< Go to the original

Creating Temporary Gmails Accounts With Python

Generate account — check inbox — read messages — restore Gmail

Better Programming · ~3 min read · January 18, 2022 (Updated: May 29, 2022) · Free: No

Sometimes you come across a number of adverts and websites that you need to signup for but don't completely trust. Temp mail is the most effective solution in this situation. These disposable email addresses replace your original ones and are expired after a certain time limit.

There are a number of online sites and services that help you make a temporary email address, however, in this tutorial, I am going to teach you how to create your own temporary Gmail addresses and receive emails at that address using Python.

Import Libraries

For this tutorial, we will be using only 2 python libraries That are `re` and `request`. If you don't have them already installed, use the commands below to install them:

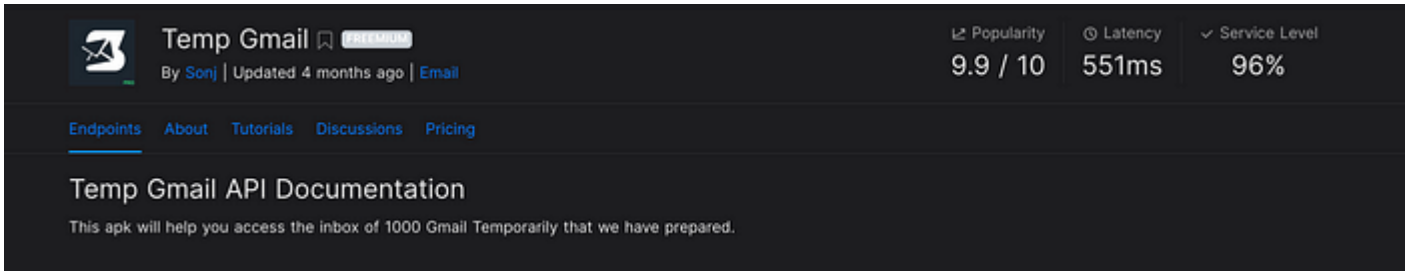
And then import them:

Copy

```
import request
import re
```

Generate Gmail

The very first step is to subscribe to the `Temp-Gmail` API via Rapid API (It's totally free).



Temp Gmail API

Once you are subscribed, you will be given a private key that will be used to access the API (by `request` library) and generate new Gmail addresses.

```python
def generate_gmail(ID: int):

  # access the API
  url = "https://temp-gmail.p.rapidapi.com/get"
  querystring = {"id":ID,"type":"alias"}
  headers = {
    'x-rapidapi-host': "temp-gmail.p.rapidapi.com",
    'x-rapidapi-key': "YOUR PRIVATE KEY"
    }

  # send a request to the API
  response = requests.request("GET", url, headers=headers, params=querystring)

  # convert the response to JSON format
  json_response = response.json()
  # get gmail address
  gmail = json_response['items']['username']
  # get gmail password
  password = json_response['items']['key']

  print('Gmail address: %s' % str(gmail))
  print('Password: %s' % str(password))
```

generate_gmail.py hosted with ❤ by GitHub view raw

```
Output Example
Gmail address: lauraburm.rs.131.9.8.8@gmail.com
Password: ***********R094ngJVIKMdXhfVCiMEEElE82Es
```

Note that the input "ID" is just an integer that corresponds to a specific Gmail address in the API dataset (contains 1000 Gmails). If you change the value of "ID", the function will generate a different Gmail address.

Check Inbox

All right, now that we have created a temporary Gmail address, we can check its inbox. The `check_inbox(gmail: str, password: str)` function scans for the new incoming mails.

If there is none, it returns a message that the inbox is empty.

Otherwise, the mail `message_id`, sender name, date, time, and the subject are printed out. We will need the `message_id` in the next section to read the body of the mail.

```python
# access the API
url = "https://temp-gmail.p.rapidapi.com/check"
querystring = {"username":gmail,"key":password}
headers = {
    'x-rapidapi-host': "temp-gmail.p.rapidapi.com",
    'x-rapidapi-key': "YOUR PRIVATE KEY"
    }

# send a request to the API
response = requests.request("GET", url, headers=headers, params=querystring)

# convert the response to JSON format
json_response = response.json()

# print the message from the API
print('API message: %s' % str(json_response['msg']))

# check whether the inbox is empty or not
if json_response['items'] == []:
  print("inbox is empty")

# if the inbox is not empty, print the details of the newest mail
else:
  message_id = json_response['items'][0]['mid']
  print('Message ID: %s' % str(message_id))
  print('From: %s' % str(json_response['items'][0]['textFrom']))
  print('Date: %s' % str(json_response['items'][0]['textDate']))
  print('Subject: %s' % str(json_response['items'][0]['textSubject']))
```

```
Input Example
MyGmail = "lauraburmr.s13198.8@gmail.com"
MyPassword = "***********QEQfM4cFqy2aie6sA6kPpxEMKGFNSQl4"
check_inbox(gmail=MyGmail, password=MyPassword)
Output Example
Message ID: 17e5dd60676eed04
From: Amir Ali Hashemi
Date: 2022-01-15 20:03:22
Subject: TEST EMAIL
```

Read inbox Message

In order to read the content of the mail, we should have the Gmail address and the `message_id`. If we send a request to the API providing these two values, the API returns the body message in the HTML format with all those HTML tags attached to it. See the example below

Copy

```
<div dir="ltr"><div>This is a test mail sent by amir-tech</div><div><br></div></div>
```

Hence, we may use the `re` library to remove the HTML tags and print the cleaned body message.

```
    # access the API
    url = "https://temp-gmail.p.rapidapi.com/read"
    querystring = {"username":gmail,"message_id":message_id}
    headers = {
        'x-rapidapi-host': "temp-gmail.p.rapidapi.com",
```

Copy

```
Input Example
MyGmail = 'lauraburmr.s13198.8@gmail.com'
MessageID = '17e5dd60676eed04'
read_message(gmail=MyGmail, message_id=MessageID)
Output Example
Body message: This is a test mail sent by amir-tech
```

Restore Gmail

As mentioned at the beginning of this tutorial, temporary emails expire after a certain time limit (Usually after 10 minutes).

Meaning that the password will be reset! Therefore, If you have created a Gmail address and want to reaccess it after a period of time, you need to restore it.

The `restore_gmail(gmail: str)` function takes your already generated Gmail address and sets a new password for it so it can be reused.

```
# access the API
url = "https://temp-gmail.p.rapidapi.com/restore"
querystring = {"username":gmail}
headers = {
    'x-rapidapi-host': "temp-gmail.p.rapidapi.com",
```

Copy

```
Input Example
restore_gmail('lauraburm.rs.131.9.8.8@gmail.com')
Output Example
Gmail address: lauraburm.rs.131.9.8.8@gmail.com
Password: ******************m7qLnoGbR094ngJVIKMdXhfVCiMEEElE82Es
```

Conclusion

To conclude, temp mails not only may be used for personal purposes but also can be integrated into programming projects to bring more functionality.

I personally created a temp Gmail Telegram bot using this API and use it on daily basis to sign up for untrusted websites.

If you are interested in how to create a Telegram bot, I have written an article about it. Feel free to check it out:

How I made my first advanced telegram bot using python

In this blog, I am going to show you how I built my first advanced Telegram bot with python and deployed it on Heroku...

medium.com

#programming #python #data-science #software-engineering #software-development