Project Report: OTP Verification System

Project Title:

OTP (One-Time Password) Verification System

Project Overview:

The OTP Verification System is a Python-based application that generates a random 6-digit OTP, simulates sending it to a user's email, and verifies the OTP entered by the user. This system ensures secure authentication by limiting the number of attempts a user can make to enter the correct OTP.

Objective:

The primary objectives of the OTP Verification System are:

1. Generate a secure 6-digit OTP.
2. Simulate sending the OTP to the user's email.
3. Allow the user to input the OTP and verify it within 3 attempts.
4. Provide feedback on the OTP verification process, including the number of remaining attempts.

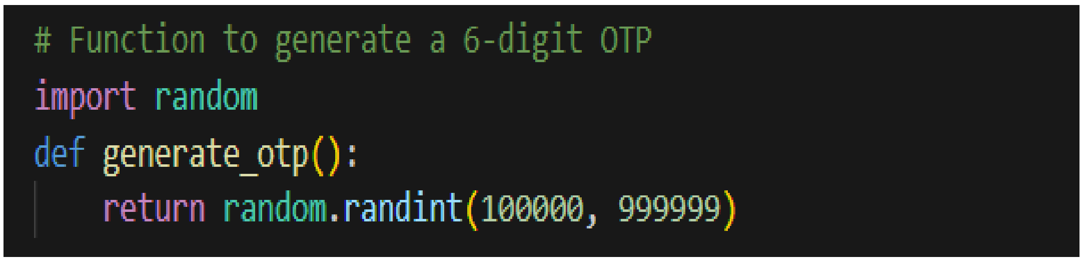
System Design:

The OTP Verification System follows these steps:

1. **Generate OTP**: A 6-digit random OTP is generated.
2. **Simulate OTP Sending**: The OTP is displayed to the console as a simulation of sending it to the user via email.
3. **Verify OTP**: The system prompts the user to enter the OTP and verifies whether it matches the generated OTP. The user has 3 attempts to enter the correct OTP.

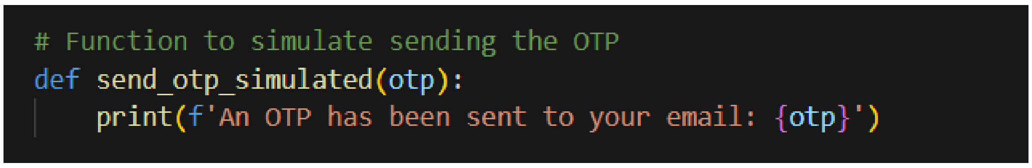
Code Explanation:

1. **OTP Generation**: The function generate_otp() generates a random integer between 100000 and 999999 using Python's random.randint() function. This 6-digit number serves as the OTP.



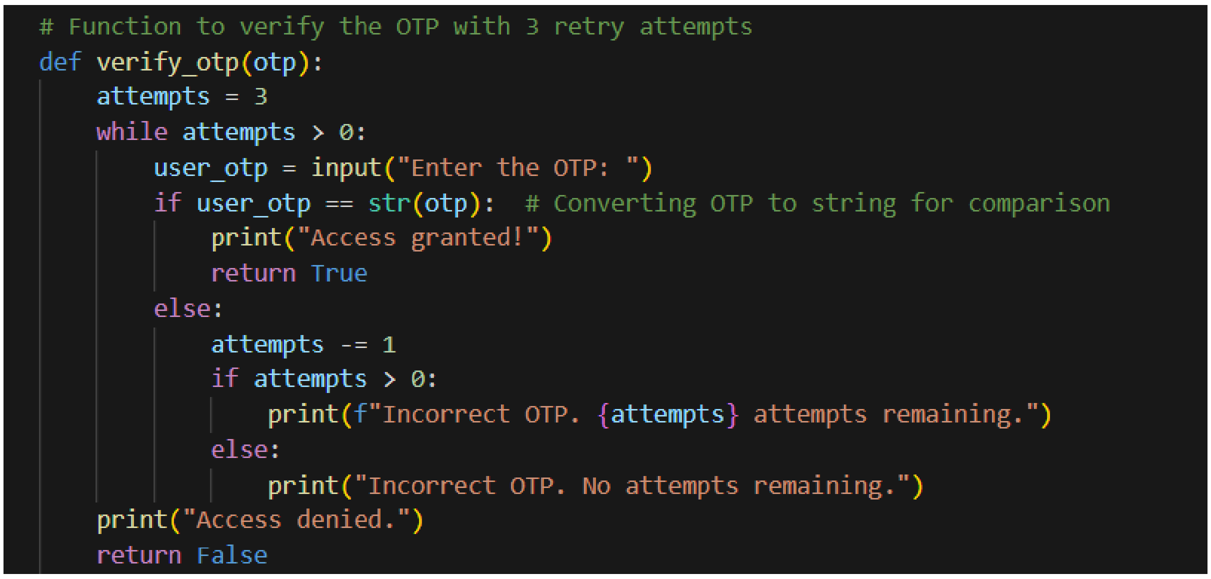
```python
# Function to generate a 6-digit OTP
import random
def generate_otp():
    return random.randint(100000, 999999)
```

2. **Simulating OTP Sending**: The function send_otp_simulated() is used to simulate the sending of the OTP. In a real-world scenario, this function would send the OTP via email or SMS, but here it simply prints the OTP to the console.



```python
# Function to simulate sending the OTP
def send_otp_simulated(otp):
    print(f'An OTP has been sent to your email: {otp}')
```

3. **OTP Verification**: The verify_otp() function compares the OTP entered by the user (user_otp) with the generated OTP. The user has 3 attempts to enter the correct OTP. If the user enters the correct OTP, the access is granted; otherwise, after 3 incorrect attempts, access is denied.



```python
# Function to verify the OTP with 3 retry attempts
def verify_otp(otp):
    attempts = 3
    while attempts > 0:
        user_otp = input("Enter the OTP: ")
        if user_otp == str(otp):  # Converting OTP to string for comparison
            print("Access granted!")
            return True
        else:
            attempts -= 1
            if attempts > 0:
                print(f"Incorrect OTP. {attempts} attempts remaining.")
            else:
                print("Incorrect OTP. No attempts remaining.")
    print("Access denied.")
    return False
```

4. **Main Function**: The otp_verification() function orchestrates the entire flow: generating the OTP, simulating its sending, and verifying the OTP entered by the user.

```python
# Main function
def otp_verification():
    print("Welcome to the OTP Verification System!")

    # Generate OTP
    otp = generate_otp()

    # Simulate sending the OTP
    send_otp_simulated(otp)

    # Attempt to verify the OTP
    success = verify_otp(otp)

    if success:
        print("Thank you!")
    else:
        print("Please try again later.")
```

5. **Running the System**: The system runs by calling the otp_verification() function, which triggers the OTP generation, sending, and verification process.

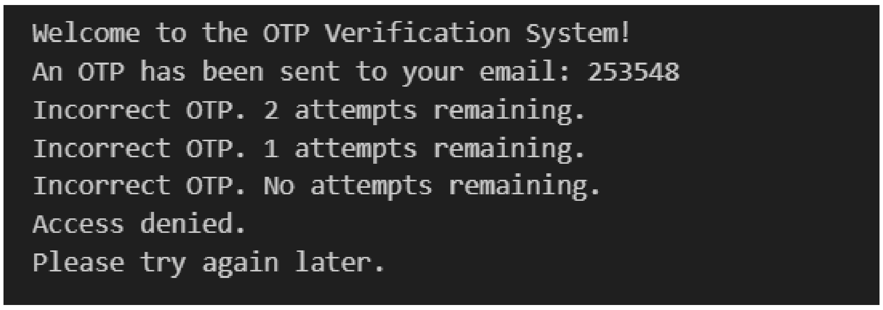
```python
# Call the main function
otp_verification()
```

Sample Output:

The following is an example of how the program behaves:

```
Welcome to the OTP Verification System!
An OTP has been sent to your email: 875109
Access granted!
Thank you!
```

If the user enters an incorrect OTP all three times:



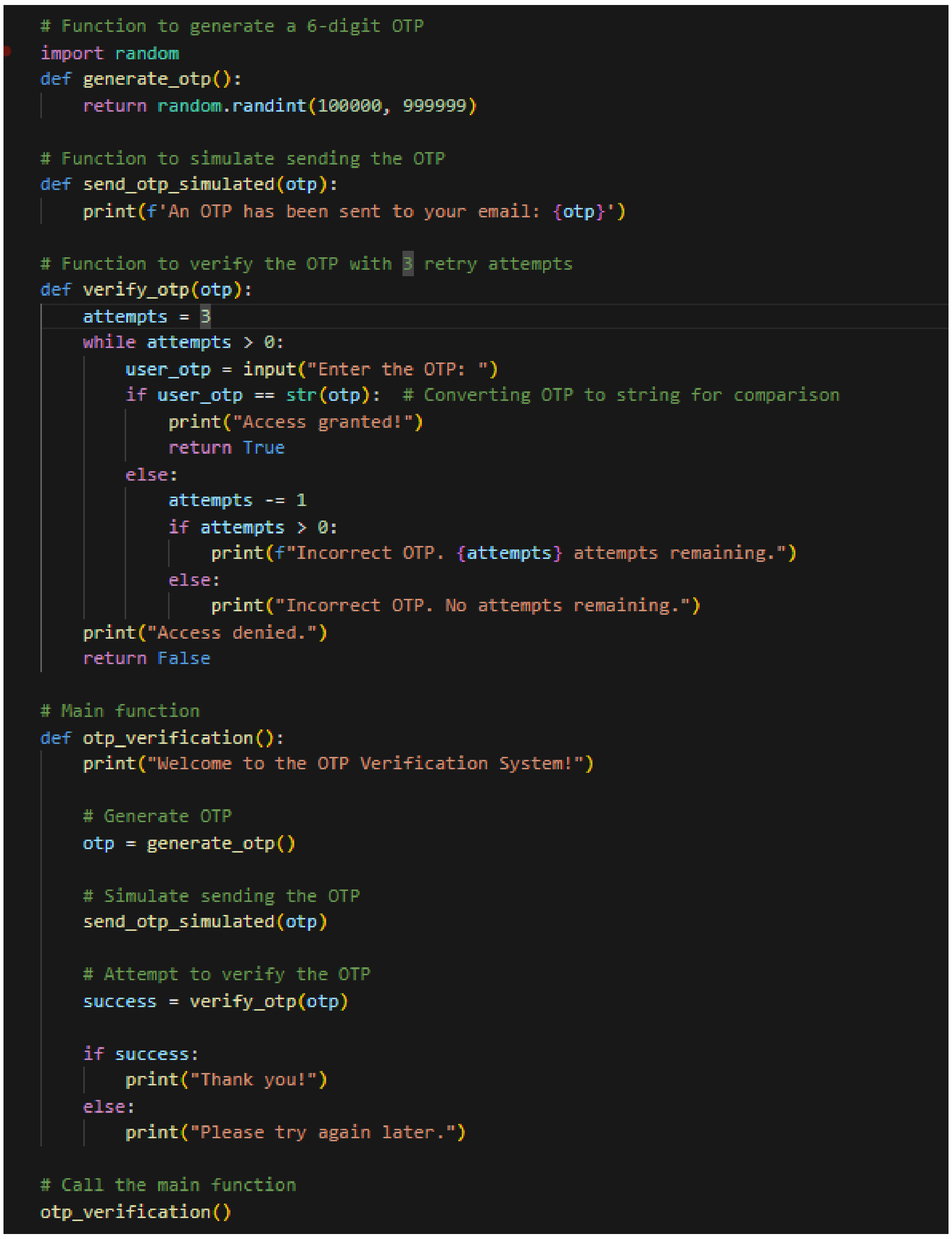
```
Welcome to the OTP Verification System!
An OTP has been sent to your email: 253548
Incorrect OTP. 2 attempts remaining.
Incorrect OTP. 1 attempts remaining.
Incorrect OTP. No attempts remaining.
Access denied.
Please try again later.
```

Features:

- **Random OTP Generation**: A 6-digit OTP is generated randomly for each verification attempt.
- **OTP Sending Simulation**: The OTP is simulated to be sent to the user's email (printed to the console).
- **Multiple Attempts**: The user has up to 3 attempts to enter the correct OTP before access is denied.
- **User Feedback**: Clear messages are provided to inform the user of the number of remaining attempts and the result of the OTP verification.

Conclusion:

This OTP Verification System provides a simple but effective demonstration of how OTPs can be used for secure user authentication. The system is easy to understand and provides a foundation for more complex verification systems, including those integrated with actual email or SMS services.



```python
# Function to generate a 6-digit OTP
import random
def generate_otp():
    return random.randint(100000, 999999)

# Function to simulate sending the OTP
def send_otp_simulated(otp):
    print(f'An OTP has been sent to your email: {otp}')

# Function to verify the OTP with 3 retry attempts
def verify_otp(otp):
    attempts = 3
    while attempts > 0:
        user_otp = input("Enter the OTP: ")
        if user_otp == str(otp):  # Converting OTP to string for comparison
            print("Access granted!")
            return True
        else:
            attempts -= 1
            if attempts > 0:
                print(f"Incorrect OTP. {attempts} attempts remaining.")
            else:
                print("Incorrect OTP. No attempts remaining.")
    print("Access denied.")
    return False

# Main function
def otp_verification():
    print("Welcome to the OTP Verification System!")

    # Generate OTP
    otp = generate_otp()

    # Simulate sending the OTP
    send_otp_simulated(otp)

    # Attempt to verify the OTP
    success = verify_otp(otp)

    if success:
        print("Thank you!")
    else:
        print("Please try again later.")

# Call the main function
otp_verification()
```