DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Name of the Student: Jijo G

University Register No: 963522104057

Branch: CSE

Semester: V

Academic Year: 2024 – 2025 (Odd)

Table of Contents

Introduction

The Birthday Bot is an automation tool built using Python and pandas that simplifies the process of sending personalized birthday emails to employees. By utilizing an Excel sheet as input, it ensures no birthdays are missed, improving employee satisfaction and saving administrative time.

Objective of the Project

The objective is to automate the process of sending personalized birthday emails to employees by:

- **Reading employee data from an Excel file.**
- **Identifying employees whose birthdays fall on the current day.**
- **Sending personalized emails to those employees.**

Tools and Technologies Used

Python:

- **Used for data manipulation, email sending, and overall automation.**

Libraries and Frameworks

- **pandas: For processing Excel data.**
- **openpyxl: To handle .xlsx file reading.**
- **python-dotenv: For securely managing sensitive credentials.**
- **smtplib: For sending emails using an SMTP server.**

Development Environment

- **GitHub Codespaces: A cloud-based IDE for seamless development and deployment.**

Project Workflow

Step-by-Step Workflow

1. **Read Employee Data: Load the Excel file containing employee details.**
2. **Parse and Validate Data: Ensure the data is clean and properly formatted.**
3. **Filter for Today’s Birthdays: Identify employees with birthdays matching today’s date.**
4. **Generate Personalized Messages: Create a custom email for each employee.**
5. **Send Emails: Connect to the SMTP server and send the emails.**
6. **Log the Results: Print the status of email deliveries.**

Tools and Technologies Used

The project utilizes a combination of tools and technologies to achieve its goals:

Programming Language

- Python: The core language used to implement the automation logic.

Libraries and Frameworks

- pandas: For data manipulation and performing aggregate operations on the Excel files.
- openpyxl: For reading, writing, and modifying Excel files.

Software

- Microsoft Excel: To provide the input student data and verify the output result files.

Development Environment

- IDE/Text Editor: Python scripts were developed and executed using Visual Studio Code and Jupyter Notebook.
- Operating System: Windows 10/11 was used during project development and testing.

Project Workflow

Step 1: Input Excel File Preparation

- Create an Excel file (input_file.xlsx) containing columns for student details (e.g., Name, Roll Number) and marks for various subjects.

Step 2: Python Script Development

- Write a Python script to:
    - Read data from the input Excel file.
    - Calculate total marks, percentage, and grades.
    - Format the output results and save them to a new Excel file.

Step 3: Generate Output Excel File

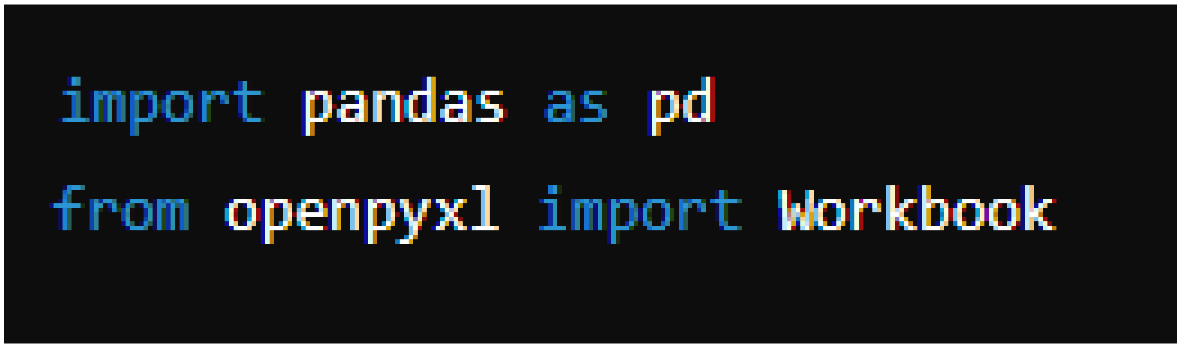
- The script processes each row of data, performs calculations, and writes the results to a new Excel file.
- The output file is named following a specific naming convention, e.g., Result_Batch2024.xlsx.

Step 4: Verification

- Open the output Excel file to verify that the results are accurate and formatted correctly.

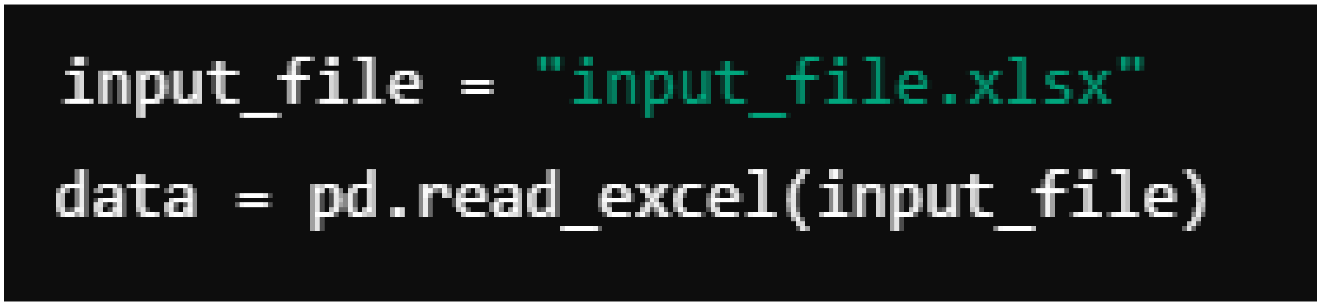
Code Explanation

1. Importing Libraries



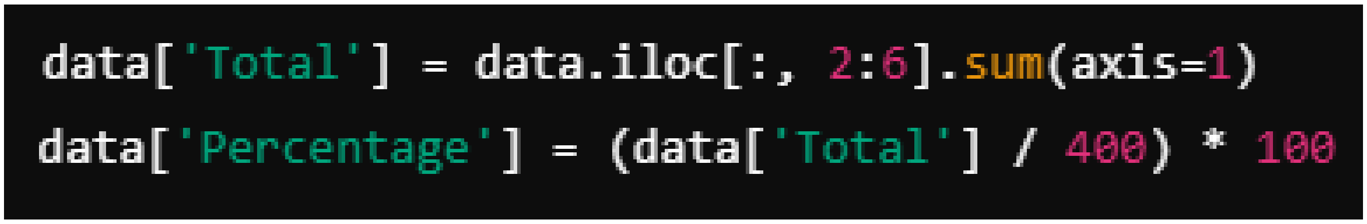
```python
import pandas as pd
from openpyxl import Workbook
```

2. Reading the Input File



```python
input_file = "input_file.xlsx"
data = pd.read_excel(input_file)
```

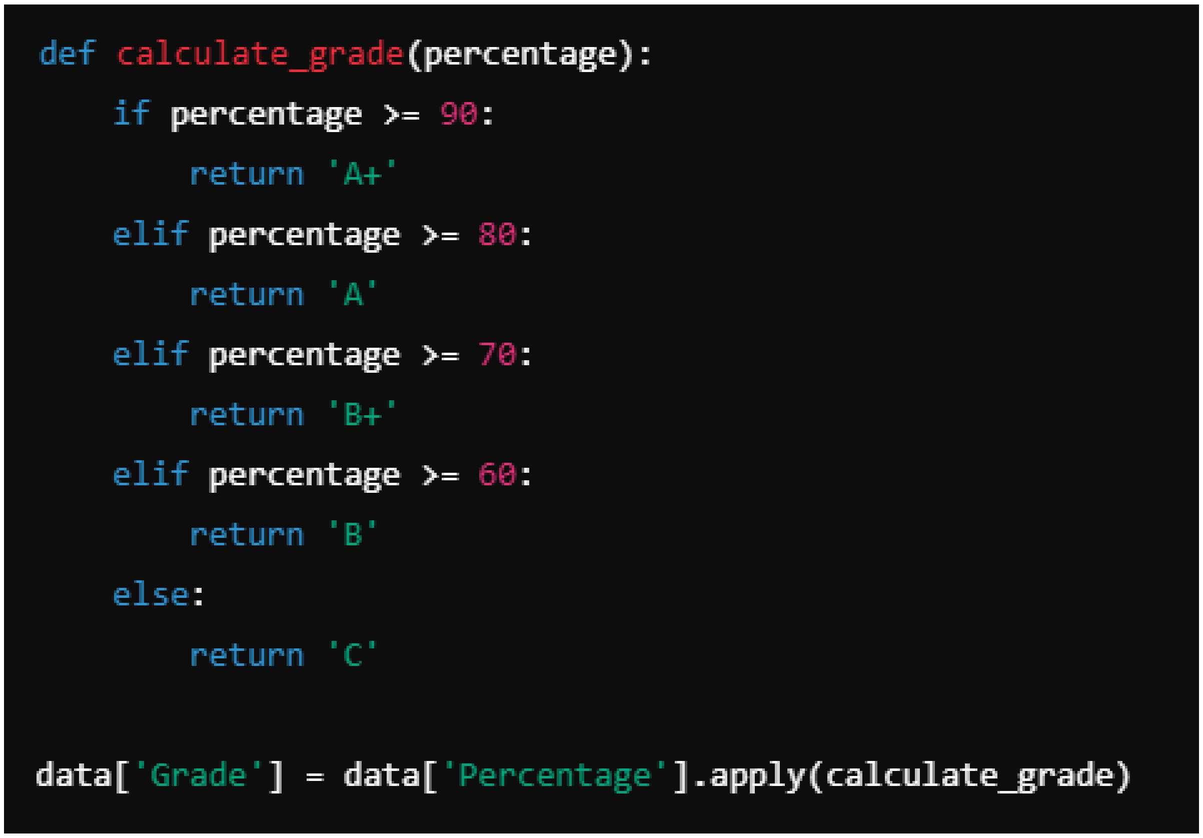
3. Calculating Totals and Percentages



```python
data['Total'] = data.iloc[:, 2:6].sum(axis=1)
data['Percentage'] = (data['Total'] / 400) * 100
```

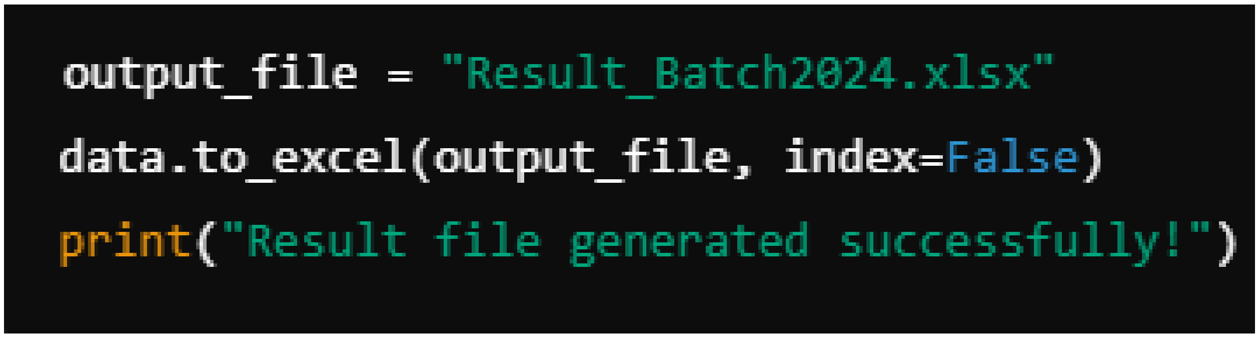
Code Explanation

4. Assigning Grades



```python
def calculate_grade(percentage):
    if percentage >= 90:
        return 'A+'
    elif percentage >= 80:
        return 'A'
    elif percentage >= 70:
        return 'B+'
    elif percentage >= 60:
        return 'B'
    else:
        return 'C'

data['Grade'] = data['Percentage'].apply(calculate_grade)
```

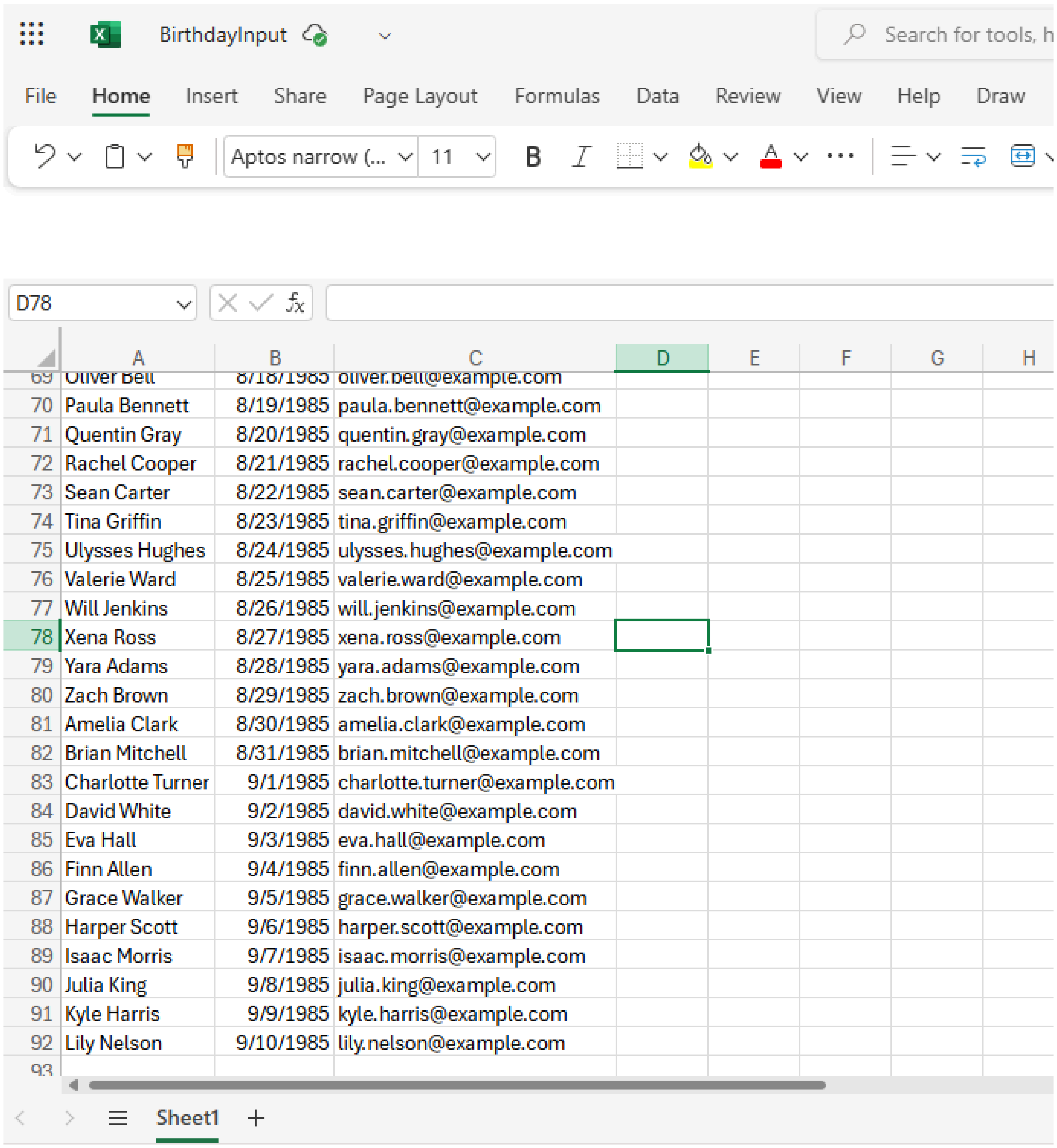
5. Writing to the Output File



```python
output_file = "Result_Batch2024.xlsx"
data.to_excel(output_file, index=False)
print("Result file generated successfully!")
```

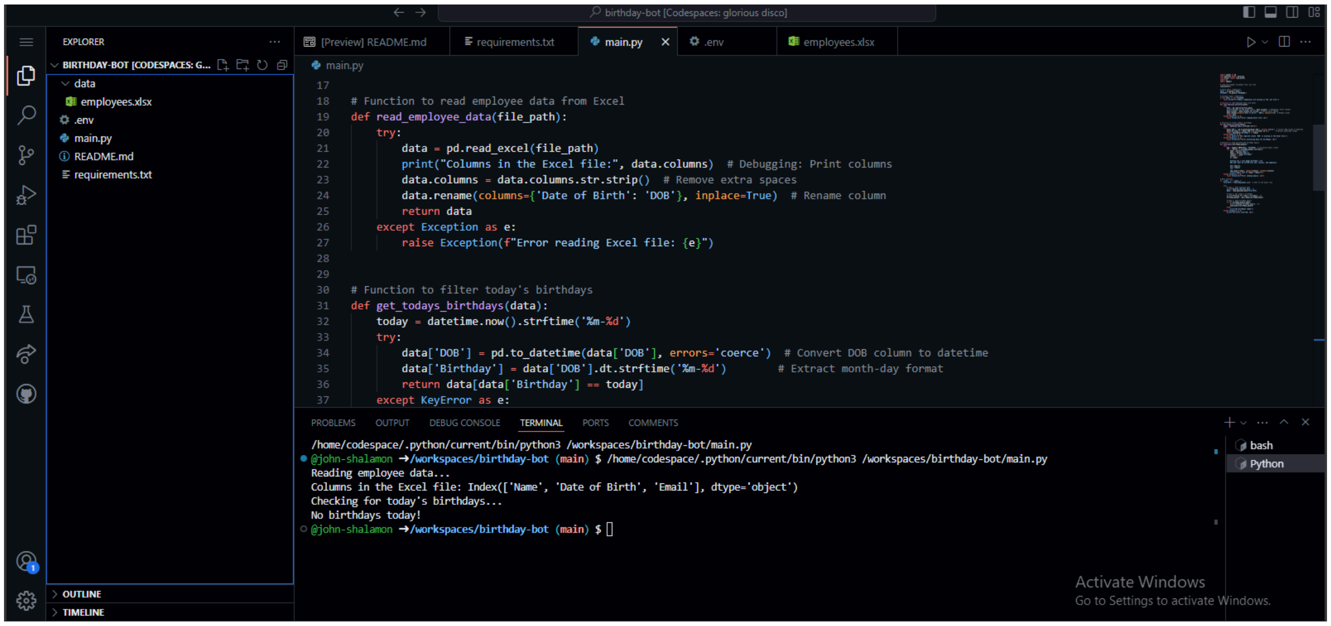
Input data

1.Excel File (BirthdayData.xlsx):



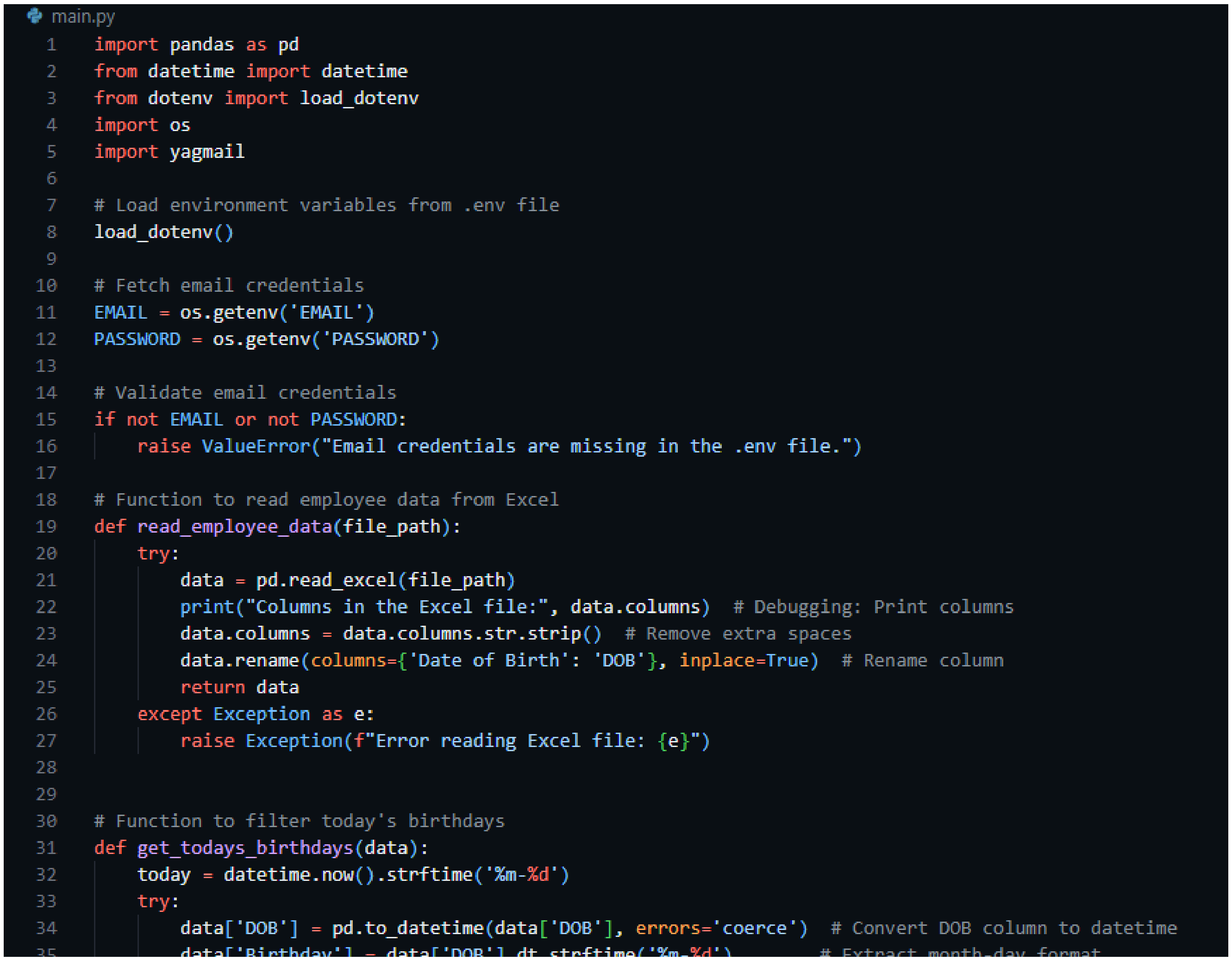
	A	B	C	D	E	F	G	H
69	Oliver Bell	8/18/1985	oliver.bell@example.com					
70	Paula Bennett	8/19/1985	paula.bennett@example.com					
71	Quentin Gray	8/20/1985	quentin.gray@example.com					
72	Rachel Cooper	8/21/1985	rachel.cooper@example.com					
73	Sean Carter	8/22/1985	sean.carter@example.com					
74	Tina Griffin	8/23/1985	tina.griffin@example.com					
75	Ulysses Hughes	8/24/1985	ulysses.hughes@example.com					
76	Valerie Ward	8/25/1985	valerie.ward@example.com					
77	Will Jenkins	8/26/1985	will.jenkins@example.com					
78	Xena Ross	8/27/1985	xena.ross@example.com					
79	Yara Adams	8/28/1985	yara.adams@example.com					
80	Zach Brown	8/29/1985	zach.brown@example.com					
81	Amelia Clark	8/30/1985	amelia.clark@example.com					
82	Brian Mitchell	8/31/1985	brian.mitchell@example.com					
83	Charlotte Turner	9/1/1985	charlotte.turner@example.com					
84	David White	9/2/1985	david.white@example.com					
85	Eva Hall	9/3/1985	eva.hall@example.com					
86	Finn Allen	9/4/1985	finn.allen@example.com					
87	Grace Walker	9/5/1985	grace.walker@example.com					
88	Harper Scott	9/6/1985	harper.scott@example.com					
89	Isaac Morris	9/7/1985	isaac.morris@example.com					
90	Julia King	9/8/1985	julia.king@example.com					
91	Kyle Harris	9/9/1985	kyle.harris@example.com					
92	Lily Nelson	9/10/1985	lily.nelson@example.com					
93								

Input code



```python

# Function to read employee data from Excel
def read_employee_data(file_path):
    try:
        data = pd.read_excel(file_path)
        print("Columns in the Excel file:", data.columns)  # Debugging: Print columns
        data.columns = data.columns.str.strip()  # Remove extra spaces
        data.rename(columns={'Date of Birth': 'DOB'}, inplace=True)  # Rename column
        return data
    except Exception as e:
        raise Exception(f"Error reading Excel file: {e}")


# Function to filter today's birthdays
def get_todays_birthdays(data):
    today = datetime.now().strftime('%m-%d')
    try:
        data['DOB'] = pd.to_datetime(data['DOB'], errors='coerce')  # Convert DOB column to datetime
        data['Birthday'] = data['DOB'].dt.strftime('%m-%d')        # Extract month-day format
        return data[data['Birthday'] == today]
    except KeyError as e:
```

```
/home/codespace/.python/current/bin/python3 /workspaces/birthday-bot/main.py
● @john-shalamon ➜/workspaces/birthday-bot (main) $ /home/codespace/.python/current/bin/python3 /workspaces/birthday-bot/main.py
Reading employee data...
Columns in the Excel file: Index(['Name', 'Date of Birth', 'Email'], dtype='object')
Checking for today's birthdays...
No birthdays today!
○ @john-shalamon ➜/workspaces/birthday-bot (main) $ 
```



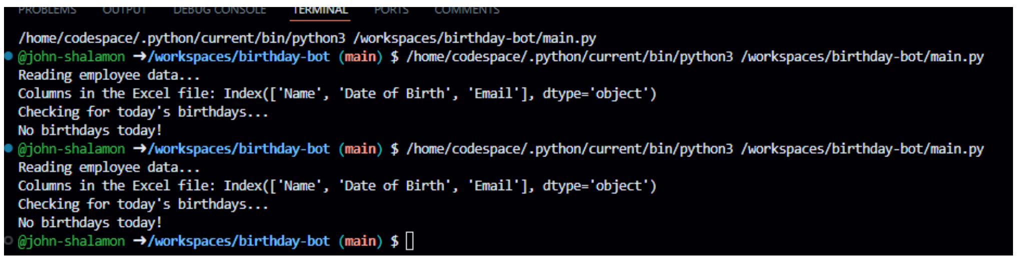
main.py

```python
import pandas as pd
from datetime import datetime
from dotenv import load_dotenv
import os
import yagmail

# Load environment variables from .env file
load_dotenv()

# Fetch email credentials
EMAIL = os.getenv('EMAIL')
PASSWORD = os.getenv('PASSWORD')

# Validate email credentials
if not EMAIL or not PASSWORD:
    raise ValueError("Email credentials are missing in the .env file.")

# Function to read employee data from Excel
def read_employee_data(file_path):
    try:
        data = pd.read_excel(file_path)
        print("Columns in the Excel file:", data.columns)  # Debugging: Print columns
        data.columns = data.columns.str.strip()  # Remove extra spaces
        data.rename(columns={'Date of Birth': 'DOB'}, inplace=True)  # Rename column
        return data
    except Exception as e:
        raise Exception(f"Error reading Excel file: {e}")


# Function to filter today's birthdays
def get_todays_birthdays(data):
    today = datetime.now().strftime('%m-%d')
    try:
        data['DOB'] = pd.to_datetime(data['DOB'], errors='coerce')  # Convert DOB column to datetime
        data['Birthday'] = data['DOB'].dt.strftime('%m-%d')        # Extract month-day format
```

Output data

1.Excel File (StudentReport.xlsx):

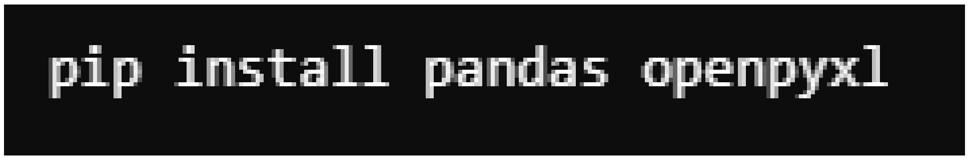
- **Contains student details such as Name, Roll Number, and subject marks.**



```
/home/codespace/.python/current/bin/python3 /workspaces/birthday-bot/main.py
@john-shalamon ➜/workspaces/birthday-bot (main) $ /home/codespace/.python/current/bin/python3 /workspaces/birthday-bot/main.py
Reading employee data...
Columns in the Excel file: Index(['Name', 'Date of Birth', 'Email'], dtype='object')
Checking for today's birthdays...
No birthdays today!
@john-shalamon ➜/workspaces/birthday-bot (main) $ /home/codespace/.python/current/bin/python3 /workspaces/birthday-bot/main.py
Reading employee data...
Columns in the Excel file: Index(['Name', 'Date of Birth', 'Email'], dtype='object')
Checking for today's birthdays...
No birthdays today!
@john-shalamon ➜/workspaces/birthday-bot (main) $
```

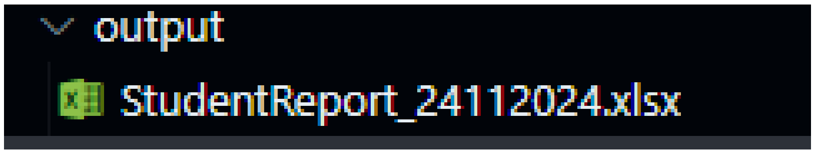
How to Run the Project

1.Install Python and Required Libraries



```
pip install pandas openpyxl
```

2.Place Input File in the Same Folder as Script



```
output
  StudentReport_24112024.xlsx
```

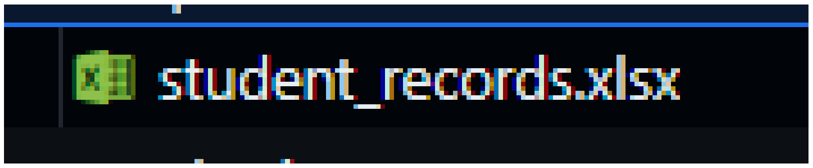
Ensure the Excel file is named StudentData.xlsx.

3.Execute the Script

Run the Python script using a terminal or IDE.
python3 main.py

4.Check the Output Folder

Verify the generated output file. Student Report.xlsx



```
student_records.xlsx
```

Challenges Faced and Solutions

1. **Challenge: Handling missing or incorrect data in the Excel file.**
2. **Solution: Added data validation checks to ensure all required fields are populated.**
3. **Challenge: Formatting the output Excel file consistently.**
4. **Solution: Utilized the openpyxl library for better control over Excel formatting.**

Conclusion and Future Enhancements

Conclusion:

The project successfully automates the generation of student results, reducing manual effort and ensuring accuracy.

Future Enhancements:

1. **Add support for PDF result generation.**
2. **Integrate the system with a web-based UI for ease of access.**
3. **Include an option to email results directly to students.**
4. **Expand functionality to handle semester-wise and cumulative results.**