이메일 자동화 보내기

이메일 설정

네이버 이메일 설정

- 네이버 메일 > 환경설정 > POP3/IAMP 설정 > IMAP/SMTP 설정 > 사용함 > 저장
- `SMTP`는 모든 설정에서 이메일을 보내는 데 사용되며, `POP3`와 `IMAP`은 이메일을 받는 방식을 정의

SMTP

- **SMTP**(Simple Mail Transfer Protocol)는 인터넷에서 이메일을 전송하기 위해 널리 사용되는 프로토콜
- **SMTP**는 "단순 메일 전송 프로토콜"이라는 이름에서 알 수 있듯이, 주로 메일을 보내는 데 사용
- **포트 번호**: SMTP는 주로 25번 포트를 사용, 보안을 위한 연결에서는 587번 포트를 사용하여 STARTTLS를 통해 암호화된 연결을 시작
  - **STARTTLS**: 메일 전송 세션 중에 TLS(전송 계층 보안)를 활성화하여 메일 데이터를 암호화

smtplib

- `smtplib`은 파이썬에서 이메일을 프로그래밍적으로 보낼 수 있도록 하는 표준 라이브러리
- 이 라이브러리를 사용하여 SMTP 프로토콜을 통해 이메일 서버에 연결하고, 메일을 보내거나 관련 작업 수행 가능
- `smtplib`은 간단한 이메일 전송부터 복잡한 MIME 기반 메시지와 보안 연결까지 다양한 이메일 관련 기능을 제공
- 파이썬에서 기본으로 가지고 있는 라이브러리로 별도 설치 필요 없음

1. **모듈 임포트**

```
import smtplib
```

2. **서버 연결**
   - SMTP 서버에 연결 시도
   - 이 때, 서버의 주소와 포트 번호가 필요
   - 일반적으로 포트 587을 사용하여 시작한 후 TLS를 사용해 암호화

```
server = smtplib.SMTP('smtp.example.com', 587) server.starttls() # 보안을 위해 TLS 사용
```

3. **로그인**
   - SMTP 서버에 로그인

```
server.login('your_email@example.com', 'your_password')
```

4. 메일 보내기 > 에러 발생 MIME 사용 필요
    - 메일을 보내는 기능을 실행
    - 발신자, 수신자, 메일 내용을 파라미터로 전달

```
server.sendmail('from@example.com', 'to@example.com', 'This is a test emai
l.')
```

5. **연결 종료**
    - 메일 전송 후 SMTP 서버와의 연결을 종료

```
server.quit()
```

MIME

- MIME(Multipurpose Internet Mail Extensions)은 인터넷에서 텍스트 이외의 데이터(예: 이미지, 비디오, 오디오, 애플리케이션 데이터 등)를 전송하기 위해 이메일 메시지에 포함시킬 수 있도록 확장한 표준
- 원래 이메일은 ASCII 텍스트 전송만을 지원했기 때문에 다양한 형태의 콘텐츠를 다루기 위해 MIME가 도입

MIME의 주요 기능

- MIME은 이메일에서 다음과 같은 기능을 지원

- **비텍스트 첨부 파일 전송**: 이미지, 동영상, 실행 파일 등 다양한 바이너리 데이터를 이메일에 첨부
- **다국어 텍스트 전송**: 다양한 문자 인코딩을 지원하여, 전 세계의 언어를 포함 가능
- **메시지 본문의 여러 부분 포맷팅**: 이메일이 텍스트와 HTML 등 다양한 포맷의 본문을 동시에 포함 가능

MIME in Python

- 파이썬의 `email.mime` 모듈은 MIME 표준을 사용하여 이메일 메시지를 생성하고 조작할 수 있는 클래스를 제공
- 주요 클래스 중 하나는 `MIMEText`

MIMEText 클래스 소개

소개

- `MIMEText` 클래스는 이메일 메시지를 MIME 형식으로 구성할 수 있게 해주는 기능을 제공
- 이 클래스를 통해 생성된 인스턴스는 이메일의 본문과 관련 메타데이터(헤더)를 적절히 관리하고 조작 가능
- 또한, 일반 텍스트나 HTML과 같은 형식의 텍스트 처리 가능

MIMEText 클래스의 파라미터

- text
  - **용도**: 이메일 본문의 내용을 문자열로 전달
  - **설명**: 이 값은 이메일에 실제로 나타날 메시지
- subtype
  - **용도**: 본문의 내용 타입을 지정
  - **기본값**: 기본적으로 'plain'으로 설정되어 일반 텍스트를 의미
  - **옵션**: 'html'과 같이 다른 값으로 설정하여 HTML 형식의 이메일 생성 가능
- charset
  - **용도**: 이메일의 문자 인코딩을 지정
  - **권장 설정**: `utf-8`은 국제적으로 사용되는 문자 인코딩 방식으로, 다양한 언어의 문자를 포함 가능, 이 인코딩을 사용함으로써 이메일 메시지가 다양한 시스템과 호환되며 문자 깨짐 현상을 방지

MIMEText 객체의 사용 예시

- `MIMEText` 클래스는 `email.mime.text` 모듈에 포함
- `MIMEText`의 생성자는 최소 한 개의 파라미터를 받음

```
from email.mime.text import MIMEText # 텍스트 이메일 생성 msg = MIMEText("T
his is a test email.", 'plain', 'utf-8') # HTML 이메일 생성 html_msg = MIME
Text("<html><body><h1>Hello, world!</h1></body></html>", 'html', 'utf-8')
```

MIMEText 객체의 주요 기능

- **이메일 본문 설정**
  - 생성자에서 전달된 `text`는 이메일의 본문으로 설정
- **헤더 관리**
  - `Subject`: 이메일의 제목을 설정. 이메일 목록에서 사용자가 가장 먼저 보게 되는 정보이며, 이메일의 내용을 요약
  - `From`: 이메일을 보내는 사람의 주소. 이 주소는 이메일을 받는 사람에게 발신자를 알려주는 역할을 하며, 회신 시 사용되는 주소
  - `To`: 이메일의 수신자 주소. 이 주소는 메시지가 전송될 목적지
  - `Cc` (Carbon Copy): 이메일의 참조 복사본을 보내고자 하는 다른 수신자들의 주소. 메일의 내용을 주 수신자 외의 다른 사람들과 공유할 때 사용

MIMEText 객체의 as_string() 메서드

- **목적**
  - `MIMEText` 객체를 완전한 문자열 형태의 이메일 메시지로 변환
- **기능**
  - 이메일의 헤더(`Subject`, `From`, `To` 등)와 본문을 포함한 전체 MIME 메시지를 하나의 문자열로 구성

일반 테스트 메일 전송

이메일 전송 자동화 스크립트

1. 필요한 모듈 임포트

```
import smtplib from email.mime.text import MIMEText
```

- `smtplib` : SMTP 프로토콜을 사용하여 이메일을 전송하기 위한 파이썬 모듈
- `email.mime.text.MIMEText` : 텍스트 기반 이메일을 생성하기 위한 클래스

2. 발신자, 수신자 및 서버 정보 설정

```
send_email = "본인 네이버 이메일 주소" send_pwd = "비밀번호" recv_email = "받는 사람 이메일 주소" smtp_name = "smtp.naver.com" smtp_port = 587
```

- **발신자 이메일 및 비밀번호**: 이메일을 보내는 계정의 주소와 비밀번호
- **수신자 이메일**: 이메일을 받을 계정의 주소
- **SMTP 서버 이름과 포트**: 이메일을 전송하는 데 사용되는 서버의 주소와 포트 번호. 포트 587은 TLS 보안 연결을 위한 표준 포트.

3. 이메일 본문 작성

```
text = """ 메일 내용 입력 텍스트1 텍스트2 """
```

- 여기서는 멀티라인 문자열을 사용하여 이메일의 본문 작성

4. MIMEText 객체 생성 및 헤더 설정

```
msg = MIMEText(text, 'plain', 'utf-8') msg['Subject'] = "메일 제목 입력" msg['From'] = send_email msg['To'] = recv_email
```

- **MIMEText 객체**: 이메일의 본문과 메타데이터(헤더) 설정

- **헤더 설정**: `Subject`, `From`, `To`는 각각 이메일의 제목, 발신자 주소, 수신자 주소 설정

5. 이메일 메시지를 문자열로 변환

```
email_string = msg.as_string() print(email_string)
```

- `as_string()` **메서드**: MIMEText 객체를 완전한 형태의 이메일 문자열로 변환. 이 문자열은 서버를 통해 전송.

6. SMTP 서버를 통한 이메일 전송

```
pythonCopy code s = smtplib.SMTP(smtp_name, smtp_port) s.starttls() s.logi
n(send_email, send_pwd) s.sendmail(send_email, recv_email, email_string)
s.quit()
```

- **서버 연결**: `smtplib.SMTP`를 사용하여 지정된 서버와 포트로 연결.
- **TLS 보안 시작**: `starttls()` 메서드를 호출하여 보안 연결 활성화.
- **서버 로그인**: `login()` 메서드를 사용하여 발신자 이메일 계정으로 로그인.
- **이메일 전송**: `sendmail()` 메서드를 사용하여 이메일을 수신자에게 전송.
- **연결 종료**: `quit()` 메서드로 SMTP 서버 연결을 종료.

7. 최종 완성 코드

```
import smtplib from email.mime.text import MIMEText from dotenv import loa
d_dotenv import os load_dotenv() send_email = os.getenv("SECRET_ID") send_
pwd = os.getenv("SECRET_PASS") recv_email = "@naver.com" smtp_name = "smt
p.naver.com" smtp_port = 587 text = "이메일 보내기 테스트입니다" msg = MIMET
ext(text, 'plain', 'utf-8') msg['Subject'] = "메일 제목 입력" msg['From'] =
send_email msg['To'] = recv_email email_string = msg.as_string() print(ema
il_string) s = smtplib.SMTP(smtp_name, smtp_port) s.starttls() s.login(sen
d_email, send_pwd) s.sendmail(send_email, recv_email, email_string) s.quit
()
```

```
#uploads 폴더의 디렉터리 정보를 수집 #파일 확장자가 txt 인 것을 리스트화 #txt 파일을 열어서 출력 import os import re import smtplib from email.mime.text import MIMEText from dotenv import load_dotenv def mail_sender(file_path, line): load_dotenv() send_email = os.getenv("SECRET_ID") send_pwd = os.getenv("SECRET_PASS") recv_email = "boanproject1234@naver.com" smtp_name = "smtp.naver.com" smtp_port = 587 text = f"중요 정보 탐지 {file_path} : 라인 {line}" msg = MIMEText(text, 'plain', 'utf-8') msg['Subject'] = f"{file_path} 내 중요 정보 탐지" msg['From'] = send_email msg['To'] = recv_email email_string = msg.as_string() print(email_string) s = smtplib.SMTP(smtp_name, smtp_port) s.starttls() s.login(send_email, send_pwd) s.sendmail(send_email, recv_email, email_string) s.quit() dir_path = "uploads" all_files = os.listdir(dir_path) txt_files = [] for file in all_files: if file.endswith(".txt"): txt_files.append(file) # 1.txt, 2.txt /uploads/1.txt for filename in txt_files: file_path = os.path.join(dir_path, filename) with open(file_path, 'r', encoding='utf-8') as file: lines = file.readlines() for index, line in enumerate(lines): if line.startswith("#") or line.startswith("//"): print(f"주석 {file_path} {index+1}라인: 탐지: {line.strip()}") mail_sender(file_path, line) if re.search(r'\d{6}\s*[-]\s*\d{7}', line): print(f"주민번호 {file_path} {index+1}라인: 탐지: {line.strip()}") mail_sender(file_path, line)
```

첨부파일을 사용할 때 양식

```
import smtplib from email.mime.multipart import MIMEMultipart from
email.mime.text import MIMEText from email.mime.application import
MIMEApplication from dotenv import load_dotenv import os load_dotenv()
send_email = os.getenv("SECRET_ID") send_pwd = os.getenv("SECRET_PASS")
recv_email = "yours@naver.com" smtp = smtplib.SMTP('smtp.naver.com', 587)
smtp.ehlo() smtp.starttls() smtp.login(send_email,send_pwd) text = f"탐지라
인:" msg = MIMEMultipart() msg['Subject'] = f"모니터 탐지:" msg['From'] =
send_email msg['To'] = recv_email text ="<b>탐지되었습니다.</b>"
contentPart = MIMEText(text) msg.attach(contentPart) etc_file_path =
r'uploads/1.txt' with open(etc_file_path, 'rb') as f : etc_part =
MIMEApplication( f.read() ) etc_part.add_header('Content-
Disposition','attachment', filename=etc_file_path) msg.attach(etc_part)
smtp.sendmail(send_email,recv_email,msg.as_string() ) smtp.quit()
```

2시 30분까지 진행해보세요!! (중간 쉬는 시간 포함)

```
import os from datetime import datetime from deep_translator import
GoogleTranslator import smtplib from email.mime.multipart import
MIMEMultipart from email.mime.text import MIMEText from
email.mime.application import MIMEApplication from dotenv import
load_dotenv def mail_sender(report_file): load_dotenv() send_email =
os.getenv("SECRET_ID") send_pwd = os.getenv("SECRET_PASS") recv_email =
"boanproject1234@naver.com" smtp = smtplib.SMTP('smtp.naver.com', 587)
smtp.ehlo() smtp.starttls() smtp.login(send_email,send_pwd) text = f"
{report_file} 파일 번역 결과" msg = MIMEMultipart() msg['Subject'] = f"
{report_file} 파일 번역 결과" msg['From'] = send_email msg['To'] =
recv_email contentPart = MIMEText(text, 'html', 'utf-8')
msg.attach(contentPart) etc_file_path = report_file with
open(etc_file_path, 'rb') as f : etc_part = MIMEApplication( f.read() )
etc_part.add_header('Content-Disposition','attachment',
filename=etc_file_path) msg.attach(etc_part)
smtp.sendmail(send_email,recv_email,msg.as_string() ) smtp.quit() # 현재 시
간 표시 now = datetime.now() day = now.strftime("%Y-%m-%d") hour =
now.strftime("%H:%M:%S") file_name = "test.txt" with open(file_name, "r",
encoding="UTF-8") as file: lines = file.read() translated =
GoogleTranslator(source="ko", target="en").translate(lines) # 번역 보고서
파일 생성 report_file = f"{file_name}_번역결과.txt" with open(report_file,
"a", encoding="UTF-8") as file: file.write(f"시간: {hour} 파일 내용
{file_name}\n") file.write(f"원문:\n{lines}\n") file.write(f"번역
문:\n{translated}\n") #번역된 것 추가 print(f"번역이 완료되었습니다. 결과는
{report_file}에 저장되었습니다.") mail_sender(report_file)
```