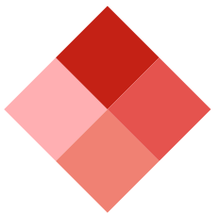


파이썬을 활용한 업무자동화

5회차: 업무자동화로 일 편하게 하기

목차

5회차: 이메일 다루기

- 복습
- 라이브러리 소개 및 설치
- 이메일은 어떻게 보내질까?
- 이메일 보내기
  - 클래스 시간에 배운 email.py 활용
  - Gmail로 구현해보기
  - 내 이메일로 테스트해보기
- 엑셀과 함께 활용
- Appendix
  - 메일 주소 유효성 검사
  - 파일 첨부해서 보내기

복습

지난 강의 때 뭐했지..?

엑셀 다루기

복습

지난 강의 때 뭐했지..?

엑셀 다루기: 읽기

```
from openpyxl import load_workbook
wb = load_workbook('simple_data.xlsx')
data = wb['sheet1']
print data['A1'].value
print data['A2'].value
print data['B1'].value
print data['B2'].value
```

복습

지난 강의 때 뭐했지..?

엑셀 다루기: 읽기

```
from openpyxl import load_workbook
wb = load_workbook('simple_data.xlsx')
data = wb['sheet1']

monster = data[1]
for mon in monster:
    print mon.value

print "-"*20

monster = data['A']
for mon in monster:
    print mon.value
```

복습

지난 강의 때 뭐

엑셀 다루

```
from openpyxl import load_workbook
wb = load_workbook('simple_data.xlsx')
data = wb['sheet1']

monster = data['A1:B2']
print monster
for row in monster:
    for cell in row:
        print cell.value

monster = data['A:B']
print monster
for row in monster:
    for cell in row:
        print cell.value

monster = data[1:2]
print monster
for row in monster:
    for cell in row:
        print cell.value
```

복습

지난 강의 때 뭐했지..?

엑셀 다루기: 쓰기

```
from openpyxl import Workbook

wb = Workbook()
ws = wb.create_sheet(title='sheet_test')

ws['A1'] = 'alghost'
ws['B1'] = 'wow!!!!'

wb.save('simple_result.xlsx')
```

복습

지난 강의 때 뭐했지..?

엑셀 다루기: 쓰기

```
from openpyxl import Workbook

wb = Workbook()
ws = wb.create_sheet(title='sheet_test')

ws.append(['Number', 'Name'])

for i in range(20):
    ws.append([i, str(i)+' data'])

wb.save('simple_result.xlsx')
```

라이브러리 소개

smtplib, email

이메일을 다루는 라이브러리

- 우리가 사용할 라이브러리는 email, smtplib
- 레퍼런스: https://docs.python.org/2/library/email.html
  - 안타깝게 영어다..

라이브러리 설치

!!!!!!!

필요없습니다!

- 필요없습니다!!!!
- 파이썬 기본 라이브러리에서 제공!!

이메일은 어떻게 보내질까?

이메일 동작 원리

메일 프로그램 사용을 해봤다면!

- 윈도우의 Outlook ?
- 리눅스에서 많이 쓰이는 Thunderbird ?
- 맥의 Mail?

필요한 설정 정보

- SMTP 서버 주소
- POP 서버 주소
- SMTP 서버 포트
- POP 서버 포트
- 계정 정보

이메일은 어떻게 보내질까?

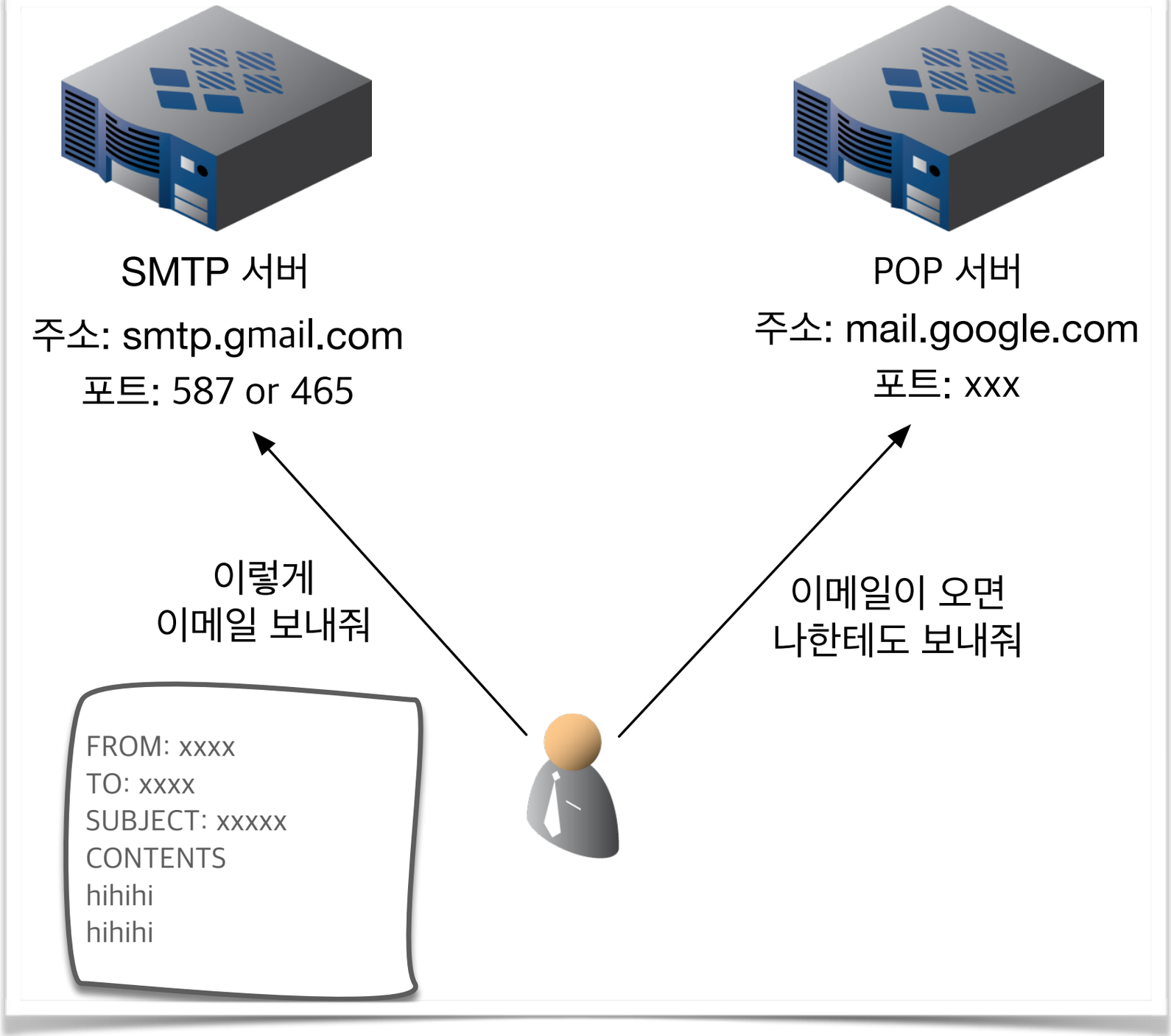
이메일 동작 원리

필요한 설정 정보

- SMTP 서버 주소: 메일 보내는 서버 주소
- POP 서버 주소: 메일 받는 서버 주소
- SMTP 서버 포트: SMTP 주소로 갈 때 쓸 길 번호
- POP 서버 포트: POP 주소로 갈 때 쓸 길 번호
- 계정 정보: 아이디, 비밀번호

이메일은 어떻게 보내질까?

이메일 동작 원리



이메일은 어떻게 보내질까?

이메일 동작 원리

우리에게 필요한 설정 정보

- SMTP 서버 주소: 메일 보내는 서버 주소
- ~~POP 서버 주소: 메일 받는 서버 주소~~
- SMTP 서버 포트: SMTP 주소로 갈 때 쓸 길 번호
- ~~POP 서버 포트: POP 주소로 갈 때 쓸 길 번호~~
- 계정 정보: 아이디, 비밀번호

이메일 보내기

클래스 수업때 사용한 Email() 활용!

클래스 함수로 구현

- 기존에 print로 사기치고 있던(..) 클래스를 동작하는 코드로!

```
class Email():
    from_email = ''
    to_email   = ''
    subject    = ''
    contents   = ''

    def send_mail(self):
        print "Send to " + to_email
```

이메일 보내기

Gmail 로 구현해보기

Gmail로 구현해보기

- 앞서 설명한 원리를 활용하기 위해 코드는 어떻게 작성되는지 보자

O

Gm

G

- 앞

자

```
# -*- coding:utf-8 -*-
from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText

SMTP_SERVER='smtp.gmail.com'
SMTP_USER=raw_input('Enter your email: ')
SMTP_PASSWORD=raw_input('Enter your password: ')
SMTP_PORT=465

def send_mail(name, addr):
    import smtplib

    msg = MIMEMultipart('alternative')

    msg['From'] = SMTP_USER
    msg['To'] = addr
    msg['Subject'] = '님에게 안내메일이 도착했습니다.'.decode('utf-8')
    contents  = name.decode('utf-8') + '''님 안녕하세요.
이 메일은 파이썬에 의해 보내지는 메일입니다.
^^
잘 따라와 주셔서 감사합니다.

- 이태화 올림'''.decode('utf-8')

    msg.attach(MIMEText(_text=contents, _charset='utf-8'))

    # 포트가 465(SSL 보안연결)일 경우 클래스 사용
    smtp = smtplib.SMTP_SSL(SMTP_SERVER, SMTP_PORT)
    smtp.login(SMTP_USER, SMTP_PASSWORD)
    smtp.sendmail(SMTP_USER, addr, msg.as_string())
    smtp.close()

send_mail('이태화', 'alghost.lee@gmail.com')
```

이메일 보내기

Gmail 로 구현해보기

```
# -*- coding:utf-8 -*-
from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText

SMTP_SERVER='smtp.gmail.com'
SMTP_USER=raw_input('Enter your email: ')
SMTP_PASSWORD=raw_input('Enter your password: ')
SMTP_PORT=465
```

라이브러리 로드와 필요한 설정 변수 생성

- MIMEMultipart, MIMEText
  - SMTP 가 사용하는 양식에 맞춰서 내용을 써주는 클래스
- 대문자로 사용된 변수들: 변수지만 상수로 쓰고싶다는 '표시'
- SMTP설정에 필요한 서버, 포트, 계정 정보를 변수에 담음

이메일 보내기

Gmail 로 구현해보기

```
def send_mail(name, addr):
    import smtplib

    msg = MIMEMultipart('alternative')

    msg['From'] = SMTP_USER
    msg['To'] = addr
    msg['Subject'] = '님에게 안내메일이 도착했습니다.'.decode('utf-8')
    contents  = name.decode('utf-8') + '''님 안녕하세요.
이 메일은 파이썬에 의해 보내지는 메일입니다.
^^
잘 따라와 주셔서 감사합니다.

- 이태화 올림'''.decode('utf-8')

    msg.attach(MIMEText(_text=contents, _charset='utf-8'))
```

보낼 이메일 정보 생성

- MIMEMultipart를 활용하여 필요한 데이터 양식을 가져옴
  - 필요한 정보인 From, To, Subject를 추가
  - 내용의 경우 MIMEText를 활용하고, attach라는 함수로 추가

이메일 보내기

Gmail 로 구현해보기

```
    # 포트가 465(SSL 보안연결)일 경우 클래스 사용
    smtp = smtplib.SMTP_SSL(SMTP_SERVER, SMTP_PORT)
    smtp.login(SMTP_USER, SMTP_PASSWORD)
    smtp.sendmail(SMTP_USER, addr, msg.as_string())
    smtp.close()

send_mail('이태화', 'alghost.lee@gmail.com')
```

이메일을 보내자!

- 앞서 설정한 SMTP 서버, 포트 정보로 서버에 연결
- 앞서 설정한 계정 정보로 서버에 로그인
- 보내는 메일주소, 내 메일을 받을 메일 주소, 메시지 정보로 sendmail
- close를 통해 서버와의 연결을 닫음

이메일 보내기

Gmail 로 구현해보기

이메일 설정시 주의사항

- MIMEMultipart로 생성한 변수에 데이터를 넣을 때 유의사항
  - From과 To는 보낸사람 받는사람의 이름을 의미
  - 실제 원하는 이름을 넣을 수 있으나 메일 서버에 따라 스팸처리 됨

```
    msg = MIMEMultipart('alternative')

    msg['From'] = SMTP_USER
    msg['To'] = addr
```

```
msg = MIMEMultipart('alternative')

msg['From'] = SMTP_USER
msg['To'] = name.decode('utf-8')
```

이메일 보내기

내 이메일로 테스트 해보기

다른 메일을 쓰려면..?

- 앞서 원리를 생각해본다면!!
- 서버 정보만 변경하면 OK!
  - SMTP_SERVER를 원하는 메일의 서버로 변경
  - SMTP_PORT를 원하는 메일 서버의 포트로 변경

```
# -*- coding:utf-8 -*-
from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText

SMTP_SERVER='smtp.gmail.com'
SMTP_USER=raw_input('Enter your email: ')
SMTP_PASSWORD=raw_input('Enter your password: ')
SMTP_PORT=465
```

이메일 보내기

내 이메일로 테스트 해보기

다른 메일을 쓸때 주의사항

- SMTP 포트는 465 이거나 587 => 차이가?
- 465: SSL(보안 연결)
- 587: 일반 연결
- 주의해야될 점은??
  - 위 연결에 따라 클래스가 다름!

```
# 포트가 465(SSL 보안연결)일 경우 클래스 사용
smtp = smtplib.SMTP_SSL(SMTP_SERVER, SMTP_PORT)

# 포트가 587(일반연결)일 경우 클래스 사용
smtp = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
smtp.ehlo()
smtp.esmtp_features['auth'] = 'LOGIN PLAIN'
```

엑셀과 함께 활용

구현한 함수를 클래스에 넣자

Email() 클래스 구현

- 클래스를 구현한 파일명 조심!
  - 앞서 사용한 라이브러리 이름이…? email
  - 근데 내가 구현한 파일명 이름이…? email
    - 문제가 발생할 수 있음 => 클래스를 가질 py 파일을 만들때 유의!
- 클래스 함수에 맞게 일부 수정이 필요
- 함수는 나에게 필요한 형태로 구현해야 함
- 하지만.. 일단 다같이 같은 형태로 구현해보자

엑

구현한

Em

- 클래

  - 앞

  - 근

  - 문

- 클래

- 함수

- 하지

```
# -*- coding:utf-8 -*-
from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText

SMTP_SERVER='smtp.gmail.com'
SMTP_USER=''
SMTP_PASSWORD=''
SMTP_PORT=465

class Email():
    from_email  = SMTP_USER
    subj_layout = '님에게 안내메일이 도착했습니다.'.decode('utf-8')
    cont_layout = '''님 안녕하세요.
이 메일은 파이썬에 의해 보내지는 메일입니다.
^^
잘 따라와 주셔서 감사합니다.

- 이태화 올림'''.decode('utf-8')

    def send_mail(self, name, to_email, attachment=None):
        import smtplib

        msg = MIMEMultipart('alternative')

        msg['From'] = self.from_email
        msg['To'] = to_email
        msg['Subject'] = name.decode('utf-8') + self.subj_layout

        contents = name.decode('utf-8') + self.cont_layout
        msg.attach(MIMEText(_text=contents, _charset='utf-8'))

        # 포트가 465(SSL 보안연결)일 경우 클래스 사용
        smtp = smtplib.SMTP_SSL(SMTP_SERVER, SMTP_PORT)
        smtp.login(SMTP_USER, SMTP_PASSWORD)
        smtp.sendmail(SMTP_USER, to_email, msg.as_string())
        smtp.close()
```

엑셀과 함께 활용

엑셀예제와 클래스를 참고하여 구현

엑셀 데이터 만들기

- 엑셀을 열어서 아래 데이터를 만들어봅시다!
- 직접 메일을 보내니 각자 개인메일을!
- 일부 이름, 메일을 공란으로!

Sheet1

이태화	lthlovelee@naver.com
김태화	lthlovelee@naver.com
김용성	lthlovelee@naver.com
장동건	lthlovelee@naver.com
원빈	lthlovelee@naver.com
강동원	lthlovelee@naver.com
공유	lthlovelee@naver.com
송혜교	lthlovelee@naver.com
한빈	lthlovelee@naver.com

엑셀과 함께 활용

엑셀예제와 클래스를 참고하여 구현

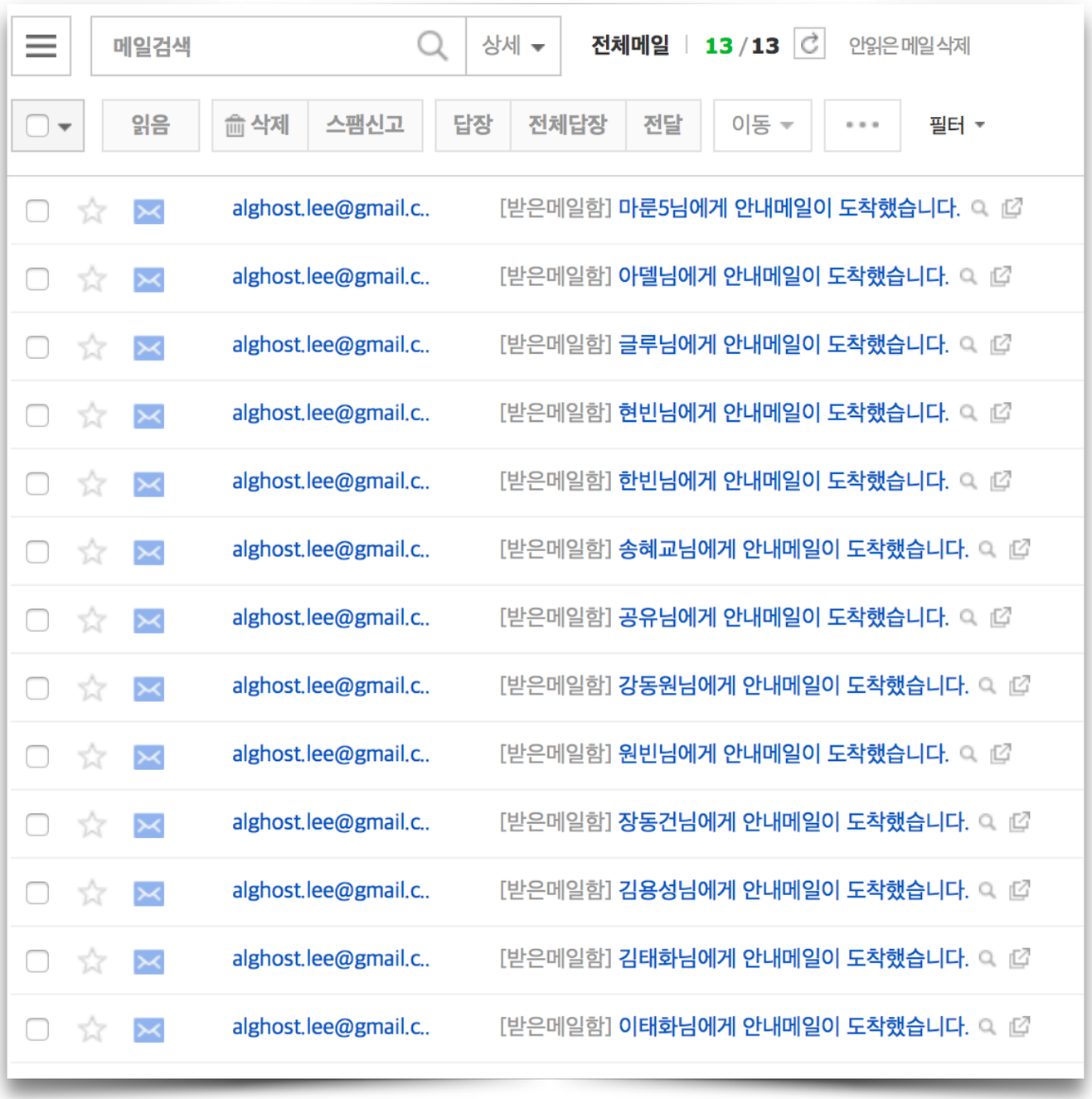
엑셀로부터 읽어서 메일보내기

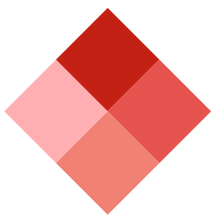
```
# -*- coding:utf-8 -*-
from openpyxl import load_workbook
from my_email import Email

email = Email()

wb = load_workbook(filename='email_list.xlsx', read_only=True)
ws = wb.active

for row in ws.rows:
    name = row[0].value
    addr = row[1].value

    if not name or not addr:
        continue

    email.send_mail(name, addr)
```

엑셀과 함께 활용

엑셀예제와 클래스를 참고하여 구현

엑셀로부터 읽어서 메일보내기





Good Bye

See you next time

Appendix

메일 주소 유효성 검사

메일 주소 유효성 검사

- 정규표현식을 사용해서 할 수 있음
- 정규표현식이란?
  - 문자열을 일정한 패턴으로 표현할 수 있는 식
  - 원하는 패턴의 문자열을 추출하거나 일치 여부 등을 확인할 때 사용됨
- 정규표현식은 정말 정말 어렵다
  - 보통 정규표현식을 작성해도 수십, 수백 차례 테스트 끝에 검증이 됨
- **그렇기 때문에, 대개 다른 사람이 잘 써논걸 가져다가 씀(내얘기)**
- **따라서! 우리는 이메일 검증용 정규표현식만을 소개**

Appendix

메일 주소 유효성 검사

메일 주소 유효성 검사

- 이메일 주소 검사용 정규표현식을 제공 사이트: http://emailregex.com
  - “(^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$)”
- **아래부터는 관심있으신분들만..**
  - ^: 문자열의 시작을 의미
  - []: []안의 문자가 올 수 있음을 의미
  - +: 바로 직전의 문자가 1번 이상 반복됨
  - @: @가 있음을 의미
  - $: 문자열이 끝남을 의미
- 즉, (알파벳,숫자,_,.,+,-)가 @앞에 나열되고 그 뒤에 (알파벳,숫자,-)가 반복되다가 .이 나오고 또 (알파벳,숫자,-,.)가 반복되다가 문자열이 끝남

Appendix

메일 주소 유효성 검사

메일 주소 유효성 검사

- ^^..어렵습니다
- 정규표현식은 어려웠지만 코드는 간단!
- re라는 파이썬의 기본 라이브러리를 사용

Appendix

메일 주소 유효성 검사

메일 주소 유효성 검사

```
def is_valid_email(addr):
    import re
    if re.match('(^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$)', addr):
        return True
    else:
        return False

print is_valid_email('alghost.lee@gmail.com')
print is_valid_email('test@asdf')
print is_valid_email('not email~~~~')
print is_valid_email('iower!@gmail.com')
print is_valid_email('asdfoa@naver.co.kr')
```

```
taehwaui-MacBook-Pro:email Alghost$ python ../../커리큘럼용\ 예제/alghost_email.py
True
False
False
False
True
```

Appendix

메일에 파일 첨부

send_mail 을 개선해보자

- 첨부파일이 있을 경우 추가하고
- 첨부파일이 없을 경우 추가하지 않고 내용만 전달하도록 개선
- 첨부파일을 추가하는 방법

```
if(attachment):
    from email.MIMEBase import MIMEBase
    from email import Encoders

    file_data = MIMEBase('application', 'octet-stream')
    file_data.set_payload(open(attachment, 'rb').read())
    Encoders.encode_base64(file_data)
    file_data.add_header('Content-Disposition', 'attachment; filename=\"' + attachment + '\"')
    msg.attach(file_data)
```

Appendix

메일에 파일 첨부

```
1 if(attachment):
2     from email.MIMEBase import MIMEBase
3     from email import Encoders
4
5     file_data = MIMEBase('application', 'octet-stream')
6     file_data.set_payload(open(attachment, 'rb').read())
7     Encoders.encode_base64(file_data)
8     file_data.add_header('Content-Disposition', 'attachment; filename=\"' + attachment + '\"')
9     msg.attach(file_data)
```

한줄 한줄 의미를 파악해보자

- 1: 파일명이 있을 경우 수행하기 위해 if문 사용
- 2: MIMEBase 라이브러리 추가 - 파일과 같은 데이터를 다루는 클래스
- 3: Encoders 라이브러리 추가 - 전송이 가능한 형태로 변경하기 위한 클래스
- 5: 빈 MIMEBase 생성
- 6: 입력받은 파일명(attachment)를 open하여 해당 데이터를 MIMEBase에 넣음
- 7: MIMEBase를 전송가능한 형태(base64)로 변경
- 8: 전송할 파일에 대한 설명을 추가 => 이 데이터는 첨부파일이고 파일명은 attachment이다
- 9: 기존에 사용했던 MIMEMultipart에다가 추가

Appendix

```
def send_mail(self, name, to_email, attachment=None):
    import smtplib

    msg = MIMEMultipart('alternative')
    if(attachment):
        msg = MIMEMultipart('mixed')

    msg['From'] = self.from_email
    msg['To'] = to_email
    msg['Subject'] = name + self.subj_layout

    contents = name + self.cont_layout
    msg.attach(MIMEText(_text=contents, _charset='utf-8'))

    if(attachment):
        from email.MIMEBase import MIMEBase
        from email import Encoders

        file_data = MIMEBase('application', 'octet-stream')
        file_data.set_payload(open(attachment, 'rb').read())
        Encoders.encode_base64(file_data)
        file_data.add_header('Content-Disposition', 'attachment; filename=\"' + attachment + '\"')
        msg.attach(file_data)

    # 포트가 465(SSL 보안연결)일 경우 클래스 사용
    smtp = smtplib.SMTP_SSL(SMTP_SERVER, SMTP_PORT)
    smtp.login(SMTP_USER, SMTP_PASSWORD)
    smtp.sendmail(SMTP_USER, to_email, msg.as_string())
    smtp.close()
```

Appendix

메일에 파일 첨부

함수 정의 부분에 추가된 부분이!

- send_mail에 3번째 입력값으로 attachment가 추가됨
- 그런데 그 변수에 None을 저장했다…?
  - None: 파이썬에서 아무것도없다(nothing)를 의미
  - 만약 send_mail을 호출했을 때 3번째 입력값을 안주면 None을 씀
  - 첨부파일이 없는 경우에도 send_mail을 호출할 수 있어야 하기 때문
  - 만약 =None을 쓰지 않고, send_mail('이름', '메일')처럼 사용한다면 입력값이 3개 필요한데 2개밖에 안넣어줬다고 에러를 발생시킴

```
def send_mail(self, name, to_email, attachment=None):
```