



Write a script to automate sending daily email reports in Python, and walk me through how I would set it up.



Sure, I'll help you set up a Python script to automate sending daily email reports. This guide will cover writing the script, setting up the email service, and scheduling the script to run daily.

## Step 1: Writing the Python Script

First, you need to write a Python script that will generate and send the email report. We'll use the `smtplib` library for sending emails and `email` library to construct the email content.

Here's a simple example script:

```
import smtplib from email.mime.multipart import MIMEMultipart from email.mime.text
import MIMEText from email.mime.application import MIMEApplication from datetime import
datetime import os def send_email(subject, body, to_emails, attachment_path=None):
from_email = 'your_email@example.com' password = 'your_email_password' # Create the
email container msg = MIMEMultipart() msg['From'] = from_email msg['To'] = ',
'.join(to_emails) msg['Subject'] = subject # Attach the email body
msg.attach(MIMEText(body, 'plain')) # Attach a file if specified if attachment_path:
with open(attachment_path, 'rb') as file: part = MIMEApplication(file.read(),
Name=os.path.basename(attachment_path)) part['Content-Disposition'] = f'attachment;
filename="{os.path.basename(attachment_path)}"' msg.attach(part) # Send the email with
smtplib.SMTP('smtp.example.com', 587) as server: server.starttls()
server.login(from_email, password) server.sendmail(from_email, to_emails,
msg.as_string()) def generate_report(): # Replace with your report generation logic
report_content = "Daily Report:\n\nAll systems operational." return report_content if
__name__ == '__main__': report = generate_report() subject = f"Daily Report -
{datetime.now().strftime('%Y-%m-%d%I%M%S')}" body = report to_emails =
['recipient1@example.com', 'recipient2@example.com'] send_email(subject, body,
to_emails)
```

## Step 2: Setting Up Your Email Service

### 1. SMTP Server Configuration:

- Replace `your_email@example.com` and `your_email_password` with your actual email address and password.
- Replace `smtp.example.com` with your email provider's SMTP server. Common ones include:
  - Gmail: `smtp.gmail.com`
  - Yahoo: `smtp.mail.yahoo.com`
  - Outlook: `smtp.office365.com`

## 2. Allow Less Secure Apps:

- For Gmail, you might need to allow less secure apps to access your account: Google Account - Less Secure Apps.
- For Outlook, you may need to create an app password if you have two-factor authentication enabled.

## Step 3: Scheduling the Script to Run Daily

To schedule the script to run daily, you can use **cron jobs** on Unix-based systems (Linux, macOS) or **Task Scheduler** on Windows.

### Using cron (Unix-based systems):

1. Open your crontab file:

```
bash
crontab -e
```



2. Add a line to schedule your script to run daily at a specific time. For example, to run the script every day at 8 AM:

```
bash
0 8 * * * /usr/bin/python3 /path/to/your_script.py
```



Replace `/usr/bin/python3` with the path to your Python interpreter and `/path/to/your_script.py` with the path to your script.

### Using Task Scheduler (Windows):

1. Open Task Scheduler and create a new task.
2. Set the trigger to run daily at your desired time.
3. Set the action to start a program and provide the path to your Python interpreter, and add the script path as an argument.

For example:

- Program/script: `C:\Path\To\Python\python.exe`
- Add arguments: `C:\Path\To\your_script.py`

## Conclusion

You've now written a script to send daily email reports, configured your email service, and scheduled the script to run daily. Make sure to test your script by running it manually to ensure everything is set up correctly.

If I want to use VS code to conduct this task and I do not want to expose my real password of gmail account, how to adjust the code accordingly thanks



To avoid exposing your Gmail password in your script, you can use environment variables to securely manage your credentials. Here's how to do it using Visual Studio Code (VS Code) and environment variables.

## Step 1: Setting Up Environment Variables

### On Unix-based Systems (Linux, macOS)

1. Open your terminal and edit your shell configuration file (e.g., `.bashrc`, `.bash_profile`, or `.zshrc`):

```
bash
nano ~/.bashrc # or ~/.bash_profile or ~/.zshrc
```

Copy code

2. Add your email credentials:

bash

 Copy code

```
export EMAIL_USER="your_email@example.com" export
EMAIL_PASSWORD="your_email_password"
```

3. Save and close the file, then reload the configuration:

bash

 Copy code

```
source ~/.bashrc # or ~/.bash_profile or ~/.zshrc
```

## On Windows

1. Open Command Prompt and set the environment variables:

cmd

 Copy code

```
setx EMAIL_USER "your_email@example.com" setx EMAIL_PASSWORD "your_email_password"
```

2. Restart your computer or the Command Prompt to apply the changes.

## Step 2: Accessing Environment Variables in Your Python Script

Modify your Python script to use environment variables:

python

 Copy code

```
import smtplib from email.mime.multipart import MIMEMultipart from email.mime.text
import MIMEText from email.mime.application import MIMEApplication from datetime import
datetime import os def send_email(subject, body, to_emails, attachment_path=None):
from_email = os.getenv('EMAIL_USER') password = os.getenv('EMAIL_PASSWORD') # Create the
email container msg = MIMEMultipart() msg['From'] = from_email msg['To'] = ',
'.join(to_emails) msg['Subject'] = subject # Attach the email body
msg.attach(MIMEText(body, 'plain')) # Attach a file if specified if attachment_path:
with open(attachment_path, 'rb') as file: part = MIMEApplication(file.read(),
Name=os.path.basename(attachment_path)) part['Content-Disposition'] = f'attachment;
filename="{os.path.basename(attachment_path)}"' msg.attach(part) # Send the email with
smtplib.SMTP('smtp.gmail.com', 587) as server: server.starttls()
server.login(from_email, password) server.sendmail(from_email, to_emails,
msg.as_string()) def generate_report(): # Replace with your report generation logic
report_content = "Daily Report:\n\nAll systems operational." return report_content if
__name__ == '__main__': report = generate_report() subject = f"Daily Report -
{datetime.now().strftime('%Y-%m-%d')}" body = report to_emails =
```

```
[ 'recipient1@example.com', 'recipient2@example.com'] send_email(subject, body,
to_emails)
```

### Step 3: Using VS Code

1. Open your project in VS Code.
2. Create a .env file in the root of your project (same directory as your script):

```
makefile
```

 Copy code

```
EMAIL_USER=your_email@example.com EMAIL_PASSWORD=your_email_password
```

3. Install the python-dotenv package to read the .env file:

```
bash
```

 Copy code

```
pip install python-dotenv
```

4. Modify your script to load the .env file:

```
python
```

 Copy code

```
import smtplib from email.mime.multipart import MIMEMultipart from email.mime.text
import MIMEText from email.mime.application import MIMEApplication from datetime
import datetime import os from dotenv import load_dotenv # Load environment
variables from .env file load_dotenv() def send_email(subject, body, to_emails,
attachment_path=None): from_email = os.getenv('EMAIL_USER') password =
os.getenv('EMAIL_PASSWORD') # Create the email container msg = MIMEMultipart()
msg['From'] = from_email msg['To'] = ', '.join(to_emails) msg['Subject'] = subject
# Attach the email body msg.attach(MIMEText(body, 'plain')) # Attach a file if
specified if attachment_path: with open(attachment_path, 'rb') as file: part =
MIMEApplication(file.read(), Name=os.path.basename(attachment_path)) part['Content-
Disposition'] = f'attachment; filename="{os.path.basename(attachment_path)}"'
msg.attach(part) # Send the email with smtplib.SMTP('smtp.gmail.com', 587) as
server: server.starttls() server.login(from_email, password)
server.sendmail(from_email, to_emails, msg.as_string()) def generate_report(): #
Replace with your report generation logic report_content = "Daily Report:\n\nAll
systems operational." return report_content if __name__ == '__main__': report =
generate_report() subject = f"Daily Report - {datetime.now().strftime('%Y-%m-%d')}"
body = report to_emails = ['recipient1@example.com', 'recipient2@example.com']
send_email(subject, body, to_emails)
```

## Conclusion

By using environment variables and the `python-dotenv` package, you can securely manage your credentials without exposing them in your code. This setup allows you to keep sensitive information out of your scripts, enhancing security.

< 1/2 >

`smtplib.SMTPAuthenticationError: (535, b'5.7.8 Username and Password not accepted. For more information, go to\n5.7.8 https://support.google.com/mail/?p=BadCredentials 38308e7fff4ca-2e95be01aecsm19690041fa.109 - gsmtplib') I set user and password in vs code, but error occurs, how to adjust accordingly?`



The `smtplib.SMTPAuthenticationError` error typically indicates that the username and password are either incorrect or that the Gmail account has security settings that prevent login from your script. Here are some steps you can follow to resolve this issue:

### Step 1: Verify Email and Password

Ensure that your email and password are correct and that they are correctly set in your environment variables.

### Step 2: Enable "Less Secure Apps"

For Gmail, you may need to allow access for "less secure apps" to enable your script to send emails. However, as of May 30, 2022, Google no longer supports "Less Secure Apps." Instead, you should use an App Password.

### Step 3: Use App Passwords

#### 1. Enable Two-Factor Authentication (2FA):

- Go to your Google Account.
- Navigate to the "Security" section.
- Set up 2-Step Verification (2FA).

#### 2. Create an App Password:

- Once 2FA is enabled, go back to the "Security" section.

- Under "Signing in to Google," select "App passwords."
- Generate an App Password for "Mail" and "Windows Computer" (or whatever label suits your setup).

### 3. Use the App Password in Your Script:

- Use the generated App Password instead of your regular account password in your environment variables.

## Step 4: Update the Environment Variables in VS Code

### 1. Set the environment variables in your .env file with the App Password:

```
makefile
EMAIL_USER=your_email@example.com EMAIL_PASSWORD=your_app_password
```

### 2. Ensure your script loads the environment variables correctly:

```
python
import smtplib from email.mime.multipart import MIMEMultipart from email.mime.text
import MIMEText from email.mime.application import MIMEApplication from datetime
import datetime import os from dotenv import load_dotenv # Load environment
variables from .env file load_dotenv() def send_email(subject, body, to_emails,
attachment_path=None): from_email = os.getenv('EMAIL_USER') password =
os.getenv('EMAIL_PASSWORD') # Create the email container msg = MIMEMultipart()
msg['From'] = from_email msg['To'] = ', '.join(to_emails) msg['Subject'] = subject
# Attach the email body msg.attach(MIMEText(body, 'plain')) # Attach a file if
specified if attachment_path: with open(attachment_path, 'rb') as file: part =
MIMEApplication(file.read(), Name=os.path.basename(attachment_path)) part['Content-
Disposition'] = f'attachment; filename="{os.path.basename(attachment_path)}"'
msg.attach(part) # Send the email with smtplib.SMTP('smtp.gmail.com', 587) as
server: server.starttls() server.login(from_email, password)
server.sendmail(from_email, to_emails, msg.as_string()) def generate_report(): #
Replace with your report generation logic report_content = "Daily Report:\n\nAll
systems operational." return report_content if __name__ == '__main__': report =
generate_report() subject = f"Daily Report - {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}"
body = report to_emails = ['recipient1@example.com', 'recipient2@example.com']
send_email(subject, body, to_emails)
```

## Step 5: Testing the Script

After setting up the App Password and updating your `.env` file, test the script again. Ensure that the `.env` file is in the correct directory (same directory as your script) and that the environment variables are being loaded correctly.

## Conclusion

Using an App Password for Gmail should resolve the `SMTPAuthenticationError`. Ensure you have set up 2FA and generated an App Password specifically for this script. This approach is more secure and complies with Google's updated security policies.

```
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.application import MIMEApplication
from datetime import datetime
import os
from DailyReport import .env

# Load environment variables from .env file
load_dotenv()

def send_email(subject, body, to_emails, attachment_path=None):
    from_email = os.getenv('EMAIL_USER')
    password = os.getenv('EMAIL_PASSWORD')

    # Create the email container
    msg = MIMEMultipart()
    msg['From'] = from_email
    msg['To'] = ', '.join(to_emails)
    msg['Subject'] = subject

    # Attach the email body
    msg.attach(MIMEText(body, 'plain'))

    # Attach a file if specified
    if attachment_path:
        with open(attachment_path, 'rb') as file:
            part = MIMEApplication(file.read(),
                                   Name=os.path.basename(attachment_path))
            part['Content-Disposition'] = f'attachment; filename="{os.path.basename(attachment_path)}"'
            msg.attach(part)
```



```
# Send the email
with smtplib.SMTP('smtp.gmail.com', 587) as server:
    server.starttls()
    server.login(from_email, password)
    server.sendmail(from_email, to_emails, msg.as_string())

def generate_report():
    # Replace with your report generation logic
    report_content = "Daily Report:\n\nAll systems operational."
    return report_content

if __name__ == '__main__':
    report = generate_report()
    subject = f"Daily Report - {datetime.now().strftime('%Y-%m-%d')}}"
    body = report
    to_emails = ['gracepan922@gmail.com']

    send_email(subject, body, to_emails)

I have this dailyReport.py in my DailyReport folder, there is also a
.env in my DailyReport folder with contents
EMAIL_USER=zpan01080124@gmail.com
EMAIL_PASSWORD=hnel yjav awtl jxps  how to adjust the code to
make the program run well
```



Let's correct and refine your script to ensure it works well with your `.env` file. Here are the adjustments you need to make:

1. **Correctly Import `load_dotenv` from `dotenv`:**  
Ensure you import `load_dotenv` correctly from the `dotenv` package.
2. **Fix Import Statement for the `.env` File:**  
Ensure you correctly load the `.env` file using the `load_dotenv` function.
3. **Correct Email Password Environment Variable in `.env` File:**  
Ensure there are no spaces in your `EMAIL_PASSWORD` value.

Here's the corrected code for your `dailyReport.py` script:

### Step 1: Ensure `.env` File is Correct

Make sure your `.env` file in the `DailyReport` folder looks like this:

makefile

 Copy code

```
EMAIL_USER=zpan01080124@gmail.com EMAIL_PASSWORD=hneljjavawt1jxps
```

## Step 2: Install the python-dotenv Package

Make sure you have `python-dotenv` installed. You can install it using pip:

bash

 Copy code

```
pip install python-dotenv
```

## Step 3: Corrected dailyReport.py Script

Here's the corrected script:

python

 Copy code

```
import smtplib from email.mime.multipart import MIMEMultipart from email.mime.text
import MIMEText from email.mime.application import MIMEApplication from datetime import
datetime import os from dotenv import load_dotenv # Load environment variables from .env
file load_dotenv(dotenv_path=os.path.join(os.path.dirname(__file__), '.env')) def
send_email(subject, body, to_emails, attachment_path=None): from_email =
os.getenv('EMAIL_USER') password = os.getenv('EMAIL_PASSWORD') if not from_email or not
password: raise ValueError("EMAIL_USER and EMAIL_PASSWORD environment variables must be
set") # Create the email container msg = MIMEMultipart() msg['From'] = from_email
msg['To'] = ', '.join(to_emails) msg['Subject'] = subject # Attach the email body
msg.attach(MIMEText(body, 'plain')) # Attach a file if specified if attachment_path:
with open(attachment_path, 'rb') as file: part = MIMEApplication(file.read(),
Name=os.path.basename(attachment_path)) part['Content-Disposition'] = f'attachment;
filename="{os.path.basename(attachment_path)}"' msg.attach(part) # Send the email with
smtplib.SMTP('smtp.gmail.com', 587) as server: server.starttls()
server.login(from_email, password) server.sendmail(from_email, to_emails,
msg.as_string()) def generate_report(): # Replace with your report generation logic
report_content = "Daily Report:\n\nAll systems operational." return report_content if
__name__ == '__main__': report = generate_report() subject = f"Daily Report -
{datetime.now().strftime('%Y-%m-%d')}" body = report to_emails =
['gracepan922@gmail.com'] send_email(subject, body, to_emails)
```

## Explanation

### 1. Import and Load Environment Variables:

python

 Copy code


```
from dotenv import load_dotenv
load_dotenv(dotenv_path=os.path.join(os.path.dirname(__file__), '.env'))
```

This ensures the `.env` file located in the same directory as your script is loaded correctly.

## 2. Check for Environment Variables:

```
python

if not from_email or not password: raise ValueError("EMAIL_USER and EMAIL_PASSWORD
environment variables must be set")
```

 Copy code

This ensures that your script raises an error if the environment variables are not found.

## Step 4: Running the Script

Run your script from the terminal or from within VS Code. Make sure you are in the **DailyReport** directory or provide the correct path to the script:

```
bash

python dailyReport.py
```

 Copy code

This setup should ensure your script loads the environment variables from the `.env` file and sends the email using the provided Gmail credentials.

ValueError: EMAIL\_USER and EMAIL\_PASSWORD environment variables must be set I get this error adjust the code to correct this error thanks



To address the `ValueError` indicating that the `EMAIL_USER` and `EMAIL_PASSWORD` environment variables are not being set properly, we need to ensure a few things:

1. The `.env` file is properly formatted and located in the correct directory.
2. The script is correctly loading and using the `.env` file.

Let's walk through each step to ensure everything is correctly set up.

## Step 1: Verify the .env File

Ensure your `.env` file in the `DailyReport` folder is correctly formatted and does not contain any trailing spaces or invisible characters. The file should look exactly like this:

```
makefile

EMAIL_USER=zpan01080124@gmail.com EMAIL_PASSWORD=hnelyjavawtljxps
```



## Step 2: Ensure dotenv is Installed

If you haven't already, install the `python-dotenv` package:

```
bash

pip install python-dotenv
```



## Step 3: Adjust the Script to Correctly Load Environment Variables

Ensure your script is correctly loading the `.env` file and handling paths properly. Here is the updated `dailyReport.py` script:

```
python

import smtplib from email.mime.multipart import MIMEMultipart from email.mime.text
import MIMEText from email.mime.application import MIMEApplication from datetime import
datetime import os from dotenv import load_dotenv # Determine the directory of the
current script script_dir = os.path.dirname(os.path.abspath(__file__)) # Load
environment variables from .env file located in the same directory as the script
load_dotenv(dotenv_path=os.path.join(script_dir, '.env')) def send_email(subject, body,
to_emails, attachment_path=None): from_email = os.getenv('EMAIL_USER') password =
os.getenv('EMAIL_PASSWORD') if not from_email or not password: raise
ValueError("EMAIL_USER and EMAIL_PASSWORD environment variables must be set") # Create
the email container msg = MIMEMultipart() msg['From'] = from_email msg['To'] = ',
'.join(to_emails) msg['Subject'] = subject # Attach the email body
msg.attach(MIMEText(body, 'plain')) # Attach a file if specified if attachment_path:
with open(attachment_path, 'rb') as file: part = MIMEApplication(file.read(),
Name=os.path.basename(attachment_path)) part['Content-Disposition'] = f'attachment;
filename="{os.path.basename(attachment_path)}"' msg.attach(part) # Send the email with
smtplib.SMTP('smtp.gmail.com', 587) as server: server.starttls()
server.login(from_email, password) server.sendmail(from_email, to_emails,
msg.as_string()) def generate_report(): # Replace with your report generation logic
report_content = "Daily Report:\n\nAll systems operational." return report_content if
__name__ == '__main__': report = generate_report() subject = f"Daily Report -
```



```
{datetime.now().strftime('%Y-%m-%d'))}" body = report to_emails =  
['gracepan922@gmail.com'] send_email(subject, body, to_emails)
```

## Explanation of the Script

### 1. Determine the Script Directory:

```
python
```

 Copy code

```
script_dir = os.path.dirname(os.path.abspath(__file__))
```

This ensures that the script can correctly find the `.env` file located in the same directory, regardless of where the script is run from.

### 2. Load Environment Variables:

```
python
```

 Copy code

```
load_dotenv(dotenv_path=os.path.join(script_dir, '.env'))
```

This explicitly sets the path to the `.env` file, ensuring it is correctly loaded.

### 3. Check Environment Variables:

```
python
```

 Copy code

```
if not from_email or not password: raise ValueError("EMAIL_USER and EMAIL_PASSWORD  
environment variables must be set")
```

This checks that the environment variables are correctly loaded and raises an error if they are not.

## Step 4: Run the Script

Make sure you are in the `DailyReport` directory or provide the correct path to the script:

```
bash
```

 Copy code

```
python dailyReport.py
```

By following these steps, your script should correctly load the environment variables from the `.env` file and send the email using the provided Gmail credentials. If you continue to encounter issues, double-check the content of your `.env` file and ensure there are no syntax errors or extra spaces.

can you adjust the code, so I can send the daily report with attachments thanks give me full code



Sure, I will provide a full, adjusted script that includes the ability to send the daily report with attachments. Here's how you can set it up:

1. Ensure your `.env` file is properly formatted and located in the correct directory.
2. Install the `python-dotenv` package if you haven't already.
3. Adjust the script to include attachment functionality.

### Step 1: Verify the `.env` File

Make sure your `.env` file in the `DailyReport` folder looks like this:

```
makefile

EMAIL_USER=zpan01080124@gmail.com EMAIL_PASSWORD=hnelyjavawtljxps
```



Copy code

### Step 2: Install the `python-dotenv` Package

Install the `python-dotenv` package:

```
bash

pip install python-dotenv
```



Copy code

### Step 3: Adjust the Script

Here's the full script (`dailyReport.py`) that includes the ability to send the daily report with attachments:

```
python

import smtplib from email.mime.multipart import MIMEMultipart from email.mime.text
import MIMEText from email.mime.application import MIMEApplication from datetime import
```



Copy code

```

datetime import os from dotenv import load_dotenv # Determine the directory of the
current script script_dir = os.path.dirname(os.path.abspath(__file__)) # Load
environment variables from .env file located in the same directory as the script
load_dotenv(dotenv_path=os.path.join(script_dir, '.env')) def send_email(subject, body,
to_emails, attachment_paths=None): from_email = os.getenv('EMAIL_USER') password =
os.getenv('EMAIL_PASSWORD') if not from_email or not password: raise
ValueError("EMAIL_USER and EMAIL_PASSWORD environment variables must be set") # Create
the email container msg = MIME multipart() msg['From'] = from_email msg['To'] = ',
'.join(to_emails) msg['Subject'] = subject # Attach the email body
msg.attach(MIMEText(body, 'plain')) # Attach files if specified if attachment_paths: for
attachment_path in attachment_paths: with open(attachment_path, 'rb') as file: part =
MIMEApplication(file.read(), Name=os.path.basename(attachment_path)) part['Content-
Disposition'] = f'attachment; filename="{os.path.basename(attachment_path)}"'
msg.attach(part) # Send the email with smtplib.SMTP('smtp.gmail.com', 587) as server:
server.starttls() server.login(from_email, password) server.sendmail(from_email,
to_emails, msg.as_string()) def generate_report(): # Replace with your report generation
logic report_content = "Daily Report:\n\nAll systems operational." return report_content
if __name__ == '__main__': report = generate_report() subject = f"Daily Report -
{datetime.now().strftime('%Y-%m-%d')}" body = report to_emails =
['gracepan922@gmail.com'] # Add paths to attachments here attachment_paths = [
os.path.join(script_dir, 'report_attachment_1.pdf'), os.path.join(script_dir,
'report_attachment_2.csv') ] send_email(subject, body, to_emails, attachment_paths)

```

## Explanation

### 1. Load Environment Variables:

```
python
```



```
load_dotenv(dotenv_path=os.path.join(script_dir, '.env'))
```

This ensures the script loads the environment variables from the `.env` file located in the same directory.

### 2. Send Email Function:

The `send_email` function now includes an optional `attachment_paths` parameter, which is a list of file paths to attach to the email. It iterates over the list of attachments and attaches each file to the email.

### 3. Generate Report Function:

The `generate_report` function contains the logic for generating the report content.

### 4. Main Execution Block:

This block calls `generate_report`, sets the email subject, body, recipients, and attachment paths,

and then sends the email.

## Running the Script

Run the script from the terminal or from within VS Code, ensuring you are in the **DailyReport** directory:

```
bash
```

 Copy code

```
python dailyReport.py
```

Make sure the paths to your attachments are correct and that the files exist in those locations. This setup should allow your script to send the daily report email with attachments.

