2024

SMTP-IMAP

NAME:YEHIAHANYALI
ID:7667

Code:

```
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import *
from PyQt5 import uic
import smtplib
from email import encoders
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
import imaplib
import email
from email.header import decode_header
import sys
from PyQt5.QtCore import QTimer
from plyer import notification


globalimap_server = None
globemail_address = None
globepassword = None
globesmtp_server = None


class EmailChecker:
    def __init__(self, email, password, imap_server):
```

```
        self.email = email
        self.password = password
        self.imap_server = imap_server
        self.imap = imaplib.IMAP4_SSL(self.imap_server)
        self.imap.login(self.email, self.password)
        self.latest_email_count = self.get_latest_email_count()

    def get_latest_email_count(self):
        self.imap.select("INBOX")
        status, msgnums = self.imap.search(None, "ALL")
        return len(msgnums[0].split())

    def check_emails(self):
        current_email_count = self.get_latest_email_count()
        if current_email_count > self.latest_email_count:
            notification.notify(
                title='New Email Notification',
                message='You have received a new email.',
                app_icon=None,  # You can set an icon path here
if needed
                timeout=10,  # Notification will automatically
close after 10 seconds
            )
            self.latest_email_count = current_email_count
```

```
class MyGUICHECK(QMainWindow):
    def __init__(self):
        super(MyGUICHECK,self).__init__()
        uic.loadUi("check.ui",self)
        self.show()
        global globalimap_server
        print (globalimap_server )
        imap_server = globalimap_server
        email_address = globemail_address
        password = globepassword

        # Connect to the IMAP server
        imap = imaplib.IMAP4_SSL(imap_server)

        # Login to the email account
        imap.login(email_address, password)
        imap.select("INBOX")

        # Search for the latest email
        status, msgnums = imap.search(None, "ALL")
        latest_email = msgnums[0].split()[-1]  # Get the ID of
the latest email
```

```
        # Fetch the latest email data
        status, data = imap.fetch(latest_email, "(RFC822)")
        raw_email = data[0][1]

        # Parse the email
        message = email.message_from_bytes(raw_email)

        # Get email headers
        sender = message.get('From', '')
        receiver = message.get('To', '')
        bcc = message.get('BCC', '')
        date = message.get('Date', '')
        subject = message.get('Subject', '')

        # Decode email headers if needed
        sender = decode_header(sender)[0][0] if
isinstance(sender, str) else sender
        receiver = decode_header(receiver)[0][0] if
isinstance(receiver, str) else receiver
        bcc = decode_header(bcc)[0][0] if isinstance(bcc, str)
else bcc
        subject = decode_header(subject)[0][0] if
isinstance(subject, str) else subject
```

```
        print(f"Message Number: {latest_email}")
        print(f"From: {sender}")
        print(f"To: {receiver}")
        print(f"BCC: {bcc}")
        print(f"Date: {date}")
        print(f"Subject: {subject}")
        self.fromlabel.setText("From:" + ' ' + sender)
        self.fromlabel.update()  # Explicitly update GUI
        self.to.setText("To:"+ ' ' +receiver)
        self.bcc.setText("BCC:"+ ' ' + bcc)
        self.date.setText("Date:" + ' ' + date)
        self.subject.setText( "Subject:" + ' ' +subject)

        print("Content:")
        # Walk through the email parts to find text/plain
content
        for part in message.walk():
            if part.get_content_type() == "text/plain":
                content = part.get_payload(decode=True)
                charset = part.get_content_charset()
                if charset:
                    content = content.decode(charset)
                else:
```

```
                content = content.decode()  # Use default
decoding
            print(content)
            self.mailText.setText(content)


        # Close the connection

        self.loginButton_3.clicked.connect(self.login_3)
    def login_3(self):
        print('here')
        widget.setCurrentIndex(1)
        widget.removeWidget(widget.widget(3))
        widget.setFixedHeight(170)
        widget.setFixedWidth(350)



class MyGUI(QMainWindow):
    def __init__(self):
        super(MyGUI,self).__init__()
        uic.loadUi("mailgui.ui",self)
        self.show()
        global globesmtp_server
        self.server = globesmtp_server
```

```
        # Basic msg that we are going to attach stuff to
        self.msg = MIMEMultipart()
        self.attachButton.clicked.connect(self.attach_sth)
        self.sendButton.clicked.connect(self.send_mail)
        self.loginButton_3.clicked.connect(self.login_3)
    def login_3(self):
            print('here')
            widget.setCurrentIndex(1)
            widget.setFixedHeight(170)
            widget.setFixedWidth(350)

    def attach_sth(self):
        # Files dialog options
        options = QFileDialog.Options()
        # File names for multiple files
        filenames, _ = QFileDialog.getOpenFileNames(self,
"Open File", "", "All Files (*.*)", options=options) #(*.*)
all file types
        if filenames != []:
            for filename in filenames:
                # Open file in reading bytes mode
                attachment = open(filename, 'rb')
                # Search for file name from right to left
                filename = filename[filename.rfind("/")+1:]
```

```
            p = MIMEBase('application', 'octet-stream')
            # Read the bytes then encode it
            p.set_payload(attachment.read())
            encoders.encode_base64(p)
            p.add_header("Content-Disposition",
f"attachment; filename={filename}")
            self.msg.attach(p)
            if not self.attachments.text().endswith(":"):
                self.attachments.setText(self.attachments.text()
+ ",")
            self.attachments.setText(self.attachments.text() +
" " + filename)

    def send_mail(self):
        dialog = QMessageBox()
        dialog.setText("Do you want to send this email?")
        dialog.addButton(QPushButton("Yes"),QMessageBox
.YesRole) # 0
        dialog.addButton(QPushButton("No"),QMessageBox.
NoRole) # 1

        if dialog.exec_() == 0:
            try:
                global globemail_address
```

```
            self.msg['From'] = globemail_address
            self.msg['To'] = self.toText.text()
            self.msg['Subject'] = self.subjectText.text()
            self.msg.attach(MIMEText(self.mailText.toPlain
Text(), 'plain'))
            text = self.msg.as_string()
            self.server.sendmail(globemail_address,
self.toText.text(), text)
            message_box = QMessageBox()
            message_box.setText("Mail Sent!")
            message_box.exec()
        except:
            message_box = QMessageBox()
            message_box.setText("Sending Mail Failed!")
            message_box.exec()
        # except Exception as e:
        #     print("Error:", e)

class MyGUIMAIN(QMainWindow):
    def __init__(self):
        super(MyGUIMAIN,self).__init__()
        uic.loadUi("Main.ui",self)
        self.show()
        self.loginButton.clicked.connect(self.login)
```

```
    def login(self):
        try:
            # Initialize server with server and port number basic
setup
            self.server = smtplib.SMTP(self.serverValue.text(),
int(self.portValue.text()))
            global globalimap_server
            globalimap_server = self.imapValue.text()
            # Check if server works
            self.server.ehlo()
            # For encryption and certificate
            self.server.starttls()
            # Make sure it works
            self.server.ehlo()
            # Start with login now
            print(self.emailText.text())
            print(self.passwordText.text())
            print(self.serverValue.text())
            print(self.portValue.text())
            print(self.imapValue.text())
            global globemail_address
            global globepassword
            globemail_address = self.emailText.text()
            globepassword = self.passwordText.text()
```

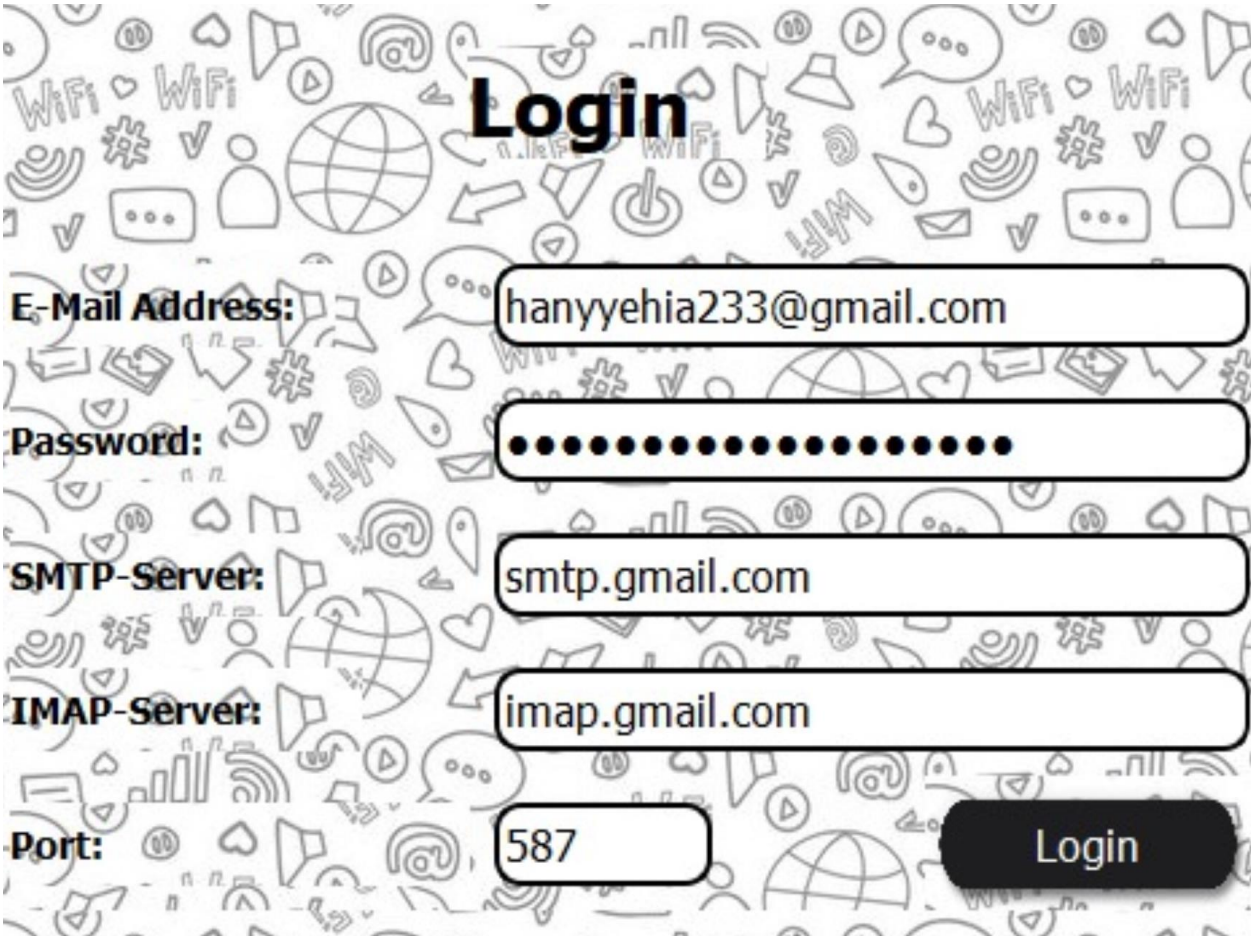
```
            self.server.login(self.emailText.text(),
self.passwordText.text())
            global globesmtp_server
            globesmtp_server = self.server
            widget.addWidget(MyGUISELECT())
            widget.setCurrentIndex(widget.currentIndex() + 1)
            widget.setFixedHeight(170)
            widget.setFixedWidth(350)
            widget.addWidget(MyGUI())

        except smtplib.SMTPAuthenticationError:
            message_box = QMessageBox()
            message_box.setText("Invalid Login Info!")
            message_box.exec()
        except:
            message_box = QMessageBox()
            message_box.setText("Login Failed!")
            message_box.exec()


class MyGUISELECT(QMainWindow):
    def __init__(self):
        super(MyGUISELECT,self).__init__()
```

```
        uic.loadUi("test.ui",self)
        self.show()
        print("here")
        self.loginButton.clicked.connect(self.login)
        self.loginButton_2.clicked.connect(self.login_2)
        self.loginButton_3.clicked.connect(self.login_3)
        self.email_checker =
EmailChecker(email=globemail_address,
                                    password=globepassword,
                                    imap_server=globalimap_server)
        self.timer = QTimer(self)
        self.timer.timeout.connect(self.check_emails_periodic
ally)
        self.timer.start(60000)  # Check for new emails every
60 seconds


    def check_emails_periodically(self):
        self.email_checker.check_emails()
    def login_3(self):
            print('here')
            widget.setCurrentIndex(0)
            widget.setFixedHeight(370)
            widget.setFixedWidth(480)
```

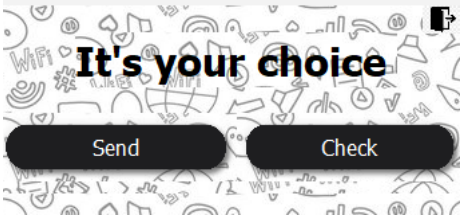
```
    def login(self):
        print('here')
        widget.setCurrentIndex(2)
        widget.setFixedHeight(880)
        widget.setFixedWidth(780)
    def login_2(self):
        print('here')
        widget.addWidget(MyGUICHECK())
        widget.setCurrentIndex(3)
        widget.setFixedHeight(750)
        widget.setFixedWidth(780)

app = QApplication([])
widget = QtWidgets.QStackedWidget()

window = MyGUIMAIN()
```

```
widget.addWidget(window)

print(widget.currentIndex() )
widget.setCurrentIndex(0)
widget.setFixedHeight(370)
widget.setFixedWidth(480)
widget.show()
app.exec_()
```

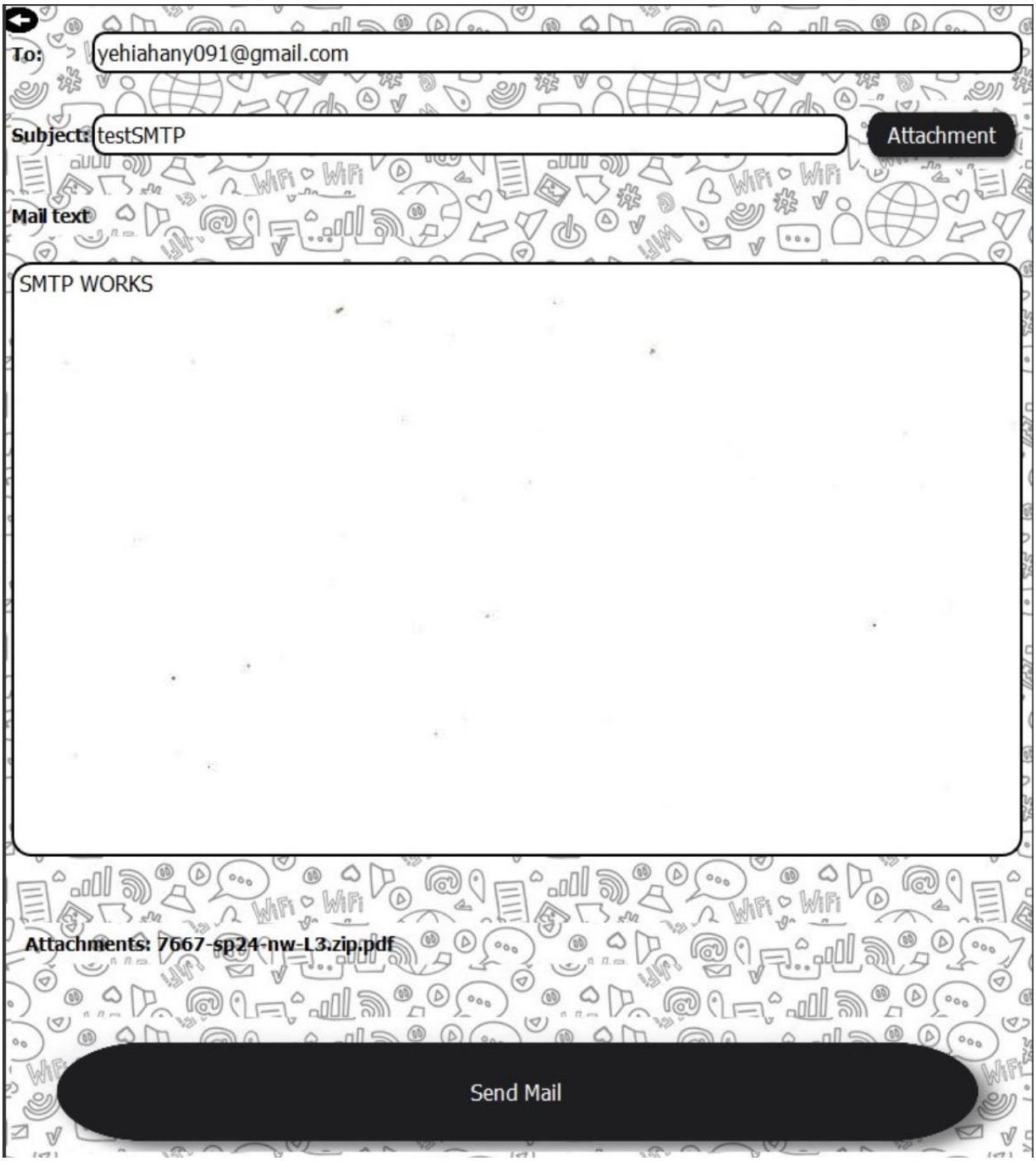
Login:



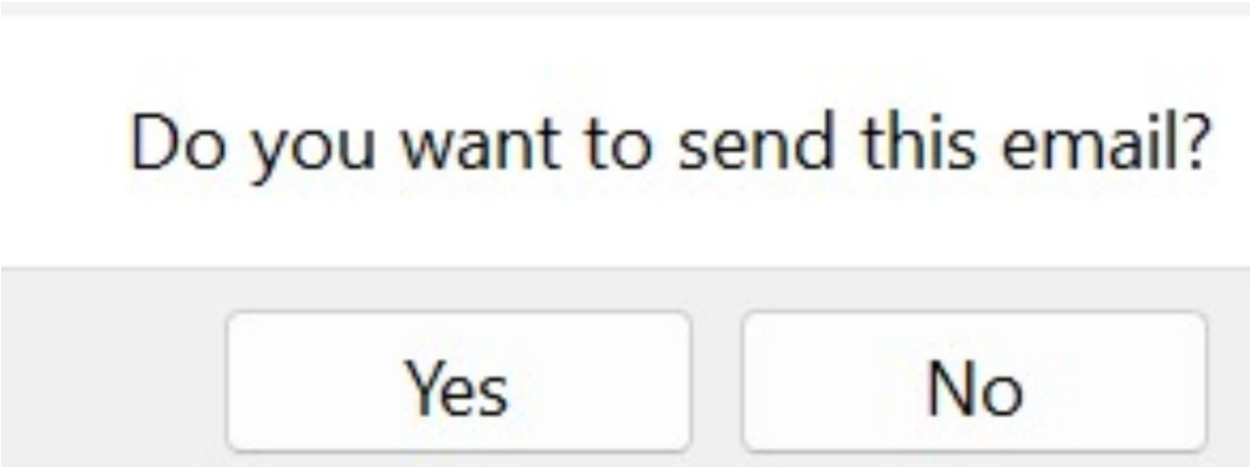
User choice:

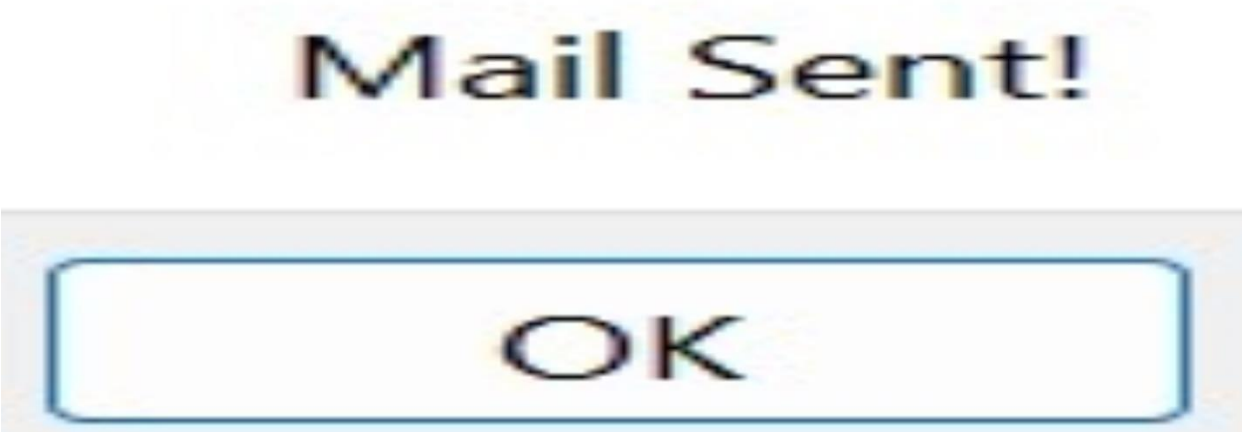


Sending email:



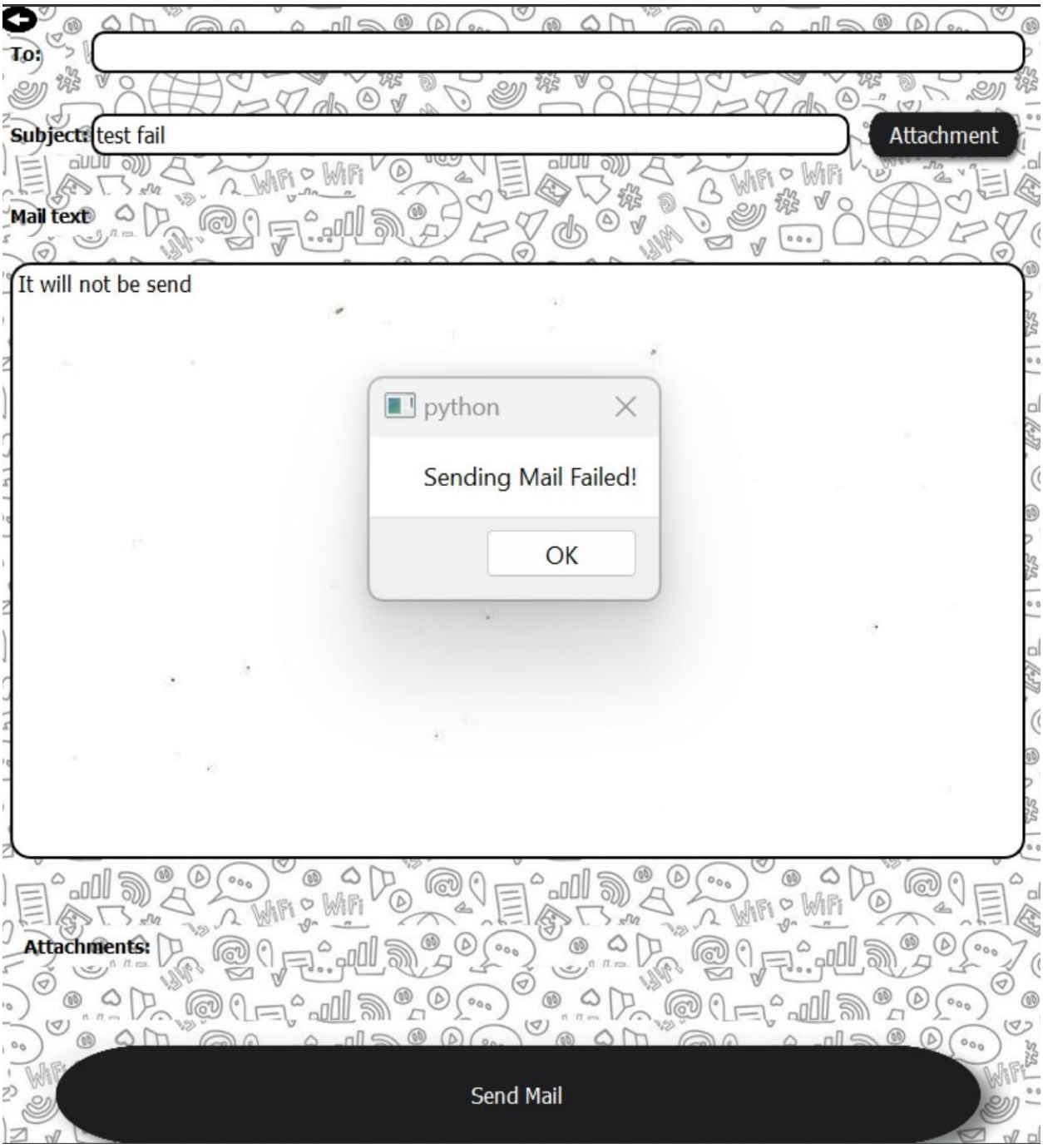
Sending verification:





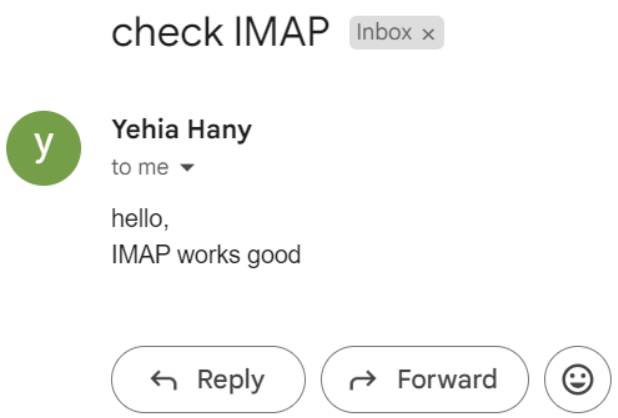
Gmail inbox of receiver:

Sending error verification:



Latest email received:

Latest email received from gmail:



How to execute in terminal:

python main.py