**Batch:** P1-2

**Group:** 10 (050, 058, 064)

**Title:** Automated mails from excel file

**Code:**

import tkinter as tk

from tkinter import filedialog

from tkinter import messagebox

from tkinter import simpledialog

import logging # logging messages and errors

import os # logging messages and errors

import re # regular expressions

import socket # network communication

import smtplib # sending emails

import time # time-related functions

import pandas as pd # working with excel files

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

def send\_emails():

    sender\_email = "SENDER\_MAIL" # email credentials

    sender\_password = os.environ.get("SENDER\_PASSWORD")

    if sender\_password is None:

        logging.error("Error: 'SENDER\_PASSWORD' environment variable not set.")

        messagebox.showerror("Error", "'SENDER\_PASSWORD' environment variable not set.")

        return

    excel\_file = excel\_file\_entry.get() # getting the Excel file path from entry field

    sheet\_name = sheet\_name\_entry.get() # getting the sheet name from entry field

    try:

        with pd.ExcelFile(excel\_file) as xls: # opens excel file

            data = pd.read\_excel(excel\_file, sheet\_name=sheet\_name) # reads excel sheet data into a dataframe object called data

        # extracts emails, names, and marks from the dataframe and converts column into respective lists

        emails = data["Email"].tolist()

        names = data["Name"].tolist()

        marks = data["Marks"].tolist()

        status = [] # initializing empty list for status

        smtp\_server = "smtp.gmail.com" # Gmail smtp server is used for sending mails

        smtp\_port = 587 # smtp server port

        email\_pattern = r'([A-Za-z0-9]+[.-\_])\*[A-Za-z0-9]+@[A-Za-z0-9-]+(\.[A-Z|a-z]{2,})+' # pattern for validating email

        try:

            smtp\_connection = smtplib.SMTP(smtp\_server, smtp\_port) # connecting to SMTP server using 'smtp.lib.SMTP'

            smtp\_connection.starttls() # to enable encryption for secure connection

            smtp\_connection.login(sender\_email, sender\_password) # authenticates email address and password with server

            for email, name, mark in zip(emails, names, marks): # iterates over 3 lists simultaneously using zip = can use email, name, marks in one iteration

                email = email.strip() # removes leading/trailing whitespace

                while not re.match(email\_pattern, email): # validate email using regular expression pattern

                    if email == "-1": # if -1, email = invalid email & exit loop

                        status.append("Invalid Email")

                        break

                    logging.warning(f"Invalid email address: {email}") # dialog box to enter valid email address

                    email = simpledialog.askstring("Invalid Email", f"Please enter a valid email address for {name} (Enter -1 to exit):")

                if email == "-1": # exit the loop if -1

                    break

                subject = f"Marks Notification for {name}" # subject of mail

                body = f"Dear {name},\n\nYou have scored {mark} marks in Python programming.\n\nRegards,\nKJSCE"

                message = f"Subject: {subject}\n\n{body}"

                try:

                    smtp\_connection.sendmail(sender\_email, email, message) # sendmail = sends mail

                    logging.info(f"Marks sent successfully to {name}!")

                    status.append("Sent")

                except smtplib.SMTPException as e:

                    logging.error(f"Failed to send email to {name}: {str(e)}")

                    status.append("Unsent")

                    for attempt in range(2): # retries to send unsent mails 2 times with 10 sec difference

                        time.sleep(10)

                        try:

                            smtp\_connection.sendmail(sender\_email, email, message)

                            logging.info(f"Marks sent successfully to {name} after retry!")

                            status[-1] = "Sent"

                            break

                        except smtplib.SMTPException as e:

                            logging.error(f"Failed to send email to {name} on retry: {str(e)}")

            data["Status"] = status # add the status column to the DataFrame

            # save updated data to new excel file

            updated\_file = filedialog.asksaveasfilename(defaultextension=".xlsx", filetypes=[("Excel files", "\*.xlsx")]) # path to updated excel file

            if updated\_file:

                data.to\_excel(updated\_file, index=False)

                logging.info(f"Email status updated and saved to '{updated\_file}'.")

                invalid\_unsent\_data = data[data["Status"].isin(["Invalid Email", "Unsent"])] # saving invalid/unsent email data to another excel file

                invalid\_unsent\_file = filedialog.asksaveasfilename(defaultextension=".xlsx", filetypes=[("Excel files", "\*.xlsx")])

                if invalid\_unsent\_file:

                    invalid\_unsent\_data.to\_excel(invalid\_unsent\_file, index=False)

                    logging.info(f"Invalid and unsent email data saved to '{invalid\_unsent\_file}'.")

        except smtplib.SMTPAuthenticationError as e:

            logging.error(f"SMTP authentication failed. Please check your email credentials: {str(e)}")

        except (smtplib.SMTPConnectError, socket.gaierror) as e:

            logging.error("Failed to establish a connection to the SMTP server.\nPlease check your internet connection.")

        finally:

            if 'smtp\_connection' in locals():

                smtp\_connection.quit() # close the smtp connection

    except pd.errors.ParserError as e: # if file is not valid excel file or issues with the file's format

        logging.error(f"Failed to parse the Excel file: {str(e)}")

    except FileNotFoundError as e: # excel file is not found

        logging.error(f"Failed to find the Excel file: {str(e)}")

    except KeyError as e: # extracting data from the DataFrame

        logging.error(f"Failed to extract data from the DataFrame: {str(e)}")

    except Exception as e: # no specific type of exception

        logging.error(f"An error occurred: {str(e)}")

def browse\_file():

    filename = filedialog.askopenfilename(filetypes=[("Excel files", "\*.xlsx")]) # opens dialog to browse & select excel file

    if filename:

        excel\_file\_entry.delete(0, tk.END) # seting selected file path in entry field

        excel\_file\_entry.insert(0, filename)

root = tk.Tk() # GUI initialization

root.title("Email Sender")

# GUI components

excel\_file\_label = tk.Label(root, text="Excel File:")

excel\_file\_entry = tk.Entry(root, width=50)

browse\_button = tk.Button(root, text="Browse", command=browse\_file)

sheet\_name\_label = tk.Label(root, text="Sheet Name:")

sheet\_name\_entry = tk.Entry(root)

send\_emails\_button = tk.Button(root, text="Send Emails", command=send\_emails)

# GUI layout

excel\_file\_label.grid(row=0, column=0, padx=5, pady=5)

excel\_file\_entry.grid(row=0, column=1, padx=5, pady=5)

browse\_button.grid(row=0, column=2, padx=5, pady=5)

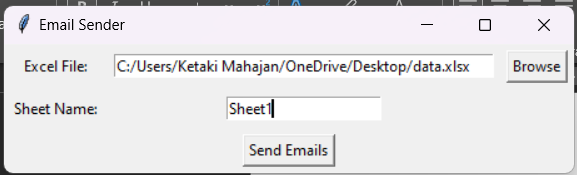
sheet\_name\_label.grid(row=1, column=0, padx=5, pady=5)

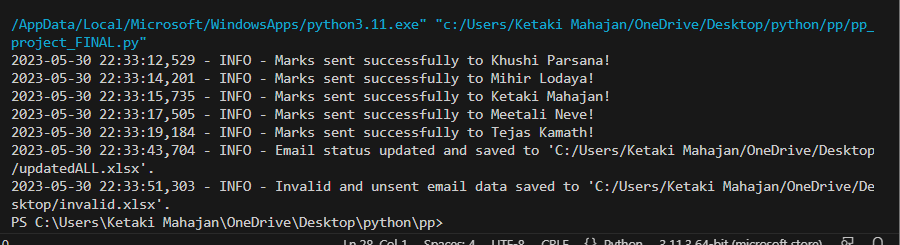
sheet\_name\_entry.grid(row=1, column=1, padx=5, pady=5)

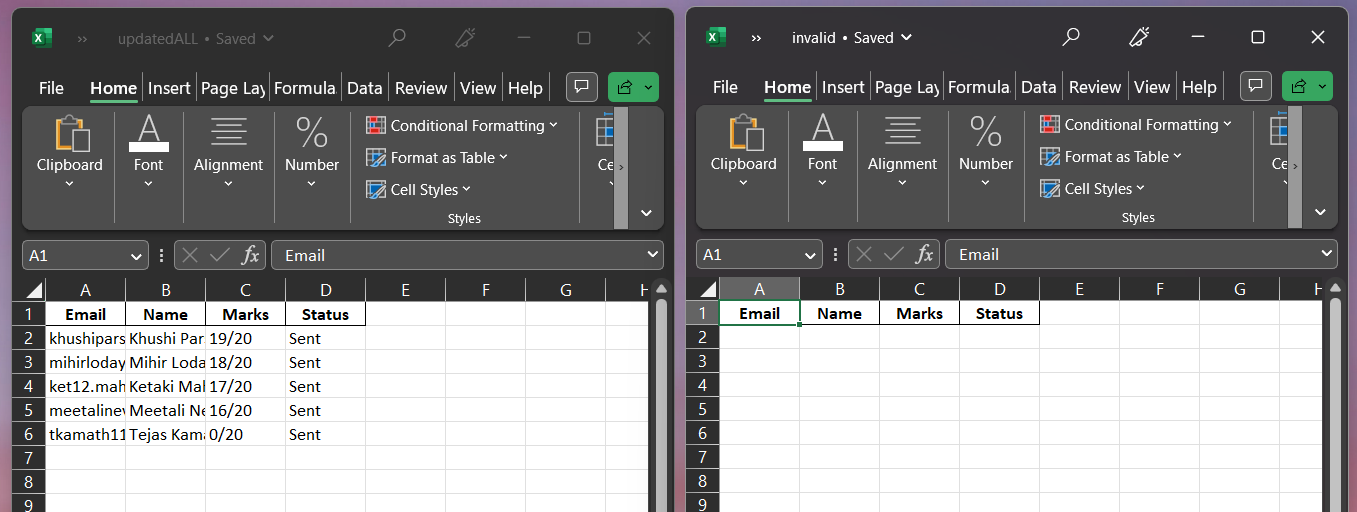
send\_emails\_button.grid(row=2, column=0, columnspan=3, padx=5, pady=5)

root.mainloop() # running GUI

**Outputs:**

****

****

****