Two-Factor Authentication using Google Authenticator in Python

Last Updated : 23 Jul, 2025

Two Factor Authentication or 2FA is an advanced method of user authentication and a subset of multi-factor authentication mechanisms. 2FA enhances the security of its user accounts by adding another layer of authenticity challenge after traditional passwords are used in single-factor authentication.

This article will show you how to implement Two-Factor Authentication using Google Authenticator App using a general-purpose programming language called Python.

Modules and Apps required

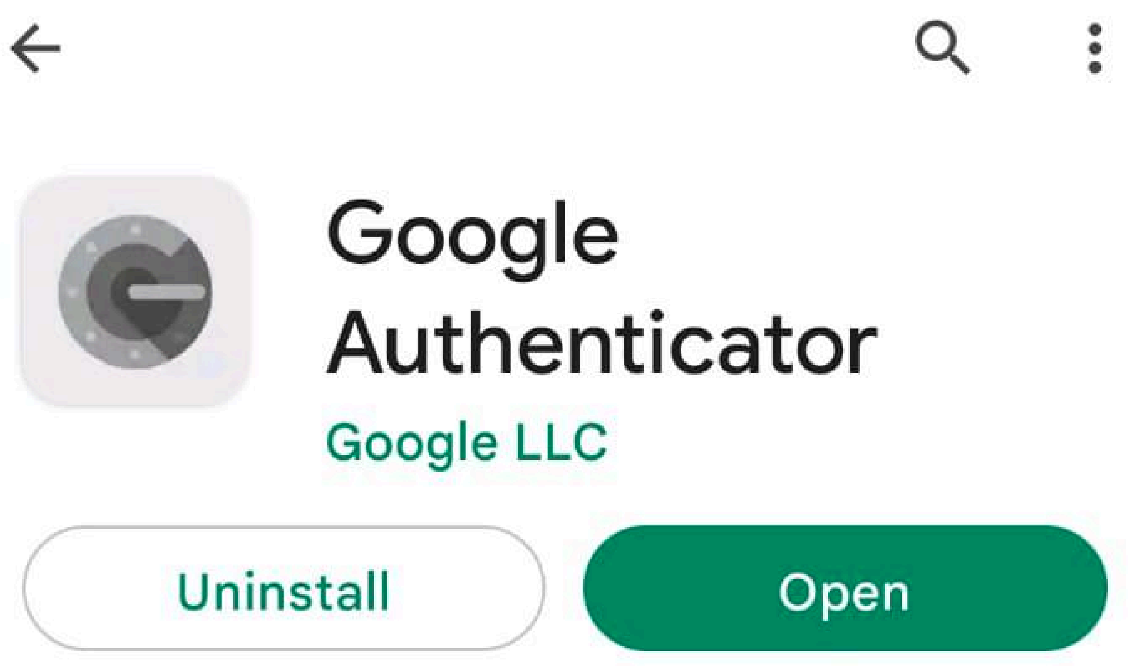
To implement this we need to use 3 modules -

- **time** - Inbuilt python module for time-related operations
- **pyotp** - to generate OTP
- **QRcode** - To generate QRcode

Run the following to install the required modules:

```
pip install pyotp qrcode
```

Users also need to Download and install the **Google Authenticator** app from the Playstore / Appstore onto their phones.



Importing required modules

Here we are going to import the required module.

```python
import time
import pyotp
import qrcode
```

Generating the Key

```python
k = pyotp.random_base32()
```

Using the **random_base32()** method of the **pyotp** module, random alphanumeric keys can be generated. Every time the code generates a new key making it impossible to recover in case it gets lost.

```python
secret_key = "GeeksforGeeksIsBestForEverything"
```

We can also define a specific secret key like the above, we just have to pass this in the TOTP method in later steps, this will never change and will be easier to maintain.

Creating a Time-based OTP (TOTP)

In the following snippet, we are passing the secret_key into the TOTP and provisioning a URI (Uniform Resource Identifier) with the name of the user and the issuer_name, this way the issuer can generate multiple keys for different users, making it easier to identify them.

```
totp_auth = pyotp.totp.TOTP(
  secret_key).provisioning_uri(
  name='Dwaipayan_Bandyopadhyay',
  issuer_name='GeeksforGeeks')

print(totp_auth)
```

Output:

```
otpauth://totp/GeeksforGeeks:Dwaipayan_Bandyopadhyay?
secret=GeeksforGeeksIsBestForEverything&issuer=GeeksforGeeks
```

The above output is the link that gets generated, but as Google Authenticator supports QR code scanning we would convert this into a QR code which we will scan through our Google Authenticator.

Generating a QR Code

```
qrcode.make(totp_auth).save("qr_auth.png")
totp_qr = pyotp.TOTP(secret_key)
```

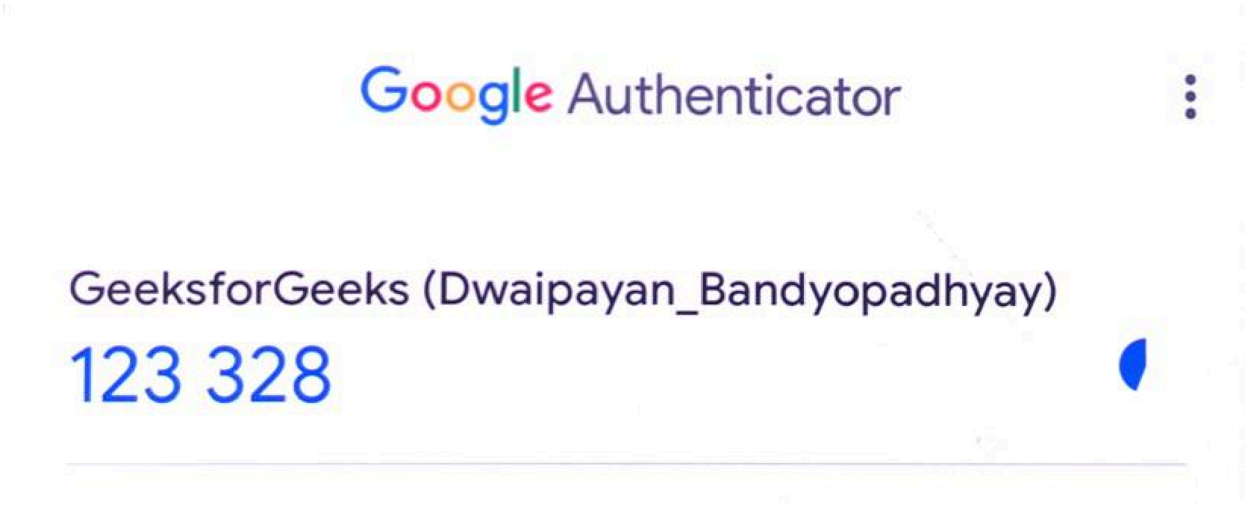
Here the QR codes get saved with the name **qr_auth** and we can scan it and get some new code every time which we can enter in our python script to verify.

Steps to Setup Google Authenticator -

1. Download the App from Playstore/AppStore.
2. Follow the initial setup procedure till a blank screen is reached.

3. Tap on the **+** sign at the lower right corner and select the **Scan a QR Code** Option**.**
4. Scan the generated QR code.
5. Now, a new account in the following format will be added with a TOTP which is valid for 30 seconds.

```
IssuerName (UserName)
<Unique Code that lasts for 30 seconds>
```



GeeksforGeeks (Dwaipayan_Bandyopadhyay)
123 328

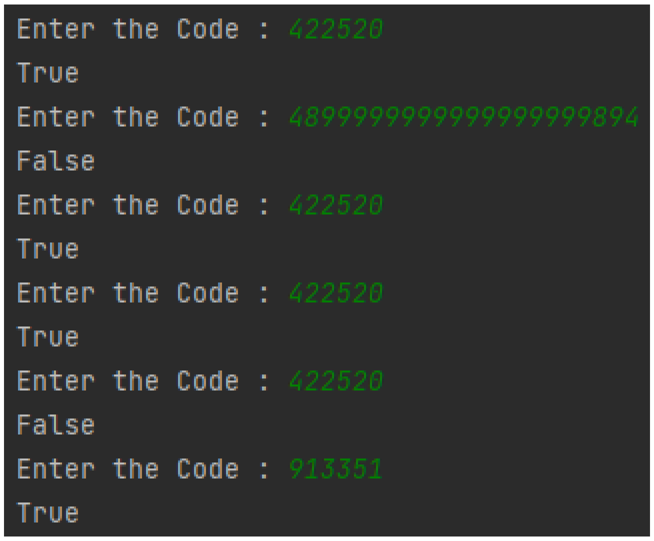
Verify the code using Python -

We can also verify the code generated using Python.

```
totp = pyotp.TOTP(secret_key)

while True:
    print(totp.verify(input(("Enter the Code : "))))
```

Output:



```
Enter the Code : 422520
True
Enter the Code : 489999999999999999894
False
Enter the Code : 422520
True
Enter the Code : 422520
True
Enter the Code : 422520
False
Enter the Code : 913351
True
```

The first code was the real one, second was to see what if we give a longer and different code result it returns, we can see that the first code after a while gives us the result **False** as it has expired, the code at the last line has taken its place for next 30 seconds.

Complete Implementation

```
import time
import pyotp
import qrcode

key = "GeeksforGeeksIsBestForEverything"

uri = pyotp.totp.TOTP(key).provisioning_uri(
    name='Dwaipayan_Bandyopadhyay',
    issuer_name='GeeksforGeeks')

print(uri)


# Qr code generation step
qrcode.make(uri).save("qr.png")

"""Verifying stage starts"""

totp = pyotp.TOTP(key)

# verifying the code
while True:
  print(totp.verify(input(("Enter the Code : "))))
```

Note: Make sure to comment out the QR code generation step after the first execution or it will keep on generating a QR code every time the code is executed.

Comment More info

Explore

Python Fundamentals

Python Data Structures

Advanced Python

Data Science with Python

Web Development with Python

Python Practice