AUTOMATED E-MAIL SENDER

Application Part

Technical Details

For this project, I developed a Python-based application to automate email reminders for important days of the year using the crontab scheduler. The project is divided into three main components:

1. **Data Collection**:
   - The data for the project includes a JSON file (data.json) containing event names and their respective dates. This file was created by scraping an online source of important days using Python's selenium library.
2. **Email Sending**:
   - The smtplib library was used to handle email sending via the SMTP protocol. The program connects to Gmail's SMTP server to send emails to a specified recipient. The email content is dynamically generated based on the event name and date.
3. **Crontab Scheduling**:
   - The Python script is executed daily at 5:10 PM using a crontab entry:

```
10 17 * * * /usr/bin/python3 /home/ck/Desktop/Projects/369/email_sender.py >>
/home/ck/Desktop/Projects/369/email_log.txt 2>&1
```

   - Crontab ensures that the script runs automatically without manual intervention.

Code Overview

The core functionality is implemented in Python. Below is a simplified outline of the code(Full code is available at the end of the report):

1. **Date Validation**:
   - The script compares today's date with the dates in data.json to identify matching events.
2. **Email Sending**:
   - A function send_email(event_name) formats the email content and sends it via the SMTP server.
3. **Automation**:
   - The datetime library is used to fetch the current date, ensuring the program works dynamically year-round.

Input and Output

- **Input**: A JSON file containing event names and dates.
- **Output**: An email reminder sent to the specified recipient's inbox.

Report Part

Summary of Work

This project automates the process of sending email reminders for significant dates. The primary motivation was to explore the intersection of web scraping, automation, and email handling, all while demonstrating the capabilities of crontab as a scheduling tool in Unix/Linux systems.

Problems Encountered

1. **Gmail Authentication Errors**:
    - Initially, Gmail blocked the script due to security settings. This was resolved by enabling less secure app access.
2. **Duplicate Entries**:
    - While scraping data, duplicate entries were included in the JSON file. This issue was addressed by using a Python set to track unique events.
3. **Manipulating Date:**
    - While scraping data, entries were Turkish while datetime library gave the date in English. This issue solved by coding a simple script to convert months to English.
4. **Scheduling Debugging**:
    - Testing crontab required careful handling of file paths and permissions.

Solutions

- Implemented a solution to convert Turkish month names to their English equivalents using a dictionary, ensuring proper date formatting for crontab scheduling.
- Handled file paths and permissions for crontab scheduling by ensuring proper directory structure and making the script executable with appropriate permissions.
- Added error handling for Gmail authentication to gracefully handle failures in email sending and alert the user with specific error messages.
- Regularly tested the entire process end-to-end to ensure no edge cases were missed, including verifying the email sending functionality and ensuring correct execution of scheduled tasks.

Personal Interest

My interest in this topic stems from a fascination with automation and its potential to simplify daily tasks. Scheduling and reminders are essential in various domains, and this project provided an opportunity to explore how they can be implemented effectively.

Creativity Part

Proposed Assignment

Title: "Automating Social Media Post Scheduling with Crontab and Python"

Assignment Description: Develop an application that automates posting to social media platforms at scheduled times. Students should:

1. Use crontab to schedule posts.
2. Use Python libraries like selenium to scrap data.
3. Use Python library smtplib to automate e-mail sending process.
4. Log all successful and failed posts for debugging purposes.

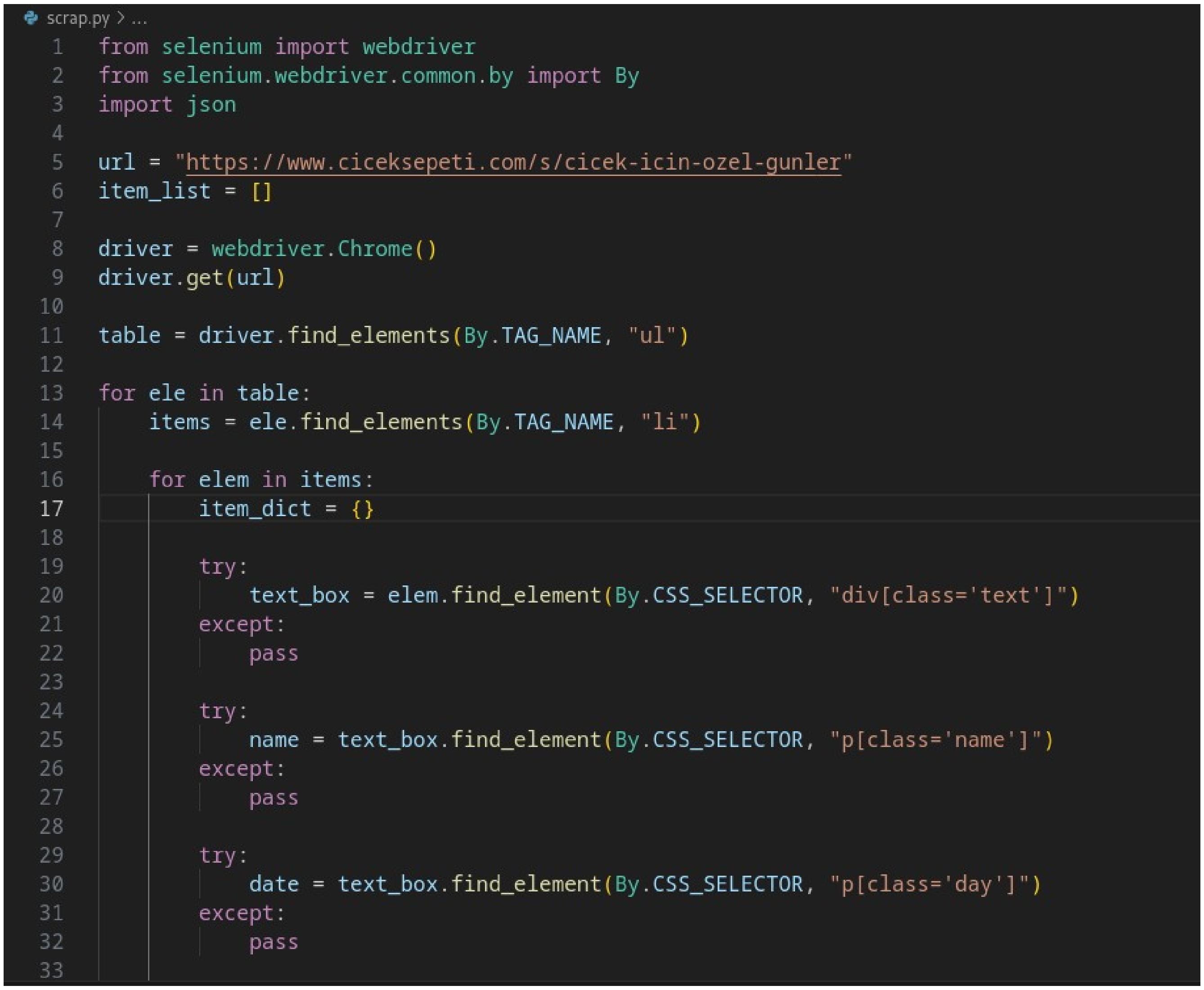
Reason for Selecting This Topic: Social media automation is a highly relevant problem in today's digital age. This assignment introduces students to:

- Web scrapping with selenium.
- E-mail automation with smtplib.
- Scheduling and logging mechanisms using crontab featues.
- Real-world applications of Python in marketing and content management.

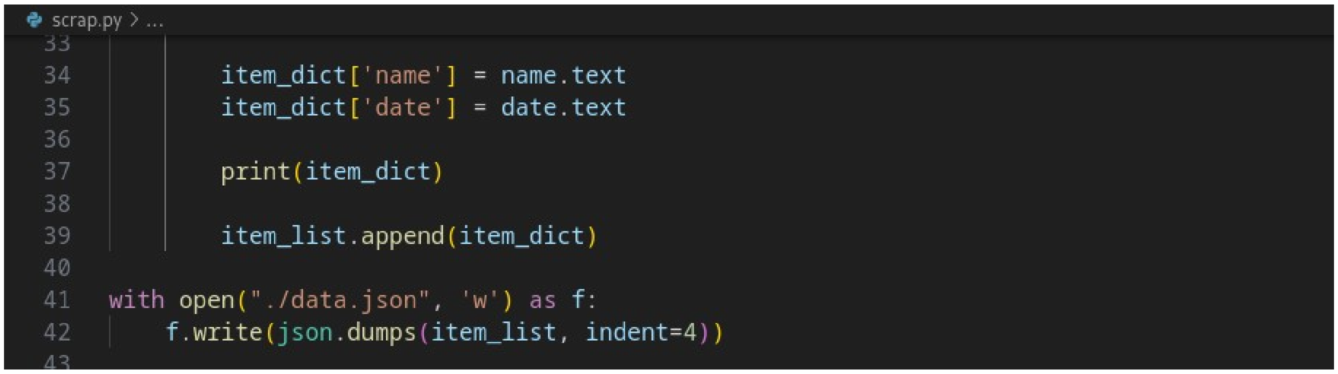
By tackling this project, students gain hands-on experience in automation while developing a practical tool applicable to real-life scenarios.

Full Code of the Program

scrap.py:

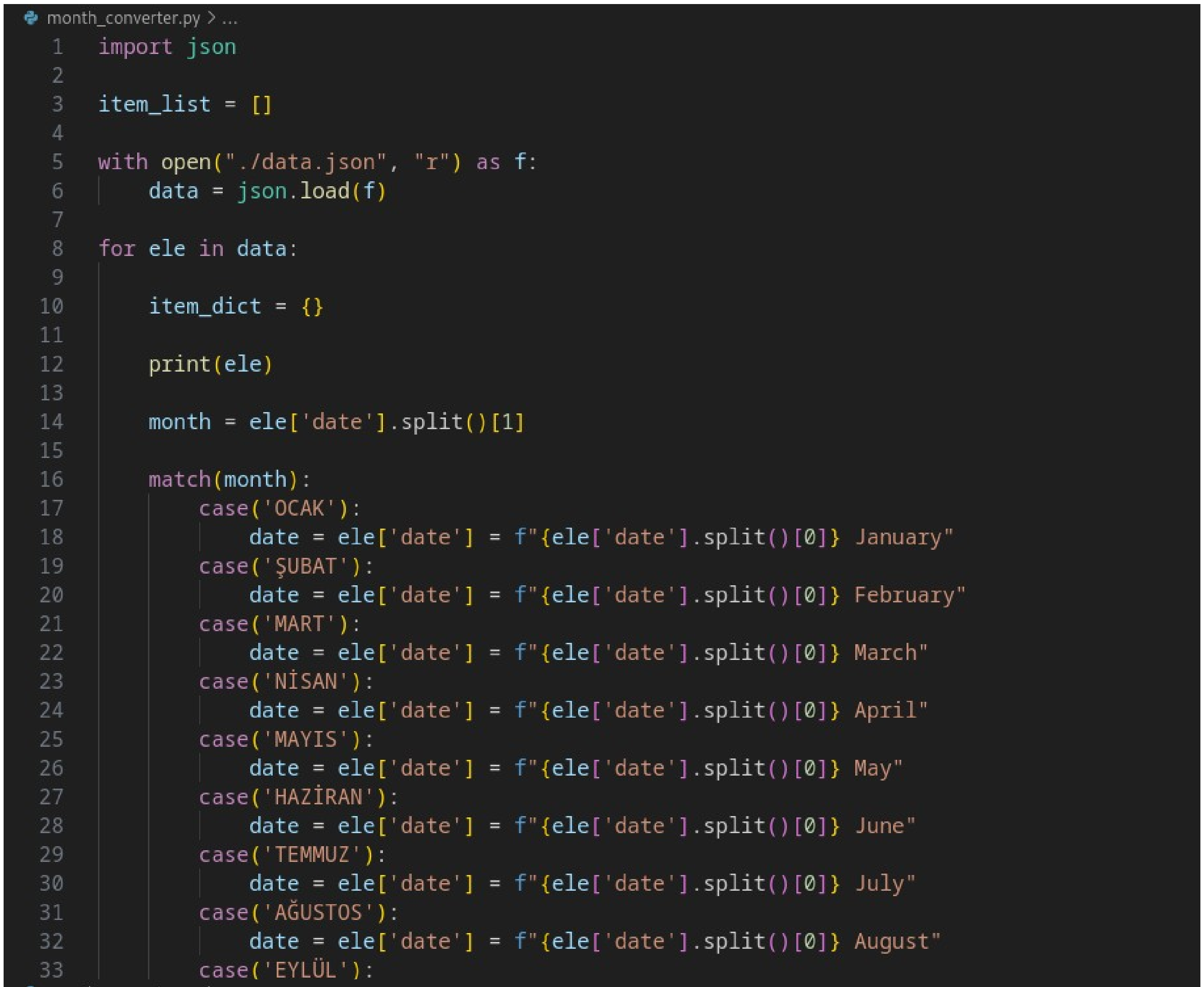

scrap.py > ...

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
import json

url = "https://www.ciceksepeti.com/s/cicek-icin-ozel-gunler"
item_list = []

driver = webdriver.Chrome()
driver.get(url)

table = driver.find_elements(By.TAG_NAME, "ul")

for ele in table:
    items = ele.find_elements(By.TAG_NAME, "li")

    for elem in items:
        item_dict = {}

        try:
            text_box = elem.find_element(By.CSS_SELECTOR, "div[class='text']")
        except:
            pass

        try:
            name = text_box.find_element(By.CSS_SELECTOR, "p[class='name']")
        except:
            pass

        try:
            date = text_box.find_element(By.CSS_SELECTOR, "p[class='day']")
        except:
            pass

```

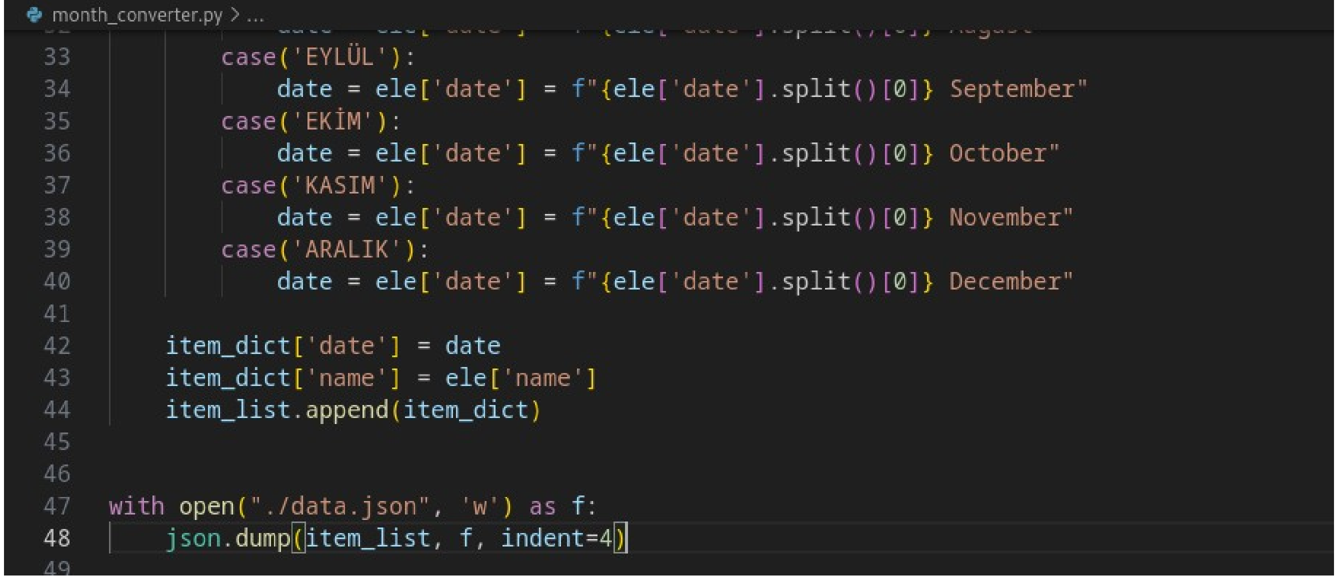

scrap.py > ...

```python

        item_dict['name'] = name.text
        item_dict['date'] = date.text

        print(item_dict)

        item_list.append(item_dict)

with open("./data.json", 'w') as f:
    f.write(json.dumps(item_list, indent=4))
```

month_converter.py:

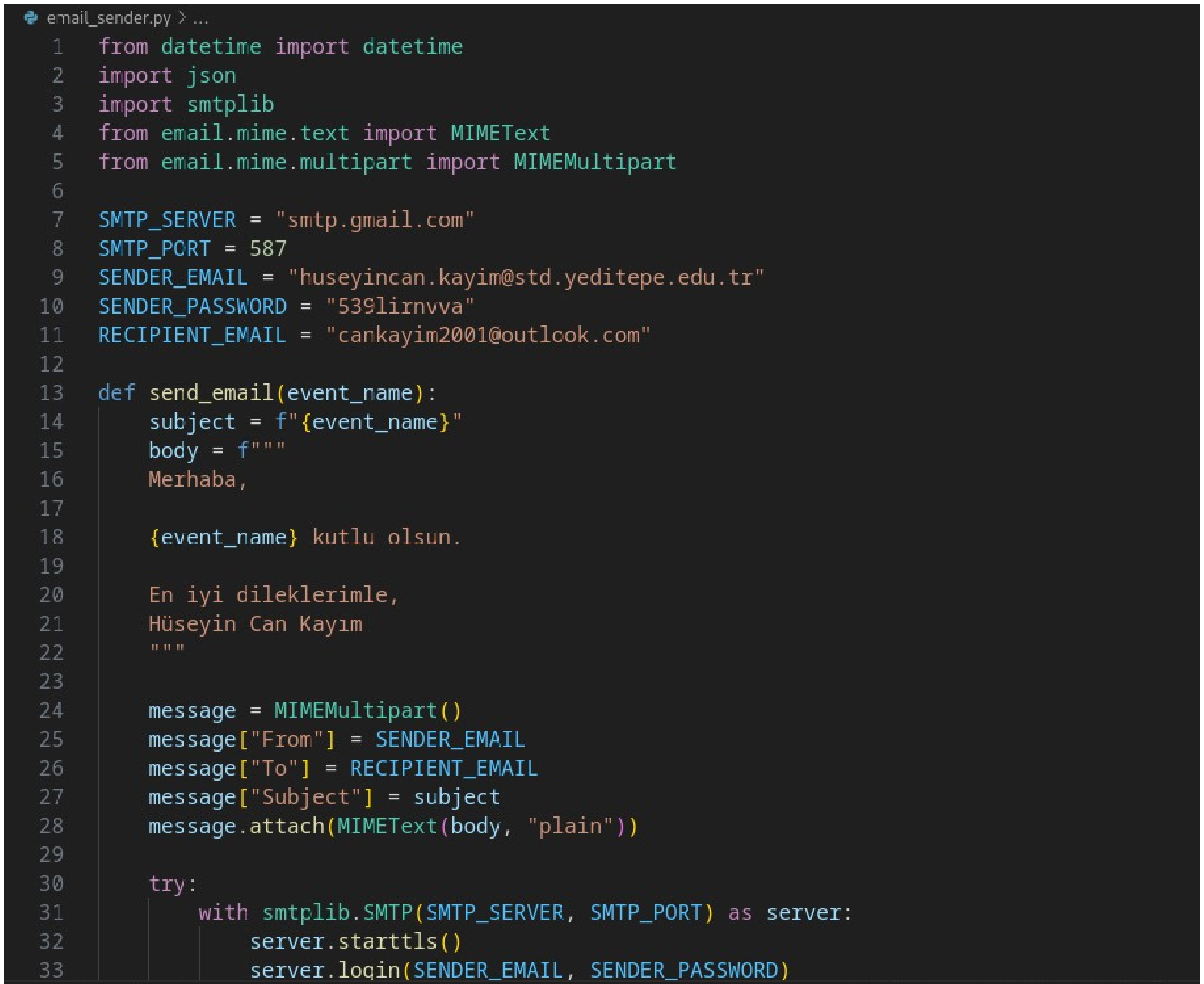


```python
import json

item_list = []

with open("./data.json", "r") as f:
    data = json.load(f)

for ele in data:

    item_dict = {}

    print(ele)

    month = ele['date'].split()[1]

    match(month):
        case('OCAK'):
            date = ele['date'] = f"{ele['date'].split()[0]} January"
        case('ŞUBAT'):
            date = ele['date'] = f"{ele['date'].split()[0]} February"
        case('MART'):
            date = ele['date'] = f"{ele['date'].split()[0]} March"
        case('NİSAN'):
            date = ele['date'] = f"{ele['date'].split()[0]} April"
        case('MAYIS'):
            date = ele['date'] = f"{ele['date'].split()[0]} May"
        case('HAZİRAN'):
            date = ele['date'] = f"{ele['date'].split()[0]} June"
        case('TEMMUZ'):
            date = ele['date'] = f"{ele['date'].split()[0]} July"
        case('AĞUSTOS'):
            date = ele['date'] = f"{ele['date'].split()[0]} August"
        case('EYLÜL'):
```



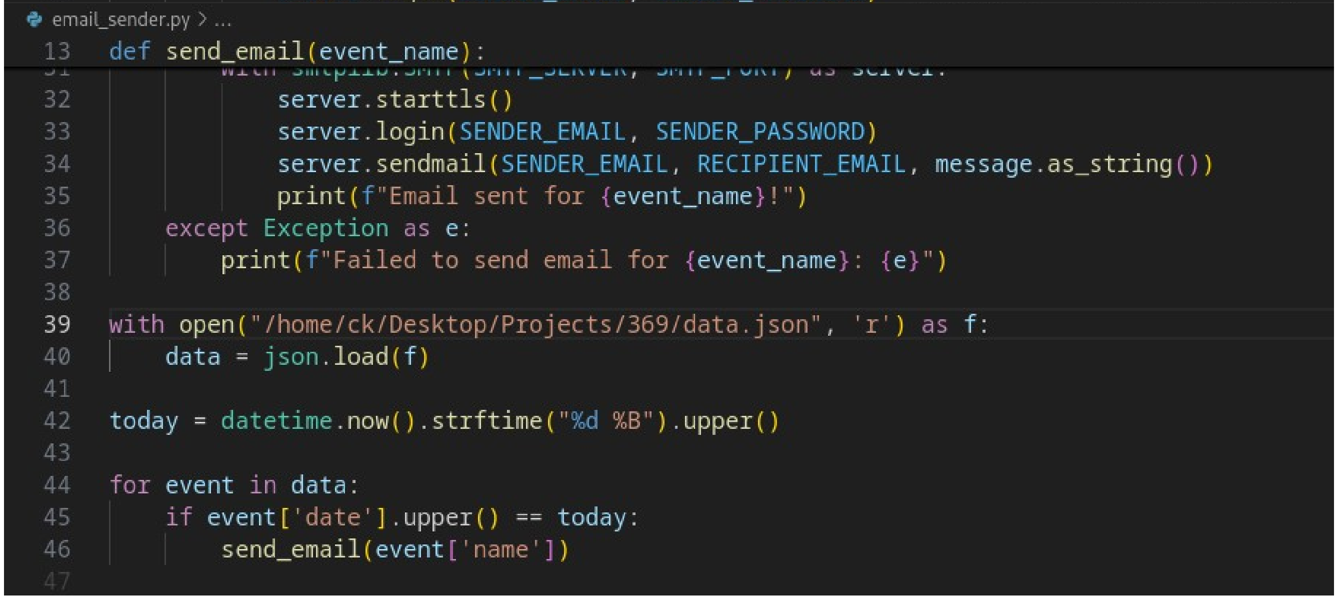
```python
        case('EYLÜL'):
            date = ele['date'] = f"{ele['date'].split()[0]} September"
        case('EKİM'):
            date = ele['date'] = f"{ele['date'].split()[0]} October"
        case('KASIM'):
            date = ele['date'] = f"{ele['date'].split()[0]} November"
        case('ARALIK'):
            date = ele['date'] = f"{ele['date'].split()[0]} December"

    item_dict['date'] = date
    item_dict['name'] = ele['name']
    item_list.append(item_dict)


with open("./data.json", 'w') as f:
    json.dump(item_list, f, indent=4)
```

email_sender.py:



email_sender.py > ...

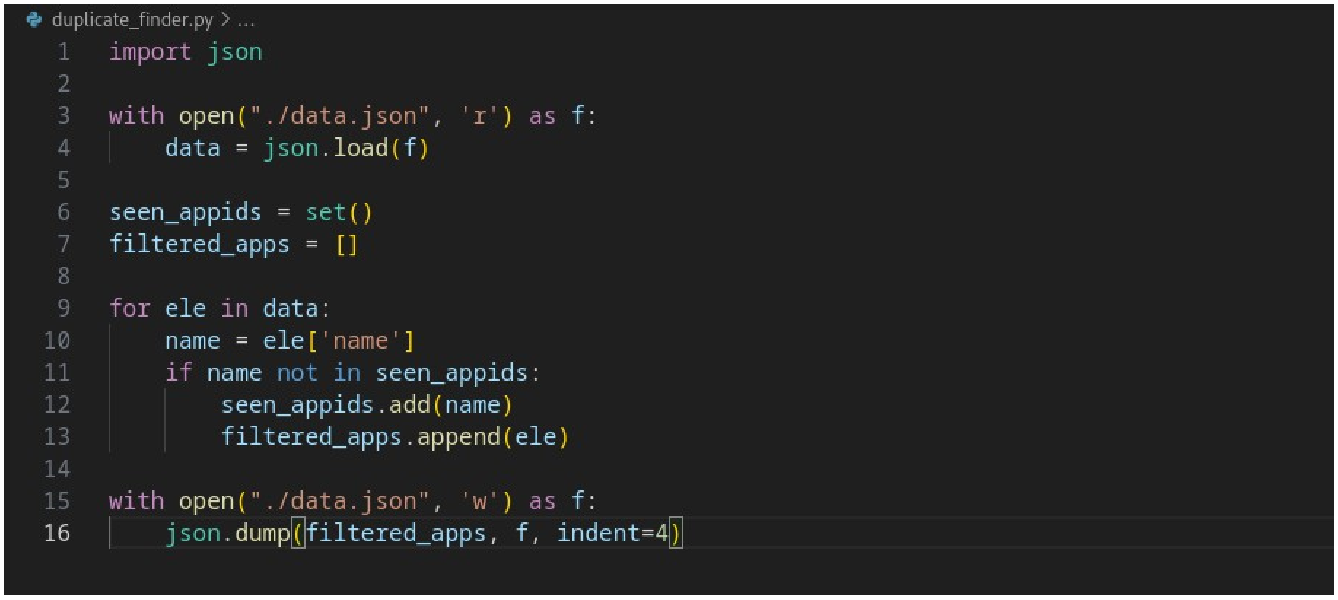
```python
from datetime import datetime
import json
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
SENDER_EMAIL = "huseyincan.kayim@std.yeditepe.edu.tr"
SENDER_PASSWORD = "539lirnvva"
RECIPIENT_EMAIL = "cankayim2001@outlook.com"

def send_email(event_name):
    subject = f"{event_name}"
    body = f"""
    Merhaba,

    {event_name} kutlu olsun.

    En iyi dileklerimle,
    Hüseyin Can Kayım
    """

    message = MIMEMultipart()
    message["From"] = SENDER_EMAIL
    message["To"] = RECIPIENT_EMAIL
    message["Subject"] = subject
    message.attach(MIMEText(body, "plain"))

    try:
        with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
            server.starttls()
            server.login(SENDER_EMAIL, SENDER_PASSWORD)
```



email_sender.py > ...

```python
def send_email(event_name):
            server.starttls()
            server.login(SENDER_EMAIL, SENDER_PASSWORD)
            server.sendmail(SENDER_EMAIL, RECIPIENT_EMAIL, message.as_string())
            print(f"Email sent for {event_name}!")
    except Exception as e:
        print(f"Failed to send email for {event_name}: {e}")

with open("/home/ck/Desktop/Projects/369/data.json", 'r') as f:
    data = json.load(f)

today = datetime.now().strftime("%d %B").upper()

for event in data:
    if event['date'].upper() == today:
        send_email(event['name'])

```

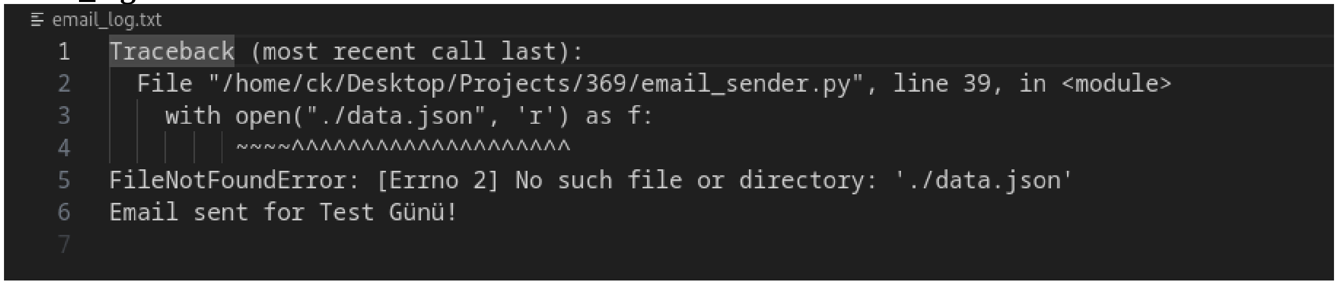
duplicate_finder.py:



duplicate_finder.py > ...

```python
import json

with open("./data.json", 'r') as f:
    data = json.load(f)

seen_appids = set()
filtered_apps = []

for ele in data:
    name = ele['name']
    if name not in seen_appids:
        seen_appids.add(name)
        filtered_apps.append(ele)

with open("./data.json", 'w') as f:
    json.dump(filtered_apps, f, indent=4)
```

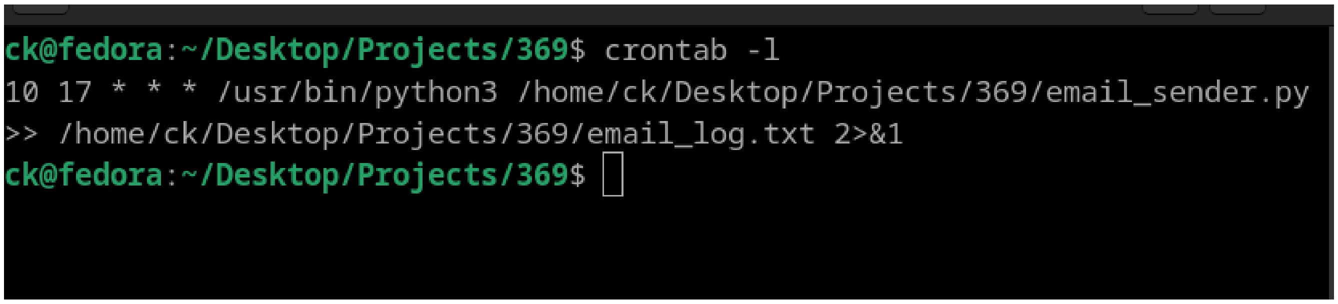
email_log.txt:



email_log.txt

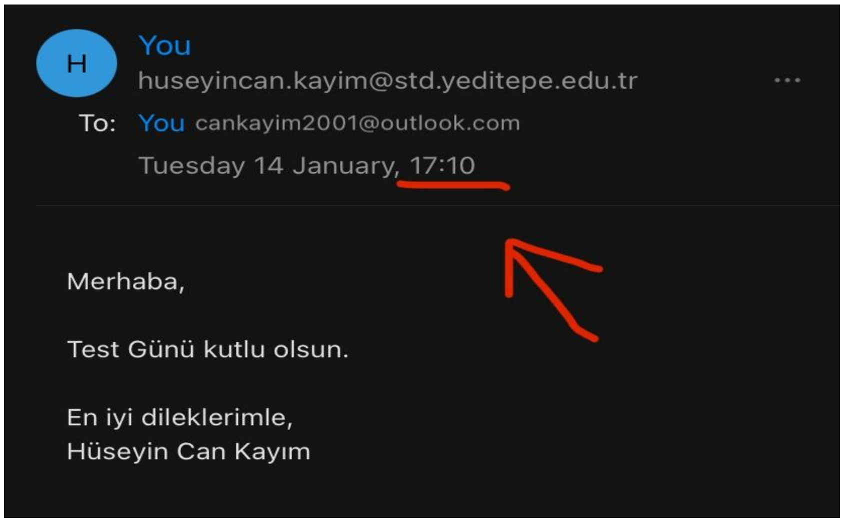
```
Traceback (most recent call last):
  File "/home/ck/Desktop/Projects/369/email_sender.py", line 39, in <module>
    with open("./data.json", 'r') as f:
         ~~~~^^^^^^^^^^^^^^^^^^^^^
FileNotFoundError: [Errno 2] No such file or directory: './data.json'
Email sent for Test Günü!
```

crontab part:



```
ck@fedora:~/Desktop/Projects/369$ crontab -l
10 17 * * * /usr/bin/python3 /home/ck/Desktop/Projects/369/email_sender.py
>> /home/ck/Desktop/Projects/369/email_log.txt 2>&1
ck@fedora:~/Desktop/Projects/369$ 
```

email proof:



H
You
huseyincan.kayim@std.yeditepe.edu.tr
To: You cankayim2001@outlook.com
Tuesday 14 January, 17:10

Merhaba,

Test Günü kutlu olsun.

En iyi dileklerimle,
Hüseyin Can Kayım