sendMail.py:

-Các thư viện hỗ trợ:

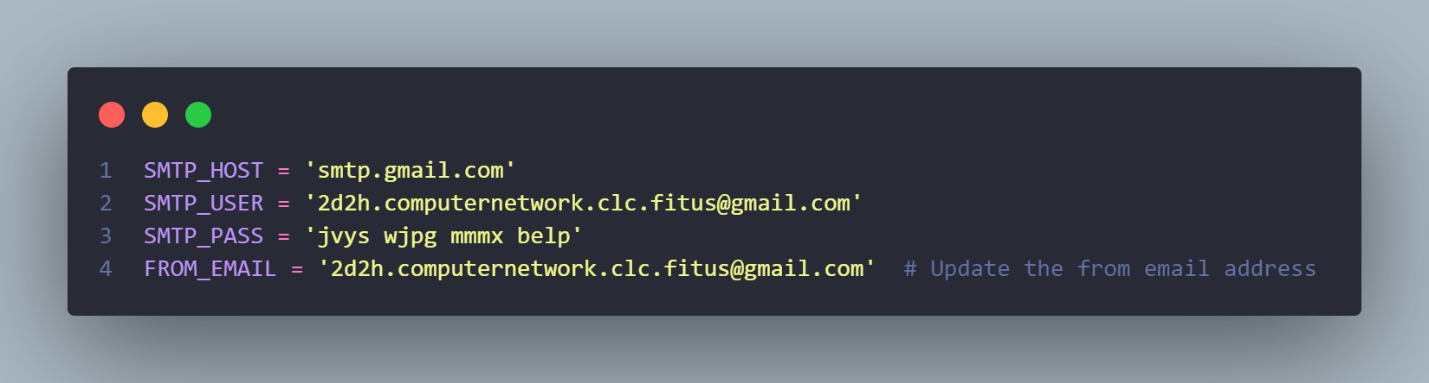
A screenshot of a computer program

Description automatically generated

**Smtplib**: cho gửi email

các lớp **MIMEMultipart**, **MIMEText**, **MIMEImage:** cho định dạng email

**ImageGrab** từ thư viện Pillow để chụp màn hình, và các lớp từ các module khác trong chương trình.



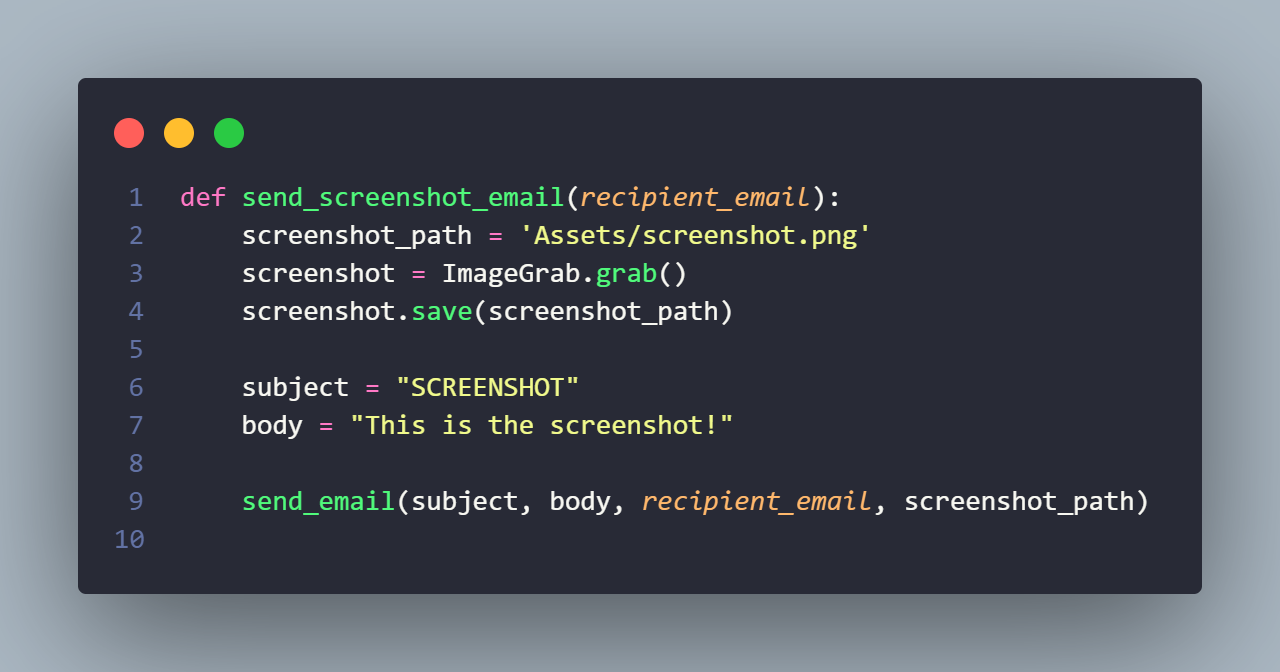
* **SMTP\_HOST = 'smtp.gmail.com'**: Đây là địa chỉ máy chủ SMTP của Gmail. Khi gửi email, phải kết nối đến máy chủ này để chuyển thư đi.
* **SMTP\_USER = '2d2h.computernetwork.clc.fitus@gmail.com'**: Đây là địa chỉ email của người gửi, sẽ được sử dụng để đăng nhập vào máy chủ SMTP.
* **SMTP\_PASS = 'jvys wjpg mmmx belp'**: Đây là mật khẩu của địa chỉ email người gửi, được sử dụng để xác thực khi đăng nhập vào máy chủ SMTP. Lưu ý rằng việc lưu mật khẩu trực tiếp trong mã nguồn không được khuyến khích do có thể tạo ra rủi ro bảo mật. Trong môi trường thực tế, nên sử dụng biện pháp bảo mật cao hơn, chẳng hạn như lưu mật khẩu trong biến môi trường hoặc sử dụng các dịch vụ quản lý mật khẩu.
* **FROM\_EMAIL = '2d2h.computernetwork.clc.fitus@gmail.com'**: Đây là địa chỉ email người gửi. Nếu cần thay đổi địa chỉ email người gửi, bạn có thể chỉnh sửa giá trị của biến này.

+hàm “sendMail”:



* **msg = MIMEMultipart()**: Tạo một đối tượng MIMEMultipart, là một đối tượng MIME sử dụng để xây dựng các thông điệp email với nhiều phần khác nhau.
* **msg['From'] = FROM\_EMAIL**: Thiết lập trường "From" của thông điệp email bằng địa chỉ email người gửi (**FROM\_EMAIL**).
* **msg['To'] = recipient\_email**: Thiết lập trường "To" của thông điệp email bằng địa chỉ email của người nhận (**recipient\_email**).
* **msg['Subject'] = subject**: Thiết lập trường "Subject" của thông điệp email bằng tiêu đề được truyền vào hàm (**subject**).
* **text = MIMEText(body, 'plain')**: Tạo một đối tượng MIMEText chứa nội dung văn bản của email (**body**) với định dạng 'plain'.
* **msg.attach(text)**: Đính kèm đối tượng MIMEText vào đối tượng MIMEMultipart để thêm nội dung văn bản vào thông điệp email.
* **if attachment\_path:**: Kiểm tra xem có đường dẫn đính kèm (**attachment\_path**) không.
* **with open(attachment\_path, 'rb') as attachment\_file:**: Mở tệp đính kèm ở chế độ đọc dưới dạng nhị phân (**'rb'**).
* **attachment = MIMEImage(attachment\_file.read(), name='screenshot.png')**: Tạo một đối tượng MIMEImage từ dữ liệu của tệp đính kèm, và đặt tên tệp là 'screenshot.png'.
* **msg.attach(attachment)**: Đính kèm đối tượng MIMEImage vào đối tượng MIMEMultipart để thêm hình ảnh hoặc bất kỳ đính kèm nào khác vào thông điệp email.
* **smtp\_server = SMTP\_SSL(SMTP\_HOST, port=SMTP\_SSL\_PORT)**: Tạo một đối tượng SMTP\_SSL để kết nối đến máy chủ SMTP của Gmail qua SSL.
* **smtp\_server.set\_debuglevel(1)**: Thiết lập chế độ debug cho đối tượng SMTP để hiển thị tất cả tương tác với máy chủ SMTP.
* **smtp\_server.login(SMTP\_USER, SMTP\_PASS)**: Đăng nhập vào máy chủ SMTP bằng tên người dùng và mật khẩu.
* **smtp\_server.sendmail(FROM\_EMAIL, recipient\_email, msg.as\_string())**: Gửi thông điệp email đến người nhận.
* **smtp\_server.quit()**: Ngắt kết nối từ máy chủ SMTP sau khi đã gửi email xong.

+hàm “send\_screenshot\_email”:



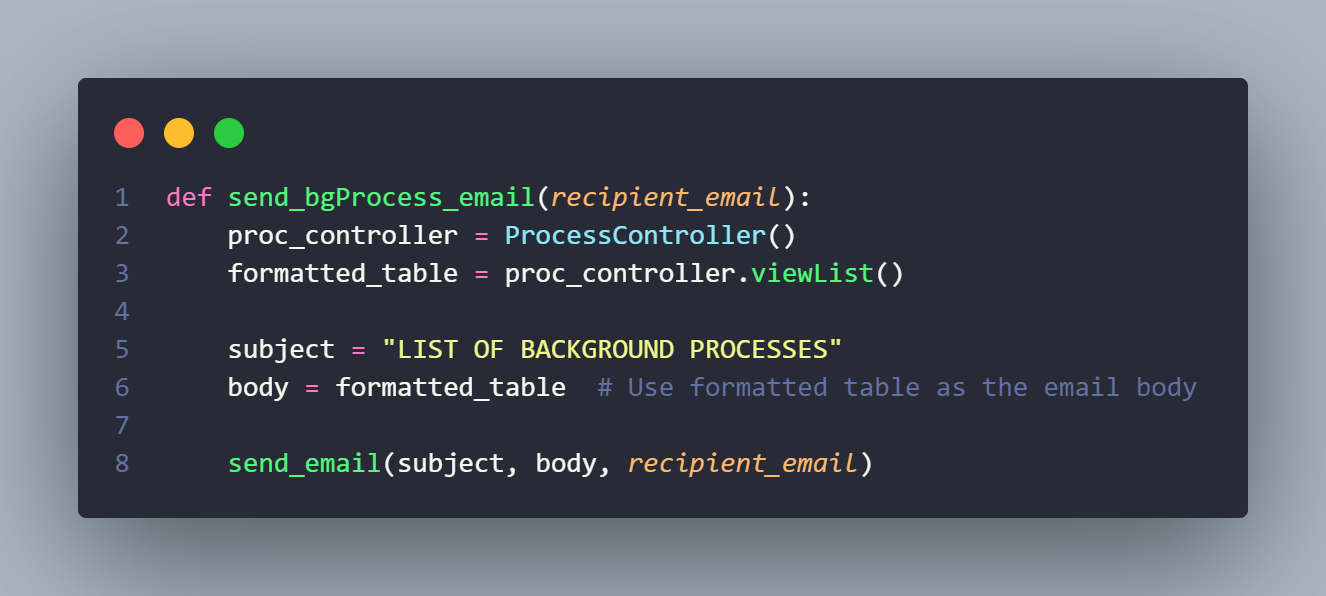
* **screenshot\_path = 'Assets/screenshot.png'**: Xác định đường dẫn nơi mà ảnh chụp màn hình sẽ được lưu trữ.
* **screenshot = ImageGrab.grab()**: Sử dụng hàm **grab()** từ thư viện PIL để chụp màn hình và lưu vào biến **screenshot**.
* **screenshot.save(screenshot\_path)**: Lưu ảnh chụp màn hình vào đường dẫn đã xác định ở bước 1.
* **subject = "SCREENSHOT"**: Đặt chủ đề của email là "SCREENSHOT".
* **body = "This is the screenshot!"**: Thiết lập nội dung văn bản của email là "This is the screenshot!".
* **send\_email(subject, body, recipient\_email, screenshot\_path)**: Gọi hàm **send\_email** để gửi email với chủ đề, nội dung và ảnh chụp màn hình đã được xác định.

+hàm “send\_process\_email”



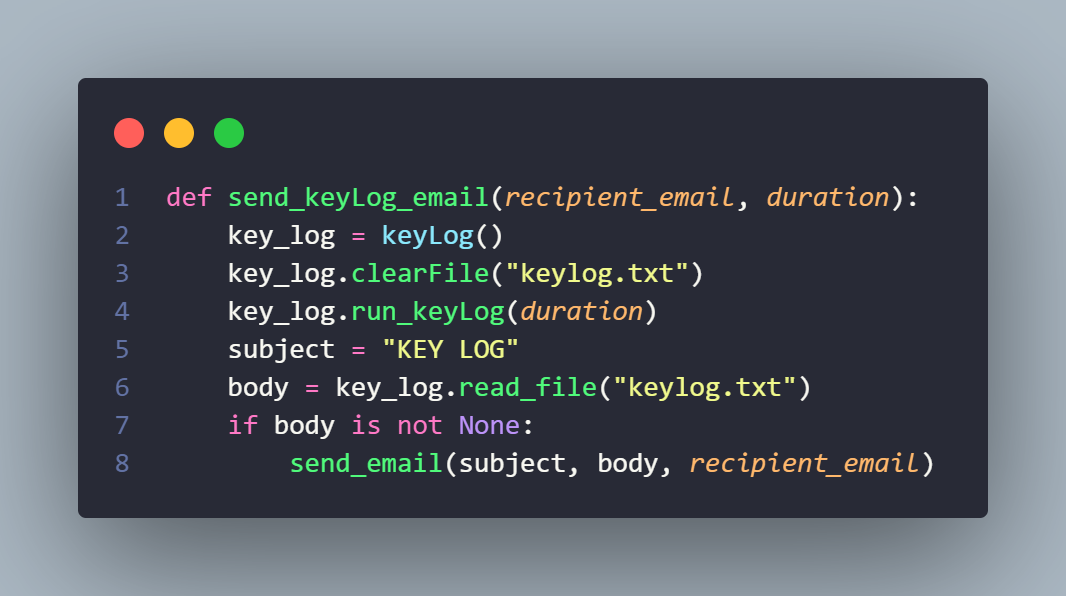
* **app\_controller = AppController()**: Tạo một đối tượng của lớp **AppController**. Lớp này chứa các phương thức để quản lý ứng dụng trên hệ thống.
* **formatted\_table = app\_controller.viewList()**: Gọi phương thức **viewList** từ đối tượng **app\_controller** để lấy danh sách các ứng dụng và định dạng nó thành một bảng.
* **subject = "LIST OF APPLICATIONS"**: Đặt chủ đề của email là "LIST OF APPLICATIONS".
* **body = formatted\_table**: Sử dụng bảng đã được định dạng như là nội dung email.
* **send\_email(subject, body, recipient\_email)**: Gọi hàm **send\_email** để gửi email với chủ đề, nội dung và không có tệp đính kèm (ảnh chụp màn hình, v.v.) cho địa chỉ email đã chỉ định.

+hàm “”send\_bgProcess\_email”



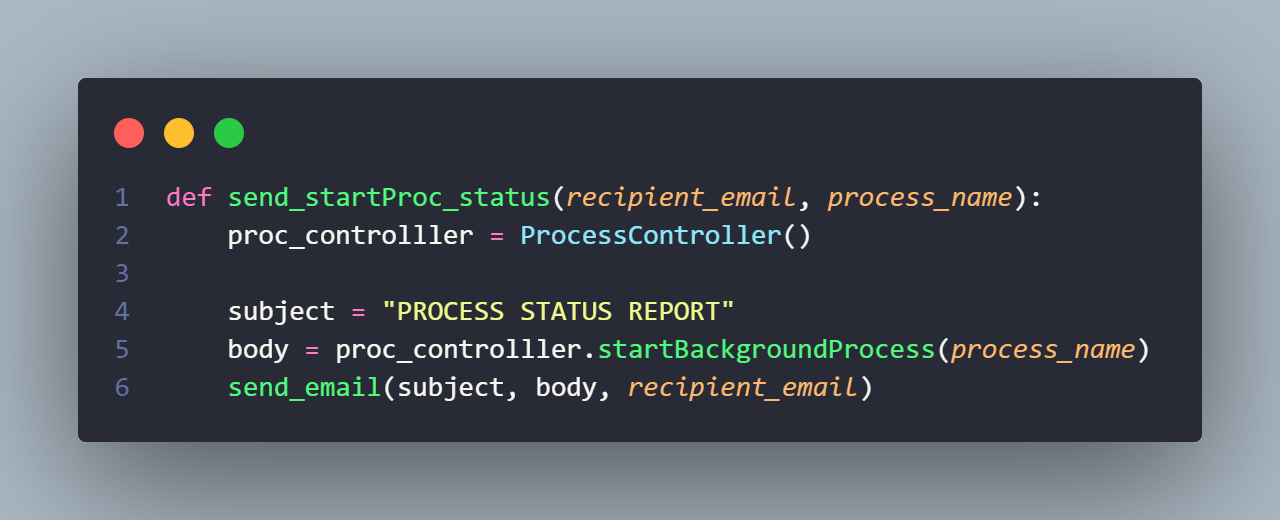
* **proc\_controller = ProcessController()**: Tạo một đối tượng của lớp **ProcessController**. Lớp này chứa các phương thức để quản lý các tiến trình nền trên hệ thống.
* **formatted\_table = proc\_controller.viewList()**: Gọi phương thức **viewList** từ đối tượng **proc\_controller** để lấy danh sách các tiến trình nền và định dạng nó thành một bảng.
* **subject = "LIST OF BACKGROUND PROCESSES"**: Đặt chủ đề của email là "LIST OF BACKGROUND PROCESSES".
* **body = formatted\_table**: Sử dụng bảng đã được định dạng như là nội dung email.
* **send\_email(subject, body, recipient\_email)**: Gọi hàm **send\_email** để gửi email với chủ đề, nội dung và không có tệp đính kèm (ảnh chụp màn hình, v.v.) cho địa chỉ email đã chỉ định.Top of Form

+hàm “send\_keyLog\_email”



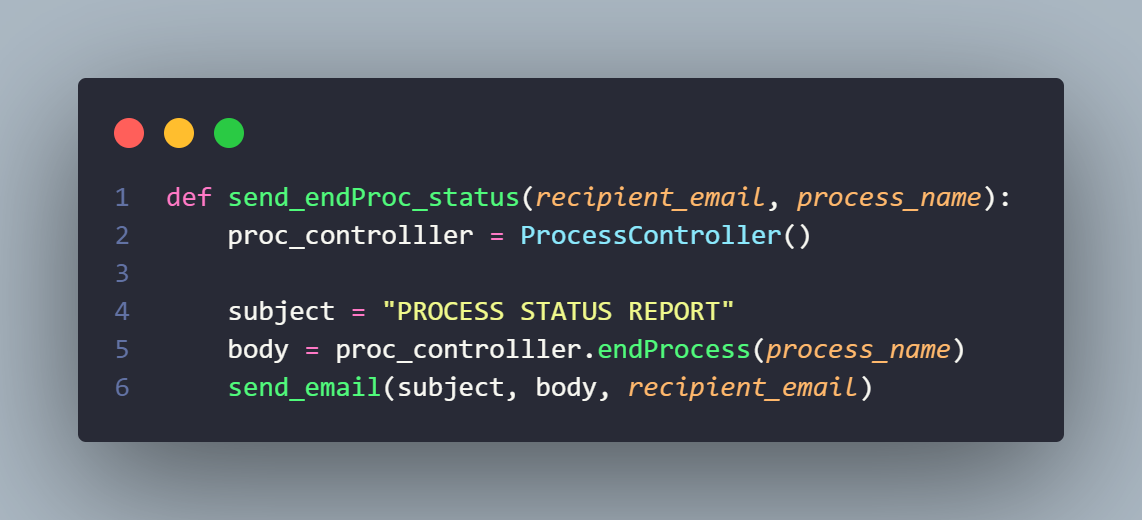
* **key\_log = keyLog()**: Tạo một đối tượng của lớp **keyLog**. Lớp này chứa các phương thức để ghi và đọc dữ liệu.
* **key\_log.clearFile("keylog.txt")**: Gọi phương thức **clearFile** của đối tượng **key\_log** để xóa nội dung của tệp "keylog.txt". Điều này giúp đảm bảo rằng chỉ có thông tin mới nhất được lưu trữ.
* **key\_log.run\_keyLog(duration)**: Gọi phương thức **run\_keyLog** để bắt đầu ghi lại các sự kiện từ bàn phím trong một khoảng thời gian **duration**.
* **subject = "KEY LOG"**: Đặt chủ đề của email là "KEY LOG".
* **body = key\_log.read\_file("keylog.txt")**: Gọi phương thức **read\_file** của đối tượng **key\_log** để đọc nội dung từ tệp "keylog.txt" và lưu trữ nó trong biến **body**.
* **if body is not None:**: Kiểm tra xem **body** có giá trị hay không.
* **send\_email(subject, body, recipient\_email)**: Gọi hàm **send\_email** để gửi email với chủ đề "KEY LOG" và nội dung là nội dung đã đọc từ tệp "keylog.txt" tới địa chỉ email đã chỉ định.

+hàm “send\_startProc\_status:



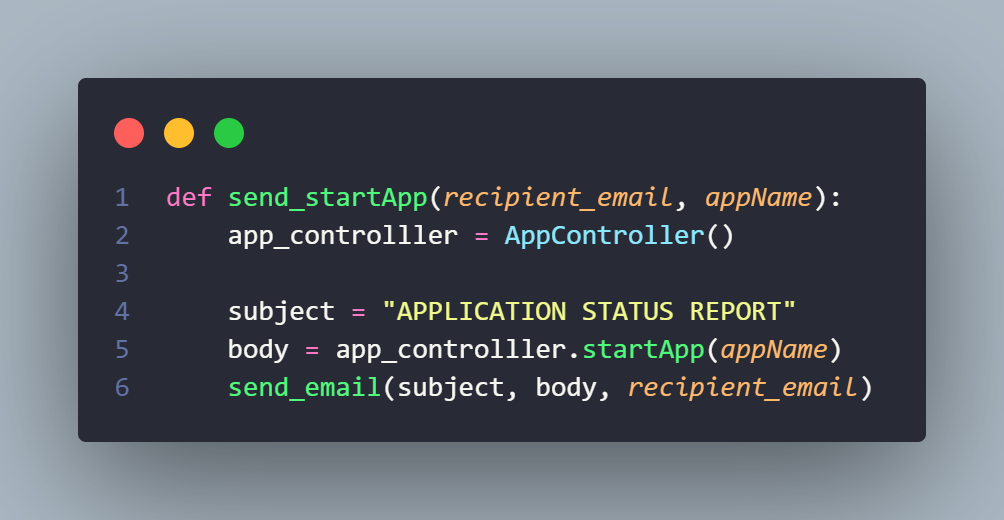
* **proc\_controller = ProcessController()**: Tạo một đối tượng của lớp **ProcessController**. Lớp này chứa các phương thức để quản lý và kiểm soát các tiến trình trên hệ thống.
* **subject = "PROCESS STATUS REPORT"**: Đặt chủ đề của email là "PROCESS STATUS REPORT".
* **body = proc\_controller.startBackgroundProcess(process\_name)**: Gọi phương thức **startBackgroundProcess** của đối tượng **proc\_controller** với **process\_name** làm đối số để lấy thông tin về việc bắt đầu một tiến trình nền và lưu trữ thông tin này trong biến **body**.
* **send\_email(subject, body, recipient\_email)**: Gọi hàm **send\_email** để gửi email với chủ đề "PROCESS STATUS REPORT" và nội dung là thông tin về việc bắt đầu một tiến trình nền tới địa chỉ email đã chỉ định.

+hàm “send\_endProc\_status”



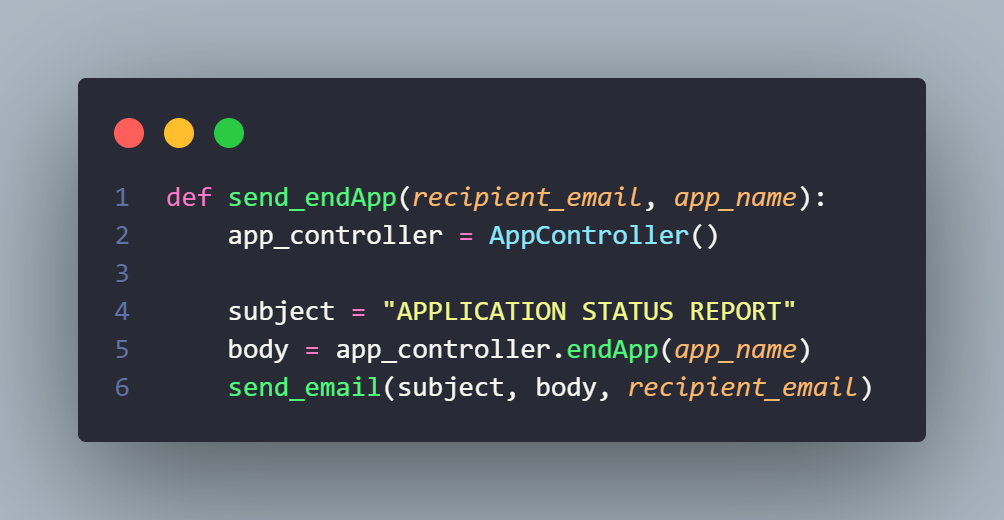
* **proc\_controller = ProcessController()**: Tạo một đối tượng của lớp **ProcessController**. Lớp này chứa các phương thức để quản lý và kiểm soát các tiến trình trên hệ thống.
* **subject = "PROCESS STATUS REPORT"**: Đặt chủ đề của email là "PROCESS STATUS REPORT".
* **body = proc\_controller.endProcess(process\_name)**: Gọi phương thức **endProcess** của đối tượng **proc\_controller** với **process\_name** làm đối số để lấy thông tin về việc kết thúc một tiến trình và lưu trữ thông tin này trong biến **body**.
* **send\_email(subject, body, recipient\_email)**: Gọi hàm **send\_email** để gửi email với chủ đề "PROCESS STATUS REPORT" và nội dung là thông tin về việc kết thúc một tiến trình tới địa chỉ email đã chỉ định.

+hàm “send\_startApp”



* **app\_controller = AppController()**: Tạo một đối tượng của lớp **AppController**. Lớp này chứa các phương thức để quản lý và kiểm soát các ứng dụng trên hệ thống.
* **subject = "APPLICATION STATUS REPORT"**: Đặt chủ đề của email là "APPLICATION STATUS REPORT".
* **body = app\_controller.startApp(appName)**: Gọi phương thức **startApp** của đối tượng **app\_controller** với **appName** làm đối số để lấy thông tin về việc khởi động một ứng dụng và lưu trữ thông tin này trong biến **body**.
* **send\_email(subject, body, recipient\_email)**: Gọi hàm **send\_email** để gửi email với chủ đề "APPLICATION STATUS REPORT" và nội dung là thông tin về việc khởi động một ứng dụng tới địa chỉ email đã chỉ định.

+hàm “send\_endApp”



* **app\_controller = AppController()**: Tạo một đối tượng của lớp **AppController**. Lớp này chứa các phương thức để quản lý và kiểm soát các ứng dụng trên hệ thống.
* **subject = "APPLICATION STATUS REPORT"**: Đặt chủ đề của email là "APPLICATION STATUS REPORT".
* **body = app\_controller.endApp(app\_name)**: Gọi phương thức **endApp** của đối tượng **app\_controller** với **app\_name** làm đối số để lấy thông tin về việc kết thúc một ứng dụng và lưu trữ thông tin này trong biến **body**.
* **send\_email(subject, body, recipient\_email)**: Gọi hàm **send\_email** để gửi email với chủ đề "APPLICATION STATUS REPORT" và nội dung là thông tin về việc kết thúc một ứng dụng tới địa chỉ email đã chỉ định.