OTP Verification System Using Python

By Kinkini Majumdar

1. Project Overview

The OTP (One-Time Password) Verification System is a Python-based project that sends a generated OTP to the user's email and allows them to verify the OTP within a given time frame. The project implements email validation, OTP generation, OTP sending via SMTP, and OTP verification within 5 minutes. The system also includes error handling, retry attempts, and logging.

Key Features:

- Email validation using regex.
- OTP generation (6-digit random number).
- OTP sent to the user's email via SMTP.
- OTP verification with 3 attempts allowed.
- 3- minute timeout for OTP entry.
- Logging of actions and errors.
- User-friendly messages for success and failure scenarios.

2. Project Requirements

2.1 Software Requirements

- **Python**: Version 3.x
- **Libraries**:
    - **random:** For generating a random OTP.
    - **smtplib:** For sending emails via SMTP.
    - **re:** For validating email format using regular expressions.
    - **time:** For handling time-based functionality (OTP timeout).
    - **logging:** For tracking and reporting messages.

2.2 Hardware Requirements

- A system capable of running Python 3.x.
- An internet connection to send OTP emails via SMTP.

3. Project Structure

The project is organized as follows:

```
otp_verification /
|
├── otp_verification.py   # Main script for OTP
|                           generation,
|                           sending, and
|                           verification.
|
└── README.me            # Project documentation
```

File Descriptions:

- **otp_verification.py:** Contains the main logic of the OTP verification system.
    - OtpVerification class: Handles the OTP generation, email sending, and verification processes.
    - Functions for email validation and OTP verification.
    - Uses logging for detailed output and error tracking.

4. Functionality Description

4.1 Email Validation

The system first validates the email entered by the user. It uses a regular expression to ensure that the email follows a standard email format.

4.2 OTP Generation

A random 6-digit OTP is generated using the random.randint() function. This OTP is then sent to the user's email.

4.3 Sending OTP

The system uses Python's smtplib.SMTP module to send the generated OTP to the user's email address. The email is sent with a subject and body message indicating the OTP.

4.4 OTP Verification

The user is prompted to enter the OTP within 5 minutes. They are given 3 attempts to enter the correct OTP. If the user provides the correct OTP, a success message is displayed. If the attempts are exhausted or the time limit is exceeded, the verification fails.

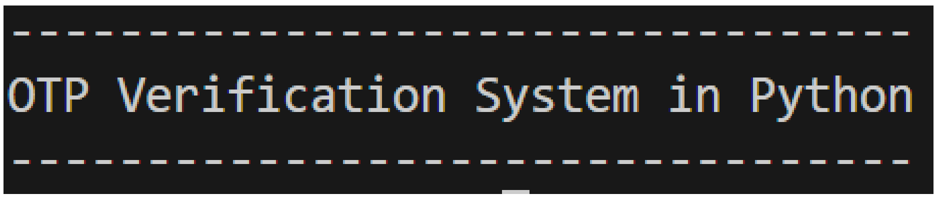
4.5 Logging

The system uses Python's built-in logging module to provide detailed information about the process. It logs the successful sending of OTPs, failed attempts, and any errors that occur during execution.

5. User Interaction Flow

Step 1: Welcome Screen

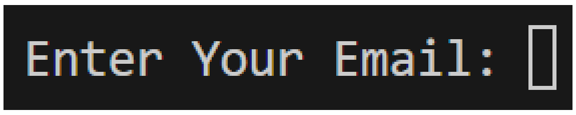
The system displays a welcome message to the user:



```
-----------------------------------
OTP Verification System in Python
-----------------------------------
```

Step 2: Email Input

The user is prompted to enter their email address:



```
Enter Your Email:
```

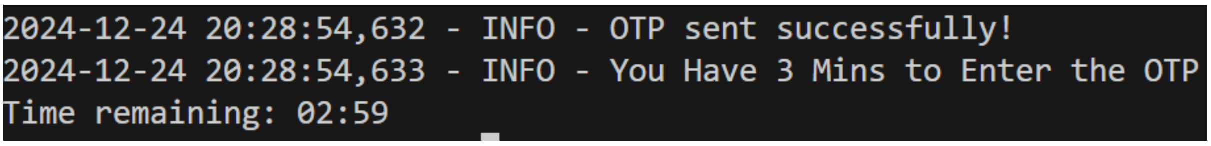
If the email is invalid, the system prompts the user to enter a valid email.

Step 3: OTP Sending

Once a valid email is entered, the system generates an OTP and sends it to the entered email. The system logs the success or failure of the OTP sending process.

Step 4: OTP Verification

The user is prompted to enter the OTP within 5 minutes:

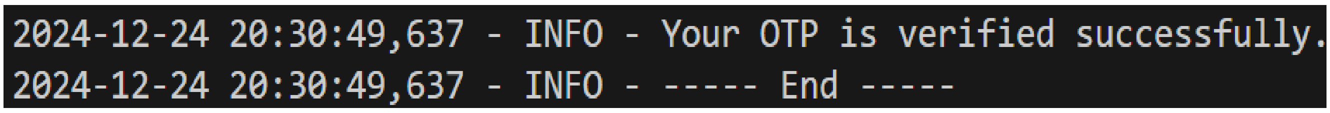


```
2024-12-24 20:28:54,632 - INFO - OTP sent successfully!
2024-12-24 20:28:54,633 - INFO - You Have 3 Mins to Enter the OTP
Time remaining: 02:59
```

The user is allowed 3 attempts to enter the correct OTP. The system will display a countdown timer and an error message after each incorrect attempt.

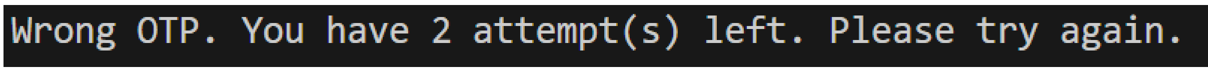
Step 5: OTP Verification Result

- If the user enters the correct OTP, the system displays:



```
2024-12-24 20:30:49,637 - INFO - Your OTP is verified successfully.
2024-12-24 20:30:49,637 - INFO - ----- End -----
```

- If the user enters the wrong OTP or exhausts all attempts, the system will display:



```
Wrong OTP. You have 2 attempt(s) left. Please try again.
```

Step 6: Logging

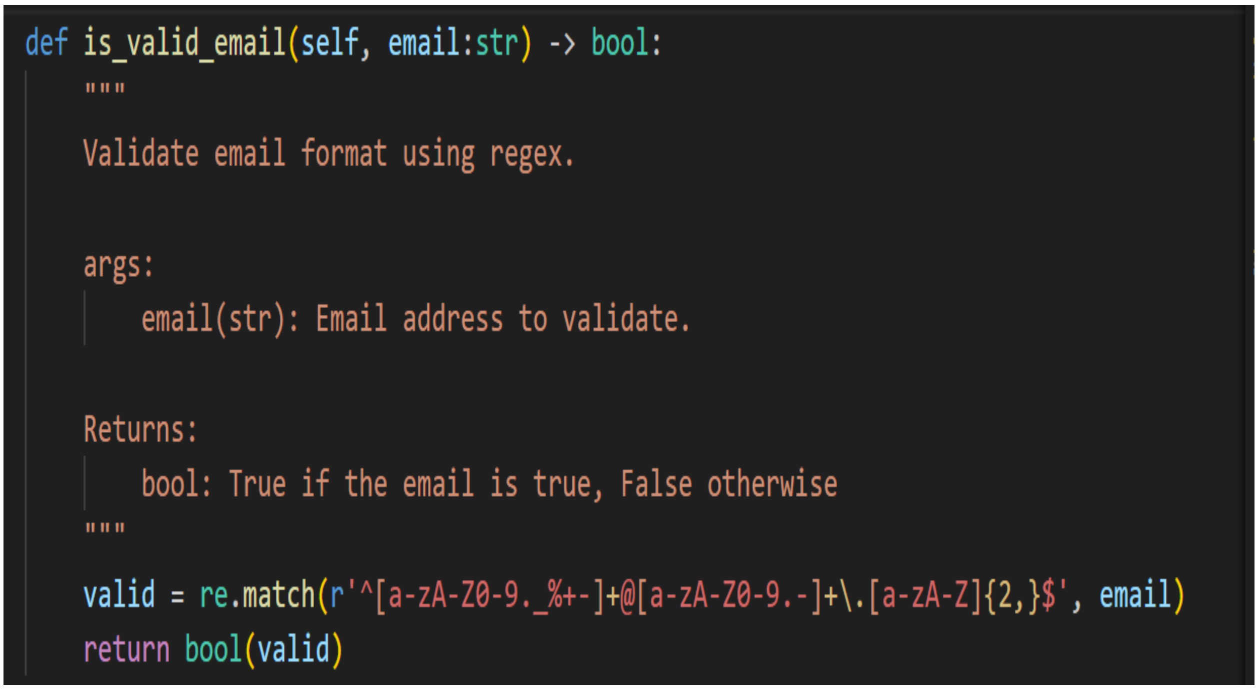
All actions, including OTP sending, validation success, and failure, are logged using the logging module.

6. Error Handling

- **Email Validation**: If the user enters an invalid email, they are prompted to try again until a valid email is provided.
- **SMTP Connection Failure**: If the OTP cannot be sent due to an SMTP error (e.g., incorrect credentials or server issues), an error message is logged.
- **Incorrect OTP**: After 3 failed attempts, the system stops and reports that the OTP verification has failed.
- **Timeout**: If the user does not enter the OTP within 5 minutes, the system stops the verification process.

7. Code Snippets

Email Validation Regex

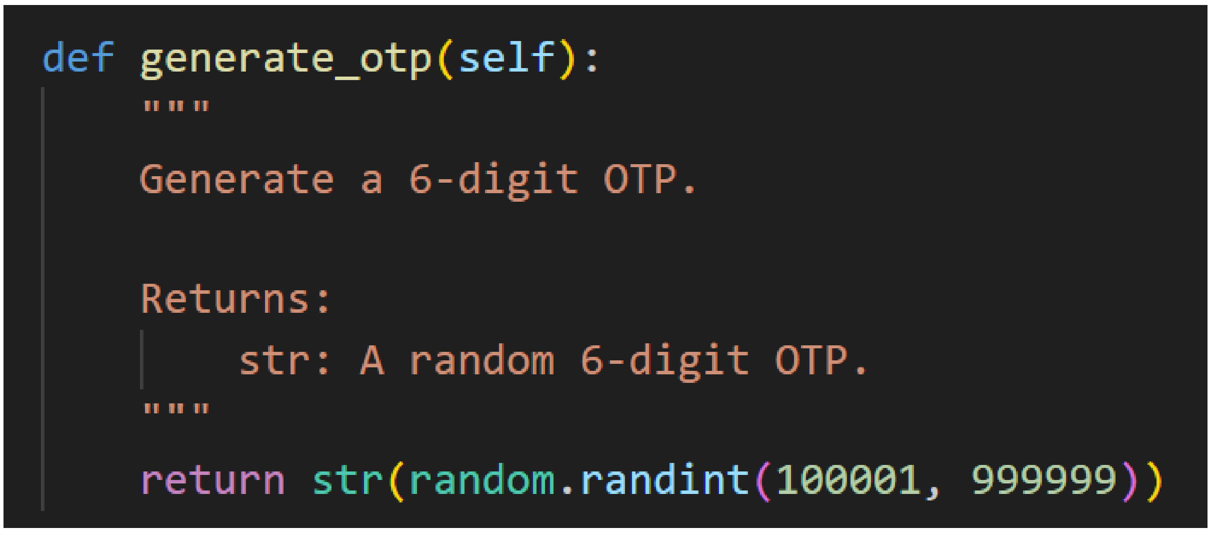


```python
def is_valid_email(self, email:str) -> bool:
    """
    Validate email format using regex.

    args:
        email(str): Email address to validate.

    Returns:
        bool: True if the email is true, False otherwise
    """
    valid = re.match(r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$', email)
    return bool(valid)
```

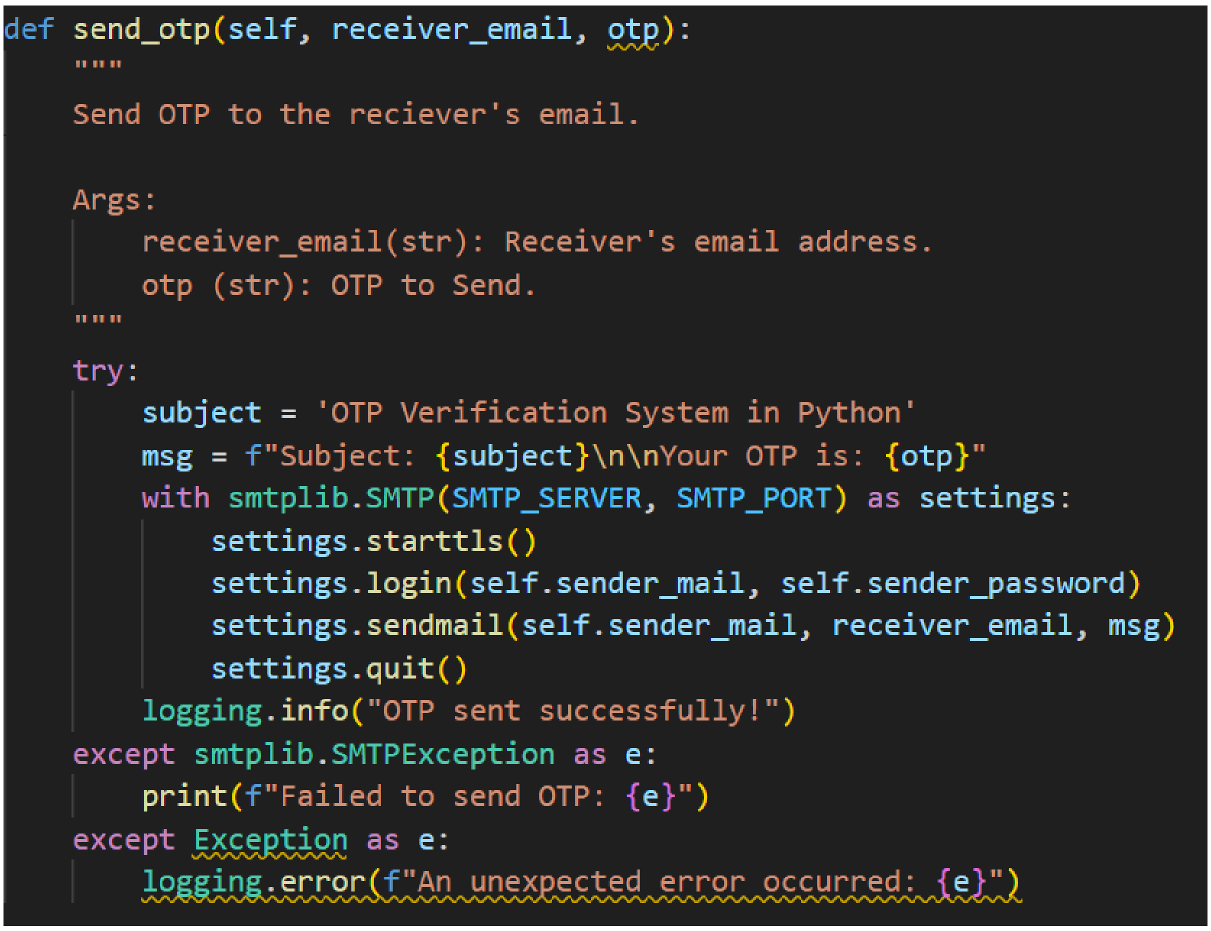
OTP Generation



```python
def generate_otp(self):
    """
    Generate a 6-digit OTP.

    Returns:
        str: A random 6-digit OTP.
    """
    return str(random.randint(100001, 999999))
```

Sending OTP via SMTP



```python
def send_otp(self, receiver_email, otp):
    """
    Send OTP to the reciever's email.

    Args:
        receiver_email(str): Receiver's email address.
        otp (str): OTP to Send.
    """
    try:
        subject = 'OTP Verification System in Python'
        msg = f"Subject: {subject}\n\nYour OTP is: {otp}"
        with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as settings:
            settings.starttls()
            settings.login(self.sender_mail, self.sender_password)
            settings.sendmail(self.sender_mail, receiver_email, msg)
            settings.quit()
        logging.info("OTP sent successfully!")
    except smtplib.SMTPException as e:
        print(f"Failed to send OTP: {e}")
    except Exception as e:
        logging.error(f"An unexpected error occurred: {e}")
```

8. Potential Enhancements

- **Email Template**: Improve the email content by adding HTML formatting and a more professional layout.
- **User Interface (UI)**: Implement a graphical user interface (GUI) using Tkinter for better user experience.
- **SMS OTP**: Extend the system to support sending OTP via SMS using a third-party API like Twilio.
- **Database**: Store OTPs and email addresses in a database for tracking and logging purposes.

9. Conclusion

This OTP Verification System project provides a complete solution for verifying users' identities via email. It includes email validation, OTP generation, and verification, along with error handling and logging. This project is a practical example of handling user verification securely and can be expanded with additional features in the future.

10. Installation and Running the Code

1. **Prerequisites**:
   - Python 3.x installed.
   - A Gmail account with "Allow less secure apps" enabled (or use an app-specific password if 2FA is enabled).
2. **Install Required Libraries**: Ensure you have the required Python libraries by running:

```
pip install smtplib
```

3. **Running the Script**: Save the script as otp_verification.py and run it:

```
python otp_verification.py
```

4. **Test the System**: Follow the prompts to enter a valid email address, receive an OTP, and verify it.