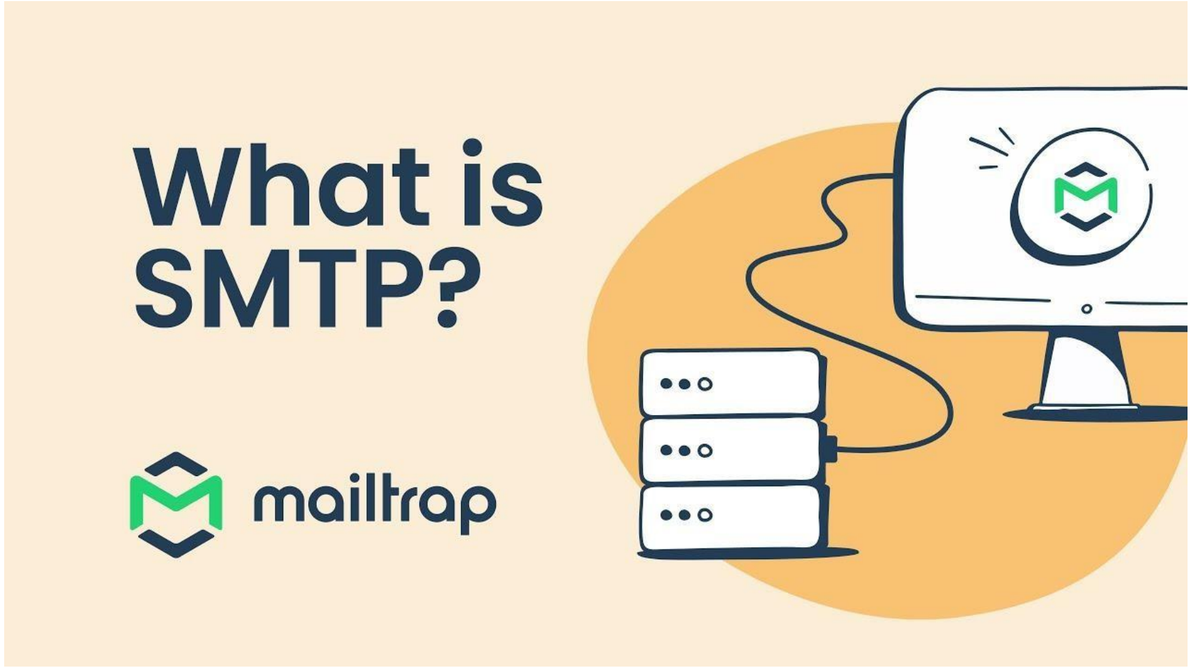

```
>>> smtpObj.ehlo()
(250, b'mx.google.com at your service, [216.172.148.131]\nSIZE 35882577\
n8BITMIME\nSTARTTLS\nENHANCEDSTATUSCODES\nCHUNKING')
```

# Iniciando una conexión SMTP segura

Cuando envías correos electrónicos a través de Python, debes asegurarte de que tu conexión SMTP esté encriptada, para que tu mensaje y las credenciales de inicio de sesión no sean fácilmente accesibles por otros. SSL (Secure Sockets Layer) y TLS (Transport Layer Security) son dos protocolos que se pueden utilizar para encriptar una conexión SMTP.

Existen dos formas de iniciar una conexión segura con tu servidor de correo electrónico:

- Iniciar una conexión SMTP que esté segura desde el principio usando SMTP_SSL().
- Iniciar una conexión SMTP no segura que luego puede ser encriptada usando .starttls().



# Iniciando el encriptado y el login

Si te estás conectando al puerto 587 en el servidor SMTP (es decir, estás utilizando encriptación TLS) con Gmail, necesitarás llamar al método starttls() a continuación. Este paso es necesario para habilitar la encriptación en tu conexión. Si te estás conectando al puerto 465 (utilizando SSL), la encriptación ya está configurada y deberías omitir este paso.

Una vez que tu conexión encriptada con el servidor SMTP esté configurada, puedes iniciar sesión con tu nombre de usuario (generalmente tu dirección de correo electrónico) y tu contraseña de correo llamando al método login().

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

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

# Mandando un email

Después de haber iniciado una conexión SMTP segura puedes enviar tu correo electrónico utilizando .sendmail(), que básicamente hace lo que su nombre indica: enviar el correo.

```python
import smtplib, ssl

port = 465  # For SSL
smtp_server = "smtp.gmail.com"
sender_email = "my@gmail.com"  # Enter your address
receiver_email = "your@gmail.com"  # Enter receiver address
password = input("Type your password and press enter: ")
message = """\
Subject: Hi there

This message is sent from Python."""

context = ssl.create_default_context()
with smtplib.SMTP_SSL(smtp_server, port, context=context) as server:
    server.login(sender_email, password)
    server.sendmail(sender_email, receiver_email, message)
```

La cadena del mensaje comienza con "Subject: Hi there" seguida de dos saltos de línea (\n). Esto asegura que "Hi there" aparezca como el asunto del correo electrónico, y el texto que sigue a los saltos de línea será tratado como el cuerpo del mensaje.

# Mandando correos con html

Si deseas dar formato al texto en tu correo electrónico (negrita, cursiva, etc.), o si deseas agregar imágenes, enlaces o contenido interactivo, el HTML es muy útil. El tipo más común de correo electrónico hoy en día es el correo Multipart MIME (Extensiones Multipropósito de Correo de Internet), que combina HTML y texto plano. Los mensajes MIME son manejados por el módulo email.mime de Python.

Dado que no todos los clientes de correo electrónico muestran contenido HTML de forma predeterminada, y algunas personas eligen recibir solo correos electrónicos en texto plano por razones de seguridad, es importante incluir una alternativa en texto plano para los mensajes en HTML.

```
import smtplib, ssl
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

sender_email = "my@gmail.com"
receiver_email = "your@gmail.com"
password = input("Type your password and press enter:")

message = MIMEMultipart("alternative")
message["Subject"] = "multipart test"
message["From"] = sender_email
message["To"] = receiver_email
```

# Mandando archivos adjuntos

Para enviar archivos binarios a un servidor de correo electrónico que está diseñado para trabajar con datos textuales, estos deben ser codificados antes del transporte. Esto se hace comúnmente utilizando base64, que codifica los datos binarios en caracteres ASCII imprimibles.

```
import email, smtplib, ssl

from email import encoders
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

subject = "An email with attachment from Python"
body = "This is an email with attachment sent from Python"
sender_email = "my@gmail.com"
receiver_email = "your@gmail.com"
password = input("Type your password and press enter:")

# Create a multipart message and set headers
message = MIMEMultipart()
message["From"] = sender_email
message["To"] = receiver_email
message["Subject"] = subject
message["Bcc"] = receiver_email  # Recommended for mass emails

# Add body to email
message.attach(MIMEText(body, "plain"))

filename = "document.pdf"  # In same directory as script
```

# Referencias bibliograficas


- Fernandez Montoro, A. (2012). Python 3 al descubierto (1ra ed.). Madrid, España: Alfaomega.
- Ortega Candel, J. M. (2018). Hacking ético con herramientas Python. Madrid, España: Ra-ma.
- Matthes, E. (2016). Python crash course: A hands-on, project-based introduction to programming. San Francisco, CA: No Starch Press.
- Sweigart, A. (2015). Automate the Boring Stuff with Python. San Francisco, CA: No Starch Press.
- Beazley, D., & Jones, B. K. (2013). Python Cookbook (3rd ed.). Sebastopol, CA: O'Reilly Media, Inc.
- Martelli, A., Ravenscroft, A. M., & Holden, S. (2017). Python in a Nutshell (3rd ed.). Sebastopol, CA: O'Reilly Media, Inc
- . Ortega Candel, J. M. (2022). Big data, machine learning y data science en Python: (1 ed.). RA-MA Editorial.

# Recursos de la unidad

| Título | Especificaciones | Enlace | Revisión |
|---|---|---|---|
| smtplib de Python | Documentación oficial del módulo smtplib de Python<br>Tiempo estimado: 40 minutos | https://docs.python.org/es/3/library/smtplib.html | Obligatoria |
| Sending emails with Python | Artículo en línea con un ejemplo de cómo utilizar el protocolo para mandar un email<br>Tiempo estimado: 30 minutos | https://realpython.com/python-send-email/ | Obligatoria |
| Enviar correos con Python usando smtplib | Video explicativo acerca del uso de smtplib<br>Tiempo estimado: 15 minutos | https://www.youtube.com/watch?v=kVKagJrZCyU | Opcional |
| | | | |

# Actividad de la semana

| | |
|---|---|
| **Nombre de la actividad y porcentaje (aplica a sumativas)** | Guía de ejercicios prácticos (Actividad Formativa) |
| **Sesión** | Presencial |
| **Forma de apropiación** | Colaborativa |
| **Indicaciones** | El estudiante con los conocimientos adquiridos deberá resolver una serie de ejercicios con lo que se ha visto hasta el momento |
| **Criterio de evaluación del bloque** | Conoce sintaxis básica de Python y la utiliza para resolver problemas sencillos mediante el estudio de estructuras claves de control y colecciones de datos, de forma individual o colaborativa |
| **Indicadores de evaluación** | No aplica |
| **Fecha de entrega** | Domingo 21 de Septiembre 23:59 p.m. |


UGB
Universidad
Gerardo Barrios