**OTP Verification System Using Python GUI**

**Objective:**

To generate a OTP (One Time Password) through Email and verify it simultaneously using python functions and modules.

**Abstract:**

One time Passwords (OTP) is a password that is valid for only one login session in a computer or in a digital device. OTPs have various applications in providing services such as Internet Banking, online transactions, etc. They are made up of a combination of 4 or 6 numeric digits or a 6-digit alphanumeric character.

Using python, we can create an OTP generation and verification GUI. This can be done by following a few steps.

* Email input screen
* Generating, Sorting and Sending OTP
* OTP validating screen



**Introduction:**

In the first step, the initial welcome screen is generated where the user enters the email id to which the OTP needs to be sent. In this screen it also contains the CONTINUE button, on clicking it moves to the second step.

In the next step the random OTP is generated using **random** module, is an in-built module of Python which is used to generate random number. Then it is stored in notepad file and it sends the OTP to the email id that the user has entered using the **smtplib** module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.

In the final step, a screen is created asking the user to enter the OTP that they received in their email. Once the user enters the OTP and clicks on the VALIDATE button, then we can validate the user provided OTP with the OTP stored in our notepad file.

# Software Required:

* Anaconda Navigator
* Juypter

**Libraries Required:**

* Import os
* Import random
* Import sys
* Import smtplib
* Import time
* Import tkinter

# Working Principle:

# The user initiates a login, account recovery, or transaction that requires identity verification.

# The system prompts the user to provide their mobile number, email address, or another predefined contact method.

# The system generates a unique, one-time-use code (the OTP). This code is typically a numeric sequence or alphanumeric characters.

# The OTP is sent to the user's provided contact method, often through SMS, email, or a mobile app. It can also be generated and displayed on a hardware token or within a mobile app (such as Google Authenticator or Authy).

# The OTP is usually valid for a short duration, often around 30 seconds to a few minutes, to ensure that it remains secure. After this time window, the OTP expires.

# The user receives the OTP and enters it into the application or website where verification is required.

# The system checks if the entered OTP matches the one generated for that specific transaction. If the OTP is correct and within the time limit, the user is granted access or authorization.

# Once the OTP has been used or has expired, it cannot be used again for any future verification or transactions.

# Program Code:

**Welcome page to enter mail id:**

# # Import necessary modules

# import tkinter

# import subprocess

# from tkinter import \*

# from tkinter import PhotoImage

# from PIL import Image, ImageTk

# root = tkinter.Tk()# Create a Tkinter main window

# root.title("OTP Mail Verification")# Set the title of the window

# root.geometry("400x400")# Set the dimensions of the window (400x400)

# root.resizable(0, 0)# Make the window non-resizable

# sky\_color = "#76c3ef"# Define colors for backgrounds

# grass\_color = "#aad207"

# output\_color = "#dcf0fb"

# input\_color = "#ecf2ae"

# large\_font = ('SimSun', 14)# Define font styles

# small\_font = ('SimSun', 11)

# sky\_frame = tkinter.Frame(root, bg=sky\_color, height=230)# Create a frame for the sky background

# sky\_frame.pack(fill="both", expand=True)

# grass\_frame = tkinter.Frame(root, bg=grass\_color)# Create a frame for the grass background

# grass\_frame.pack(fill="both", expand=True)

# # Create a frame for displaying output (e.g., the image)

# output\_frame = tkinter.LabelFrame(sky\_frame, bg=output\_color, height=225, width=325)

# output\_frame.pack(pady=30)

# output\_frame.pack\_propagate(0)

# # Create a frame for user input

# input\_frame = tkinter.LabelFrame(grass\_frame, bg=input\_color, width=325)

# input\_frame.pack(pady=15)

# input\_frame.pack\_propagate(0)

# # Function to perform email verification

# def verify():

# cmd = str(mail\_entry.get()) # Get the email address from the entry field

# # Specify the path to the Python script for sending mail

# script\_path = "C:\\Users\\surya\\OneDrive\\Desktop\\mini project\\sendmail.py"

# # Create a list with the command to run the script with the email address as an argument

# cmd\_list = ['python', script\_path, cmd]

# print("Executing command:", cmd\_list) # Print the command being executed

# try:

# subprocess.run(cmd\_list, check=True)# Execute the command and capture the result

# except subprocess.CalledProcessError as e:

# print("Error:", e) # Handle errors by printing an error message

# # Load and display an image on the GUI

# png\_image = Image.open("C:\\Users\\surya\\OneDrive\\Desktop\\mini project\\main.jpg")

# image = ImageTk.PhotoImage(png\_image)

# image\_label = tkinter.Label(output\_frame, image=image)

# image\_label.pack()

# l = tkinter.Label(input\_frame, text='Email')# Create a label for the "Email" input field

# l.grid(row=0, column=0, padx=10, pady=10)

# # Create an entry field for the user to enter their email

# mail\_entry = tkinter.Entry(input\_frame, width=25, font=small\_font)

# mail\_entry.grid(row=0, column=1, padx=10, pady=10)

# # Create a "Submit" button that calls the "verify" function when clicked

# submit\_but = tkinter.Button(input\_frame, font=large\_font, text='Submit', command=verify)

# submit\_but.grid(row=0, column=2, padx=2, pady=10)

# root.mainloop()# Start the Tkinter main event loop

**Sending OTP to mail id:**

# import subprocess # Import the subprocess module for running external commands

# import math # Import the math module for mathematical operations

# import random # Import the random module for generating random numbers

# import smtplib # Import the smtplib module for sending emails

# import sys # Import the sys module for accessing command-line arguments

# mailid = sys.argv[1] # Get the email address as a command-line argument

# digits = "0123456789" # Define a string of digits for OTP generation

# OTP = "" # Initialize an empty string to store the OTP

# # Generate a 6-digit OTP by randomly selecting digits from the 'digits' string

# for i in range(6):

# OTP += digits[math.floor(random.random() \* 10)]

# # Create a message containing the OTP

# msg = 'Your OTP Verification for app is ' + OTP + ' Note.. Please enter OTP within 2 minutes and 3 attempts, otherwise, it becomes invalid'

# file2 = open("otp.txt", "w") # Open a file named "otp.txt" in write mode and write the OTP to it

# file2.write(OTP)

# file2.close()

# smtp\_server = 'smtp.gmail.com' # Set up SMTP server configuration

# smtp\_port = 587

# email\_address = 'suryah6009@gmail.com' # Enter your Gmail credentials here (replace with actual email and password)

# email\_password = 'cgihtohlufurhreh'

# try:

# s = smtplib.SMTP(smtp\_server, smtp\_port) # Create an SMTP client and connect to the SMTP server

# s.starttls() # Start a secure TLS connection

# s.login(email\_address, email\_password) # Log in to your Gmail account

# s.sendmail(email\_address, mailid, msg) # Send the email containing the OTP

# s.quit() # Close the SMTP connection

# # Specify the path to another Python script to be executed

# script\_path = "C:\\Users\\surya\\OneDrive\\Desktop\\mini project\\otpveripage.py"

# cmd\_list = ['python', script\_path] # Create a list with the command to execute the script

# print("Executing command:", cmd\_list) # Print a message indicating the command about to be executed

# try:

# subprocess.run(cmd\_list, check=True) # Execute the specified command (run the other Python script)

# except subprocess.CalledProcessError as e:

# print("Error:", e)

# except Exception as e:

# print("An error occurred:", str(e)) # Handle any exceptions that may occur during the execution of the script

# OTP verifying page:

# from tkinter import \* # Import all modules from the tkinter library

# from tkinter import messagebox # Import the messagebox module for displaying messages

# import tkinter # Import the tkinter module for creating a GUI

# import time # Import the time module for timing functionality

# import subprocess # Import the subprocess module for running external commands

# from tkinter import \* # Import all modules from the tkinter library (this line is redundant)

# from tkinter import PhotoImage # Import the PhotoImage module from tkinter

# from PIL import Image, ImageTk # Import modules from the Python Imaging Library (PIL)

# root = tkinter.Tk() # Create a tkinter window

# root.title("Verification Screen") # Set the window title, dimensions, and make it non-resizable

# root.geometry("400x400")

# root.resizable(0, 0)

# sky\_color = "#76c3ef" # Define colors for various elements

# grass\_color = "#aad207"

# output\_color = "#dcf0fb"

# input\_color = "#ecf2ae"

# large\_font = ('SimSun', 14) # Define font styles

# small\_font = ('SimSun', 11)

# sky\_frame = tkinter.Frame(root, bg=sky\_color, height=230)# Create a frame for the sky background

# sky\_frame.pack(fill="both", expand=True)

# grass\_frame = tkinter.Frame(root, bg=grass\_color) # Create a frame for the grass background

# grass\_frame.pack(fill="both", expand=True)

# # Create an output frame within the sky frame

# output\_frame = tkinter.LabelFrame(sky\_frame, bg=output\_color, height=225, width=325)

# output\_frame.pack(pady=30)

# output\_frame.pack\_propagate(0)

# # Create an input frame within the grass frame

# input\_frame = tkinter.LabelFrame(grass\_frame, bg=input\_color, width=325)

# input\_frame.pack(pady=15)

# input\_frame.pack\_propagate(0)

# count = 3 # A global variable for counting attempts. Initially, there are 3 attempts.

# def verify(): # Define a function for OTP verification

# global count

# global root

# end = time.time() # Timer ends when the user clicks 'verify'

# t = format(end - start) # Calculate the time difference

# print(float(t)) # Print the time in seconds

# if float(t) >= 120: # Check if the user takes more than 2 minutes

# messagebox.showinfo("Time out", "Session Expired... Time out Please regenerate OTP")

# root.destroy()

# else:

# cmd1 = str(otp\_entry.get()) # Get the entered OTP

# script\_path = "C:\\Users\\surya\\OneDrive\\Desktop\\mini project\\verify.py"

# cmd\_list = ['python', script\_path, cmd1]

# print("Executing command:", cmd\_list)

# try:

# subprocess.run(cmd\_list, check=True)

# except subprocess.CalledProcessError as e:

# print("Error:", e)

# ok = 'Invalid OTP: ' + str((count - 1)) + ' attempts remaining'

# count = count - 1

# f1 = open("status.txt", "r")

# bh = f1.read()

# if count >= 1 and bh != "success":

# tkinter.messagebox.askretrycancel("Error", ok)

# f1.close()

# elif count == 0 and bh != "success":

# f = open("otp.txt", "w")

# f.write("")

# f.close()

# messagebox.showinfo("Oooo", "Your 3 attempts were over. Please regenerate OTP")

# f1.close()

# root.destroy()

# elif bh == "success":

# f1.close()

# root.destroy()

# start = time.time() # Start the timer once the screen is entered

# png\_image = Image.open("C:\\Users\\surya\\OneDrive\\Desktop\\mini project\\otp.jpg") # Load and display an image

# image = ImageTk.PhotoImage(png\_image)

# image\_label = tkinter.Label(output\_frame, image=image)

# image\_label.pack()

# # Create labels and an entry field for OTP

# min = tkinter.Label(output\_frame, text='Enter OTP within 2 min', font=('arial', 14))

# min.pack()

# l = tkinter.Label(input\_frame, text='Enter OTP')

# l.grid(row=0, column=0, padx=10, pady=10)

# otp\_entry = tkinter.Entry(input\_frame, width=25, font=small\_font)

# otp\_entry.grid(row=0, column=1, padx=10, pady=10)

# # Create a 'Verify' button that calls the 'verify' function

# submit\_but = tkinter.Button(input\_frame, font=large\_font, text='Verify', command=verify)

# submit\_but.grid(row=0, column=2, padx=2, pady=10)

# root.mainloop() # Start the main tkinter event loop

**Display verification page:**

# import sys # Import the sys module for accessing command-line arguments

# from tkinter import messagebox # Import the messagebox module from tkinter for displaying messages

# from tkinter import \* # Import all modules from the tkinter library

# import time # Import the time module for timing functionality

# b = sys.argv[1] # Get the OTP value from the command-line argument passed when the script is executed

# f1 = open("otp.txt", "r") # Open the "otp.txt" file in read mode to read the stored OTP

# b1 = f1.read() # Read the OTP from the file

# f1.close() # Close the file

# if b == b1: # Check if the provided OTP (b) matches the OTP from the file (b1)

# f = open("status.txt", "w") # If the OTPs match, write "success" to the "status.txt" file

# f.write("success")

# f.close()

# # Display a success message using a message box

# messagebox.showinfo("Congratulations", "Your OTP was verified Successfully!!")

# else:

# f = open("status.txt", "w") # If the OTPs do not match, write "failure" to the "status.txt" file

# f.write("failure")

# f.close()

**Flowchart:**

start

User Initiates Verification

System Requests User Contact Method

User Provides Contact Method

System Generates OTP

System Sends OTP to User

User Receives OTP

User Enters OTP

System Validates OTP

Enter Valid OTP

Is OTP Correct and Within Time Limit

No

Yes

Grant Access

End

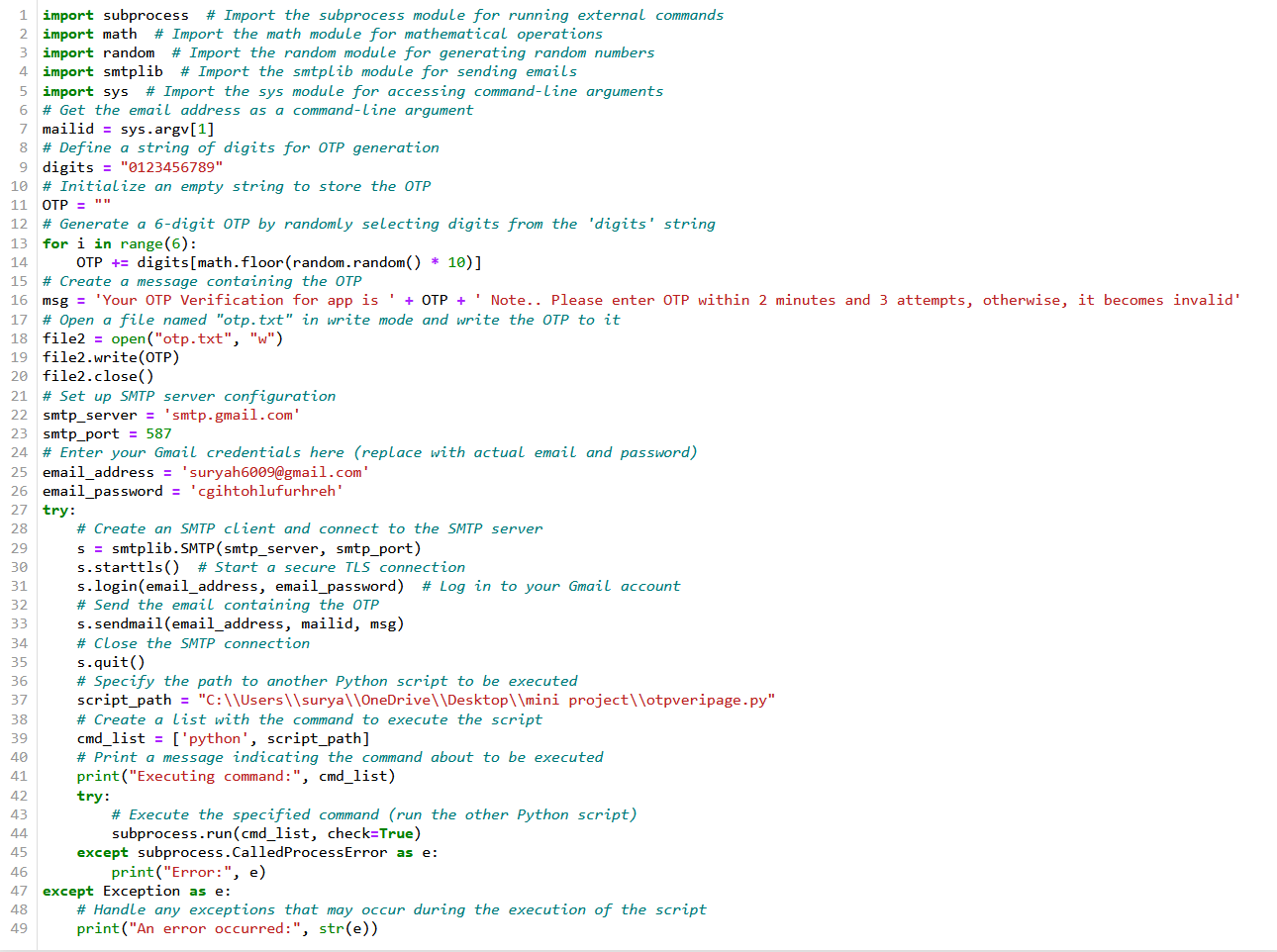
# Simulation Results

**Code:**

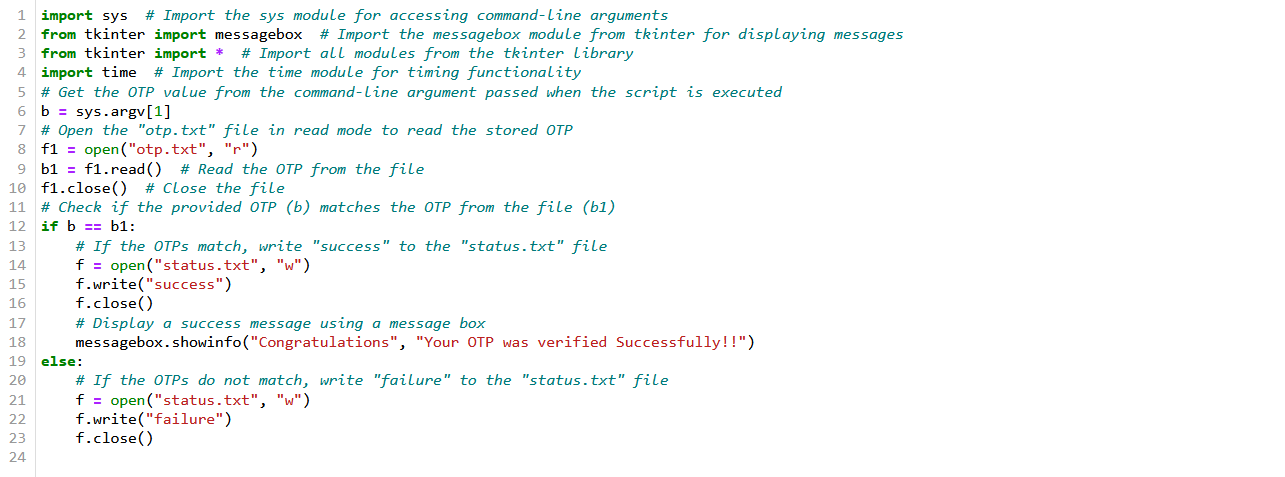
1. **Welcome page to enter mail id:**

****

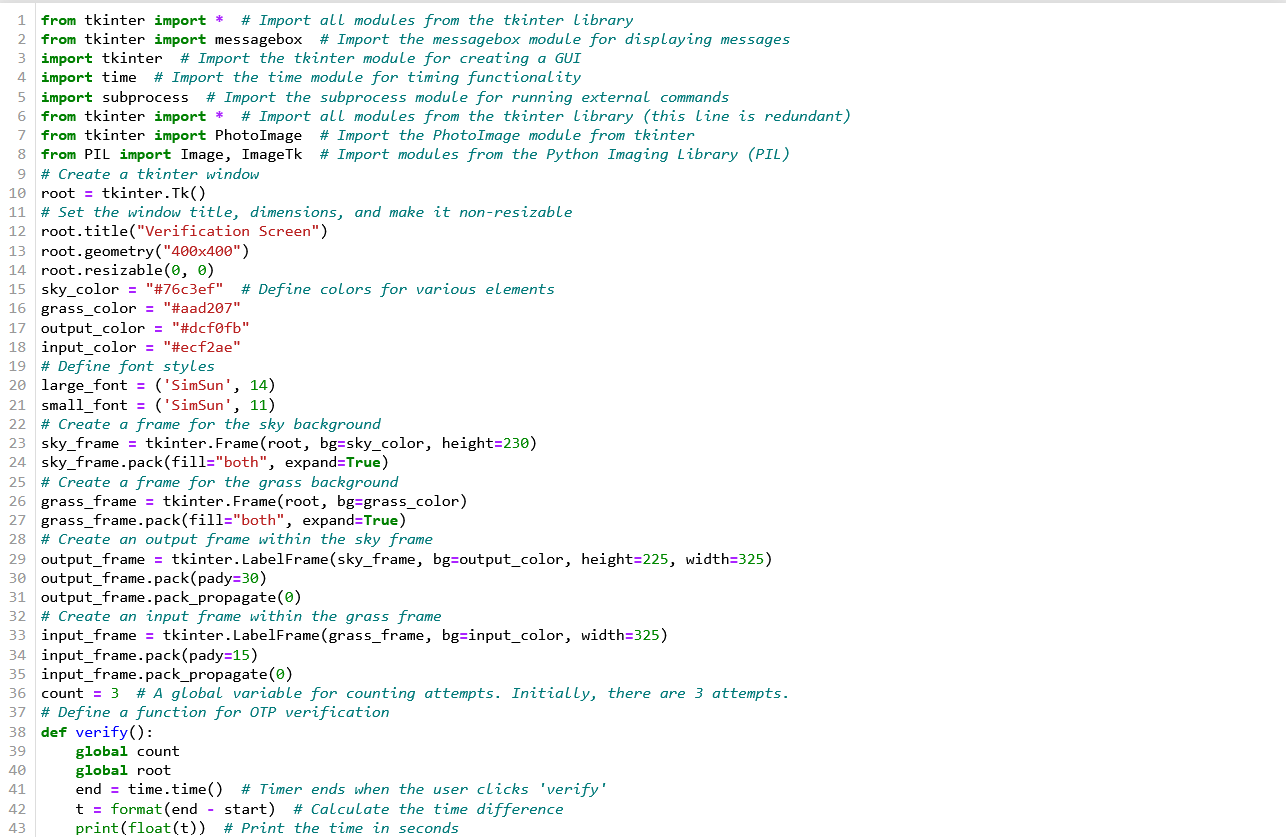
1. **Sending OTP to mail id:**

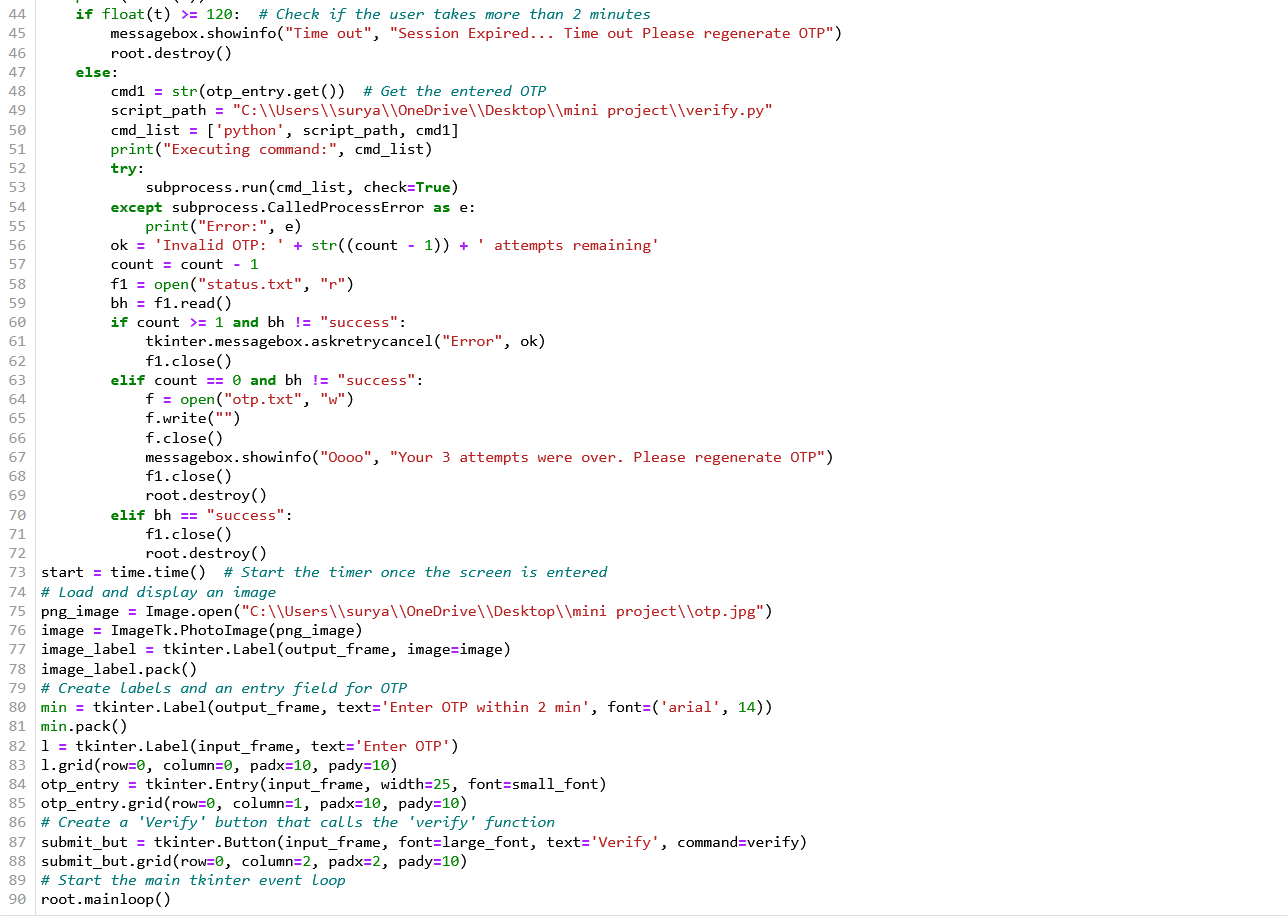
****

1. **Display verification page:**

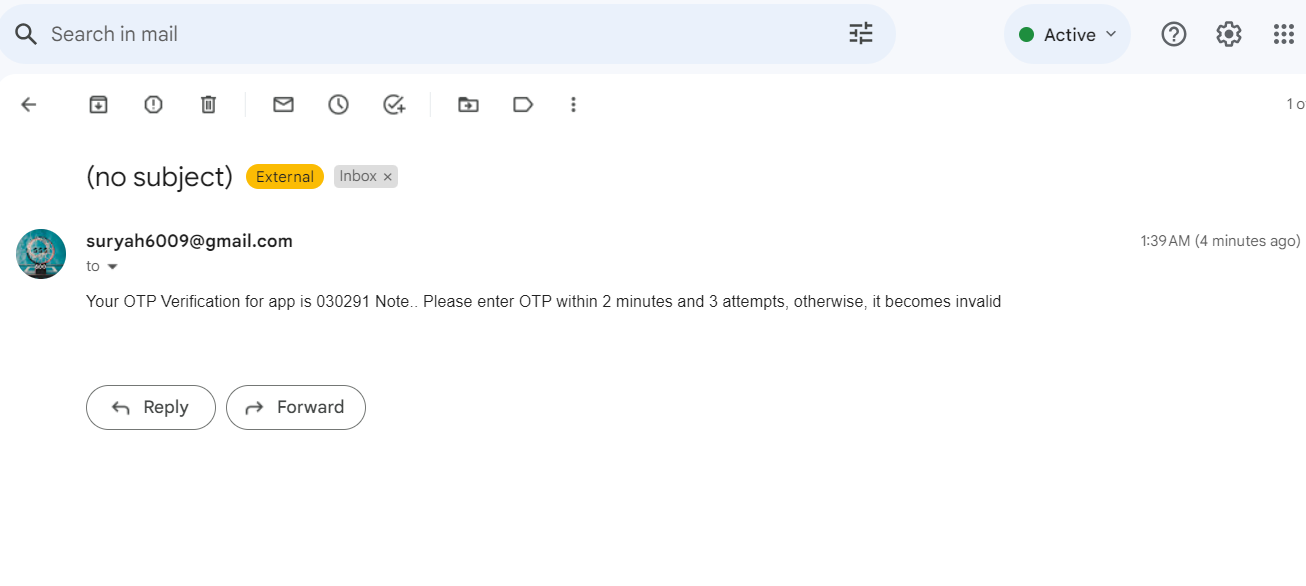
****

**D) OTP verifying page:**

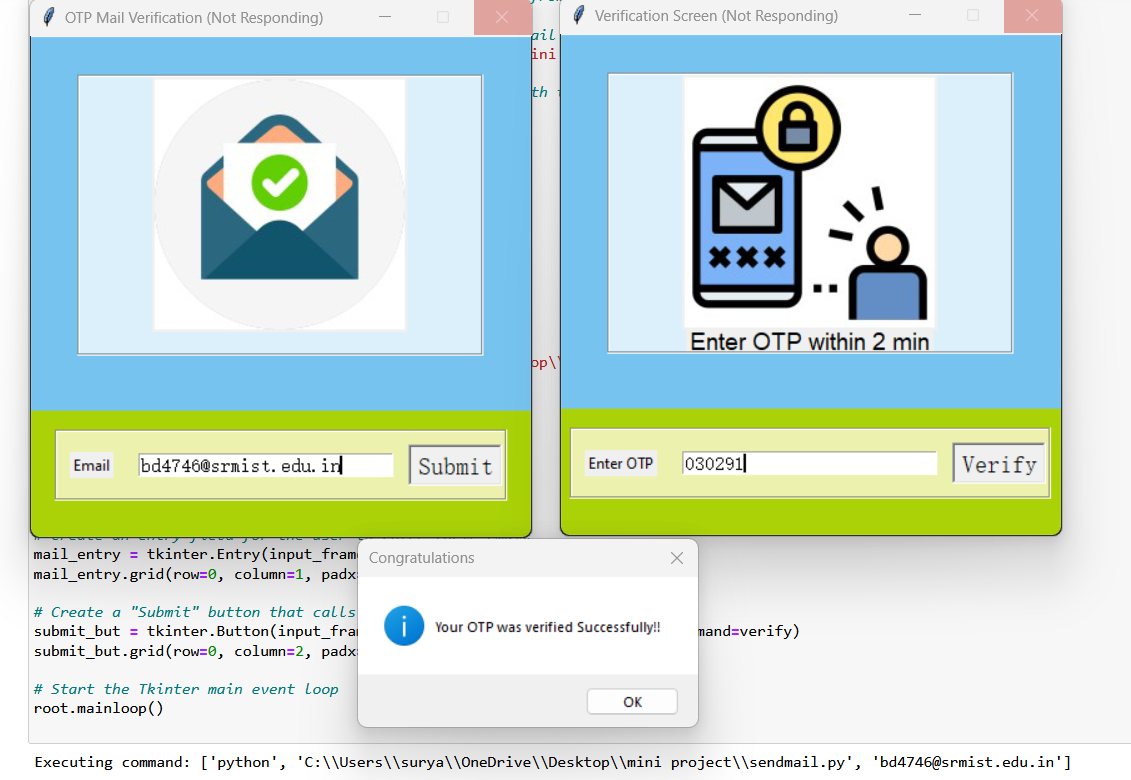
****



**Output:**

****

* This screen shows that the OTP is generated to the respected mail id that is entered.

****

* This shows that the OTP entered is correct and verified successfully.

**Inference from Result:**

The output of our code after execution shows that a page is created to enter the mail id and OTP is the generated to that mail id and the OTP is entered to verify and if the OTP is matched then it shows your OTP was verified successfully.

# Applications:

* Using OTP’s to secure payment or confirming transactions allows real time communications and can significantly reduce costly fraud resolution.
* OTP can be used to verify that the person trying to attain access to an account is the original owner of that account and not a hacker trying to steal your account and information.
* OTP's can also be used to confirm that you want to secure multiple account devices to one account. This ensures that all your devices are secure and are linked to one account of your choosing, which also increases the security of your devices and account.
* Making changes to your profile can be confirmed with an SMS message to the mobile number withholding an OTP. This step will verify the changes made to your profile.
* When a user logs in to an app or website from an unknown or alternative device and requests a password reset, sending an OTP via SMS to verify the user’s identity can help reduce fraud and identity theft.
* When a user of an application or website attempts to sign in after a long period of inactivity, an OTP can help ensure once again that your user is genuine and not a hacker or spammer.

# Result:

The OTP Verification System Using Python GUI code was successfully executed and the desired output was obtained.