import imaplib  
import email  
from email.header import decode\_header  
import os  
  
# Email account configuration  
imap\_server = "edu.mailserver.ro" # IMAP server address (e.g., Gmail's IMAP server)  
email\_user = "janoska@edutus.mailserver.ro"  
email\_pass = "012345"  
  
# Local directory to save emails and attachments  
save\_directory = r"D:\janoskaemails" # Replace with your desired directory  
  
# Ensure the directory exists  
os.makedirs(save\_directory, exist\_ok=True)  
  
# Folders to synchronize  
folders = ["INBOX", "Sent", "Trash", "Junk", "Drafts","Archive" ] # Adjust for your IMAP server  
  
def safe\_decode\_header(header\_value):  
 *"""Safely decode an email header."""* if header\_value:  
 decoded, encoding = decode\_header(header\_value)[0]  
 if isinstance(decoded, bytes):  
 return decoded.decode(encoding if encoding else "utf-8")  
 return decoded  
 return "No Value"  
  
  
def save\_email(email\_message, folder\_name, email\_id):  
 *"""Save email content and attachments."""* # Create a folder for this email  
 email\_dir = os.path.join(save\_directory, folder\_name, f"Email-{email\_id}")  
 os.makedirs(email\_dir, exist\_ok=True)  
  
 # Safely decode the subject  
 subject = safe\_decode\_header(email\_message.get("Subject"))  
 subject = subject.replace("/", "\_") # Sanitize the subject for filenames  
  
 # Save email body  
 body\_file = os.path.join(email\_dir, "email.txt")  
 with open(body\_file, "w", encoding="utf-8") as f:  
 f.write(f"Subject: {subject}\n")  
 f.write(f"From: {safe\_decode\_header(email\_message.get('From'))}\n")  
 f.write(f"To: {safe\_decode\_header(email\_message.get('To'))}\n\n")  
  
 # Extract the email body  
 body = None  
 if email\_message.is\_multipart():  
 for part in email\_message.walk():  
 if part.get\_content\_type() == "text/plain":  
 body = part.get\_payload(decode=True)  
 if body:  
 body = body.decode(errors="ignore")  
 break  
 else:  
 body = email\_message.get\_payload(decode=True)  
 if body:  
 body = body.decode(errors="ignore")  
  
 # Write the body to the file  
 f.write(body if body else "No content available")  
 print(f"Saved email: {subject}")  
  
 # Save attachments  
 for part in email\_message.walk():  
 if part.get\_content\_disposition() == "attachment":  
 filename = part.get\_filename()  
 if filename:  
 filename = safe\_decode\_header(filename)  
 attachment\_path = os.path.join(email\_dir, filename)  
 with open(attachment\_path, "wb") as f:  
 f.write(part.get\_payload(decode=True))  
 print(f"Saved attachment: {filename}")  
  
  
try:  
 # Connect to the IMAP server  
 mail = imaplib.IMAP4\_SSL(imap\_server)  
 mail.login(email\_user, email\_pass)  
 print("Logged in successfully!")  
  
 for folder in folders:  
 # Select the folder  
 status, \_ = mail.select(folder)  
 if status != "OK":  
 print(f"Could not open folder: {folder}")  
 continue  
  
 folder\_name = folder.replace("[Gmail]/", "").replace("/", "\_") # Clean up folder name  
 print(f"Processing folder: {folder\_name}")  
  
 # Search for all emails in the folder  
 status, messages = mail.search(None, "ALL")  
 if status != "OK":  
 print(f"Could not retrieve messages from folder: {folder\_name}")  
 continue  
  
 # Iterate over each email  
 email\_ids = messages[0].split()  
 for email\_id in email\_ids:  
 # Fetch the email  
 status, msg\_data = mail.fetch(email\_id, "(RFC822)")  
 if status != "OK":  
 print(f"Could not fetch email ID: {email\_id}")  
 continue  
  
 for response\_part in msg\_data:  
 if isinstance(response\_part, tuple):  
 # Parse the email  
 msg = email.message\_from\_bytes(response\_part[1])  
 save\_email(msg, folder\_name=folder\_name, email\_id=email\_id.decode())  
  
 # Close the connection  
 mail.logout()  
 print("Logged out successfully. All emails and attachments saved!")  
  
except Exception as e:  
 print(f"Error: {e}")

import smtplib  
from email.mime.text import MIMEText  
from email.mime.multipart import MIMEMultipart  
from email.mime.base import MIMEBase  
from email import encoders  
  
# Email configuration  
smtp\_server = "edu.mailserver.ro" # Replace with your SMTP server  
smtp\_port = 587 # Standard port for STARTTLS  
sender\_email = "janos@edutus.mailserver.ro"  
sender\_password = "012345"  
recipient\_email = "janoska@edutus.mailserver.ro"  
  
# Create email content  
subject = "Test Emailke"  
body = "This is a test email sent from Pythonnnfvsdjfgnsdfjklnsdjkfgnsdfnkgsdjkfngksdfngjkadfnjksndfjksgdf."  
  
# Setting up MIME  
message = MIMEMultipart()  
message["From"] = sender\_email  
message["To"] = recipient\_email  
message["Subject"] = subject  
message.attach(MIMEText(body, "plain"))  
  
  
file\_path = "D:/IMG\_20241026\_200124\_7.jpg" # Replace with the actual file path  
try:  
 with open(file\_path, "rb") as attachment:  
 # Add file as application/octet-stream  
 part = MIMEBase("application", "octet-stream")  
 part.set\_payload(attachment.read())  
  
 # Encode file in ASCII characters for email  
 encoders.encode\_base64(part)  
  
 # Add header for file attachment  
 part.add\_header(  
 "Content-Disposition",  
 f"attachment; filename={file\_path.split('/')[-1]}" # Extracts the file name  
 )  
  
 # Attach the file to the message  
 message.attach(part)  
  
 # Send the email  
 with smtplib.SMTP(smtp\_server, smtp\_port) as server:  
 server.starttls() # Upgrade the connection to secure  
 server.login(sender\_email, sender\_password)  
 server.sendmail(sender\_email, recipient\_email, message.as\_string())  
 print("Email with attachment sent successfully!")  
except Exception as e:  
 print(f"Error: {e}")