# THƯ VIỆN

* imaplib: Thư viện này cung cấp chức năng truy cập và thao tác với các email được lưu trữ trên máy chủ IMAP. Nó thường được sử dụng trong các ứng dụng email và các tập lệnh tự động cần tương tác với email.
* email: Thư viện này cung cấp các lớp để làm việc với email bằng Python. Nó có thể được sử dụng để phân tích, tạo và sửa đổi email.
* email.header: Thư viện này cung cấp các hàm để giải mã header email có thể chứa các ký tự không phải ASCII. Nó thường được sử dụng trong việc phân tích và xử lý email.
* threading: Thư viện này cung cấp chức năng tạo và quản lý các luồng trong Python. Nó cũng có thể được sử dụng để chạy nhiều tác vụ đồng thời.
* time: Thư viện này cung cấp các hàm để làm việc với thời gian trong Python. Nó có thể được sử dụng để đo đạc khoảng thời gian, delay một khoảng thời gian nhất định,…
* re: Thư viện này cung cấp các phép toán khớp biểu thức chính quy trong Python. Nó có thể được dùng để tìm kiếm và thao tác chuỗi dựa trên các mẫu.
* bs4: Thư viện này cung cấp chức năng phân tích cú pháp HTML và XML trong Python. Nó có thể được dùng để trích xuất dữ liệu từ các trang web và các tài liệu có cấu trúc khác.

# GIẢI THÍCH CODE

## *readMail.py*

* Hàm extract\_email\_information:



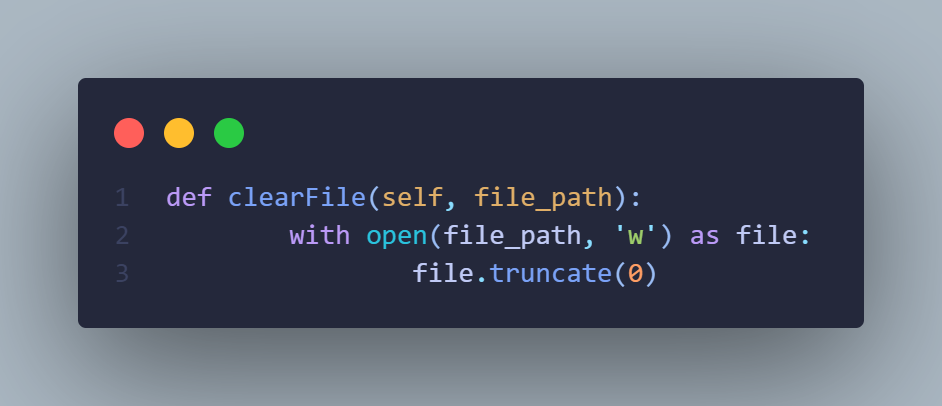
* + email\_message = email.message\_from\_bytes(raw\_email): Chuyển đổi email thô thành đối tượng email.
  + subject, encoding = decode\_header(email\_message["Subject"])[0]: Giải mã tiêu đề email.
  + if isinstance(subject, bytes): subject = subject.decode(encoding): Nếu tiêu đề là bytes, giải mã nó.
  + from\_, encoding = decode\_header(email\_message["From"])[0]: Giải mã tên người gửi.
  + if isinstance(from\_, bytes): from\_ = from\_.decode(encoding): Nếu tên người gửi là bytes, giải mã nó.
  + sender\_email = email\_message["From"]: Lấy địa chỉ email của người gửi.
  + match = re.search(r'<([^>]+)>', sender\_email): Tìm địa chỉ email thực sự từ trường "From".
  + if match: sender\_email = match.group(1): Nếu tìm thấy, cập nhật địa chỉ email của người gửi.
  + print("Subject:", subject), print("From:", from\_): In tiêu đề và người gửi.
  + if email\_message.is\_multipart(): Kiểm tra xem email có nhiều phần không.
  + for part in email\_message.walk(): Duyệt qua từng phần của email.
  + content\_type=part.get\_content\_type(), content\_disposition = str(part.get("Content-Disposition")): Lấy loại nội dung và cách hiển thị nội dung.
  + try: body = part.get\_payload(decode=True).decode() except: pass: Cố gắng giải mã nội dung của phần.
  + if content\_type == "text/plain" and "attachment" not in content\_disposition: print(body): Nếu phần là văn bản đơn giản và không phải là tệp đính kèm, in nội dung.
  + return sender\_email, body: Trả về địa chỉ email của người gửi và nội dung email.
* Hàm process\_email:
  + def process\_email(sender\_email, body): Định nghĩa một hàm tên là process\_email nhận vào địa chỉ email của người gửi và nội dung email.
  + start = 0 và end = len(body): Đặt vị trí bắt đầu và kết thúc cho việc tìm kiếm trong nội dung email.
  + Tìm kiếm các từ khóa cụ thể trong nội dung thông qua cấu trúc if body.find(“”, start, end) != -1
  + Gọi hàm tương ứng từ các module để thực hiện hành động.
* Hàm read\_email:



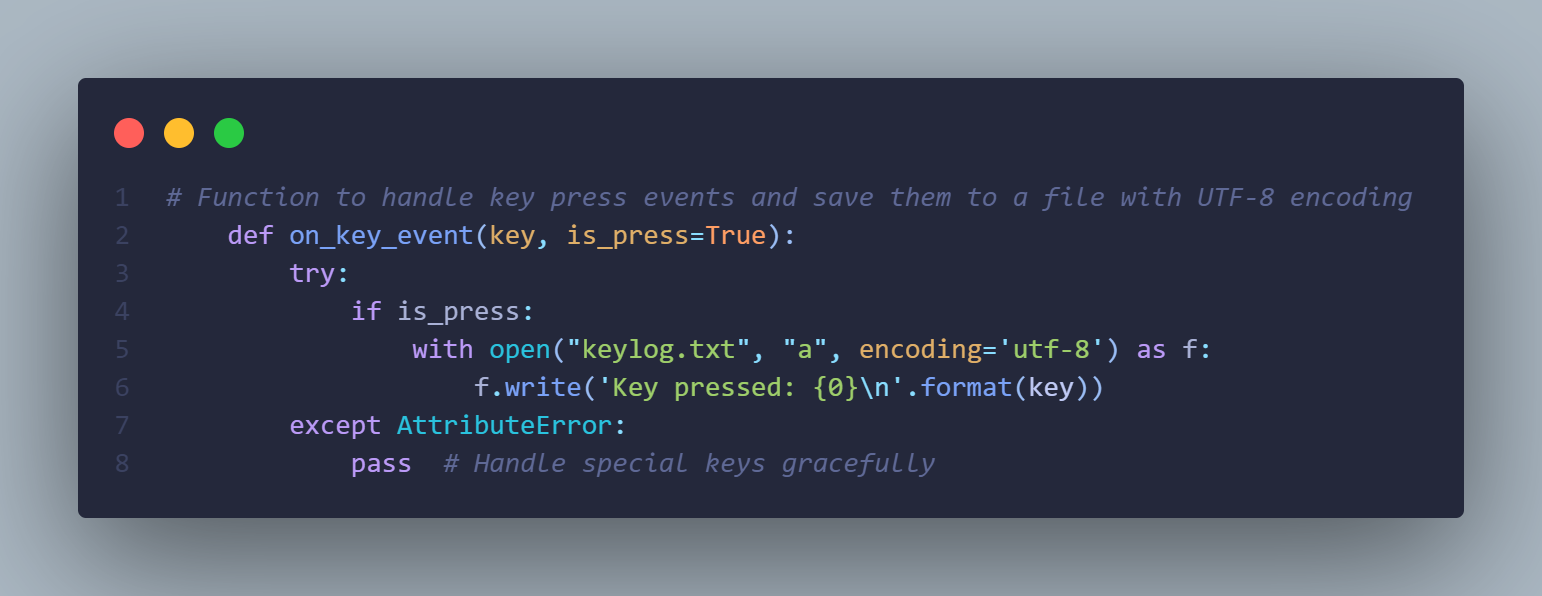
* + def read\_email(): Định nghĩa một hàm tên là read\_email.
  + username = "2d2h.computernetwork.clc.fitus@gmail.com" và password = "jvys wjpg mmmx belp": Đặt tên đăng nhập và mật khẩu cho tài khoản Gmail.
  + imap = imaplib.IMAP4\_SSL("imap.gmail.com"): Tạo một đối tượng IMAP4 với SSL để kết nối với máy chủ Gmail.
  + imap.login(username, password): Đăng nhập vào tài khoản Gmail.
  + imap.select("inbox"): Chọn hộp thư đến.
  + status, email\_ids = imap.search(None, 'UNSEEN'):Tìm tất cả các email chưa đọc.
  + if not email\_ids[0]: Kiểm tra xem có email chưa đọc nào không.
  + latest\_email\_id = email\_ids[0].split()[-1]: Lấy ID của email chưa đọc mới nhất.
  + status, email\_data = imap.fetch(latest\_email\_id, "(RFC822)"): Lấy dữ liệu của email bằng ID.
  + sender\_email, body = extract\_email\_information(email\_data[0][1]): Trích xuất thông tin từ email.
  + process\_email(sender\_email, body): Xử lý email.
  + imap.logout(): Đăng xuất khỏi tài khoản Gmail.

## *keyLog.py*

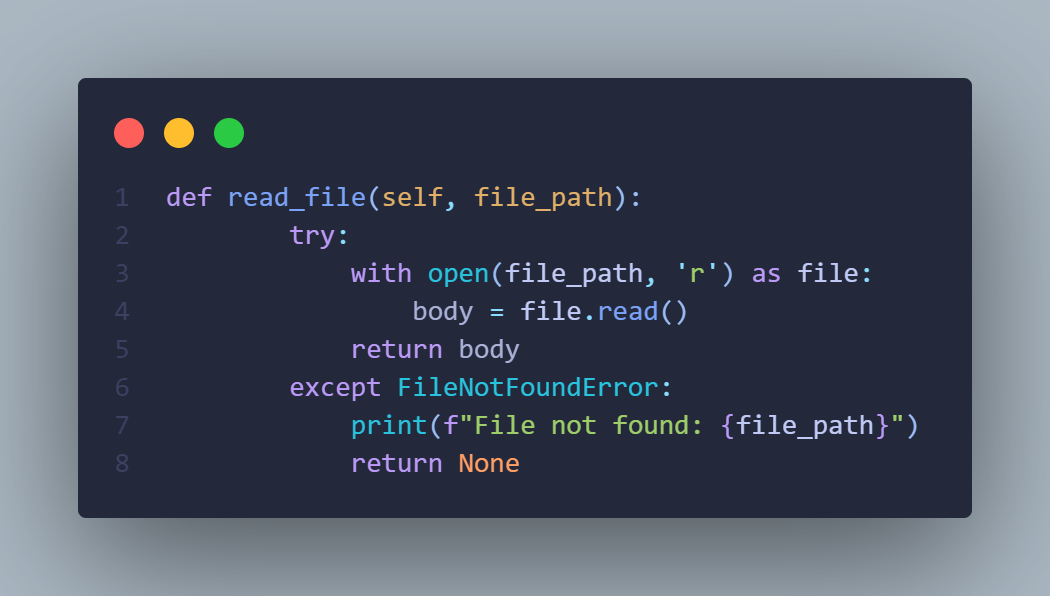
* Hàm clearFile:



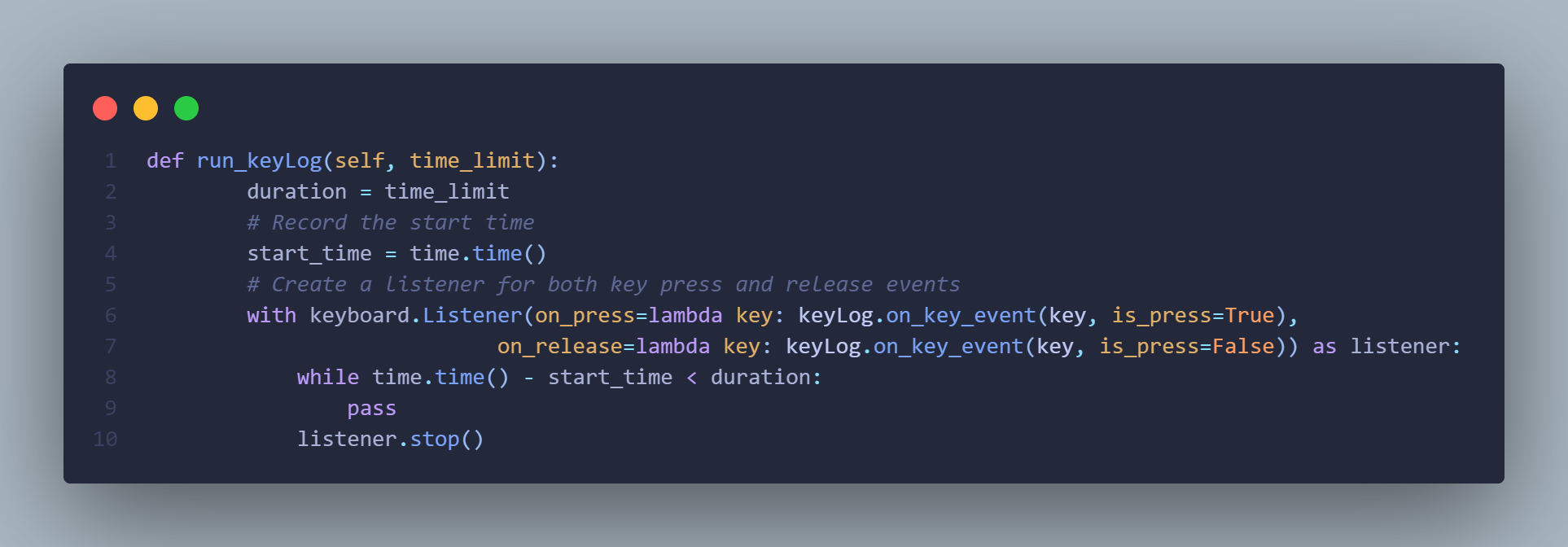
* + with open(file\_path, 'w') as file: Mở tệp đường dẫn được chỉ định trong chế độ ghi ('w'). Sử dụng từ khóa with giúp đảm bảo tệp sẽ được đóng lại sau khi hoàn thành, ngay cả khi có lỗi xảy ra.
  + file.truncate(0): Gọi phương thức truncate với tham số là 0. Phương thức này cắt độ dài của tệp xuống còn 0, tức là xóa tất cả nội dung của tệp.
* Hàm on\_key\_event:



* + def on\_key\_event(): Định nghĩa một hàm tên là on\_key\_event nhận vào hai tham số: key và is\_press.
  + try: Bắt đầu một khối try để bắt các ngoại lệ có thể xảy ra.
  + if is\_press: Kiểm tra nếu sự kiện là nhấn phím.
  + with open("keylog.txt", "a", encoding='utf-8') as f: Mở tệp "keylog.txt" trong chế độ ghi tiếp ('a') với mã hóa UTF-8. Tệp này sẽ được tạo nếu nó không tồn tại.
  + f.write('Key pressed: {0}\n'.format(key)): Ghi chuỗi 'Key pressed: {0}\n' vào tệp, với {0} được thay thế bằng giá trị của key. Điều này ghi lại phím đã được nhấn.
  + except AttributeError: Bắt ngoại lệ AttributeError, có thể xảy ra khi cố gắng truy cập một thuộc tính không tồn tại của đối tượng key.
  + pass: Nếu có AttributeError, hàm sẽ không làm gì và tiếp tục thực thi. Điều này giúp xử lý các phím đặc biệt mà không gây ra lỗi.
* Hàm read\_file:



* + def read\_file(self, file\_path): Định nghĩa một hàm read\_file nhận vào một tham số file\_path.
  + try: Bắt đầu một khối try để bắt các ngoại lệ có thể xảy ra.
  + with open(file\_path, 'r') as file: Mở tệp đường dẫn được chỉ định trong chế độ đọc ('r'). Sử dụng từ khóa with giúp đảm bảo tệp sẽ được đóng lại sau khi hoàn thành, ngay cả khi có lỗi xảy ra.
  + body = file.read(): Đọc toàn bộ nội dung của tệp và gán nó vào biến body.
  + return body: Trả về nội dung của tệp.
  + except FileNotFoundError: Bắt ngoại lệ FileNotFoundError, có thể xảy ra khi tệp không tồn tại.
  + print(f"File not found: {file\_path}"): In ra thông báo lỗi với đường dẫn tệp không tìm thấy.
  + return None: Trả về None khi không tìm thấy tệp.
* Hàm run\_keyLog:



* + def run\_keyLog(self, time\_limit): Định nghĩa một hàm run\_keyLog nhận vào tham số time\_limit.
  + duration = time\_limit: Đặt thời gian chạy cho keylogger.
  + start\_time = time.time(): Ghi lại thời gian bắt đầu.
  + with keyboard.Listener(...) as listener: Tạo một listener để theo dõi cả sự kiện nhấn phím và thả phím.
  + on\_press=lambda key: keyLog.on\_key\_event(key, is\_press=True): Khi một phím được nhấn, hàm on\_key\_event sẽ được gọi với tham số is\_press được đặt là True.
  + on\_release=lambda key: keyLog.on\_key\_event(key, is\_press=False): Khi một phím được thả, hàm on\_key\_event sẽ được gọi với tham số is\_press được đặt là False.
  + while time.time() - start\_time < duration: Vòng lặp sẽ tiếp tục cho đến khi thời gian hiện tại trừ đi thời gian bắt đầu vượt quá thời gian chạy đã định.
  + listener.stop(): Sau khi vòng lặp kết thúc, listener sẽ được dừng lại.