

Sending Emails from Python

✓

Reading: Module 3 Introduction

10 min

📝

Discussion Prompt: Discussion Prompt

10 min

✓

Reading: Introduction to Python Email Library

10 min

✓

Reading: Adding Attachments

10 min

✓

Reading: Sending the Email Through an SMTP Server

10 min

Generating PDFs from Python

Module 3 Project

Adding Attachments

Remember, email messages are made up completely of strings. When you add an attachment to an email, whatever type the attachment happens to be, it’s encoded as some form of text. The ***Multipurpose Internet Mail Extensions (MIME)*** standard is used to encode all sorts of files as text strings that can be sent via email.

Let's dive in and break down how that works.

In order for the recipient of your message to understand what to do with an attachment, you need to label the attachment with a ***MIME type*** and ***subtype*** to tell them what sort of file you’re sending. The ***Internet Assigned Numbers Authority (IANA)*** (iana.org) hosts a registry of valid MIME types. If you know the correct type and subtype of the files you'll be sending, you can use those values directly. If you don't know, you can use the Python **mimetypes** module to make a good guess!

```
1 >>> import os.path
2 >>> attachment_path = "/tmp/example.png"
3 >>> attachment_filename = os.path.basename(attachment_path)
4 >>> import mimetypes
5 >>> mime_type, _ = mimetypes.guess_type(attachment_path)
6 >>> print(mime_type)
7 image/png
```

Alright, that **mime_type** string contains the MIME type and subtype, separated by a slash. The **EmailMessage** type needs a MIME type and subtypes as separate strings, so let's split this up:

```
1 >>> mime_type, mime_subtype = mime_type.split('/', 1)
2 >>> print(mime_type)
3 image
4 >>> print(mime_subtype)
5 png
6
```

Now, finally! Let's add the attachment to our message and see what it looks like.

```
1 >>> with open(attachment_path, 'rb') as ap:
2 ...     message.add_attachment(ap.read(),
3 ...                             maintype=mime_type,
4 ...                             subtype=mime_subtype,
5 ...                             filename=os.path.basename(attachment_path))
6 ...
7 >>> print(message)
8 Content-Type: multipart/mixed; boundary="====5350123048127315795=="
9
10 -----5350123048127315795==
11 Content-Type: text/plain; charset="utf-8"
12 Content-Transfer-Encoding: 7bit
13
14 Hey there!
15
16 I'm learning to send email using Python!
17
18 -----5350123048127315795==
19 Content-Type: image/png
20 Content-Transfer-Encoding: base64
21 Content-Disposition: attachment; filename="example.png"
22 MIME-Version: 1.0
23
24 iVBORw0KGgoAAAANSUhEUgAAASIAAABSCAYAAADw69nDAAAACXBIWXMAAAsTAAALEwEampwYAAAg
25 AE1EQVR4nO2dd3wUZf7HP8/M9k2nKIJA4BCUNJKgNJWIBUgEggCiSgeVhA8jzv05Gc5z4KHiqn
26 eBZIIBDKIXggKIECRCAhjQAqx4UiCARSt83uzDy/PzazTDZbwy4BnHde+9qZydNn97Pf5/uUIZRS
27 (...We deleted a bunch of lines here...)
28 wgAAAABJRUSerkJggg==
29
30 -----5350123048127315795===
31
```

The entire message can still be serialized as a text string, including the image that we attached! The email message as a whole has the MIME type "multipart/mixed". Each **part** of the message has its own MIME type. The message body is still there as a "text/plain" part, and the image attachment is a "image/png" part. Cool!

Now, how do we *send* this email message? That's coming up!

✓ Completed

Go to next item

👍 Like

👎 Dislike

📄 Report an issue