**Project Title: OTP Verification System (Python)**

**Step 1**

Import all the important libraries

import random

import smtplib

import ssl

from email.message import EmailMessage

# -------------------------------------------

# Step 1: Generate a dynamic 6-digit OTP

# -------------------------------------------

def generate\_otp():

    otp = random.randint(100000, 999999)

    return str(otp)

This generates a random integer between 100000 and 999999, inclusive.

So the result will always be a 6-digit number (e.g., 523491, 847203, etc.).

str(otp)

Converts the number to a string, since OTPs are often handled as text (especially when emailed or compared with user input).

return str(otp)

Returns the OTP as a string so it can be used elsewhere in your program (e.g., sent via email or printed to the console).

Why use str instead of returning an integer?

So you can easily:

Include it in email messages ("Your OTP is: " + otp)

Compare it with user input from input() (which is always a string)

**Step 2**

import numpy as np  # For generating a random OTP

import smtplib      # For sending email using SMTP

from email.mime.text import MIMEText  # For formatting the email message

from getpass import getpass

**import numpy as np**

What it does: Imports the numpy library and gives it the alias np.

Why it's used here:

To generate a random number (OTP) using np.random.randint(100000, 999999), which works similarly to Python's built-in random module, but is part of the powerful NumPy library (mostly used in data science and numerical computing).

**import smtplib**

What it does: Imports Python's built-in SMTP (Simple Mail Transfer Protocol) library.

Why it's used:

To send emails from your Python script. It's the core tool for connecting to Gmail (or other email services) and sending messages from one email to another.

**from email.mime.text import MIMEText**

What it does: Imports the MIMEText class from the email.mime.text module.

Why it's used:

To create text email messages that can be properly formatted and sent using smtplib. This wraps your email body in a standard format that email servers can read.

**from getpass import getpass**

What it does: Imports the getpass() function from the getpass module.

Why it's used:

To securely enter the sender's email password (App Password) without showing it on screen.

Useful when sharing notebooks or coding in public (e.g., Google Colab or classrooms).

Import Purpose

numpy Generate OTP

smtplib Connect to email server & send email

MIMEText Format the email content

getpass Securely enter email password

**Step 3:**

# generating OTP

def Generate\_otp():

    OTP = np.random.randint(100000, 999999)  # Generate a random integer between 100000 and 999999

    return str(OTP)

Generates a random 6-digit number using NumPy. Converts the number to a string so it can be easily sent by email or compared with user input.

**Step 4**

 # Function to send the OTP email to the recipient

def send\_otp\_to\_email(otp, sender\_email, sender\_password, recipient\_email):

    # Creating the email content

    msg = MIMEText(f"Your OTP is: {otp}")

    msg["Subject"] = "OTP Verification"

    msg["From"] = sender\_email

    msg["To"] = recipient\_email

    try:

        # Connect to Gmail's SMTP server and send the email

        with smtplib.SMTP("smtp.gmail.com", 587) as server:

            server.starttls()  # Upgrade to secure connection

            server.login(sender\_email, sender\_password)  # Login using credentials

            server.send\_message(msg)

            print("✅ OTP sent successfully.")

    except Exception as e:

        print("❌ Failed to send email:", e)

**Creates the body of the email.**

Sets the subject, sender, and recipient info.

Connects to Gmail’s SMTP server, upgrades to a secure connection, logs in, and sends the email.

Catches and prints errors if sending fails.

This function formats and sends the OTP via email using Gmail's secure SMTP server.

**Step 5**

**# Function to prompt the user to enter the received OTP**

def prompt\_to\_enter\_otp():

    return input("Enter the OTP you received: ")

**Step 6**

**# Function to verify if the entered OTP matches the generated one**

def verify\_otp(generated\_otp, entered\_otp):

    return generated\_otp == entered\_otp

This function:

Displays a message asking the user to enter the OTP.

Returns the user's input as a string for verification.

Compares the OTP generated by the system with the OTP entered by the user.

Returns True if they match, otherwise returns False.

In short:

Checks if the entered OTP is correct.

**Step 7**

**# Function to handle the OTP verification process**

def otp\_verification():

    print("OTP Verification\n")

**# Get sender and recipient email credentials**

    sender\_email = input("Enter your Gmail address: ")

    sender\_password = getpass("Enter your Gmail app password: ")

    recipient\_email = input("Enter the recipient's email address: ")

**# Generate and send the OTP**

    otp = Generate\_otp()

    send\_otp\_to\_email(otp, sender\_email, sender\_password, recipient\_email)

**# Allow up to 3 attempts to enter the correct OTP**

    for attempts\_left in range(3, 0, -1):

        if verify\_otp(otp, prompt\_to\_enter\_otp()):

            print("\u2705 OTP verified successfully!")  # Success message

            return

        else:

            print(f"\u274C Incorrect OTP. You've ({attempts\_left - 1}) Attempts left.")  # Error message

**otp\_verification()**

This function manages the entire OTP verification process.

Sender's Gmail and App Password

Recipient’s email address

Generates and sends the OTP to the recipient via email.

Gives the user 3 chances to enter the correct OTP.

Verifies if entered OTP matches:

✅ If correct → prints success message.

❌ If wrong → shows remaining attempts.

✅ In short:Handles email input, sends OTP, and verifies it with up to 3 tries.

**Short Explanation of What Happened:**

✅ The system generated and sent an OTP to your email successfully.

❌ On the first attempt, you entered 255507, but it didn't match the generated OTP (likely a typo or mismatch).

✅ On the second attempt, you entered 255507 again — and this time, it matched the OTP.

✅ The system verified the OTP and granted access.