<https://www.thepythoncode.com/article/reading-emails-in-python>

Being able to create an application that is able to read your emails and automatically downloading attachments is a handy tool. In this tutorial, you will learn how to use the built-in [imaplib](https://docs.python.org/3/library/imaplib.html" \o "imaplib — IMAP4 protocol client" \t "_blank) module to list and read your emails in Python, we gonna need the help of **IMAP protocol**.

[IMAP](https://en.wikipedia.org/wiki/Internet_Message_Access_Protocol) is an Internet standard protocol used by email clients to retrieve email messages from a mail server. Unlike the [POP3](https://en.wikipedia.org/wiki/Post_Office_Protocol) protocol which downloads email and delete them from the server (and then read them offline), with **IMAP**, the message does not remain on the local computer, it stays on the server.

To get started, we don't have to install anything, all the modules used in this tutorial are the built-in ones:

import imaplib

import email

from email.header import decode\_header

import webbrowser

import os

# account credentials

username = "youremailaddress@provider.com"

password = "yourpassword"

We've imported the necessary modules, and then specified the credentials of our email account.

First, we gonna need to connect to the IMAP server:

# create an IMAP4 class with SSL

imap = imaplib.IMAP4\_SSL("imap.gmail.com")

# authenticate

imap.login(username, password)

Since I'm testing this on a gmail account, I've used imap.gmail.com server, check [this link](https://www.systoolsgroup.com/imap/) that contains list of IMAP servers for most commonly used email providers.

Also, if you're using a Gmail account and the above code raises an error indicating that the credentials are incorrect, make sure you [allow less secure apps](https://myaccount.google.com/lesssecureapps?pli=1)on your account.

If everything went okey, then you have successfully logged in to your account, let's start getting emails:

status, messages = imap.select("INBOX")

# number of top emails to fetch

N = 3

# total number of emails

messages = int(messages[0])

We've used imap.select() method, which selects a mailbox (Inbox, spam, etc.), we've chose INBOX folder, you can use imap.list() method to see the available mailboxes.

messages variable contains number of total messages in that folder (inbox folder), and status is just a message that indicates whether we received the message successfully. We then converted messages into an integer so we can make a for loop.

N variable is the number of top email messages you want to retrieve, I'm gonna use 3 for now, let's loop over each email message, extract everything we need and finish our code:

# if the email message is multipart

if msg.is\_multipart():

# iterate over email parts

for part in msg.walk():

# extract content type of email

content\_type = part.get\_content\_type()

content\_disposition = str(part.get("Content-Disposition"))

try:

# get the email body

body = part.get\_payload(decode=True).decode()

except:

pass

if content\_type == "text/plain" and "attachment" not in content\_disposition:

# print text/plain emails and skip attachments

print(body)

elif "attachment" in content\_disposition:

# download attachment

filename = part.get\_filename()

if filename:

if not os.path.isdir(subject):

# make a folder for this email (named after the subject)

os.mkdir(subject)

filepath = os.path.join(subject, filename)

# download attachment and save it

open(filepath, "wb").write(part.get\_payload(decode=True))

else:

# extract content type of email

content\_type = msg.get\_content\_type()

# get the email body

body = msg.get\_payload(decode=True).decode()

if content\_type == "text/plain":

# print only text email parts

print(body)

if content\_type == "text/html":

# if it's HTML, create a new HTML file and open it in browser

if not os.path.isdir(subject):

# make a folder for this email (named after the subject)

os.mkdir(subject)

filename = f"{subject[:50]}.html"

filepath = os.path.join(subject, filename)

# write the file

open(filepath, "w").write(body)

# open in the default browser

webbrowser.open(filepath)

print("="\*100)

# close the connection and logout

imap.close()

imap.logout()

A lot to cover here, the first thing to notice is we've used range(messages, messages-N, -1), which means going from the top to the bottom, the newest email messages got the highest id number, the first email message has an ID of 1, so that's the main reason, if you want to extract the oldest email addresses, you can change it to something like range(N).

Second, we used the imap.fetch() method, which fetches the email message by ID using the standard format specified in [RFC 822](https://tools.ietf.org/html/rfc822).

After that, we parse the bytes returned by the fetch() method to a proper Message object, and used decode\_header() function from email.header module to decode the subject of the email address to human readable unicode.

After we print the email sender and the subject, we want to extract the body message. We look if the email message is multipart, which means it contains multiple parts. For instance, an email message can contain the text/html content and text/plain parts, which means it has the HTML version and plain text version of the message.

It can also contain file attachments, we detect that by the Content-Disposition header, so we download it under a new folder created for each email message named after the subject.

The msg object, which is email module's Message object, has many other fields to extract, in this example, we used only From and the Subject, write msg.keys() and see available fields to extract, you can for instance, get the date of when the message was sent using msg["Date"].

After I ran the code for my test gmail account, I got this output:

After I ran the code for my test gmail account, I got this output:

Subject: Thanks for Subscribing to our Newsletter !

From: rockikz@thepythoncode.com

====================================================================================================

Subject: An email with a photo as an attachment

From: Python Code **<rockikz@thepythoncode.com>**

Get the photo now!

====================================================================================================

Subject: A Test message with attachment

From: Python Code **<rockikz@thepythoncode.com>**

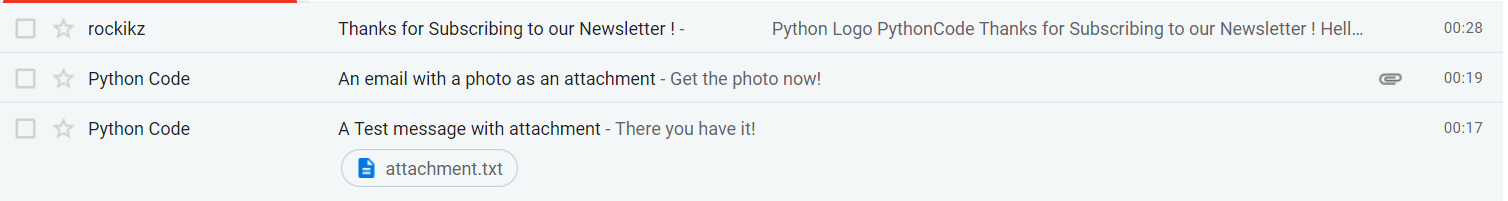
There you have it!

====================================================================================================

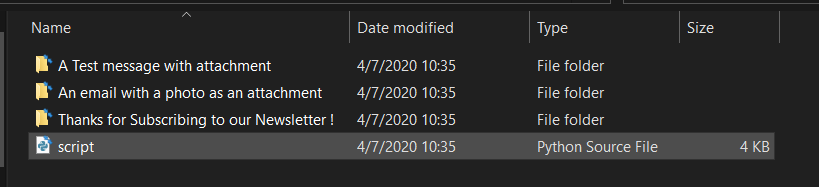
Copy

So the code will only print text/plain body messages, it will create a folder for each email, in which it contains the attachment and the HTML version of the email, it also opens the HTML email in your default browser for each email extracted that has the HTML content.

Going to my Gmail, I see the same emails that were printed in Python:



Awesome, I also notice the folders created for each email:

Each folder now has the HTML message (if available) and all the files attached with the email.

## Conclusion

Awesome, now you can build your own email client using this recipe, for example, instead of opening each email on a new browser tab, you can build a [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface) program which reads and parses HTML just like a regular browser, or maybe you want to send notifications whenever a new email is sent to you, the possibilities are endless !

A note though, we haven't covered everything that imaplib module offers, for example, you can search for emails and filter by the sender address, subject, sending date, and more using imap.search() method.

Here are other Python email tutorials:

* [How to Send Emails in Python](https://www.thepythoncode.com/article/sending-emails-in-python-smtplib).
* [How to Delete Emails in Python.](https://www.thepythoncode.com/article/deleting-emails-in-python)

Here is the documentation of modules used for this tutorial:

* [imaplib — IMAP4 protocol client](https://docs.python.org/3/library/imaplib.html)
* [email — An email and MIME handling package](https://docs.python.org/3/library/email.html)
* [webbrowser — Convenient Web-browser controller](https://docs.python.org/3/library/webbrowser.html)

Finally, if you're a beginner and want to learn Python, I suggest you take [Master Python in 5 Online Courses from University of Michigan](https://click.linksynergy.com/fs-bin/click?id=Bjs2xp3zg0Y&offerid=759505.31&type=3&subid=0), in which you'll learn a lot about Python, good luck!

*Learn also*: [*How to Handle Files in Python using OS Module*](https://www.thepythoncode.com/article/file-handling-in-python-using-os-module).