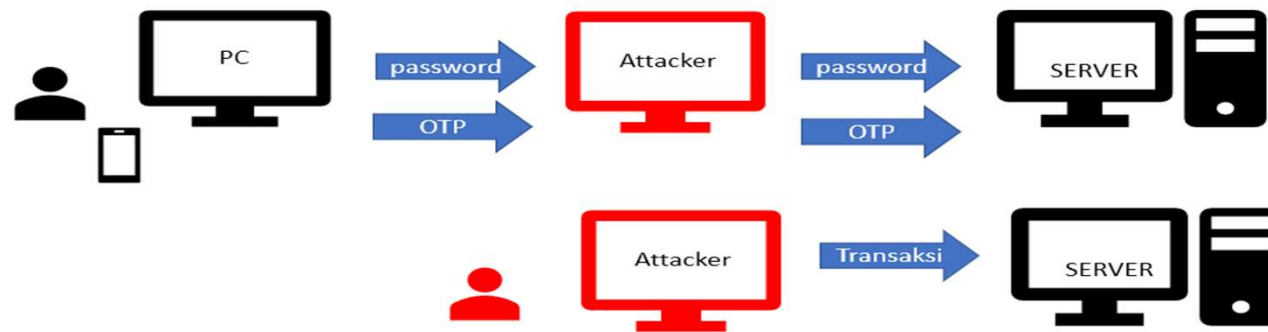


OTP VERIFICATION SYSTEM

OTP VERIFICATION SYSTEM



What is a one-time password (OTP)?

A one-time password (OTP) is an **automatically** generated **numeric** or **alphanumeric string of characters** that authenticates a user for a single transaction or [login](#) session.

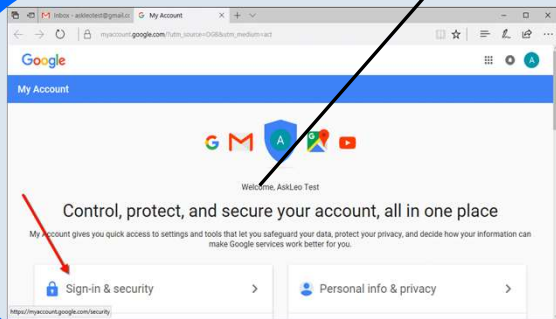
An OTP is more secure than a static password, especially a user-created password, which can be weak and reused across multiple accounts.

OTPs might replace traditional [authentication](#) login information or may be used in addition to it to add another layer of security.

One-time password examples

OTP [security tokens](#) are microprocessor-based [smart cards](#) or pocket-size key fobs that produce a numeric or alphanumeric code to authenticate access to the system or transaction. This secret code changes every 30 or 60 seconds, depending on how the token is configured.

Mobile device apps, such as [Google Authenticator](#), rely on the token device and PIN to generate the one-time password for [two-step verification](#).

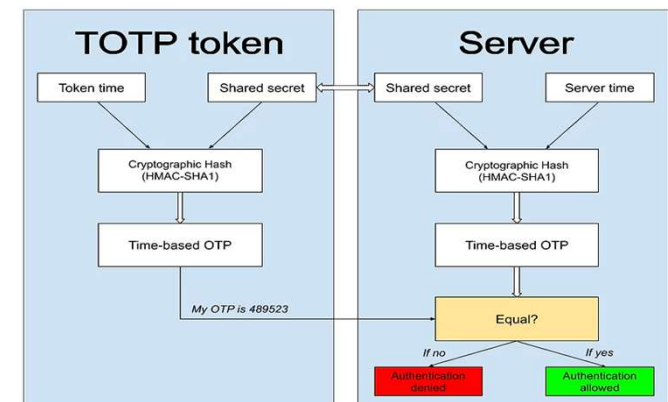
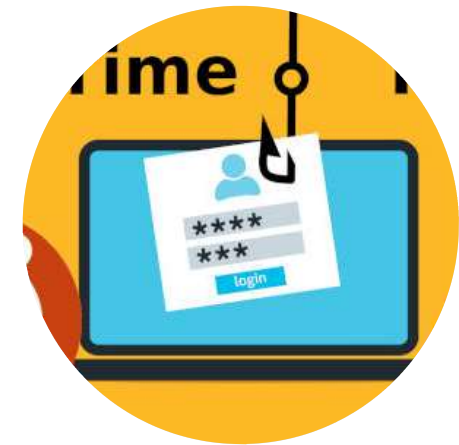


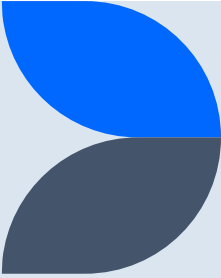
How to get a one-time password

When an **unauthenticated** user attempts to access a system or perform a transaction on a device, an authentication manager on the network server generates a number or shared secret, using one-time password algorithms. The same number and algorithm are used by the security token on the smart card or device to match and validate the one-time password and user.

Many companies use Short Message Service ([SMS](#)) to provide a temporary passcode via text for a second authentication factor. **The temporary passcode is obtained out of band through cellphone communications after the user enters his username and password on networked information systems and transaction-oriented web applications.**

For two-factor authentication ([2FA](#)), the user enters **a user ID, traditional password** and **temporary passcode** to access the account or system.



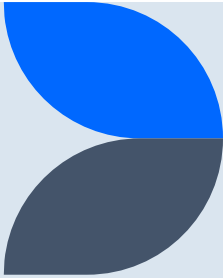


Benefits of a one-time password

The one-time password **avoids some common pitfalls of password security**. With OTPs, IT administrators and security managers do not have to worry about composition rules, known-bad and weak passwords, sharing of credentials or **reuse of the same password on multiple accounts and systems**.

Another advantage of one-time passwords is that they become invalid in minutes -- in the case of TOTP -- or once they have been used -- in the case of HOTP. In this way, one-time passwords prevent attackers from obtaining the secret codes and reusing them.

We can easily create an application for the task of OTP verification using Python by following the steps mentioned below:



1. First, create a 6-digit random number
2. Then store the number in a variable
3. Then we need to write a program to send emails
4. When sending email, we need to use OTP as a message
5. Finally, we need to request two user inputs; first for the user's email and then for the OTP that the user has received.

OTP Verification using Python

I hope you now have understood what is an OTP and how we can create an application for the task of OTP verification.

Now let's follow the steps mentioned above by using Python to create an application for the task of OTP verification.

I will start by importing the necessary Python library that we need for this task:

↑
IMPORT



1

Import math

Python has also a built-in module called `math`, which extends the list of mathematical functions.

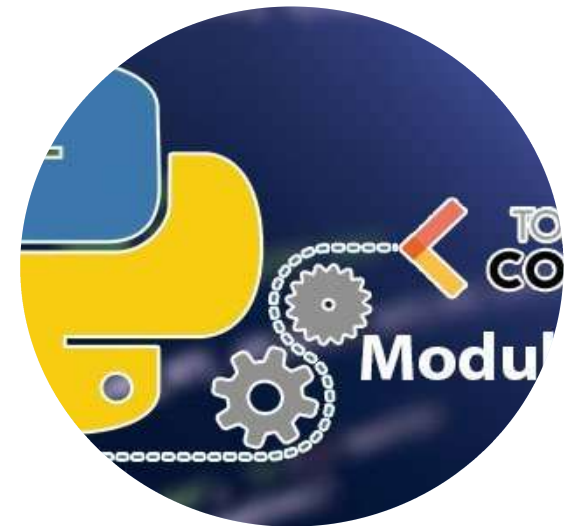
To use it, you must import the `math` module:

```
import math
```

Example
`import` math

```
→ x = math.sqrt(64)
```

```
print(x)
```



2

Import random

Python has a built-in module that you can use to make random numbers.

The random module has a set of methods:

- 1). Seed() → initialize the random number generator.
- 2). Randint() → return the random number between the given range.
- 3). random() → return the random float number between 0 and 1.

And so on.....

```
import random
```

```
print(random.randint(3, 9))
```

#returns a number between 3 and 9
(both included)

```
import random
```

```
print(random.random())
```

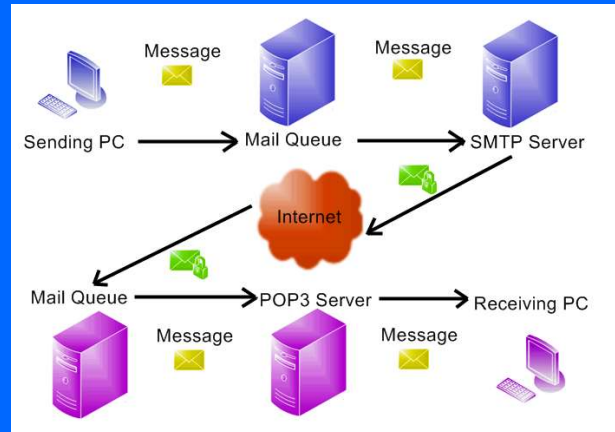


3 Import smtplib (Simple Mail Transfer Protocol)

[Python](#) comes with the built-in [smtplib](#) module for sending emails using the ***Simple Mail Transfer Protocol (SMTP)***. `smtplib` uses the [RFC 821](#) protocol for SMTP.

The rules of the internet are defined by the many protocols used on it. For instance, we have ***HTTP*** which is used in the transfer of web pages. Likewise, we have a protocol called ***SMTP*** or Simple Mail Transfer Protocol ***for sending emails across the internet.***

The Python SMTP Library defines how email messages should be formatted, encrypted, and relayed between mail servers, and any other small details that your computer has to deal with



Now I will generate a random number and store it in a variable which I will be using while sending emails to the users:



The screenshot shows a Jupyter Notebook window titled 'otp ver sys.ipynb'. The interface includes a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu bar, there are tabs for '+ Code' and '+ Text'. The code editor displays the following Python code:

```
1 import math # importing libraries.
2 import random
3 import smtplib
4
5 ### Now I will generate a random number and store it in a variable which I will be using while sending emails to the users:
6 digits = "0123456789"
7 OTP = ""
8 for i in range(6):
9     OTP += digits[math.floor(random.random() * 10)]
10
```

floor() method in Python returns the floor of x i.e., the largest integer not greater than x.

Now, before we go ahead, you need to have your Google app password to be able to send emails using your Gmail account. For this task, you need to follow the steps mentioned [here](#).

After you create your own {app password} for your Gmail account you will get a **key**. Copy that key and paste in the code below to send emails for **OTP verification using Python**:



```
otp ver sys.ipynb ☆
File Edit View Insert Runtime Tools Help

+ Code + Text

19
20 s = smtplib.SMTP('smtp.gmail.com', 587)
21 s.starttls()
22 s.login("16rehan687@gmail.com", "blig gtwq qfxq efjt") ## you have to submit your login credentials. 1) email-id. 2) App password.
23
24 emailid = input("Enter your email: ")
25 s.sendmail('Your Email Address', emailid, msg.encode('utf-8'))
26
27 a = input("Enter Your OTP >>: ")
28 if a == OTP:
29     print("Verified")
30 else:
31     print("Please Check your OTP again")
32
33 s.quit()

*** Enter your email: 
```

CODE EXECUTION

A screenshot of a Jupyter Notebook interface. The top bar shows the file name 'otp ver sys.ipynb' and a star icon. Below it is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. The main area is a code editor with a light gray background. It contains Python code for sending an email using the smtplib library. The code includes comments and prompts for user input. At the bottom, there is a prompt 'Enter your email:' followed by an empty text input field.

```
19
20 s = smtplib.SMTP('smtp.gmail.com', 587)
21 s.starttls()
22 s.login("16rehan687@gmail.com", "blig gtwa qfxq efjt") ## you have to submit your login credentials. 1) email-id. 2) App password.
23
24 emailid = input("Enter your email: ")
25 s.sendmail('Your Email Address', emailid, msg.encode('utf-8'))
26
27 a = input("Enter Your OTP >>: ")
28 if a == OTP:
29     print("Verified")
30 else:
31     print("Please Check your OTP again")
32
33 s.quit()
```

*** Enter your email:



(no subject) Inbox x



16rehan687@gmail.com

to bcc: me ▼


001691 is your OTP

↩ Reply

➦ Forward



VERIFICATION



```
14 ### Now, before we go ahead, you need to have your Google app password to be able to send emails using your Gmail account.
15 ### For this task, you need to follow the steps mentioned [[here]].
16 ### After you create your app password for your Gmail account you will get a key.
17 ### Copy that key and paste in the code below to send emails for OTP verification using Python:
18
19
20 s = smtplib.SMTP('smtp.gmail.com', 587)
21 s.starttls()
22 s.login("16rehan687@gmail.com", "blig gtwq qfxq efjt") ## you have to submit your login credentials. 1) email-id. 2) App password.
23
24 emailid = input("Enter your email: ")
25 s.sendmail('Your Email Address', emailid, msg.encode('utf-8'))
26
27 a = input("Enter Your OTP >>: ")
28 if a == OTP:
29     print("Verified")
30 else:
31     print("Please Check your OTP again")
32
33 s.quit()
```

Enter your email: 16rehan687@gmail.com

Enter Your OTP >>: 001691

Verified

