**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное автономное образовательное учреждение

высшего образования «Уральский федеральный университет

имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий - РТФ

Центр ускоренного обучения

**ПРОГРАММА ПРОВЕРКИ СТРУКТУРЫ ДОКУМЕНТОВ**

КУРСОВОЙ ПРОЕКТ

по теме «Реализация функции рассылки почтовых сообщений для дальнейшей интеграции в приложения»

Руководитель ст. пр. О.Л. Чагаева

Нормоконтpолер ст. пр. О.Л. Чагаева

Студенты гр. РИЗ-230939у Н.Д. Кандыба

М.Е. Устюгов

**Екатеринбург 2025**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc195100346)

[1 Постановка задачи 6](#_Toc195100347)

[1.1. Описание проблемы 7](#_Toc195100348)

[2 Описание методов 8](#_Toc195100349)

[3 Руководство пользователя 10](#_Toc195100350)

[3.1. Требования 10](#_Toc195100351)

[3.2. Структура программы 10](#_Toc195100352)

[3.3. Запустите программу: 11](#_Toc195100353)

[3.4. Пример вывода программы: 11](#_Toc195100354)

[3.5. Отладка и тестирование 13](#_Toc195100355)

[3.6. Заключение 14](#_Toc195100356)

[ЗАКЛЮЧЕНИЕ 15](#_Toc195100357)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 18](#_Toc195100358)

[ПРИЛОЖЕНИЕ А 20](#_Toc195100359)

# **ВВЕДЕНИЕ**

Электронная почта один из самых используемых инструментов для обмена информацией, постановки и выполнения задач, но зачастую рассылка и ответ на поступающие сообщения может быть весьма объемным, вед количество запросов может превышать сотни, каждый из которых ожидает структурированный ответ. В таком случае использование автоматической рассылки посредством SMTP — Simple Mail Transfer Protocol (Простой протокол передачи почты) может стать отличным решение для разработки функции рассылки сообщений. Реализация почтового клиента с поддержкой вложений представляет собой комплексную задачу интеграции сетевых протоколов, обработки данных и обеспечения кроссплатформенной совместимости. Актуальность подобной функции будет высока и спустя множество времени, ведь использовать для подобного действия сотрудника просто не представляется возможным ввиду большого количества данных, требуемых не только для составления ответа на запрос, но и указания верного адреса получателя.

Сложность реализации почтового клиента заключается в необходимости интеграции множества разнородных компонентов, требующих точной настройки и обработки исключительных ситуаций. Основные вызовы включают:

* Работу с SMTP-протоколом, так как разные почтовые сервисы (Gmail, Яндекс, Mail.ru) используют уникальные настройки портов (587 для TLS, 465 для SSL) и требования к аутентификации, сетевую ненадёжность в виде таймаутов подключения, блокировки портов фаерволом, динамические IP-ограничения со стороны почтовых провайдеров.
* Формирование MIME-сообщений, включающая в себя корректную обработку кириллических имён файлов через email.utils.encode\_rfc2231, а так же определение MIME-типов для редких расширений (например, .pptx, .csv) и гарантию отображения вложений в разных почтовых клиентах (Outlook, Gmail, Thunderbird).
* Безопасность с реализацией поддержки OAuth2 для сервисов вроде Gmail, генерацию и валидацию паролей приложений для аккаунтов с двухфакторной аутентификацией, предотвращение утечки логинов/паролей через логгирование или отладочные сообщения.
* Кроссплатформенность кодировки консоли с реализацией автоматического переключение между CP1251 (Windows) и UTF-8 (Linux/macOS), обработку корректного чтение файлов с пробелами или спецсимволами в именах (например, C:/Документы/Отчёт №3.xlsx).
* Диагностику сбоев ошибок сети (например, socket.gaierror), SMTP (например, smtplib.SMTPAuthenticationError) и файловой системы.

Цель данной курсовой работы — провести разработку функции отправки и пересылки файлов и почтовых сообщений для дальнейшей интеграции её в различные приложения. В исследовании будут рассмотрены возможности функции, её интеграция в готовое приложение на платформе Android, а также возможные запросы и выводы как отдельно реализованной функции, так и готового приложения:

1. Запрос текстового сообщения от пользователя для интеграции в отправляемое письмо
2. Предложение указать файл любого формата для включение его в состав отправленного письма
3. Вывод возможных ошибок в случае невозможности доставки

Ожидается, что результаты данного исследования помогут разработать универсальную функцию, возможность интеграции в приложение которой будет одинакова полезна у проста в применении.

# **Постановка задачи**

Целью данной курсовой работы является разработка функции рассылки почтовых сообщений для дальнейшей интеграции в различные проекты. Для достижения этой цели необходимо решить следующие задачи:

1. Выбор и описание методов:

- Определить перечень методов, которые будут использоваться для решения задачи рассылки;

- Кратко описать каждый метод, выделяя его основные принципы и особенности.

2. Разработка и реализация алгоритмов:

- Реализовать выбранные методы на языке программирования Python;

- Убедиться в корректности реализации путем тестирования.

3. Проведение экспериментов:

- Сформировать сообщение, включающее как набор простого текста, так и добавление одного или нескольких файлов разного формата;

- Запустить отправку сообщение, добавив различные ошибки как в самом сообщении, так и в протоколах и адресах для проверки вывода исключений с текстом ошибки.

4. Анализ результатов:

Проанализировать полученные данные, сопоставив отправленное и полученное письмо по следующим критериям:

- Верность адресата при составлении письма;

- Содержание отправленного и полученного текста;

- Наименование отправленного и полученного файла, его расширение, а также содержание;

- Вывод ошибок в консоль в случае обнаружения конфликтов.

5. Подготовка отчета:

- Составить отчет по проделанной работе, включающий введение, постановку задачи, описание методов, результаты экспериментов и их анализ, а также выводы и рекомендации.

- Представить результаты анализа в виде скриншотов и примеров вывода ответов в консоль для наглядности.

## **Описание проблемы**

В современном программировании часто возникает необходимость автоматизированной отправки электронных писем, будь то для уведомлений, рассылок или передачи документов. При этом важно обеспечить не только передачу текстовой информации, но и возможность прикрепления файлов различных форматов (например, .txt, .pdf, .docx, .jpg и др.). Несмотря на наличие готовых решений и библиотек, разработка собственной функции на языке Python, использующей протокол SMTP, позволяет глубже понять принципы работы с электронной почтой, а также реализовать гибкость под конкретные нужды приложения или сервиса.

Проблема заключается в создании универсальной и надежной функции, способной отправлять сообщения на почтовые ящики, корректно обрабатывать текстовые данные и различные вложения, а также обеспечивать совместимость с популярными почтовыми сервисами. Также необходимо учитывать вопросы безопасности (например, шифрование соединения), кодировки и корректной работы с файлами разных типов.

# **Описание методов**

Для реализации функции рассылки почтовых сообщений в данной курсовой работе были использованы следующие методы:

1. Основной метод «send\_email»:

Выполняет автоматизированную отправку электронного письма с текстом и вложениями через SMTP-сервер, его основные функции:

- Аутентификация на почтовом сервере (логин/пароль или токен);

- Формирование MIME-сообщений (добавление текста с обязательной подписью, прикрепление файлов (Excel, Word, тексты) с корректными именами и MIME-типами).

- Кодирование данных, таких как кириллица в именах файлов и бинарные вложения.

- Отправка через SMTP-протокол с обработкой сетевых ошибок и таймаутов.

- Вывод статуса в консоль (успех/ошибка).

1. Метод «get\_mime\_type»:

Определяет MIME-тип файла по его расширению, чтобы корректно указать тип содержимого при отправке вложения. Это необходимо для того, чтобы почтовые клиенты и серверы правильно интерпретировали формат вложенных файлов.

- Анализ расширения файла (например, .xlsx, .docx, .txt);

- Сопоставление с MIME-типом (использует стандартную библиотеку mimetypes для базовых форматов. Вручную обрабатывает специфические типы (например, Excel/Word).

- Возвращает строку MIME-типа (например .xlsx, .docx, .txt. Если тип неизвестен, то application/octet-stream (общий бинарный тип).

1. Функции проверки кодировок консоли sys.platform и sys.stdout.encoding.lower:

Функция sys.platform перенастраивает стандартные потоки вывода (stdout) и ошибок (stderr) на кодировку UTF-8 в Windows, где по умолчанию часто используется CP-1251, чтобы кириллические символы (например, в путях к файлам или тексте письма) корректно отображались в консоли и не вызывали ошибок вида UnicodeEncodeError

Функция sys.stdout.encoding.lower проверяет, установлена ли кодировка консоли в UTF-8. Если нет — выводит предупреждение и советы по исправлению.

Эти методы были реализованы и протестированы путем задания необходимых параметров в консоли, представляющих различные случаи.

# **Руководство пользователя**

Данное руководство описывает процесс использования программного обеспечения для реализации функции отправки сообщений на адреса почтовых ящиков. Программа реализована на языке программирования Python и позволяет пользователю получить сообщение с указанными данными и файлами на указанный почтовый адрес.

## **Требования**

Для запуска программы потребуется:

- Компьютер с операционной системой Windows, macOS или Linux.

- Компилятор Python (рекомендуется использование VS Code или PyCharm).

- Стандартные библиотеки и модули Python: smtplib для работы с SMTP, email.mime для формирования писем, os и mimetypes для обработки вложений, socket для сетевого взаимодействия, а также системные модули для работы с консолью.

## **Структура программы**

Программа состоит из следующих основных частей:

- Реализация функции отправки почтовых сообщений.

- Функция отправки файлов если таковые будут указаны при запросе.

- Вывод исключений в консоль если будут обнаружены ошибки при отправке или взаимодействии с сервером почты.

## **Запустите программу:**

Программа выполнит следующие действия:

- Сверит кодировку консоли для корректного отображения выводимых символов.

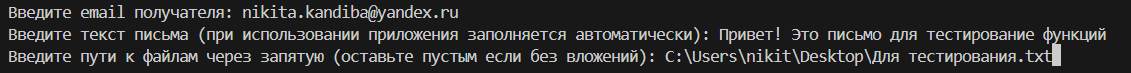
- Запросит у пользователя через диалоговое окно консоли ввод почтовый адрес получателя, текстовое сообщение, а также предложит выбрать файл;

- Выведет в диалоговое окно консоли информацию об успешной отправке почтового сообщения или ошибку с возможными рекомендациями по её устранению.

## **Пример вывода программы:**

Ниже предоставлены примеры выводов программы в зависимости от предоставленных пользователем данных.

Вывод диалогового окна консоли с предложение ввести данные для отправки, продемонстрировано на рисунке 1:



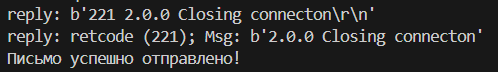
Результат выполнения кода (рис. 1)

Вывод в окно консоли IP-адреса SMTP-сервера Яндекс почты для проверки его состояния, продемонстрировано на рисунке 2:



Результат выполнения кода (рис. 2)

Если сообщение успешно отправлено, будет выведено сообщение, представленное на рисунке 3:



Результат выполнения кода (рис. 3)

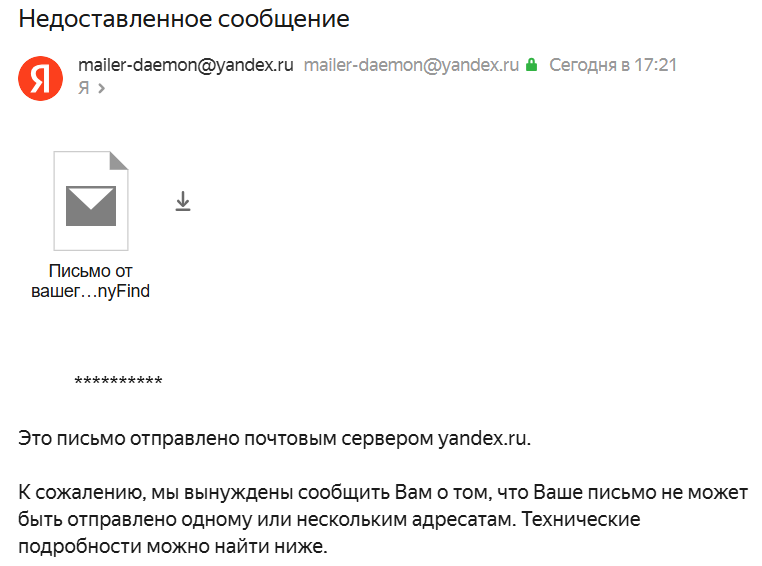
На следующих рисунках с 4 по 11 будут представлены обработки возможных ошибок и вывод соответствующего оповещения в консоль:



Ошибка в названии файла или пути к файлу (рис. 4)

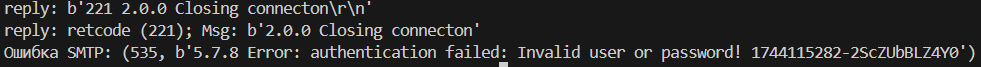
Введен ошибочный почтовый адрес получателя, вместо правильного «nikita.kandiba@yandex.ru» указан ошибочный «nikita.kandibaaa@yandex.ru»:





Неправильно указан почтовый ящика получателя (рис. 5)

В случае, если пользователь ввёл ошибочный пароль:



Неправильно указан пароль приложения от почтового ящика (рис. 6)

Если пользователь ввёл неверный адрес SMTP-сервера или порт:



Неверный SMTP-сервер или порт (рис. 7)

Если сервер по какой-либо причине не активен или запрос к нему не может быть выполнен по техническим причинам сервера в течении 60 секунд:



Истекло время запрос к серверу (рис. 8)

В случае, если пользователь использует кодировку, отличную от UTF-8:



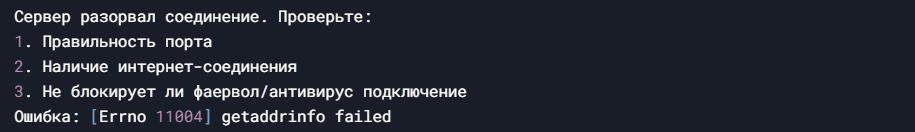
Предупреждение об использовании кодировки отличной от рекомендованной (рис. 9)

Если пользователь желает отправить тип файла, отличный от стандартных:



Предупреждение о неизвестном формате файла (рис. 10)

Если сервер разорвал соединение, пользователю будет предложены варианты устранения ошибки, а также код ошибки:



Предупреждение о неизвестном формате файла (рис. 11)

## **Отладка и тестирование**

Если программа не компилируется или не работает корректно:

- Убедитесь, что все библиотеки и заголовочные файлы подключены.

- Проверьте синтаксис и правильность написания кода.

- Используйте средства отладки компилятора для выявления ошибок.

## **Заключение**

Данная программа предоставляет функцию автоматической рассылки сообщений для простого и удобного интегрирования в любой программный продукт. Различные функции и обработки ошибок позволят удобно наладить работу в случае неполадок, а также предоставить полноценный функционал взаимодействия с сервисом различных почтовых серверов.

# **ЗАКЛЮЧЕНИЕ**

В данной курсовой работе была произведена разработка функции рассылки почтовых сообщений посредством взаимодействия с SMTP-сервером. Основной целью исследования было рассмотрение возможности функции, её интеграции в готовые приложение, а также возможные запросы и выводы как отдельно реализованной функции, так и готового приложения:

Были реализованы и протестированы следующие методы и функции:

1. Метод «send\_email»;

2. Метод «get\_mime\_type», определяющий MIME-тип файла по его расширению, чтобы корректно указать тип содержимого при отправке вложения;

3. Функции проверки кодировок.

**Результаты анализа**

1. Метод «send\_email»:

Выполняет автоматизированную отправку электронного письма с текстом и вложениями через SMTP-сервер, его основные функции.

2. Метод «get\_mime\_type»:

Определяет MIME-тип файла по его расширению, чтобы корректно указать тип содержимого при отправке вложения

3. Функции проверки кодировок:

Функция sys.platform перенастраивает стандартные потоки вывода (stdout) и ошибок (stderr) на кодировку UTF-8 в Windows, где по умолчанию часто используется CP-1251, чтобы кириллические символы (например, в путях к файлам или тексте письма) корректно отображались в консоли и не вызывали ошибок вида UnicodeEncodeError

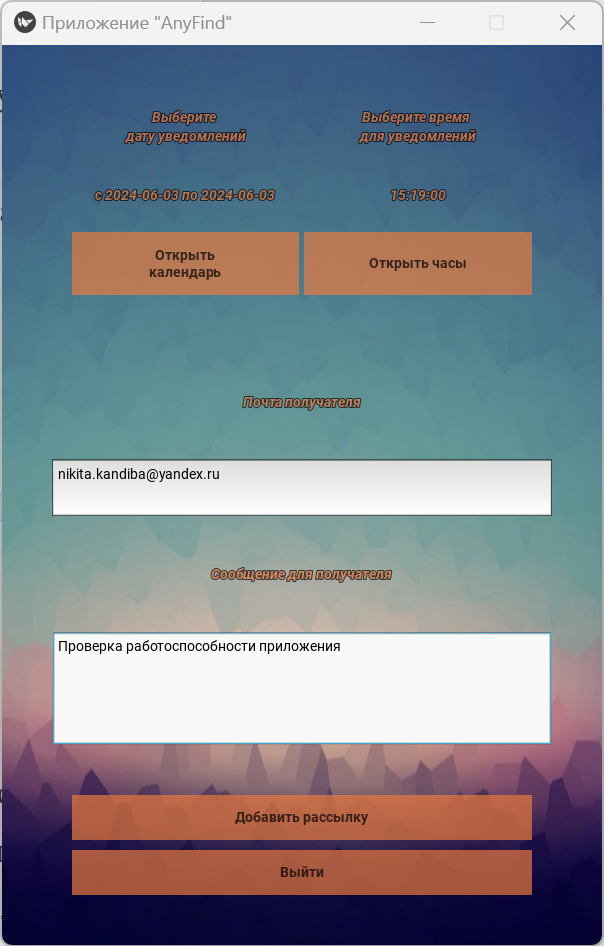
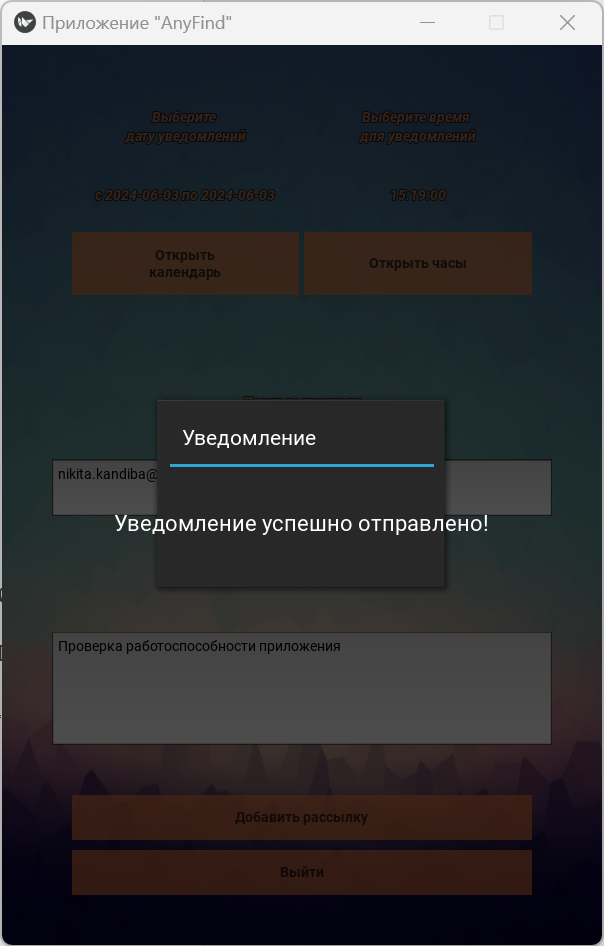
Функция sys.stdout.encoding.lower проверяет, установлена ли кодировка консоли в UTF-8. Если нет — выводит предупреждение и советы по исправлению.

**Интеграция разработанной функции**

Разработанная функция была интегрирована в приложение на базе Android, выполняющее парсинг различных сайтов по заданным пользователями критериям и формирование отчета в виде структурированных результатов поиска в Excel или Word. В приложении подобная функция может быть применена при желании пользователя выслать полученные таблицы на свою почту.

Функция рассылки реализована в меню окна «Рассылки».

Она позволяет, предоставляет пользователю возможность указать почту для получения рассылки, а также указать время, в которое данная рассылка будет приходить на указанную почту. Продемонстрировано на рисунке 12.

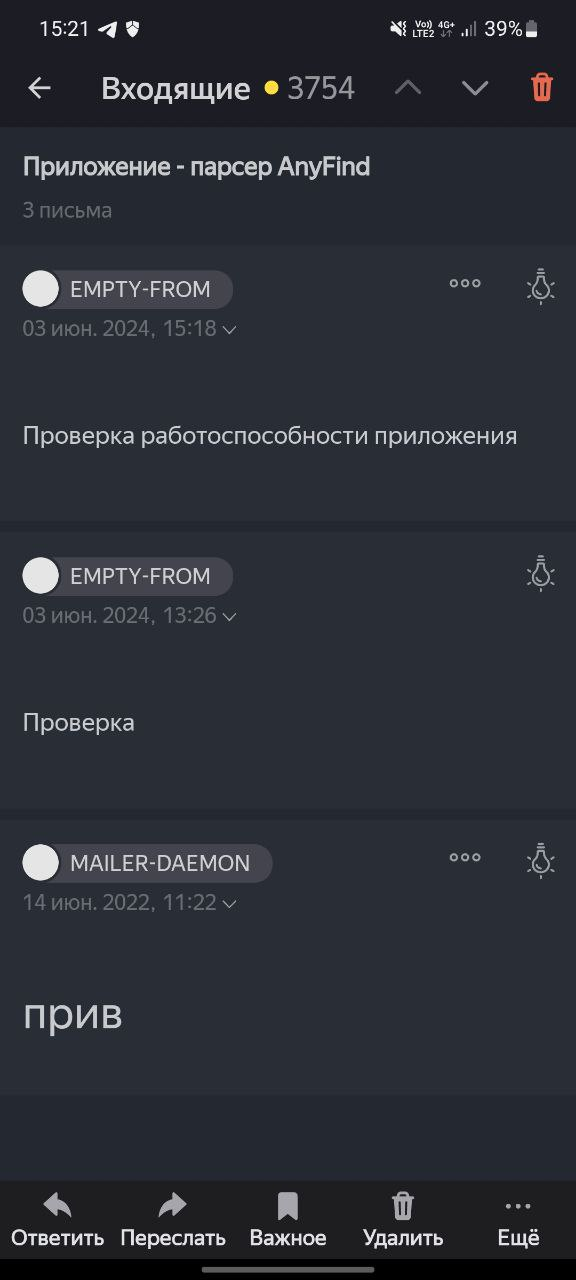


Рисунок 12 – Окно рассылки и проверка входящих сообщений на почту от приложения

**Заключительные выводы**

Результаты экспериментов показали, что разработанную функцию можно успешно интегрировать в различные приложения и успешно использовать полученные методы для достижения корректной рассылки с возможностью в любой момент получить от функции обратную связь для устранения ошибок.

**Ссылка на github:**

https://github.com/AfsAlik/Kurs\_Proj\_Pocht\_rass.git

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Codeby.net. Пример кода отправки email с вложением [Электронный ресурс]. — 2021. — Режим доступа: https://codeby.net/threads/python-email-smtp.73198/, свободный. — Дата обращения: 03.04.2025.
2. Django Software Foundation. Отправка email в Django [Электронный ресурс]. — 2023. — Режим доступа: https://django.fun/docs/django/ru/4.0/topics/email/, свободный. — Дата обращения: 06.04.2025.
3. GeekBrains. Отправка писем с Python: краткое руководство [Электронный ресурс]. — 2023. — Режим доступа: https://gb.ru/posts/python-mail, свободный. — Дата обращения: 31.03.2025.
4. GeekBrains. Что такое SMTP и как он работает? [Электронный ресурс]. — 2022. — Режим доступа: https://gb.ru/posts/chto-takoe-smtp, свободный. — Дата обращения: 30.03.2025.
5. Mail.ru Group. Настройка SMTP для отправки почты [Электронный ресурс]. — 2022. — Режим доступа: https://help.mail.ru/mail-help/settings/smtp, свободный. — Дата обращения: 04.04.2025.
6. Python Software Foundation. Модуль email.message [Электронный ресурс]. — 2023. — Режим доступа: https://docs.python.org/3/library/email.message.html, свободный. — Дата обращения: 07.04.2025.
7. Python Software Foundation. Модуль smtplib [Электронный ресурс]. — 2023. — Режим доступа: https://docs.python.org/3/library/smtplib.html, свободный. — Дата обращения: 28.03.2025.
8. PythonWorld. Работа с электронной почтой и вложениями в Python [Электронный ресурс]. — 2023. — Режим доступа: https://pythonworld.ru/tipy-dannyx-v-python/pis’ma-s-vlozhenijami.html, свободный. — Дата обращения: 01.04.2025.
9. Skillbox. Как отправить письмо через SMTP-сервер на Python [Электронный ресурс]. — 2023. — Режим доступа: https://skillbox.ru/media/code/python\_smtp\_email/, свободный. — Дата обращения: 05.04.2025.
10. Stepik. Курс «Программирование на Python» — раздел об отправке писем [Электронный ресурс]. — 2023. — Режим доступа: https://stepik.org/course/67, свободный. — Дата обращения: 30.03.2025.
11. Журнал «Хакер». Как отправить письмо с вложением на Python [Электронный ресурс]. — 2022. — № 10. — Режим доступа: https://xakep.ru/2022/10/12/python-email/, свободный. — Дата обращения: 02.04.2025.
12. Культин Н. Б. Современный Python. Полное руководство. — СПб.: БХВ-Петербург, 2021. — 704 с.
13. Лутц М. Изучаем Python. — СПб.: Питер, 2021. — 1216 с.
14. Мельников И. В. Компьютерные сети: учебное пособие. — М.: Академия, 2018. — 320 с.
15. Хабр. SMTP для начинающих: теория и практика [Электронный ресурс]. — 2021. — Режим доступа: https://habr.com/ru/post/524908/, свободный. — Дата обращения: 06.04.2025.
16. Хабр. Как отправить email с вложением на Python [Электронный ресурс]. — 2022. — Режим доступа: https://habr.com/ru/post/678456/, свободный. — Дата обращения: 29.03.2025.
17. Хабр. Разбор MIME-структуры email-сообщений в Python [Электронный ресурс]. — 2020. — Режим доступа: https://habr.com/ru/post/481704/, свободный. — Дата обращения: 01.04.2025.
18. Яндурин А. Н. Основы программирования на Python. — СПб.: БХВ-Петербург, 2020. — 384 с.

# **ПРИЛОЖЕНИЕ А**

import os

import smtplib

import socket

import sys

import locale

import io

import mimetypes

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.application import MIMEApplication

# Для сложных имен файлов используется библиотека email.utils:

from email.utils import make\_msgid, formatdate, formataddr

from email.utils import encode\_rfc2231

# Принудительная установка кодировки

if sys.platform == "win32":

    sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')

    sys.stderr = io.TextIOWrapper(sys.stderr.buffer, encoding='utf-8')

# Проверка системной кодировки

if sys.stdout.encoding.lower() != 'utf-8':

    print("Внимание! Системная кодировка может вызвать проблемы с кириллицей.")

    print(f"Текущая кодировка консоли: {locale.getpreferredencoding()}")

    print("Рекомендуется:")

    print("1. Для Windows: изменить кодировку консоли на UTF-8")

ПРИЛОЖЕНИЕ А

    print("2. Использовать IDE с поддержкой UTF-8 (VS Code, PyCharm)")

def get\_mime\_type(file\_path):

    """Определение MIME-типа с использованием стандартной библиотеки"""

    mime\_type, encoding = mimetypes.guess\_type(file\_path)

    return mime\_type or 'application/octet-stream'

def send\_email():

    # Ввод данных

    sender\_email = "AnyFind@yandex.ru" # Почтовый адрес бота

    sender\_password = input("Введите пароль приложения для доступа к боту рассылки: ") # Специальный пароль приложения для входа в почту

    smtp\_server = "smtp.yandex.ru" # SMTP сервер для взаимодействия с почтой Яндекса

    smtp\_port = "587" # Используемый порт для взаимодействия с почтой Яндекса

    receiver\_email = input("Введите email получателя: ")

    subject = "Письмо от вашего помощника AnyFind"

    body = input("Введите текст письма (при использовании приложения заполняется автоматически): ")

    file\_paths = input("Введите пути к файлам через запятую (оставьте пустым если без вложений): ").split(',')

    # Формирование письма

    msg = MIMEMultipart()

    msg['From'] = sender\_email

ПРИЛОЖЕНИЕ А

    msg['To'] = receiver\_email

    msg['Subject'] = subject

    # Добавление текста с подписью

    full\_body = f"{body}\n\n--\n Письмо сформировано ботом AnyFind"

    msg.attach(MIMEText(full\_body, 'plain'))

    # Добавление вложений

    for file\_path in file\_paths:

        file\_path = file\_path.strip()

        if not file\_path:

            continue

        if not os.path.isfile(file\_path):

            print(f"Ошибка: файл {file\_path} не найден!")

            return

        try:

            # Получаем имя файла с кириллицей

            file\_name = os.path.basename(file\_path)

            # Определяем MIME-тип

            mime\_type = get\_mime\_type(file\_path)

            # Читаем файл в бинарном режиме

            with open(file\_path, 'rb') as f:

                file\_data = f.read()

            # Создаем MIME-часть

ПРИЛОЖЕНИЕ А

            part = MIMEApplication(file\_data, Name=file\_name)

            # Кодируем имя файла для заголовков

            encoded\_name = encode\_rfc2231(file\_name, charset='utf-8')

            # Устанавливаем заголовки

            part['Content-Disposition'] = f'attachment; filename\*=\'\'{encoded\_name}'

            part.add\_header('Content-Type', mime\_type, name=encoded\_name)

            msg.attach(part)

        except Exception as e:

            print(f"Ошибка при обработке файла {file\_path}: {str(e)}")

            return

    server = None

    try:

        # Явное указание протокола в зависимости от порта

        if smtp\_port == 465:

            server = smtplib.SMTP\_SSL(

                host=smtp\_server,

                port=smtp\_port,

                timeout=60  # Указание таймаута

            )

        else:

            server = smtplib.SMTP(

                host=smtp\_server,

                port=smtp\_port,

ПРИЛОЖЕНИЕ А

                timeout=60  # Указание таймаута

            )

            server.starttls()  # Принудительный STARTTLS для порта 587

        # Включение debug-режима для просмотра обмена с сервером

        server.set\_debuglevel(1)

        server.login(sender\_email, sender\_password)

        server.sendmail(sender\_email, receiver\_email, msg.as\_string())

        print("Письмо успешно отправлено!")

    # Исключения в случае ошибок и вывод сообщения для пользователя

    except (socket.timeout, smtplib.SMTPServerDisconnected) as e:

        print(f"Сервер разорвал соединение. Проверьте:\n"

              f"1. Правильность порта\n"

              f"2. Наличие интернет-соединения\n"

              f"3. Не блокирует ли фаервол/антивирус подключение\n"

              f"Ошибка: {str(e)}")

    except socket.gaierror as e:

        print(f"Ошибка подключения: Проверьте правильность SMTP-сервера и порта. ({str(e)})")

    except smtplib.SMTPException as e:

        print(f"Ошибка SMTP: {str(e)}")

    except Exception as e:

        print(f"Неожиданная ошибка: {str(e)}")

    finally:

        if server is not None:

            try:

                server.quit()

            except:

                pass

ПРИЛОЖЕНИЕ А

if \_\_name\_\_ == "\_\_main\_\_":

    send\_email()