**Capstone Project – Python**  
  
**Name: Sanjiban Hati   
Student ID: S8895**

**Project Title: OTP Verification System  
  
Problem Statement:**

We have been tasked with developing an OTP (One-Time Password) verification system in Python. The system should generate a 6-digit OTP and send it to the user's email address for verification. Upon receiving the OTP, the user should enter it into the system for validation. If the entered OTP matches the generated OTP, access should be granted; otherwise, access should be denied.

**Project Requirements:**

* Implement a function to generate a 6-digit OTP randomly.
* Develop a function to simulate sending the OTP to the user's email address.
* Create a function to prompt the user to enter the OTP received in their email.
* Implement a function to verify if the entered OTP matches the generated OTP.
* Ensure proper error handling and user-friendly prompts throughout the system.
* Allow the user to retry OTP entry in case of incorrect input.

**Step by Step Analysis of the Project:**

**Step 1: Importing the Libraries**

**random:** It’s used for generating random numbers (OTP in this case) **smtplib:** It’s used for defining an SMTP client session object to send email using the Simple Mail Transfer Protocol (SMTP).  
**email.mime.text:** Using the **MIMEText** class from this package, email messages are created with text content.  
**email.mime.multipart:** Provides the **MIMEMultipart** class, which is used to create email messages that can contain multiple parts, such as text and attachments.  
**getpass:** It provides a way to securely handle passwords and other sensitive information by hiding input as it is entered (for inserting the password in this case)   
**re:** It’s used to match the pattern of the email address with the desired format given in the code. Basically its function here is to validate the correctness of the user’s email address.  **os:** It’s being used to retrieve the environment variables like **sender\_email** & **sender\_password**. For security reasons we would prefer to keep these values as environment variables instead of hardcoding them in the functions.   
  

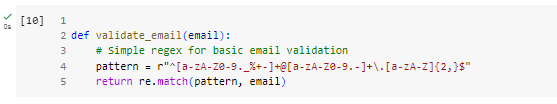

**Step 2: Function to generate a 6-digit OTP randomly**  
  
**generate\_otp( )** is the function that will create a 6-digit OTP in this project. The code along with the output for the same is being shared below.   


**Step 3: Function to simulate sending the OTP to the user's email address**

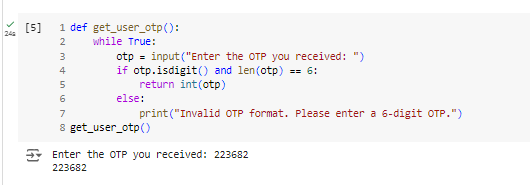
**send\_otp\_via\_mail( )** is the function used for sending the OTP to the user’s mail box. The code along with the output for the same is being shared below.   



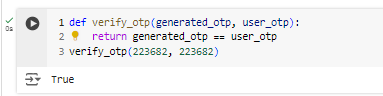

The first two commented lines can be used if the deployment happens through AWS. In that case the variables along with the values will be kept as environment variables only on the lambda and the way they are hardcoded now inside the function, will not be needed in that case. This approach will ensure of a much reliable privacy for important data such as account credentials.   
Also, inside the **send\_otp\_via\_email( )** function has been used another function named **validate\_email( )** whose function is to validate the correctness of the email address shared by the user.   
Code for the same is being shown below.



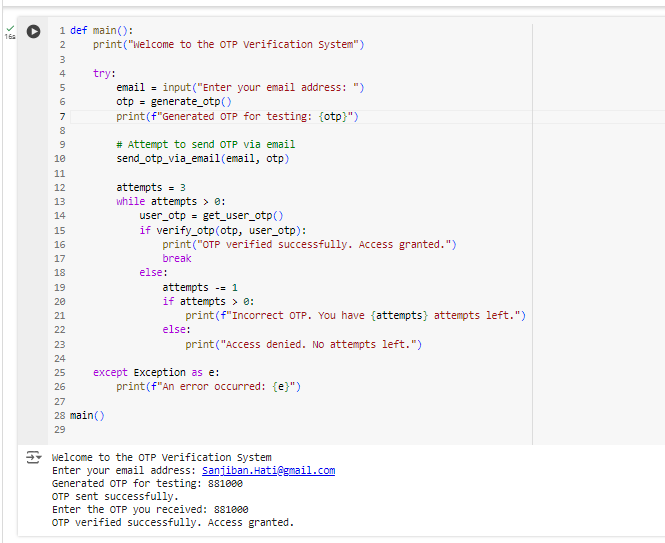
**Step 4: Function to prompt the user to enter the OTP received in their email**

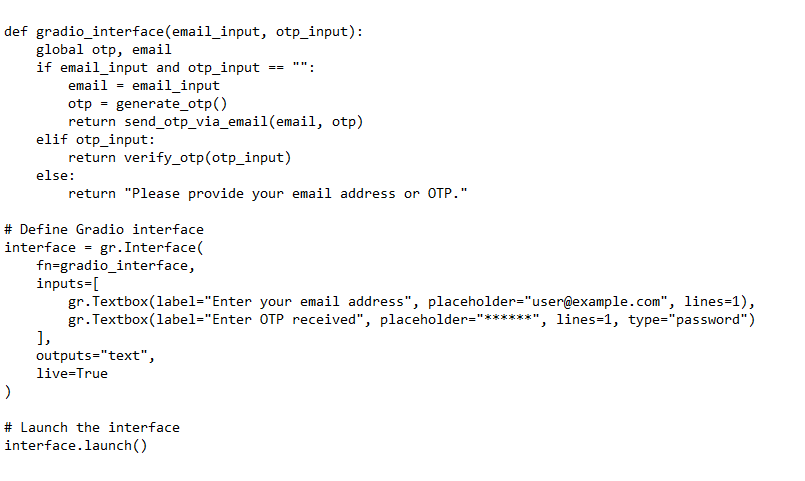
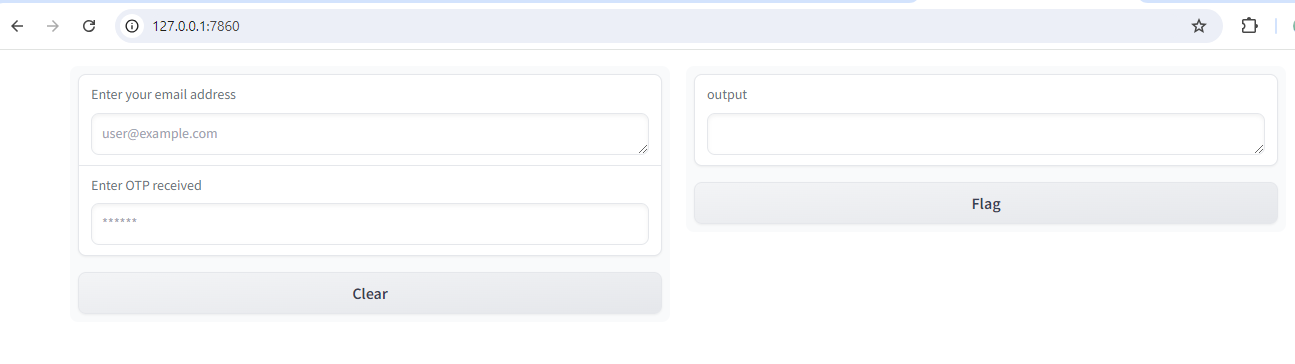
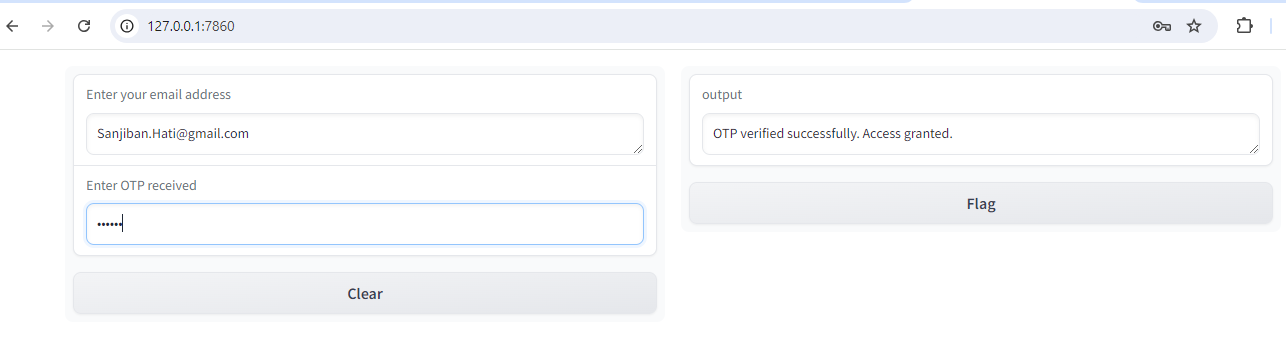
**get\_user\_otp( )** is the function which will prompt the user to enter the OTP received in their mail. The code along with the output for the same is being shared below.  
  


**Step 5: Function to verify if the entered OTP matches the generated OTP**

**verify\_otp( )** function will be used to check if the OTP entered by the User is same as the OTP generated or not. The code along with the output for the same is being shared below.  


**Step 6: main Function to integrate the functions defined earlier**

In this segment, all the functions written so far will be tied together to achieve the desired output. The code along with the output for the same is being shared below.  


**Step 7: Creating simple GUI interface using Gradio**  
  
The code along with the output for the same is being shared below.  
  
  
**gradio** is used for creating quick Graphical User Interface (GUI).   
  
Once the python script (**OTP\_Verification\_System.py**) was prepared, it was run from the command prompt and the following screen appeared.   
  
  
Now on accessing the local URL (<http://127.0.0.1:7860>), we were redirected to the GUI as shown below.  
  
  
  
After inserting the necessary inputs, we were able to get this page showing the following message.  
  
 **Tools used as part of this project:  
1. Google Colab  
2. Command Prompt  
3. Notepad**