

How to Read Emails in Python?



To read emails from an email server, we use the [Internet Message Access Protocol \(IMAP\) protocol](#) . Although you can visit the email service provider website like gmail.com and look up the emails present in your Inbox, it would be cool to write a Python script that can read or fetch emails from your Inbox. And here, in this Python tutorial, we will walk you through the steps you need to follow to fetch emails from your Gmail Inbox.

How to Read Emails in Python?

For this tutorial, we will be fetching the mails preset in a Gmail inbox, and to be more specific, we will be fetching only those emails that are sent from a specific email address. When we try to access our Gmail account with any programming language or third-party package, we receive the following error.

```
imaplib.IMAP4.error: b' [ALERT] Application-specific password required
```

You get this error because Gmail blocks the request of third-party packages or applications if 2 Step Verification is active for your account. You can simply solve this error by deactivating the 2 Step verification, but we would not suggest that. Instead, you can use the Gmail App Password and generate an alternative app password for your Gmail account. With this, you do not have to deactivate 2 Step Verification and you can access your Gmail account with Python or any other third-party package. This Python tutorial is divided into three sections:

1. In **Section 1**, you will learn how to generate or set up a Gmail App password.

2. **Section 2** will detail the libraries required to write the Python program to read emails.
3. In **Section 3**, we will walk you through the Python program to read the emails in your Gmail account.

If you already know how to set up or generate a Gmail App password or you are using a different Email service, you can skip sections 1 and 2 and go directly to Section 3 for the Python program. Otherwise, start from Section 1.

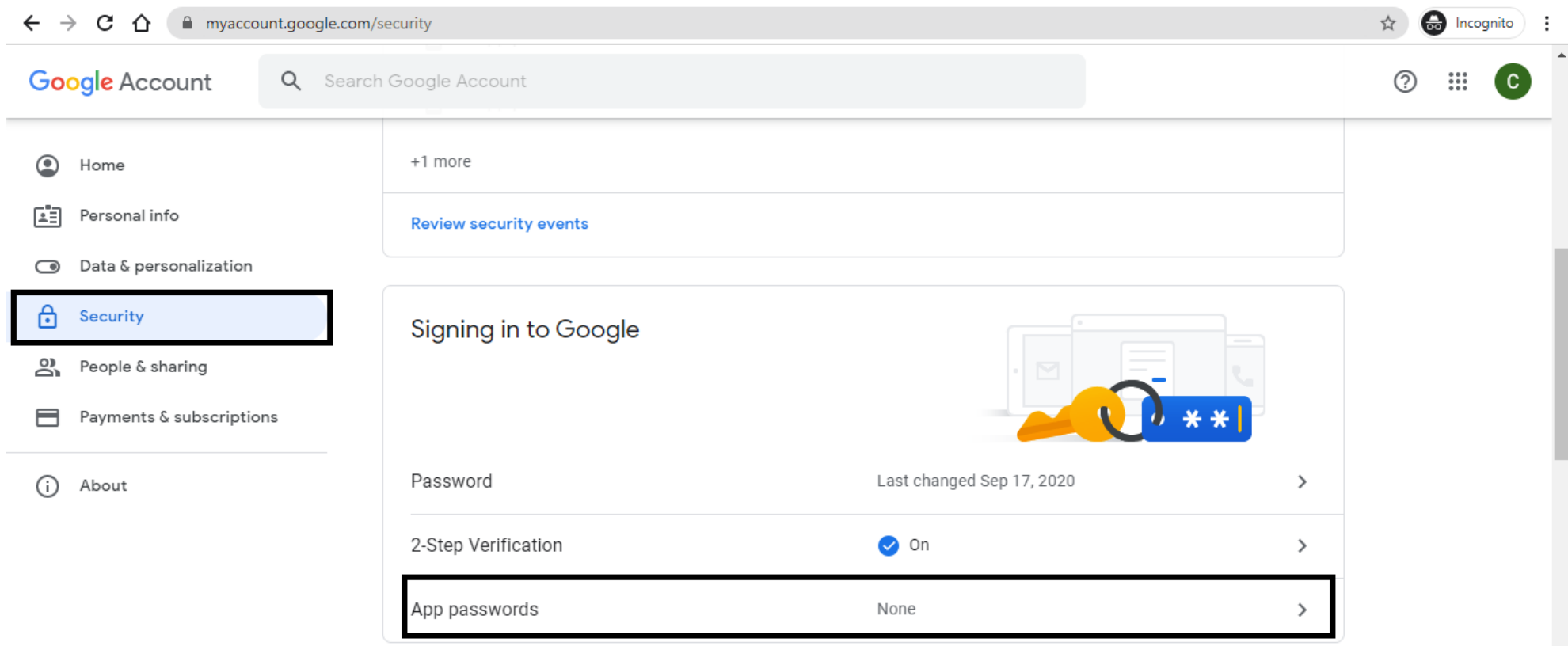
Section 1: Set up App Password for Gmail

Again, we are generating the App Password because Gmail does not allow us to log in to the Gmail account with third-party packages if the 2 Step Verification feature is on. The generated App Password will thus, help us to log in to our Gmail account with an alternative generated password.

Step 1: Go to the Google [My Account Settings](#) , and you will see a screen similar as shown below:

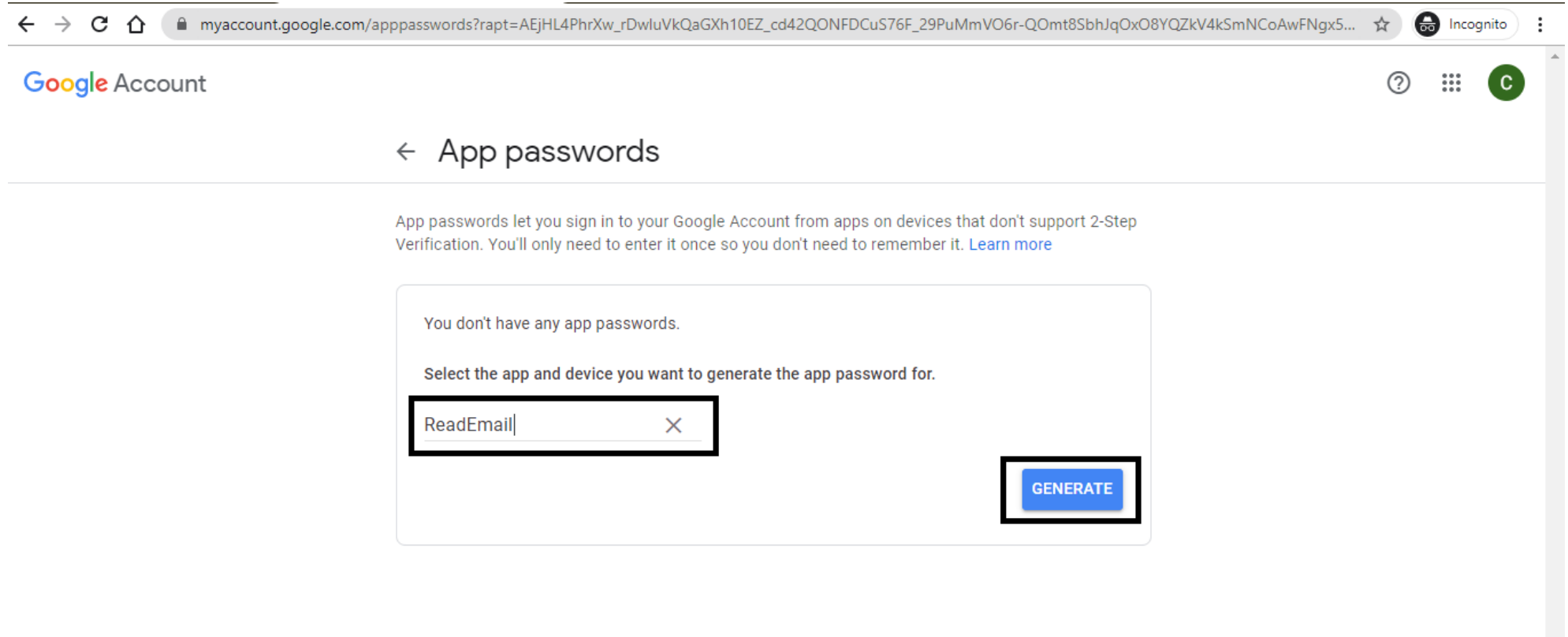
The screenshot shows the Google Account management interface. At the top, the browser address bar displays 'myaccount.google.com' in an Incognito window. The Google Account header includes a search bar and navigation icons. A left sidebar lists account management categories: Home (selected), Personal info, Data & personalization, Security, People & sharing, Payments & subscriptions, and About. The main content area features four tiles: 'Privacy & personalization' with a 'Manage your data & personalization' link; 'Security issues found' with a 'Secure account' link; 'Account storage' showing '0 GB of 15 GB used' and a 'Manage storage' link; and 'Privacy suggestions available' with a 'Review suggestions (3)' link. The footer contains links for Privacy, Terms, Help, and About.

Step 2: Navigate to **Security>>>App passwords** .



When you click on the App passwords, Google might ask you to enter your password. Do it to proceed.

Step 3: Select App to **Other(Custom Name)** option and give a Custom name to your app. We have given our App name "**ReadEmail.**" After specifying the name, hit the **GENERATE** button.



← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

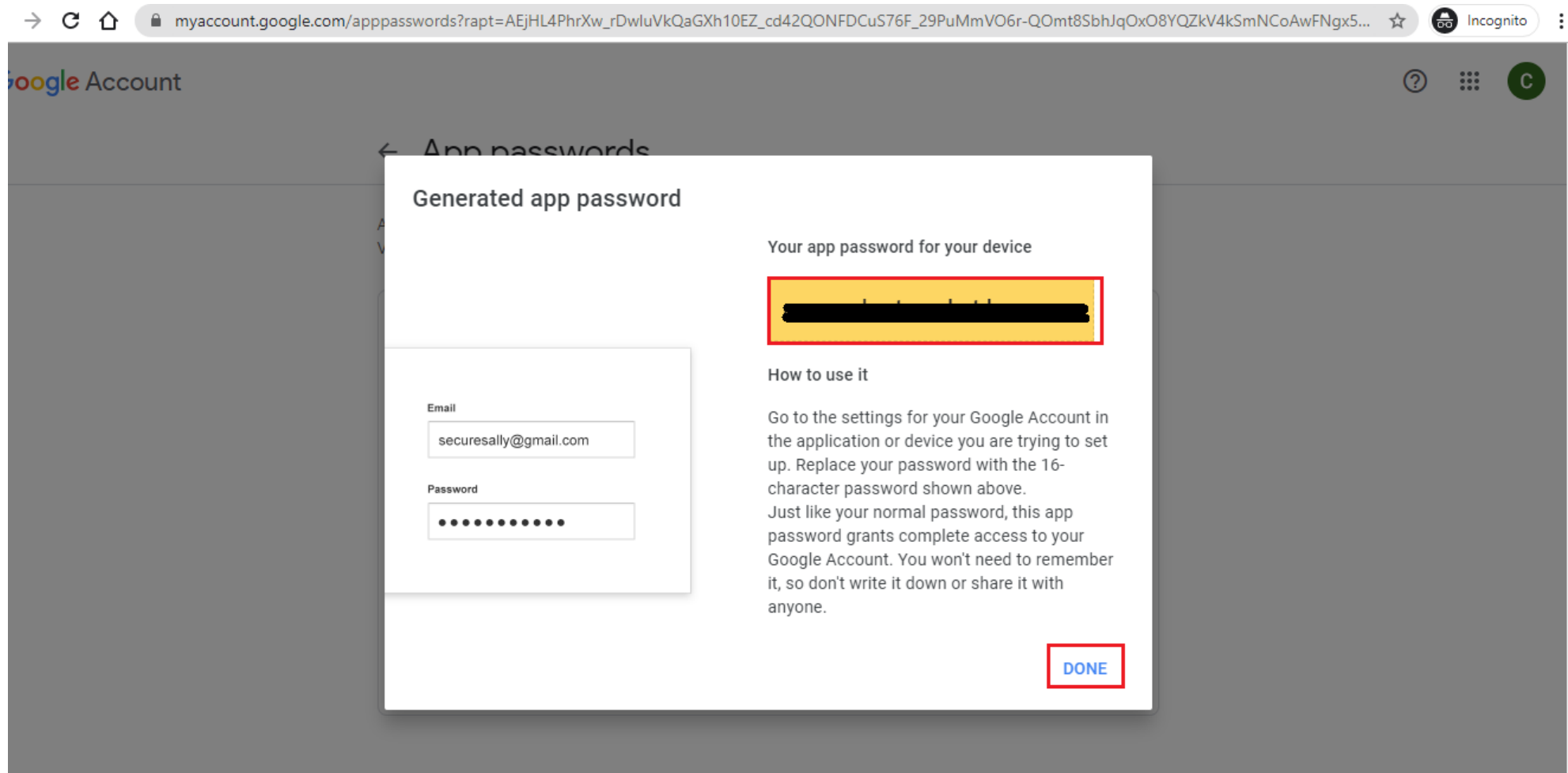
You don't have any app passwords.

Select the app and device you want to generate the app password for.

ReadEmail

GENERATE

Step 4: When you hit on the **GENERATE** button, a small window will prompt containing a 16-character password. **Copy the password** and do not share it with anyone.



Now you have successfully generated the App Password. Do not forget to copy it on your Notepad or clipboard of your Python IDE.

Section 2: Importing the Python imaplib Library

The `imaplib` library is a standard Python library for handling IMAP protocols. As it is a part of Python Standard Libraries, you do not have to worry about installing it because it comes preinstalled with Python. We will be using this library to connect with the email service provider server (Gmail in our case) and log in to the server with the login credentials.

Python email Library

The Python `read_email` library is also a standard Python library that is used to handle [Multipurpose Internet Mail Extensions](#) (MIME) standards. A mail contains multiple pieces of information, so we will be using this library to extract information from an email, such as subject, date, from, and message. We are done with sections 1 and 2. Thus, it's time to write the code to read emails in Python. Open your [best Python IDE or text editor](#) and follow along.

Section 3: How to Read Emails in Python?

Let's start with importing `imaplib` and `email` modules and also declare the credentials and host provider server.

```
#modules
import imaplib
import email

#credentials
username = "codehundred100@gmail.com"

#generated app_password
app_password= "aqwertyuiopasdfa"

# https://www.systoolsgroup.com/imap/
gmail_host= 'imap.gmail.com'
```

In this tutorial, we will be reading a Gmail Inbox. Thus, our host server is `'imap.gmail.com'`, but if you are trying to access a different email provider, such as Hotmail or Outlook, [click here](#) to know the server name for your host. Next, let's set a connection to the Gmail host server with the help of `imaplib`. `IMAP4SSL()` library, and log in to the server with the `login()` method credentials.

```
#set connection
mail = imaplib.IMAP4_SSL(gmail_host)
```

```
#login
mail.login(username, app_password)
Now we are successfully logged in to the Gmail server with our Gmail account. Next, let's select the "INBOX" to read the messages. To select the Inbox, we will use the mail.select() method.
```

```
#select inbox
mail.select("INBOX")
```

Here, we are reading messages from the Inbox, and that's why we specify `"INBOX"` as an argument to the `select()` function. You can also read messages from other mailboxes present on your mail server. To list out all available mailboxes, you can use the `mail.list()` method. Our Inbox is full of emails, so for this tutorial, we will only be reading mail from **"noreply@kaggle.com."** To specify the mails that we will read, we will use the `mail.search()` method.

```
#select specific mails
_, selected_mails = mail.search(None, ' (FROM "noreply@kaggle.com")')

#total number of mails from specific user
print("Total Messages from noreply@kaggle.com:" , len(selected_mails[0].split()))
```

If you wish, you can also fetch the UNSEEN messages using the `mail.search(None, 'UNSEEN')` statement. The `mail.search()` method returns a list of single binary data representing emails ID in bytes. Now we will split that single binary data and loop through every email ID and access its content using the `email` module.

```
for num in selected_mails[0].split():
    _, data = mail.fetch(num, ' (RFC822)')
    _, bytes_data = data[0]

    #convert the byte data to message
    email_message = email.message_from_bytes(bytes_data)
    print("\n=====")

    #access data
    print("Subject: ", email_message["subject"])
    print("To: ", email_message["to"])
    print("From: ", email_message["from"])
```

```

print("Date: ", email_message["date"])
for part in email_message.walk():
    if part.get_content_type()=="text/plain" or part.get_content_type()=="text/html":
        message = part.get_payload(decode=True)
        print("Message: %n", message.decode())
        print("===== %n")
        break

```

Now it's time to put all the code together and execute it. **#Python program to read emails from Gmail.**

```

#modules
import imaplib
import email

#credentials
username = "codehundred100@gmail.com"

#generated app_password
app_password= "aqwertyuiopasdfa"

# https://www.systoolsgroup.com/imap/
gmail_host= 'imap.gmail.com'

#set connection
mail = imaplib.IMAP4_SSL(gmail_host)

#login
mail.login(username, app_password)

#select inbox
mail.select("INBOX")

#select specific mails
_, selected_mails = mail.search(None, ' (FROM "noreply@kaggle.com")')

#total number of mails from specific user
print("Total Messages from noreply@kaggle.com:" , len(selected_mails[0].split()))

for num in selected_mails[0].split():
    _, data = mail.fetch(num , ' (RFC822)')
    _, bytes_data = data[0]

    #convert the byte data to message
    email_message = email.message_from_bytes(bytes_data)
    print("%n===== %n")

    #access data
    print("Subject: ", email_message["subject"])
    print("To: ", email_message["to"])
    print("From: ", email_message["from"])
    print("Date: ", email_message["date"])
    for part in email_message.walk():
        if part.get_content_type()=="text/plain" or part.get_content_type()=="text/html":
            message = part.get_payload(decode=True)
            print("Message: %n", message.decode())
            print("===== %n")
            break

```

Output

Total Messages from noreply@kaggle.com: 7

```

=====
Subject: Competition Recap: Google Research Football Simulation
To: codehundred100@gmail.com
From: Kaggle <noreply@kaggle.com>
Date: Tue, 12 Jan 2021 15:55:36 -0800
Message:
=====

```



```
=====
Subject: Competition Recap: NFL Impact Detection
To: codehundred100@gmail.com
From: Kaggle <noreply@kaggle.com>
Date: Fri, 15 Jan 2021 09:52:27 -0800
Message:
```

```
=====
Subject: Competition Recap: Riiid! Answer Correctness Prediction
To: codehundred100@gmail.com
From: Kaggle <noreply@kaggle.com>
Date: Tue, 19 Jan 2021 13:14:53 -0800
Message:
```

Conclusion

In this Python tutorial, you learned **"How to Read Emails in Python?"**. Using Python we read the emails from a Gmail account, without deactivating the 2 step verification. We used Google App Password to connect our Python script to the Gmail account, so our Python program could read the email from the inbox.

You do not need to do it if you are using a different email provider or server. There, you can log in to your account just with your email id and password with the Python program. We would urge you to go through the official documentation of the Python [imaplib](#) and [email](#) modules to know more about these two libraries.

All the best!

People are also reading: