Mô tả source code: main.py

-Các thư viện sử dụng

1. **tkinter**: để tạo giao diện người dùng.
2. **threading.Thread**: để tạo và quản lý các luồng (threads) độc lập, giúp thực hiện các tác vụ đồng thời.
3. **time**: thao tác với thời gian trong Python.
4. **imaplib**: để truy cập và thao tác với email qua giao thức IMAP.
5. **email**:để xử lý và phân tích cú pháp của email.
6. **email.header**: để giải mã tiêu đề email.
7. **re**: xử lý biểu thức chính quy.
8. **bs4 (BeautifulSoup)**: để phân tích và trích xuất thông tin từ HTML hoặc XML.
9. **sendMail**: một module được tạo ra để gửi email.
10. **powerController**: một module để tắt máy tính hoặc đăng xuất
11. **readMail**: một module để đọc và xử lý email đến.

* Các hàm đã viết

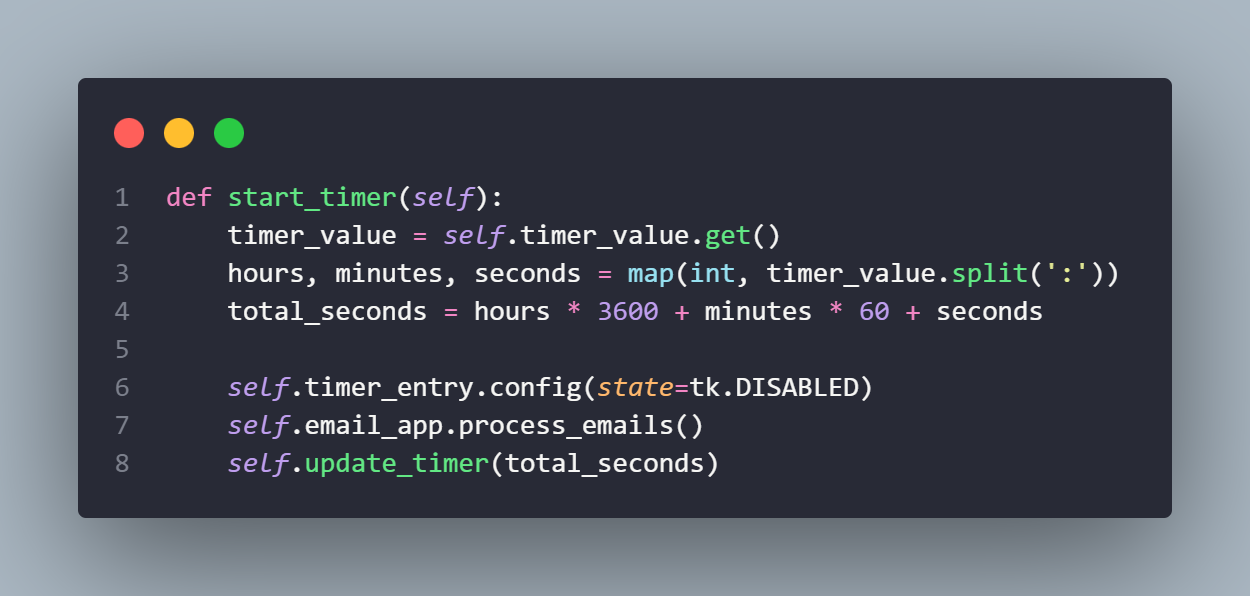
+ class TimeApp:

+hàm “\_\_init\_\_”:



* **def \_\_init\_\_(self, root, email\_app):**: Đây là phương thức khởi tạo của lớp **TimerApp**.
* **self.root = root**: Lưu trữ tham chiếu đến cửa sổ chính của ứng dụng Tkinter.
* **self.email\_app = email\_app**: Lưu trữ tham chiếu đến đối tượng **EmailProcessingApp** để giao tiếp giữa các phần của ứng dụng.
* **self.root.title("Timer App")**: Thiết lập tiêu đề của cửa sổ ứng dụng là "Timer App".
* **self.timer\_value = tk.StringVar()**: Tạo một đối tượng **StringVar** trong Tkinter để lưu trữ và cập nhật giá trị thời gian.
* **self.timer\_value.set("00:00:00")**: Khởi tạo giá trị mặc định cho thời gian là "00:00:00".
* **self.timer\_label = tk.Label(root, text="Set Timer:")**: Tạo nhãn (Label) để hiển thị thông báo "Set Timer:" trên giao diện.
* **self.timer\_label.pack()**: Đặt nhãn vào cửa sổ giao diện.
* **self.timer\_entry = tk.Entry(root, textvariable=self.timer\_value)**: Tạo một ô nhập liệu (Entry) trong giao diện, liên kết với đối tượng **StringVar** để hiển thị và cho phép người dùng nhập giá trị thời gian.
* **self.timer\_entry.pack()**: Đặt ô nhập liệu vào cửa sổ giao diện.

+hàm “start\_timer”:



* **timer\_value = self.timer\_value.get()**: Lấy giá trị thời gian nhập vào từ thành phần giao diện người dùng.
* **hours, minutes, seconds = map(int, timer\_value.split(':'))**: Tách giá trị thời gian thành các giờ, phút và giây.
* **total\_seconds = hours \* 3600 + minutes \* 60 + seconds**: Chuyển đổi thời gian thành tổng số giây.
* **self.timer\_entry.config(state=tk.DISABLED)**: Vô hiệu hóa ô nhập liệu thời gian để người dùng không thể thay đổi thời gian trong khi đếm ngược đang diễn ra.
* **self.email\_app.process\_emails()**: Gọi phương thức **process\_emails** của đối tượng **email\_app**, cho phép xử lý email trong khi bắt đầu đếm ngược.
* **self.update\_timer(total\_seconds)**: Bắt đầu đếm ngược với tổng số giây được truyền vào.

+hàm “update\_timer”:



* **if remaining\_seconds >= 0:**: Kiểm tra nếu thời gian còn lại lớn hơn hoặc bằng 0.
* **hours = remaining\_seconds // 3600**: Tính toán số giờ còn lại từ tổng số giây.
* **minutes = (remaining\_seconds % 3600) // 60**: Tính toán số phút còn lại từ số giây còn lại sau khi loại bỏ giờ.
* **seconds = remaining\_seconds % 60**: Tính toán số giây còn lại.
* **timer\_value = f"{hours:02d}:{minutes:02d}:{seconds:02d}"**: Tạo chuỗi thời gian mới với định dạng HH:MM:SS.
* **self.timer\_value.set(timer\_value)**: Cập nhật giá trị thời gian trong đối tượng **StringVar** để hiển thị trên giao diện.
* **self.root.after(1000, self.update\_timer, remaining\_seconds - 1)**: Sử dụng **after** để lên lịch gọi lại chính nó sau 1000ms (1 giây) với thời gian còn lại giảm đi 1 giây, tạo hiệu ứng đếm ngược. Nếu thời gian còn lại là 0, đếm ngược kết thúc và ô nhập liệu thời gian được kích hoạt trở lại để người dùng có thể thay đổi thời gian.

+class EmailProcessingApp:

+hàm “\_\_init\_\_”

* **self.root = root**: Lưu trữ tham chiếu đến cửa sổ chính của ứng dụng Tkinter.
* **self.root.title("Email Processing App")**: Thiết lập tiêu đề của cửa sổ là "Email Processing App".
* **self.is\_running = False**: Biến đánh dấu xem quá trình xử lý email đang chạy hay không. Ban đầu được đặt là **False**.
* **self.is\_first\_run = True**: Cờ đánh dấu cho biết đây có phải lần chạy đầu tiên không. Trạng thái ban đầu là **True**.
  + **self.start\_stop\_button**: Button để bắt đầu hoặc dừng xử lý email, khi nhấn sẽ gọi đến phương thức **start\_timer**.
  + **self.progress\_label**: Nhãn để hiển thị tiến độ của lệnh (command progress).
  + **self.progress\_bar**: Thanh tiến trình để thể hiện tiến độ của công việc.
  + **self.command\_label**: Nhãn để hiển thị thông tin về lệnh đang được thực thi.
* **self.timer\_app = TimerApp(root, self)**: Tạo một đối tượng **TimerApp** để quản lý tính năng đếm ngược thời gian.

+hàm “start\_timer



* **def start\_timer(self):**: Phương thức để bắt đầu đếm ngược thời gian từ **TimerApp**.
* **self.timer\_app.start\_timer()**: Gọi phương thức **start\_timer** từ đối tượng **TimerApp** để bắt đầu đếm ngược.

+hàm “process\_emails”:



* **def process\_emails(self):**: Phương thức để xử lý các email.
* **if not self.is\_running:**: Kiểm tra xem quá trình xử lý email có đang chạy hay không.
* **self.is\_running = True**: Đặt biến **is\_running** thành **True** để bắt đầu quá trình xử lý.
* **self.start\_stop\_button.config(text="Processing...", state=tk.DISABLED)**: Cấu hình button để hiển thị "Processing..." và vô hiệu hóa nó để tránh khả năng nhấn nút khi quá trình đang chạy.
  + **thread = Thread(target=self.process\_emails\_thread)**: Tạo một luồng mới để thực thi phương thức **process\_emails\_thread**.
  + **thread.start()**: Khởi động luồng để bắt đầu xử lý email trong nền.

+hàm “process\_emails\_thread”:



* **self.start\_timer()**: Bắt đầu đếm ngược thời gian từ **TimerApp**.
  + **command = readMail.read\_email()**: Đọc email và lấy lệnh từ email.
  + **Xử lý lệnh và cập nhật tiến độ**: Hiển thị tiến độ xử lý lệnh trên thanh tiến trình.
  + **time.sleep(random.uniform(0.01, 0.1))**: Tạm dừng một khoảng thời gian ngẫu nhiên để mô phỏng việc xử lý lệnh.
* **self.stop\_processing()**: Kết thúc quá trình xử lý email khi thời gian đếm ngược kết thúc hoặc khi lệnh là "Die".

+hàm “stop\_processing”:

* **self.is\_running = False**: Đặt biến **is\_running** thành **False** để kết thúc quá trình xử lý email.
* **self.progress\_value = 0**: Đặt giá trị tiến độ về 0.
  + **self.progress\_bar["value"] = 0**: Thiết lập giá trị thanh tiến trình về 0.
  + **self.command\_label.config(text="Command: ")**: Xóa thông tin lệnh đang thực thi.
  + **self.start\_stop\_button.config(text="Start", state=tk.NORMAL)**: Thiết lập lại nút bắt đầu để có thể nhấn và bắt đầu lại quá trình xử lý email.
  + **self.timer\_app.timer\_value.set("00:00:00")**: Đặt giá trị thời gian trong **TimerApp** về "00:00:00".

+hàm “update\_progress**”:**

****

* **self.progress\_value = value**: Cập nhật giá trị tiến độ.
* **self.progress\_bar["value"] = self.progress\_value**: Cập nhật giá trị thanh tiến trình trên giao diện.
* **self.command\_label.config(text="Command: " + command)**: Hiển thị thông tin lệnh đang thực thi lên giao diện người dùng.