**Як відправляти імейли у Python: приклади для різних потреб**

[*Підписуйтеся на Telegram-канал «DOU #tech», щоб не пропустити нові технічні статті*](https://t.me/dou_tech)

Бадьорого дня! На звʼязку Олександр, я — Full Stack інженер у продуктовій студії Railsware. Мій колега вже ділився порадами щодо [листів у Ruby](https://dou.ua/forums/topic/41039/?from=company_posts), а я сьогодні хочу поговорити про те, як налагодити відправки імейлів у Python.

Попри свій вже солідний вік, ця мова програмування, схоже, вічно молода і знаходить нові сфери застосування. Тож цей матеріал стане в пригоді тим, хто працює з вебзастосунками на Python і хоче розширити їхню функціональність шляхом сповіщень чи інших електронних листів.

Зауважу, що приклади протестовані з Python 3.6.9, але мають працювати і з найсвіжішими версіями Python 3.

**Як можна надсилати імейли з Python**

Зазвичай, йдеться про два основні способи, якими надсилають листи у Python. Це використання SMTP або ж транзакційного поштового сервісу. Розгляньмо їх ближче і розберімося, як вони працюють.

* SMTP

Тут уже є хороша новина. У стандартну бібліотеку Python вбудовано модуль smtplib, який використовується для надсилання електронних листів через SMTP-з’єднання. Модуль працює зі [стандартним протоколом RFC 821](https://www.rfc-editor.org/rfc/rfc821), тому ніяких додаткових налаштувань та хитрощів не потрібно.

* Транзакційні поштові сервіси

Якщо вам потрібно надсилати імейли з Python-застосунку, не використовуючи вбудований SMTP, ви можете інтегрувати сторонній API транзакційної електронної пошти. Чому йти цим шляхом? Найчастіше, такий вибір роблять, коли висока швидкість доставки — критично важлива. Крім того, більшість сторонніх API-сервісів пропонують аналітичні інструменти та можливості масштабування. Це може знадобитися, коли ваш проєкт зростатиме.

**Відправляємо імейли у Python: крок за кроком**

**Через SMTP**

Імпортуємо вбудований модуль smtplib за допомогою наступного оператора:

import smtplib

Далі, щоб надіслати лист, створіть один SMTP-об’єкт:

smtpObj = smtplib.SMTP( [host [, port]] )

Що в параметрах?

* *host* — необов’язковий аргумент, який вказує на хост, де запущений ваш SMTP-сервер. Можна вказати IP-адресу хоста або доменне ім’я;
* *port* — вкажіть порт, на якому працює SMTP-сервер якщо він відрізняється від стандартного STMP порта (465).

Об’єкт SMTP має метод екземпляра *sendmail*, який використовується для надсилання повідомлення. У нього є три параметри:

* *sender* — рядок з адресою відправника;
* *receivers* — список рядків, по одному для кожного отримувача;
* *message* — повідомлення у вигляді рядка, відформатованого за RFC 822.

smtpObj.sendmail(sender, receivers, message)

Щоб переконатися, що модуль електронної пошти імпортувався, як треба, і отримати повний опис його класів та аргументів, ви можете запустити команду в інтерактивному сеансі роботи з Python:

help(smtplib)

У [документації Python](https://docs.python.org/3/library/smtplib.html) можна більше дізнатися про решту об’єктів SMTP (наприклад, smtp.ehlo; smtp.send\_message(msg) і т.д.) і способи їхнього застосування.

А ось приклад простого скрипта на Python, який показує, як можна надіслати імейл з локального SMTP.

import smtplib

sender = 'from@example.com'

receivers = ['to@example.com']

message = """From: From Person <from@example.com>

To: To Person <to@example.com>

Subject: SMTP email example

This is a test message.

"""

try:

    smtpObj = smtplib.SMTP('localhost')

    smtpObj.sendmail(sender, receivers, message)

    print("Successfully sent email")

except SMTPException:

    pass

Спойлер: цей код не спрацює.

Передусім, вам необхідний робочий SMTP-сервер. Крім того, більшість одержувачів відхиляють листи, які надходять з ненадійних джерел. Та й отримати всі необхідні підтвердження та сертифікати, щоб інші сервери могли схвалювати ваші імейли, не так і просто.

Використання стороннього API для функцій відправки у вашому застосунку на Python — найкращий спосіб вирішити цю проблему. У таких сервісів більшість «нудної» роботи для доставки імейлів вже зроблено, тож вам не потрібно витрачати час на налаштування власного SMTP-серверу.

**Через сервіс доставки імейлів**

На ринку є низка пропозицій таких API. Для нашого прикладу візьму розробку колег, [Mailtrap Email Sending](https://mailtrap.io/" \t "_blank). До того ж приємно користуватися українським продуктом :-)

Перед тим, як надсилати будь-які типи листів через Mailtrap, потрібно підключити та підтвердити своє доменне ім’я. Про те, як це зробити, є [окремий гайд](https://help.mailtrap.io/article/69-sending-domain-verification).

А вже після цього повертайтеся у свій застосунок на Python і встановіть [Python-клієнт Mailtrap](https://pypi.org/project/mailtrap/" \t "_blank) (потрібна версія від 2.0.0):

pip install mailtrap

Далі — створюйте об’єкт імейлу і заповнюйте змінні (наприклад, email та ім’я) своїми обліковими даними Mailtrap. Важливо, що тут треба вказати адресу, де домен відправника підтверджений.

import mailtrap as mt

*# створюємо обʼєкт імейлу*

mail = mt.Mail(

    sender=mt.Address(email="mailtrap@example.com", name="Mailtrap Test"),

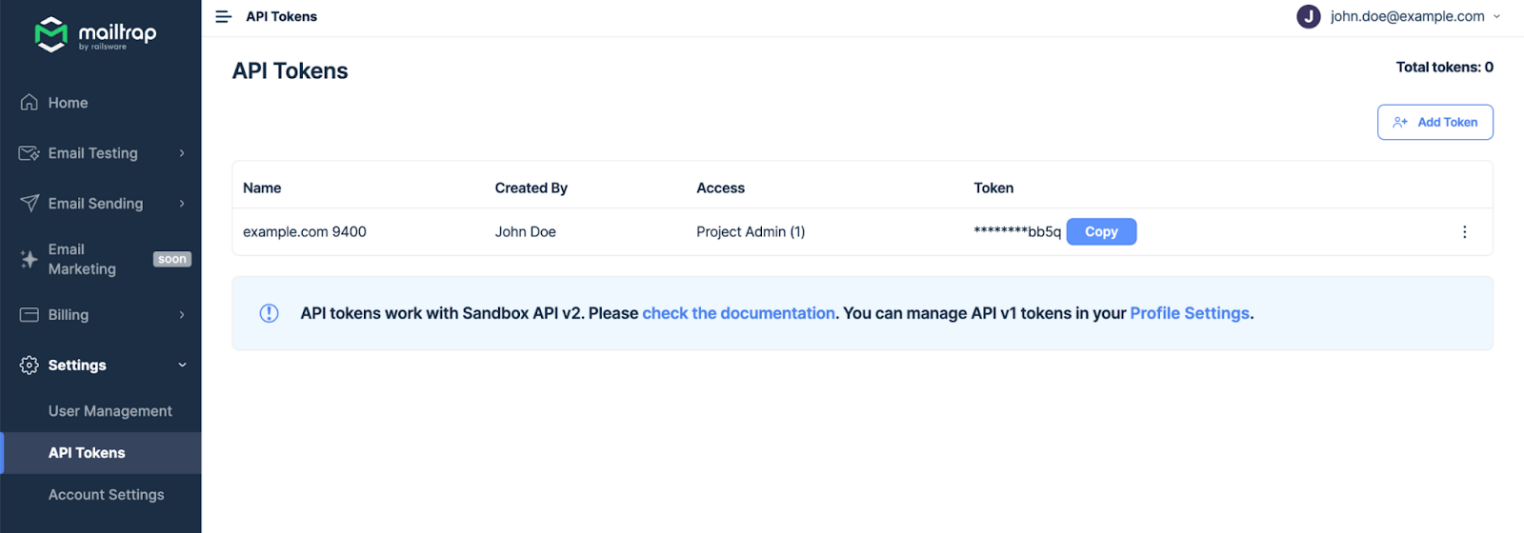
    to=[mt.Address(email="your@email.com")],

    subject="Тестовий імейл для спільноти ДОУ",

    text="Українські інженери круті!",

)

Щоб надіслати простий лист, потрібно також створити клієнт з вашим API-токеном. Це робиться в акаунті Mailtrap: у меню Settings ліворуч, вкладка API Tokens. Дані токену можна скопіювати просто звідти.



Коли все готово, відправляйте листа за допомогою скрипта:

*# створюємо клієнт та відправляємо*

client = mt.MailtrapClient(token="your-api-key")

client.send(mail)

Ось повніший приклад коду для складніших імейлів (приміром, з отримувачами в копії та прихованій копії, HTML-тілом, вкладеннями тощо):

import base64

from pathlib import Path

import mailtrap as mt

welcome\_image = Path(\_\_file\_\_).parent.joinpath("welcome.png").read\_bytes()

mail = mt.Mail(

    sender=mt.Address(email="mailtrap@example.com", name="Mailtrap Test"),

    to=[mt.Address(email="your@email.com", name="Your name")],

    cc=[mt.Address(email="cc@email.com", name="Copy to")],

    bcc=[mt.Address(email="bcc@email.com", name="Hidden Recipient")],

    subject="Тестовий імейл для спільноти ДОУ",

    text="Вітаю, ви відправили тестовий імейл!",

    html="""

    <!doctype html>

    <html>

      <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

      </head>

      <body style="font-family: sans-serif;">

        <div style="display: block; margin: auto; max-width: 600px;" class="main">

          <h1 style="font-size: 18px; font-weight: bold; margin-top: 20px">

            Вітаю, ви відправили тестовий імейл!

          </h1>

      </body>

    </html>

    """,

    category="Test",

    attachments=[

        mt.Attachment(

            content=base64.b64encode(welcome\_image),

            filename="welcome.png",

            disposition=mt.Disposition.INLINE,

            mimetype="image/png",

            content\_id="welcome.png",

        )

    ],

    headers={"X-MT-Header": "Custom header"},

    custom\_variables={"year": 2023},

)

client = mt.MailtrapClient(token="your-api-key")

client.send(mail)

**Як відправляти HTML-мейли у Python**

Ми розібрали, як надсилати HTML-листи за допомогою Python-клієнта Mailtrap. Утім, й інші пакети, наприклад, пакет *email* у Python, також підтримують HTML-листи. Попрацюймо тепер з типом повідомлення MIME, який може поєднувати HTML/CSS та звичайний текст. У Python за це відповідає модуль *email.mime*.

Краще написати текстову і HTML-версії окремо, а потім об’єднати їх через екземпляр MIMEMultipart("alternative"). Тобто, таке повідомлення матиме два варіанти відображення: якщо HTML-версія чомусь не відобразиться, текстова все одно буде доступна.

Що на вході:

*# спочатку імпортуйте потрібні компоненти*

import smtplib, ssl

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

port = 2525

smtp\_server = "live.smtp.mailtrap.io"

login = "api"

password = "1a2b3c4d5e6f7g" *# вставте згенрований Mailtrap пароль*

sender\_email = "mailtrap@example.com"

receiver\_email = "new@example.com"

message = MIMEMultipart("alternative")

message["Subject"] = "комбіноване повідомлення"

message["From"] = sender\_email

message["To"] = receiver\_email

*# вставте текстову частину*

text = """\

Привіт,

Радимо почитати нову статтю в блозі Mailtrap про відмінності імейл-протоколів:

https://mailtrap.io/blog/imap-vs-pop3-vs-smtp-email-protocols/

Сподіваємося, цей матеріал буде корисним!

"""

*# вставте HTML-частину*

html = """\

<html>

  <body>

    <p>Привіт,<br>

       Радимо почитати нову статтю в блозі Mailtrap <a href="https://mailtrap.io/blog/imap-vs-pop3-vs-smtp-email-protocols/">про відмінності імейл-протоколів</a></p>

    <p> Сподіваємося, цей матеріал буде <strong>корисним</strong>!</p>

  </body>

</html>

"""

*# конвертуйте обидві частини в обʼєкти MIMEText та додайте їх до повідомлення MIMEMultipart*

part1 = MIMEText(text, "plain")

part2 = MIMEText(html, "html")

message.attach(part1)

message.attach(part2)

context=ssl.create\_default\_context()

*# відправляєте імейл*

with smtplib.SMTP(smtp\_server, port) as server:

    server.starttls(context=context)

    server.login(login, password)

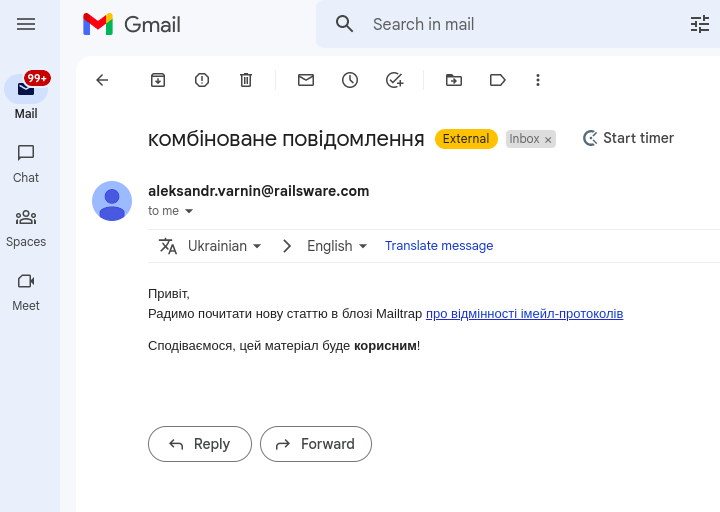
    server.sendmail(

        sender\_email, receiver\_email, message.as\_string()

    )

print('Відправлено')

І отримуємо такий лист на пошту:



**Як надіслати імейл з вкладеннями**

Ще один крок до опанування імейлів у Python — прикріплення файлів. Вкладення — це також MIME-об’єкти, але нам потрібно кодувати їх за допомогою [модуля base64](https://docs.python.org/3/library/base64.html), який кодує всі двійкові дані в ASCII-символи.

Кілька важливих зауважень про вкладення:

* Python дозволяє прикріплювати стандартний .txt, RTF та інші популярні текстові файли, зображення, аудіофайли і навіть застосунки. Вам просто потрібно використовувати відповідний клас типу: email.mime.audio.MIMEAudio або email.mime.image.MIMEImage.
* Більше інформації та [прикладів](https://docs.python.org/3/library/email.examples.html) — як завжди, у [документації](https://docs.python.org/3/library/email.mime.html) Python.
* У цьому випадку, розмір має значення: надсилати файли більші за 20 МБ — погана ідея.

Найчастіше у транзакційних листах використовуються PDF-файли: нам зазвичай надсилають квитанції, квитки, підтвердження замовлень, медичні аналізи та багато інших документів, пов’язаних з автентифікацією. Сподіваючись, що вже скоро цивільна авіація знову літатиме над Україною, розглянемо, як відправити посадковий талон у PDF.

На вході:

import smtplib, ssl

*# імпортуємо відповідні модулі*

from email import encoders

from email.mime.base import MIMEBase

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

port = 2525

smtp\_server = "live.smtp.mailtrap.io"

login = "api"

password = "1a2b3c4d5e6f7g" *# вставте згенерований Mailtrap пароль*

subject = "Приклад посадкового талону"

sender\_email = "mailtrap@example.com"

receiver\_email = "new@example.com"

message = MIMEMultipart()

message["From"] = sender\_email

message["To"] = receiver\_email

message["Subject"] = subject

*# Додаємо тіло імейлу*

body = "Ось приклад того, як можна надіслати посадковий талон у вкладенні за допомогою Python"

message.attach(MIMEText(body, "plain"))

filename = "Ваш\_Посадковий\_Талон.pdf"

*# Відкриваємо PDF-файл у бінарному режимі*

*# Припускаємо, що цей файл лежить у тій само директорії, звідки ви запускаєте свій Python-скрипт*

with open(filename, "rb") as attachment:

*# Тип контенту "application/octet-stream" означає, що MIME-вкладення – бінарний файл*

    part = MIMEBase("application", "octet-stream")

    part.set\_payload(attachment.read())

*# Кодуємо у base64*

encoders.encode\_base64(part)

*# Додаємо гедер*

part.add\_header(

    "Content-Disposition",

    f"attachment; filename= {filename}",

)

*# Додаємо вкладення до повідомлення та конвертуємо його у string*

message.attach(part)

text = message.as\_string()

context = ssl.create\_default\_context()

*# відправляємо імейл*

with smtplib.SMTP(smtp\_server, port) as server:

    server.starttls(context=context)

    server.login(login, password)

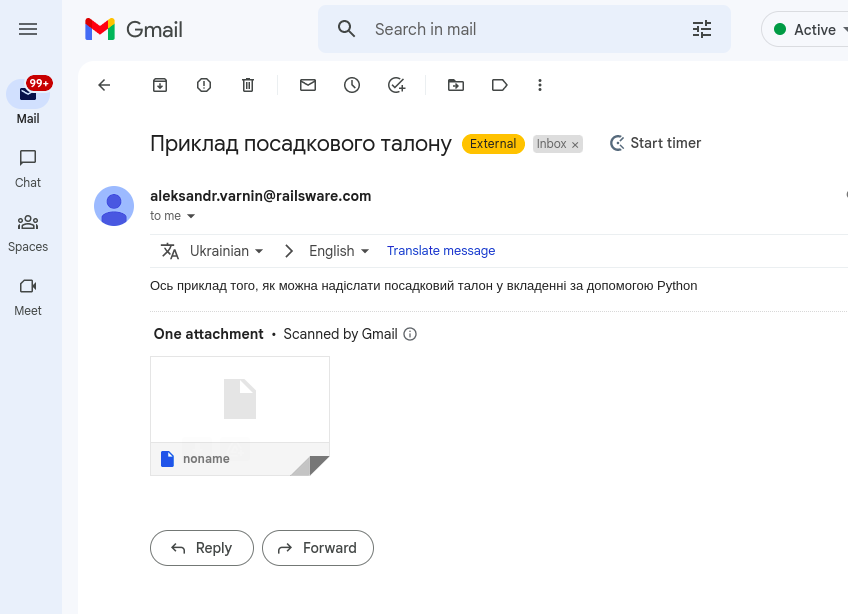
    server.sendmail(

        sender\_email, receiver\_email, text

    )

print('Відправлено')

І на виході отримуємо таке:



Щоб прикріпити декілька файлів, достатньо викликати метод message.attach() відповідну кількість разів.

**Як надіслати імейл багатьом відправникам**

Потрібно надсилати кілька листів різним одержувачам і персоналізувати їх? У Python є для цього свої фішки. Наприклад, щоб додати ще кілька отримувачів, ви можете просто ввести їхні адреси через кому і додати cc та bcc.

Утім, якщо ви робите масову розсилку, Python врятує вас за допомогою циклів. Один з варіантів — створити базу даних у форматі .csv (припустимо, що вона лежить у тій же папці, що і ваш Python-скрипт).

Транзакційні чи маркетингові імейли нерідко звертаються до нас на імʼя. Ось як це можна зробити у Python. Спершу, організуймо список у вигляді простої таблиці з двома стовпчиками: ім’я та електронна адреса. Виглядатиме це так:

#імʼя,імейл Тарас Григорович,taras@shevchenko.ua Іван Якович,ivan@franko.ua

Код нижче відкриє файл і буде проходитися ним рядок за рядком, замінюючи {name} на значення зі стовпчика «name».

import csv, smtplib, ssl

from email.mime.text import MIMEText

port = 2525

smtp\_server = "live.smtp.mailtrap.io"

login = "api"

password = "1a2b3c4d5e6f7g" *# вставте згенерований Mailtrap пароль*

sender = "mailtrap@example.com"

message = """Привіт {name}, дякуємо за ваше замовлення! Ми вже працюємо з ним і скоро з вами звʼяжемося."""

context = ssl.create\_default\_context()

with smtplib.SMTP(smtp\_server, port) as server:

   server.starttls(context=context)

   server.login(login, password)

   with open("contacts.csv") as file:

       reader = csv.reader(file)

       next(reader)  *# пропускаємо рядок гедеру*

       for name, email in reader:

           mime\_message = MIMEText(message.format(name=name), \_charset="UTF-8")

           mime\_message['Subject'] = 'Підтвердження замовлення'

           mime\_message['To'] = email

           mime\_message['From'] = sender

           server.sendmail(

               sender,

               email,

               mime\_message.as\_string()

           )

           print(f'Відправлено до {name}')

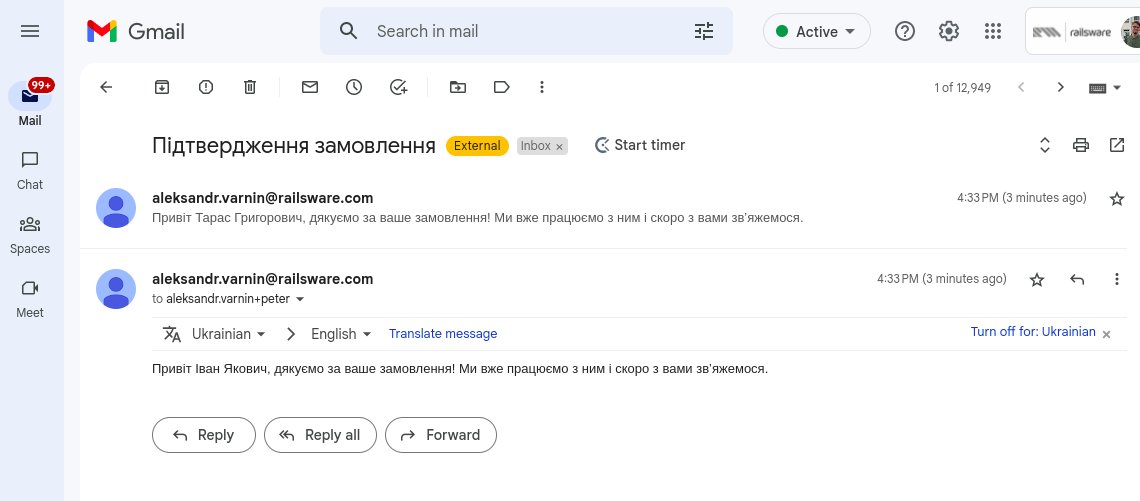
Після запуску скрипту отримаємо наступну відповідь:

Відправлено до Тарас Григорович

Відправлено до Іван Якович

>>>

І тоді в нашому тестовому інбоксі в Mailtrap побачимо два листи: один для Тараса Григоровича, один для Івана Яковича. Вони надійдуть одночасно.



**Як відправляти імейли із зображеннями**

Зображення, навіть якщо вони входять до тіла листа, все одно лишаються по суті вкладеннями. Вони бувають трьох типів:

* CID-вкладення (вбудовуються як MIME-об’єкт) ;
* Base64-зображення через Data URL (вбудовуються в текст);
* зображення за зовнішнім посиланням.

Кожен з цих способів працює по-різному в різних імейл-клієнтах. Наприклад, CID-вкладення не працює в Thunderbird, а Data URL зображення не працює в Gmail. При цьому зображення за посиланнями працюють всюди, але деякі клієнти ховають їх за замовчуванням.

Тому ми розглянемо перші два способи, між якими ви можете обирати в залежності від того, що вам відомо про вашу цільову аудиторію. Зображення за посиланням окремо розглядати не будемо, оскільки це звичайний HTML і з точки зору кодування нічого не потребує.

Щоб додати CID-вкладення, створюємо багатокомпонентне MIME-повідомлення за допомогою компонента MIMEImage:

*# import all necessary components*

import smtplib, ssl

from email.mime.text import MIMEText

from email.mime.image import MIMEImage

from email.mime.multipart import MIMEMultipart

port = 2525

smtp\_server = "live.smtp.mailtrap.io"

login = "api"

password = "1a2b3c4d5e6f7g" *# вставте згенерований Mailtrap пароль*

sender\_email = "mailtrap@example.com"

receiver\_email = "new@example.com"

message = MIMEMultipart("alternative")

message["Subject"] = "Тест для CID-зображення"

message["From"] = sender\_email

message["To"] = receiver\_email

*# write the HTML part*

html = """\

<html>

 <body>

   <img src="cid:catimage">

 </body>

</html>

"""

part = MIMEText(html, "html")

message.attach(part)

*# Припускаємо, що файл зображення лежить у тій само директорії, звідки ви запускаєте свій Python-скрипт*

fp = open('cat.jpg', 'rb')

image = MIMEImage(fp.read())

fp.close()

*# Вказуємо ID відповідно до img src у HTML-частині*

image.add\_header('Content-ID', '<catimage>')

message.attach(image)

context = ssl.create\_default\_context()

*# відправляємо імейл*

with smtplib.SMTP(smtp\_server, port) as server:

    server.starttls(context=context)

    server.login(login, password)

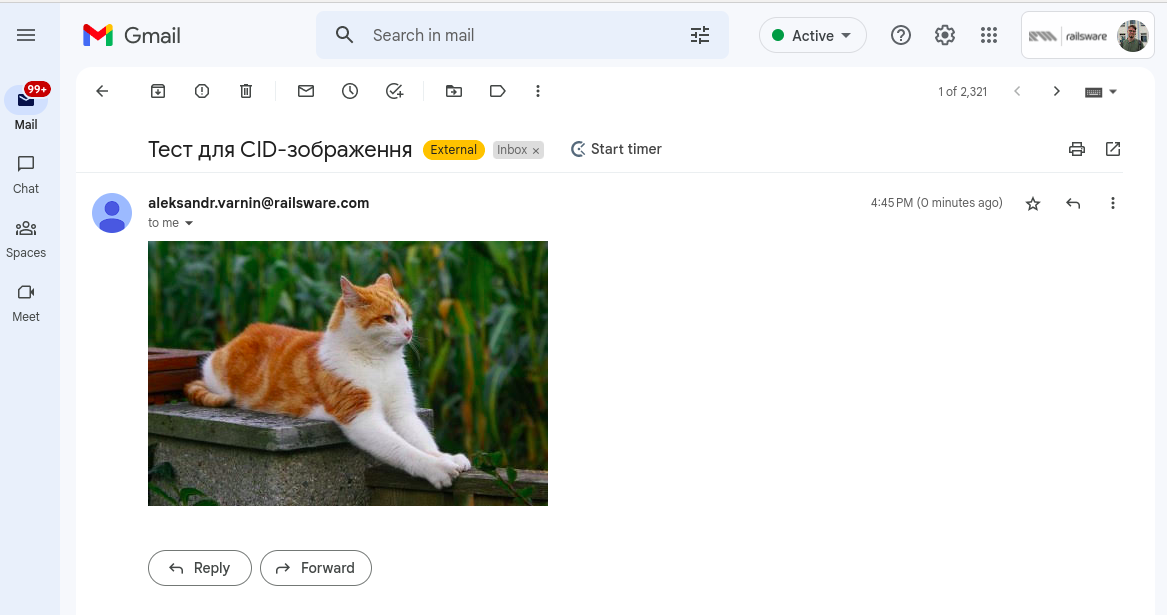
    server.sendmail(

        sender\_email, receiver\_email, message.as\_string()

    )

print('Відправлено')

На виході отримуємо наступне:



Тепер розгляньмо, як вбудувати зображення в кодуванні base64 через Data URL.

Для цього використаємо модуль base64 і поекспериментуємо з одним і тим самим файлом зображення:

*# спочатку імпортуємо потрібні компоненти*

import smtplib, ssl

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

import base64

port = 2525

smtp\_server = "live.smtp.mailtrap.io"

login = "api"

password = "1a2b3c4d5e6f7g" *# вставте згенерований Mailtrap пароль*

sender\_email = "mailtrap@example.com"

receiver\_email = "new@example.com"

message = MIMEMultipart("alternative")

message["Subject"] = "Вбудовування в текст листа"

message["From"] = sender\_email

message["To"] = receiver\_email

*# Припускаємо, що файл зображення лежить у тій само директорії, звідки ви запускаєте свій Python-скрипт*

encoded = base64.b64encode(open("mailtrap.jpg", "rb").read()).decode()

html = f"""\

<html>

 <body>

   <img src="data:image/jpg;base64,{encoded}">

 </body>

</html>

"""

part = MIMEText(html, "html")

message.attach(part)

context = ssl.create\_default\_context()

*# відправляємо імейл*

with smtplib.SMTP(smtp\_server, port) as server:

    server.starttls(context=context)

    server.login(login, password)

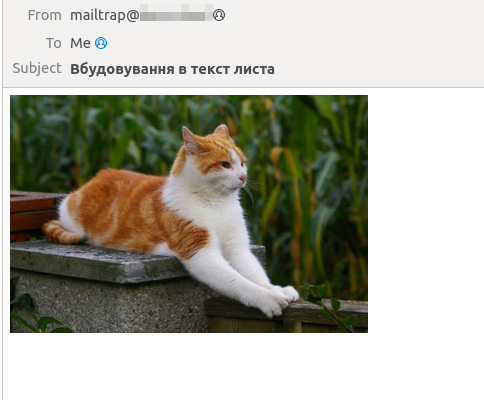
    server.sendmail(

        sender\_email, receiver\_email, message.as\_string()

    )

print('Відправлено')

І тоді на виході отримуємо ось таке:



У цьому випадку зображення вбудовується безпосередньо в тіло HTML-повідомлення. Python закодував наше зображення у форматі jpg. Тому, якщо перейдемо на вкладку HTML Source, побачимо довгий рядок даних зображення у файлі *img src*.

**Як відправляти імейли за допомогою Python через Gmail**

Налаштовувати продакшн-сервер варто, коли ви вже готові надсилати свої імейли на реальні адреси одержувачів. Це також залежить від ваших потреб, цілей та вподобань: ваш локальний хост або будь-який зовнішній SMTP.

Gmail — один з найпопулярніших варіантів, тому розгляньмо його докладніше. Через високі стандарти безпеки, Google відмовився від стандартної автентифікації SMTP.

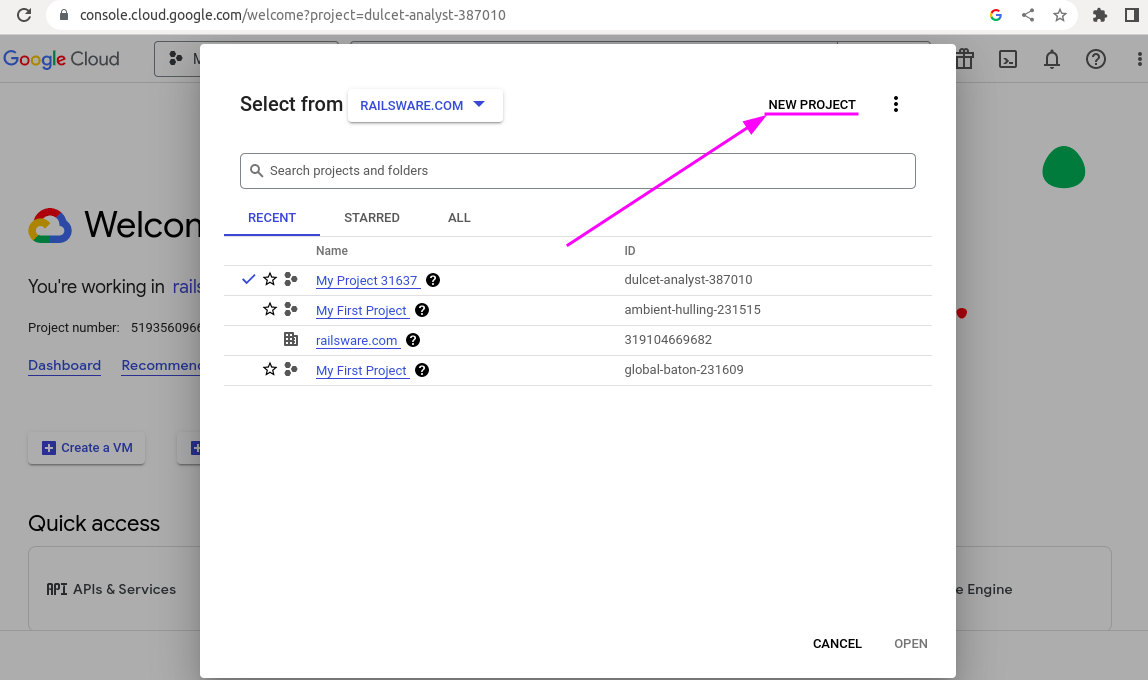
Колись звичайну автентифікацію з ім’ям користувача та паролем можна було використати для автентифікації в Google через SMTP, але зараз ця функціональність вже не доступна. Небезпека полягає в тому, що цей спосіб змушує вас зберігати пароль від облікового запису для використання в SMTP-клієнті.

Замість цього Google радить використовувати OAuth2-автентифікацію. Вона складніша з точки зору розробки і вимагає реєстрації вашого застосунку в Google. Але якщо уважно все імплементувати, ви матимете змогу користуватися протоколом SMTP для відправки листів, як і з будь-яким іншим SMTP-провайдером.

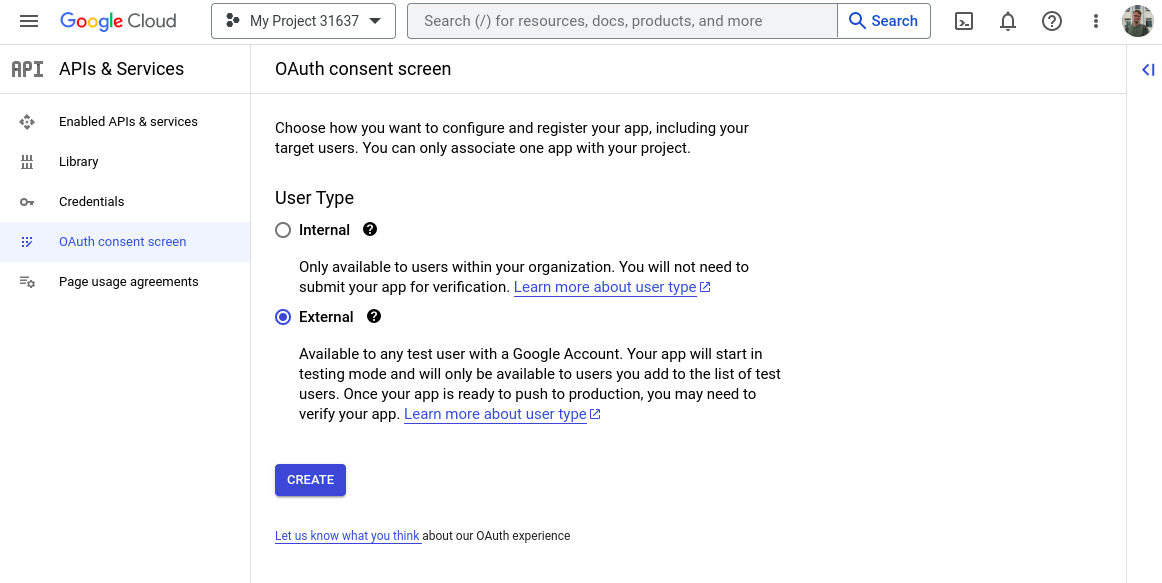
**Крок 1. Реєстрація застосунку в Google Cloud Console**

Щоб мати змогу надсилати листи через екаунт Gmail, потрібно зареєструвати свій застосунок в Google Cloud Console. Для цього потрібно скористатися протоколом авторизації OAuth2. Як це зробити?

* Зайдіть у свою [Google Cloud Console](https://console.developers.google.com/). Там оберіть свій проєкт чи створіть новий.



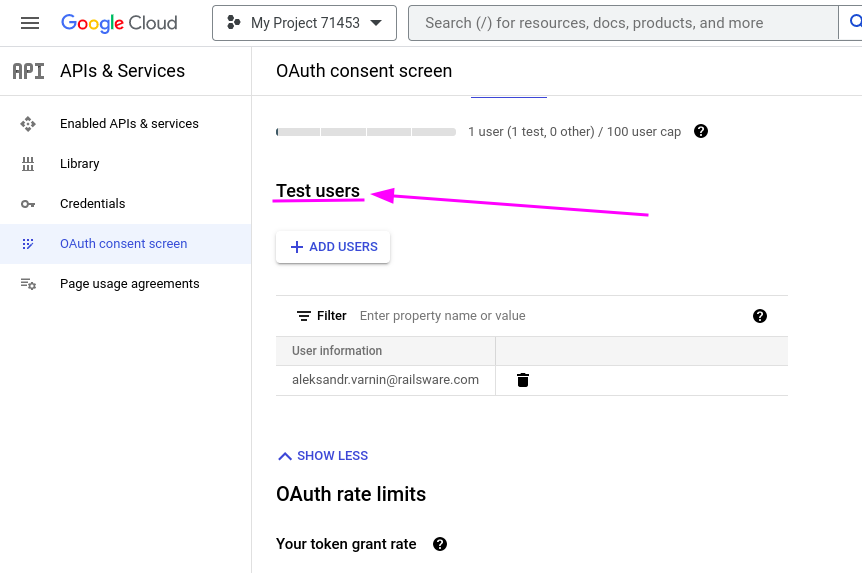
* Якщо ви створили новий проєкт, вам буде потрібно налаштувати OAuth consent screen. Спочатку треба обрати цільову аудиторію вашого застосунку.



Якщо ви користувач GSuite в організації, вам буде доступний тип Internal, який означає, що вашим застосунком зможуть користуватися (тобто, відправляти листи в нашому випадку), тільки користувачі з вашої організації.

Тип External передбачає, що застосунок буде доступним всім в інтернеті. Але для цього Google має перевірити ваш застосунок: поки він не буде авторизованим, ним зможуть користуватися тільки користувачі зі списку тестувальників.

У цій статті не будемо проговорювати процес верифікації застосунку, тому припустимо, що ви додали необхідних користувачів до списку тестувальників.



Зверніть увагу, що користувач якій створив застосунок (власник, розробник) в Google Cloud і юзер, якій буде ним користуватися для відправки листів, можуть бути різними користувачами. Наприклад, ви можете створити застосунок за допомогою вашого особистого акаунту, але відправляти листи за допомогою акаунту, пов’язаного з вашим бізнесом, як-от [support@example.ua](mailto:support@example.ua).

* У вкладці Credentials оберіть Create credentials — і там створіть новий OAuth client ID (якщо вже маєте такий, можна використовувати його).
* На питання про Application type для цього ID оберіть Desktop app та оберіть зрозумілу назву (наприклад «Test Python sending»).
* На наступному екрані отримаєте свої client ID та секретний ключ. Збережіть їх (і, звісно, ні з ким не діліться).

**Крок 2. Отримання refresh token**

Автентифікація OAuth складається з двох кроків. Спочатку ваш застосунок має отримати так званий refresh token. Це довгостроковий токен, якій ви маєте зберігати в безпечний спосіб. Маючи refresh token, ви можете отримати короткостроковий access token, якій буде вже безпосередньо використовуватися для автентифікації SMTP-з’єднання.

Для простоти ми напишемо код для отримання refresh token як окремої програми, але ви можете поєднати цей код в одному застосунку з відправкою листа.

import urllib.parse

import urllib.request

import json

oauth2\_client\_id = '....apps.googleusercontent.com' *# вставте з Google консолі*

oauth2\_client\_secret = '...' *# вставте з Google консолі*

auth\_code\_request\_params = {

   'client\_id': oauth2\_client\_id,

   'redirect\_uri': 'urn:ietf:wg:oauth:2.0:oob',

   'scope': 'https://mail.google.com/',

   'response\_type': 'code'

}

permission\_url = f'https://accounts.google.com/o/oauth2/auth?{urllib.parse.urlencode(auth\_code\_request\_params)}'

print('Перейдить за посиланням', permission\_url)

authorization\_code = input('Введить код авторизації: ')

refresh\_token\_request\_params = {

   'client\_id': oauth2\_client\_id,

   'client\_secret': oauth2\_client\_secret,

   'code': authorization\_code,

   'redirect\_uri': 'urn:ietf:wg:oauth:2.0:oob',

   'grant\_type': 'authorization\_code'

}

params\_encoded = urllib.parse.urlencode(refresh\_token\_request\_params).encode('UTF-8')

response = urllib.request.urlopen('https://accounts.google.com/o/oauth2/token', params\_encoded).read().decode('UTF-8')

response\_data = json.loads(response)

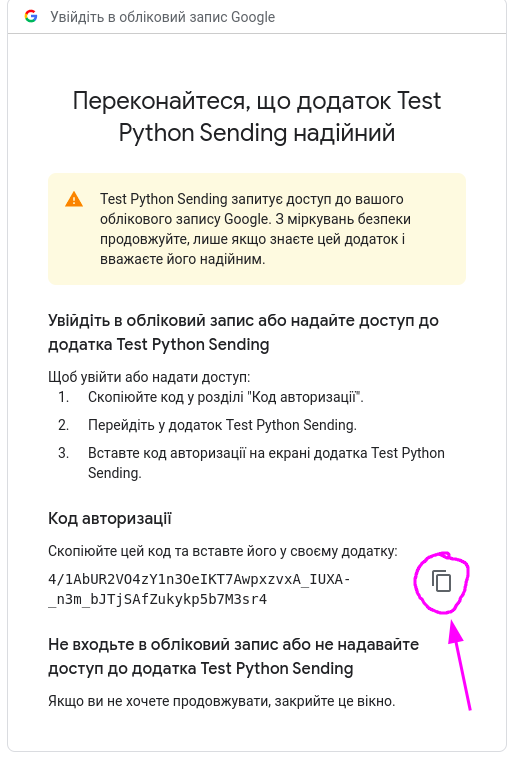
refresh\_token = response\_data['refresh\_token']

print('Ось ваш refresh token:', refresh\_token)

Коли ви запустите цей код, програма запропонує перейти за посиланням. За ним Google поставить вам декілька запитань, щоб ви підтвердили свій намір використання застосунку для відправки імейлів, а також дали дозвіл на використання функціональності відправки імейлів.

Перейдіть за посиланням: https://accounts.google.com/o/oauth2/auth?client\_id= [...]

Введіть код авторизації:



Вставляємо цей код в промпт вище — і отримуємо токен.

Ось ваш refresh token:: [...]

Збережіть цей refresh token для подальшого застосування. Він знадобиться нам на наступному кроці.

**Крок 3. Авторизація за допомогою refresh token і відправка листа**

Підсумуємо: щоб використовувати SMTP-сервер Gmail, вам потрібно знати:

* ім’я сервера = [*smtp.gmail.com*](http://smtp.gmail.com/);
* порт = *465 для SSL/TLS*-з’єднання (бажано);
* або порт = *587* для з’єднання *STARTTLS*;
* *OAuth2 Client ID і Client Secret*;
* *Refresh token*.

*# спочатку імпортуємо потрібні компоненти*

import smtplib, ssl

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

import urllib.parse

import urllib.request

import json

import base64

port = 587

smtp\_server = "smtp.gmail.com"

oauth2\_client\_id = '....apps.googleusercontent.com' *# вставте з Google консолі*

oauth2\_client\_secret = '...' *# вставте з Google консолі*

oauth2\_refresh\_token = '...' *# вставте значення, отримане на минулому кроці*

sender\_email = "mailtrap@example.com" *# має збігатися з юзером, для якого ми отримали refresh token у минулому кроці*

receiver\_email = "new@example.com"

message = MIMEMultipart("alternative")

message["Subject"] = "Вітання через Gmail"

message["From"] = sender\_email

message["To"] = receiver\_email

def make\_authorization\_string():

   params = {

       'client\_id': oauth2\_client\_id,

       'client\_secret': oauth2\_client\_secret,

       'refresh\_token': oauth2\_refresh\_token,

       'grant\_type': 'refresh\_token'

   }

   encoded\_params = urllib.parse.urlencode(params).encode('UTF-8')

   response = urllib.request.urlopen('https://accounts.google.com/o/oauth2/token', encoded\_params).read().decode(

       'UTF-8')

   response\_data = json.loads(response)

   access\_token = response\_data['access\_token']

*# ми не використовуватимемо це значення зараз, але воно може допомогти вам зрозуміти, коли підійшов час оновити токен*

   expires\_in = response\_data['expires\_in']

   auth\_string = 'user=%s\1auth=Bearer %s\1\1' % (sender\_email, access\_token)

   return base64.b64encode(auth\_string.encode('ascii')).decode('ascii')

html = f"""\

<html>

<body>

  Відправлено за допомогою Gmail SMTP і авторизацію через OAuth2!

</body>

</html>

"""

part = MIMEText(html, "html")

message.attach(part)

auth\_string = make\_authorization\_string()

context = ssl.create\_default\_context()

with smtplib.SMTP(smtp\_server, port) as server:

   server.ehlo(oauth2\_client\_id)

   server.starttls(context=context)

   server.docmd('AUTH', 'XOAUTH2 ' + auth\_string)

   server.sendmail(

       sender\_email, receiver\_email, message.as\_string()

   )

print('Відправлено')

**Використання модуля Yagmail**

Якщо хочеться чогось простого, можете використовувати [yagmail](https://yagmail.readthedocs.io/en/latest/index.html" \t "_blank), спеціальний клієнт Gmail/SMTP. Він значно полегшує надсилання електронних листів. Просто порівняйте наведені вище приклади з цими кількома рядками коду:

import yagmail

yag = yagmail.SMTP(

  "mailtrap@example.com", *# вказуємо акаунт відправника*

  oauth2\_file='oauth\_creds.json'

)

contents = [

    "Я – тіло листа, а це просто текст http://somedomain/image.png",

    "У вкладенні – аудіофайл.", '/local/path/to/Anthem\_of\_European\_Union.mp3'

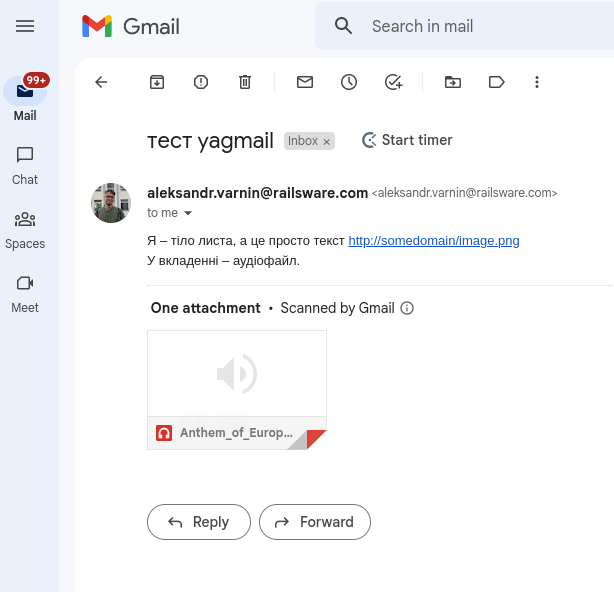
]

yag.send('to@someone.com', 'тест yagmail', contents)

Ми так само використаємо OAuth2 для автентифікації. Вказаний файл oauth\_creds.json зберігатиме необхідні дані для авторизації. При першому запуску такої програми цього файлу ще не буде існувати.

Тому yagmail запросить у вас всі ті самі дані через консольний інтерфейс: OAuth2 client id, client secret, а потім попросить вас пройти авторизацію застосунку (як ми вже робили вище), після чого ввести URL з кодом авторизації в промпт.

При наступних запусках yagmail запам’ятає креденшели і вже буде відправляти листи без взаємодії з користувачем.



**Що ще допоможе з імейлами у Python**

Ми розглянули лише базові варіанти надсилання листів за допомогою Python, щоб описати логіку та спектр його можливостей. Раджу ознайомитися з документацією Python і поекспериментувати з власним кодом, щоб отримати справді круті результати.

Існує чимало різноманітних фреймворків та бібліотек Python, які роблять створення застосунків елегантнішим та цілеспрямованішим. Зокрема, деякі з них можуть допомогти з надсиланням електронних листів.

* [Flask](http://flask.pocoo.org/) — з простим інтерфейсом для надсилання електронних листів Flask Mail.
* [Django](https://www.djangoproject.com/) добре працює для створення HTML-шаблонів.
* [Zope](http://www.zope.org/en/latest/) стане в пригоді для розробки сайтів.
* [Marrow Mailer](https://github.com/marrow/mailer) — це спеціальний фреймворк для доставки пошти, який додає різні корисні конфігурації.
* [Plotly](https://plot.ly/python/email-reports/) та його [Dash](https://plot.ly/products/dash/" \t "_blank) допоможуть з графіками та звітами про розсилки.

І, звісно, не забувайте все спочатку протестувати. Тут допоможе, зокрема, Mailtrap зі своєю широкою функціональністю для тестування. Успішних експериментів!