소프트웨어 개발보안 기술문서

20418 정연서

1. 환경 설정

1.1. python 설치

- python.org 접속
- 상단의 Downloads 메뉴에 마우스 커서 놓기
- 3.0 이상의 python 설치 파일 다운 (18.03.06 기준 Windows latest release version : Python 3.6.4)
- 설치 파일 실행
- 설치 시 환경 변수 체크

1.2. sublimetext 3 설치

- https://www.sublimetext.com/3 접속 > sublimetext3 download

1.3. SublimeText 단축키

- sublime 전체 주석 단축키 : Ctrl + /
-
- 이전 명령 재실행 > Command Palette : Ctrl + Shift + P

```
print("Hello World!")       #test code
```

```
결과 :
Hello World!
```

2. Python 기본

2.1. Python

- 컴파일 없이 실행 가능한 스크립트 언어
- 인터프리터 사용

컴파일러와 인터프리터

컴파일러	인터프리터
프로그램 단위 번역	명령 줄 단위 번역
번역 속도 느림	번역 속도 빠름
실행 속도 빠름	실행 속도 느림
큰 메모리 필요	적은 메모리 필요

2.2. Python 기초문법

2.2.1 자료형

- 데이터를 변수에 담는 순간 자료형이 정해진다.
- Python에서는 자료형마다 연산을 지원한다.

2.2.2 출력

```
>>> a=1
>>> a
1
>>> print(a)
1
>>> print(a,“test”)
1 test
```

2.2.3 연산자

- 정수와 실수의 연산의 결과는 실수이다.
- 음수를 나누었을 때 나머지가 있는 경우는 -1이 추가된다.

ex) -11 // 2 => -6 (-1이 추가된 값)

2.2.4 문자열 선언 방법

```
>>> a="asdf"                # "" 포함
>>> a='asdf'                # ''포함
>>> a='''asdf'''
>>> a="""asdf"""
```

(탭: Ctrl + TAB)

2.2.5. 인덱싱
- 문자열에서 특정 문자를 가리키는 것을 의미한다.
- 인덱스 값은 0부터 시작한다.

```
>>> a="2018-03-13"
>>> print(a[0])
2
```

2.2.6. 슬라이싱
- 문자열에서 특정 문자열을 자르는 것

```
>>> a="2018-03-13"
>>> print(a[2:7])
18-03
>>> print(a[:7])
2018-03
>>> print(a[5:])
03-13
```

2.2.7. 리스트
- C언어에서 배열과 비슷한 개념

```
>>> a=[0, 123, "aaa", 111, 1.1]
>>> a
[0, 123, 'aaa', 111, 1.1]
>>> print(a[2:])
['aaa', 111, 1.1]
>>> print(a[1], "and", a[3:5])
123 and [111, 1.1]
```

2.2.8 딕셔너리
- 대응 관계를 나타낼 수 있는 자료형
- key와 value가 연결된 자료형
- 현실 세계의 사전과 비슷하다.
- 딕셔너리 자체의 연산은 없다.
- 인덱싱은 가능하지만 슬라이싱은 불가능하다.

```
>>> dic={'1':'aaaa', '2':'bbbb', '3':'cccc'}
>>> dic
{'1':'aaaa', '2':'bbbb', '3':'cccc'}
>>> dic['1']
'aaaa'
```

3. Python 함수

3.1. 리스트 함수

3.1.1. count(x) : 리스트 내에 x의 개수를 반환

```
a ='programming'
res = a.count('m')
print(res)

결과 : 2
```

3.1.2. index : 찾고자 하는 내용이 없으면 error 출력

```
a ='programming'
res = a.index('m')
print(res)

결과 : 6
```

3.1.3. find : 찾고자 하는 내용이 없으면 -1 출력

```
a ='programming'
res = a.find('m')
print(res)

결과 : 6
```

3.1.4. join : 리스트를 문자열로 반환

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

결과 : A_m-_-m_B_m-_-m_C
```

```
# join 함수가 메소드로 지정된 변수의 문자를 find함수의 인자(문자) 사이에 삽입한다.
# 숫자는 불가능하고, 굳이 하고 싶다면 따옴표 사용
# 한 줄씩 띄우게 하려면 \n, \t와 같은 escape 문자 사용
ab = [1,'asdf',123, 'ssfa']
print(ab[2:4])

결과 : [123, 'ssfa']
```

3.1.5. append : 인자를 리스트에 추가한다.

```
>>> a = [1, 2, 3]
>>> a.append(4)
>>> print(a)
[1, 2, 3, 4]
```

3.1.6. sort : 리스트 내의 요소들을 정렬한다.

```
res1=['e','a','h']
res2=[1,6,2]
res1.sort()
res2.sort(reverse = True)       # reverse = True : 내림차순 정렬
res3 = sorted(res1, reverse = True)
res4 = sorted(res1)                          # 외부정렬
print(res1)
```

```
print(res2)
print(res3)
print(res4)

#sorted는 딕셔너리 정렬이 가능.
a={'2':'B','1':'A','3':'U'}
a1=sorted(a)
print(a1)

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

3.1.7. insert : 특정 인덱스의 값이 되도록 요소를 추가한다.

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

결과 : [100, 2, 123, 523]
```

3.1.8. remove : 함수의 인자값을 찾아서 삭제한다.

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

결과 : [20, 30, 40, 10]
```

3.1.9. pop : 마지막 요소를 삭제한다. 반환값이 있으므로 변수에 pop한 값을 저장할 수 있다.

```
res = [10,20,30,40]
res.pop()                                    # 반환값이 있는 함수이다
print(res)

결과:
40
[10, 20 30]
```

3.2. 문자열 함수

3.2.1. upper : 소문자를 대문자로 변환

3.2.2. lower : 대문자를 소문자로 변환

```
a ='karKat VanTaS'
res = a.upper()
res1 = a.lower()
print(res,"\t",res1)

결과 :
KARKAT VANTAS
karkat vantas
```

3.2.3. replace : 문자열 바꾸기

```
>>> a="aabbcc"
>>> a.replace("bb","dd")
>>> print(a)
'aaddcc'
```

3.2.4. split : 문자열 나누기

```
>>> a="Life is too short"
>>> a.split()
>>> print(a)
['Life', 'is', 'too', 'short']
```

```
>>> a="a:b:c:d"
>>> a.split(':')
>>> print(a)
['a', 'b', 'c', 'd']
```

3.2.5. strip : 양쪽 공백 지우기

3.2.6 lstrip : 왼쪽 공백 지우기

3.2.7 rstrip : 오른쪽 공백 지우기

```
>>> a="  asdf  "
>>> print(a.strip())
'asdf'
>>> print(a.lstrip())
'asdf  '
>>> print(a.rstrip())
'  asdf'
```

3.3. 내장 함수

3.3.1. type : 자료형 반환

```
>>> a="DSM"
>>> type(a)
<class 'str'>
```

3.3.2. int

- 문자열 형태의 소수점이 있는 숫자 등을 정수 형태로 반환
- 정수를 입력으로 넣으면 그대로 반환한다.

```
>>> int(3)
3
>>> int(3.141592653597932)
3
```

3.3.3. str : 문자열 형태로 객체를 변환하여 반환

```
>>> str(555)
'555'
```

3.3.4. ord : 문자의 아스키 코드값 반환

```
>>> ord('A')
65
```

3.3.5. chr : 아스키 코드값에 해당하는 문자 반환

```
>>> chr(65)
'A'
```

3.4. 딕셔너리 함수

3.4.1. keys : 딕셔너리의 key들을 반환한다.

- 출력 형태 : dict_keys([key1, key2, ...])

```
a = {'a':123,'b':456}
res = a.keys()
print(res)

결과 : dict_keys(['a', 'b'])
```

3.4.2. values : 딕셔너리에서 각 키들에 대응하는 값을 반환한다.

- 출력 형태 : dict_values([value1, value2, ...])

```
res = a.values()
print(res)

결과 : dict_values([123, 456])
```

3.4.3. items : 딕셔너리의 각 키와 그것에 대응하는 값을 반환한다.

- 출력 형태 : dict_items([(key1, value1), (key2, value2), ...])

```
res = a.items()
print(res)

결과 : dict_items([('a', 123), ('b', 456)])
```

3.4.4. get : 인자로 준 키에 대응하는 값을 반환한다.

```
a = {'q':123,'w':456}
res=a.get('q')
print(a['q'])
print(res)
# get함수를 쓸때 키가 없는 값을 찾으면 none을 반환한다.
# 그냥 딕셔너리에서 키 값으로 찾았을 때 없으면 오류
# 찾고 싶은 키 값이 없는 것이라면 기본값을 저장하여 반환할 수 있다.

결과 :
123
123
```

3.4.5. in : 키 값이 있는지 검사하고 있으면 True, 없으면 False를 반환한다.

```
a = {'q':123,'w':456}
print('q'in a)

결과 : True
```

4. Python 코드 응용

4.1. Python 코드축약

- 순서 : 문제 작성 ▶ 파일해결 논리에 의한 정상 코드 ▶ Python 기능을 활용하여 축약 코드 제시

> 문제1 : 키보드를 사용하여 두 정수를 입력하면 덧셈, 뺄셈, 곱셈, 나눗셈, 몫, 나머지가 출력되는 계산기 프로그램을 작성하시오.

```
#1
a = input()
b = input()
a =int(a)
b =int(b)
print(a + b)
print(a - b)
print(a * b)
```

```
print("%.2f" %(a / b))          # formatting C언어랑 다른 점은 % 붙인다는 것
print(int(a / b))
print(a % b)
```

```
#2
a,b = input().split()           # 한 줄 입력이 가능해진다
a =int(a)
b =int(b)
print(a + b)
print(a - b)
print(a * b)
print("%.2f" %(a / b))
print(int(a / b))
print(a % b)
```

```
#3
a,b = map(int, input().split())          # 입력을 받아서 각 변수에 정수로 mapping
print(a + b)
print(a - b)
print(a * b)
print("%.2f" %(a / b))
print(int(a / b))
print(a % b)
```

```
#4
a,b = map(int, input().split())
print(a +b, a -b, a *b, "%.2f" %(a /b), a //b, a%b, sep ='\n')
```

> 문제2 : 영어 문자열을 입력받아서 단어의 개수를 출력하는 프로그램을 작성하시오.

```
#1
s = input("Insert an English sentence : ")
s.strip()
cnt = s.count(' ') +1
print(cnt)
```

```
#2
s = input("Insert an English sentence : ").strip()
cnt = s.count(' ') +1
print(cnt)
```

```
#3
cnt = input("Insert an English sentence : ").strip() +1
print(cnt)
```

```
#4
print(input("Insert an English sentence : ").strip().count(' ') +1)
```

4.2. Python 제어구조

> 문제1 : 나이를 정수로 입력받아 20세 이상이면 “party tonight”을, 20세 미만이면 “Study tonight”을 출력하는 프로그램을 작성하시오.

```
# python의 코드 구분은 들여쓰기로 수행된다.
# 함수, 조건, 반복 구조 등 내포가 필요한 구문은 콜론:으로 구분한다.
```

```
#1
age = input("나이를 입력하세요 : ")
if int(age) >=20:      # 중괄호 써도 되고 안 써도 됨
            print("Party tonight")
else: print("Study tonight")
```

```
# 삼항 연산자
# format : 명령문 if 조건 else 거짓일 때의 명령문

#2
age = input("나이를 입력하세요 : ")
print ("party tonight"if int(age) >=20 else "Study tonight")
```

```
결과 :
나이를 입력하세요 : 22
party tonight
--------------------
나이를 입력하세요 : 16
Study tonight
```

> 문제2 : a라는 변수에 아무 입력이나 계속 받고(숫자/문자) a가 1이 아닐 경우 This is not the one을, a가 1일 경우 The end를 출력하는 프로그램을 작성한다.

```
while True:
            a=input("input a : ")
            if a =='1'and int(a)==1:
                        print("The end")
                        break
            else: print("This is not one")
```

```
결과 :
input a : 3
This is not one
input a : 9
This is not one
input a : -5
This is not one
input a : 1
The end
```

> 문제3 : for문을 활용한 구구단 출력

```
for x in range(2, 10):
    for y in range(1, 10):
        print(x, "*", y, "=", x *y)
```

```
결과 :
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
```

3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48

```
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
```

> 문제4 : for문을 리스트에 내포하기

```
list_a=[3,6,9,12]
res1=[]
for n in list_a:
            res1.append(n-1)
print(res1)
res2=[n -2 for n in list_a]
print(res2)
```

```
결과 :
[2, 5, 8, 11]
[1, 4, 7, 10]
```

5. Python Problem Solving (문제 해결)

문제 해결 단계

- 추상화 : 문제 분해, 핵심 요소 추출
  - > 현재상태 정의
  - > 목표상태 정의
  - > 문제 분해
  - > 핵심요소(조건) 추출

- 알고리즘 : 절차적 사고 표현
  - > 0개 이상의 입력
  - > 1개 이상의 결과
  - > 명확성/유한성/실행 가능성

- 자동화 : 프로그래밍 (코딩)

5.1. 하노이의 탑 문제

5.1.1. 추상화

```
#1

현재상태 : 탑이 모두 A기둥이 있음
목표상태 : 탑이 모두 C기둥에 있음
문제분해 : ?
핵심요소 : ?
```

```
#2
```

```
현재상태 : 탑이 모두 A기둥이 있음
목표상태 : 제일 큰 원판이 A기둥에 있고 나머지 원판이 B기둥에 있음
문제분해 : 위 과정이 문제 분해임
핵심요소 : 제일 큰 원판 -> n
           나머지 원판 -> 1 ~ n-1
```

```
#3

현재상태 : 제일 큰 원판이 A기둥에 있고, 나머지 원판들이 B기둥에 있음
목표상태 : 제일 큰 원판이 C기둥에 있고, 나머지 원판들이 B기둥에 있음
문제분해 : 위의 과정이 문제 분해임
핵심요소 : 제일 큰 원판 옮기는 횟수 -> +1
```

```
#4

현재상태 : 제일 큰 원판이 C기둥에 있고, 나머지 원판들이 B기둥에 있음
목표상태 : 모든 원판이 C기둥에 있음
문제분해 : 위의 과정이 문제 분해임
핵심요소 : 나머지 기둥 옮기는 횟수 -> n-1
           n-1이 1이면 종료
```

5.1.2. 알고리즘 : 추상화 단계를 토대로 알고리즘을 설정한다.
5.1.3. 자동화
(코드 축약 포함)

```
#1
cnt =0
def hanoi(num, A, B, C):
        if num ==1:
                print(A, '->', C)
                cnt +=1
        else:
                hanoi(num -1, A, C, B)
                hanoi(1, A, B, C)
                hanoi(num -1, B, A, C)
num =int(input())
hanoi(num, 'A', 'B', 'C', cnt)
```

```
#2
def hanoi(n):
        if(n ==1): return 1
        else: return hanoi(n -1) +1 + hanoi(n -1)
num =int(input())
hanoi(num, 'A', 'B', 'C', cnt)
```

```
#3
print(2 **int(input("input hanoi column's number : "))-1)
```

```
결과 :
input hanoi column's number : 10
1023
--------------------------------
input hanoi column's number : 56
72057594037927935
```

5.2. 다른 문제 해결 예제

> 문제1 : n개의 계단을 오를 때 한 번에 1계단 또는 2계단으로 오를 수 있는 방법의 수 구하기

#추상화

```
현재상태 : 올라갈 계단이 1칸
목표상태 : 계단을 모두 올라감
문제분해 : -
핵심요소 : 올라가는 경우의 수 = 1개
```

...

#알고리즘 및 자동화

```
def fib(n):
        if n ==1: return 1
        elif n ==2: return 2
        else: return fib(n -1) + fib(n -2)
print(fib(int(input("input the number of stairs : "))))
```

```
결과 :
input the number of stairs : 10
89
------------------------------
input the number of stairs : 20
10946
```

> 문제2 : 코드표를 통한 암호해독

- 주어지지 않은 10개의 코드를 가진 암호코드표가 주어지고, 각각의 암호에는 0부터 9까지의 숫자가 매칭된다. 암호문이 주어졌을 때 이 암호코드를 기반으로 이 암호문을 복호화하는 알고리즘(Python 코드)를 작성하시오. (단, 암호문은 공백을 허용하며, 암호코드에 있는 문자만으로 암호문을 입력한다.

ex) input1 : lohcgpdabk // 암호코드표
    input2 : cdp // 암호문
    출력 : 365

#추상화

```
현재상태 : 암호문 목록(암호코드표)
목표상태 : 암호문의 문자가 암호코드표의 코드와 매칭되었는지 확인 후 인덱스를
문제분해 : -
핵심요소 : 암호코드표를 순차탐색하여 암호문의 문자가 있는지 확인
          없는 문자(공백)에 대해서는 동장하지 않음
```

#알고리즘 및 자동화 (코드축약 단계 포함

```
#1
a = str(input())
b = str(input())
i =0
while(i <len(b)):
        if b[i] !=" ": print(a.find(b[i]), end ="")
        else: print(b[i], end ="")
        i = i +1
```

```
#2
a = str(input())
b = str(input())
i = 0
while(i < b[i]):
        print(a.find(b[i]) if b[i] !='' else b[i], end ="")
        i = i +1
```

```
#3
a = str(input())
b = str(input())
i = 0
for i in b: print(a.find(i) if i !='' else i, end ="")
```

```
결과 :
10
30
-11
```

6. 함수

> 정의 : 입력 값을 받아서 특정 연산(작업)을 수행한 후에 결과를 출력하는 것
- 입력 값을 받아서 특정 연산(작업)을 수행한 결과이기도 하다.
- 프로그래밍에서는 조금 다른 의미를 가진다. 결과보다는 어떤 기능을 하느냐가 더 주안점이다.

6.1. 사용자 함수

6.1.1. 주의사항
- 함수 호출 전까지는 함수 안의 문장들은 수행되지 않는다.
- 함수는 호출되기 전에 먼저 생성되어져 있어야 한다.

> 양식 : def {function name}(parameter 1, parameter 2 ...): ...

6.1.2. 동작과정
- 반환 유무를 생각해서 사용자 함수를 작성하여야 한다.

```
#function with no return value
def func(*a):
        print(a *3)
print('start')
func('y','s','w')
#함수는 선언하고 호출하는 위치가 중요하다.
#Python은 C언어에서처럼 프로토타입을 선언할 수 없다.

결과 :
start
('y', 's', 'w', 'y', 's', 'w', 'y', 's', 'w')
```

```
#function with return value
def sum(a,b):
        return a +b
a=int(input())
b=int(input())
res=sum(a,b)
print(res)

결과 :
```

```
10
20
30
```

6.1.3. 사용자 함수 작성

> 조건 : 문자열을 입력받은 후 문자열 내에 skip이라는 단어가 있으면 rejected를 출력하고, 없으면 그대로 출력한다. 매 출력마다 '-'를 10개 출력해서 각 출력을 구분하고 사용자가 quit을 입력하여 끝낼 때까지 프로그램을 무한반복한다.

```
def read(line):
        if 'skip'in line:
                print('rejected')
                return
        else:
                print(line)
        print('-'*10)
user_input=' '
while(user_input !='quit'):
        user_input=input()
        read(user_input)
```

```
결과 :
DSM
DSM
----------
the last straw
the last straw
----------
skip
rejected
quit
quit
----------
```

7. 클래스

- 일종의 템플릿
- C언어의 구조체와 유사하다.
- 구조체와의 차이점은 구조체는 변수만 담을 수 있지만 클래스는 함수까지 담을 수 있다.

#형식 :

```
class {class name}():
        변수명=변수값
        def {function name}(self, instance, ...): ...
```

7.1. 생성

- C언어의 구조체처럼 '.'을 이용하여 클래스 내부를 활용할 수 있다.
- 클래스도 함수처럼 호출되기 전에는 수행되지 않는다.

```
class SimpleTest():
        a=0
        postfix='\t DSM'
        def print_with(self, string):
                print(string)
```

```
                    print(self.a)
                    print(str(self.a)+string +self.postfix)
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('\tOMG')
s2.print_with('\tWWW')
```

```
결과 :
0
0
10
20
        OMG
10
10      OMG     DSM
        WWW
20
20      WWW     DSM
```

7.2. self
- 클래스의 변수에 접근하기 위해 Python에서 제공하는 변수
- 클래스 내에서 함수를 정의할 때 잊지 않고 사용하여야 한다.

7.3. 생성자
- 클래스 변수가 생성될 때 자동으로 호출되는 함수
- 클래스 내부에 정의된 변수 등을 초기화할 때 사용된다.

```
class SimpleTest():
        my_data =0
        def __init__(self):
                self.my_data =100
                print('Call init')
simple=SimpleTest()
print(simple.my_data)
```

```
결과 :
Call init
100
```

8. Python 메일링 및 Exel 파일 응용

8.1. 이메일 보내기

필요한 설정 정보
- SMTP 서버(메일을 보내는 서버) 주소
- POP3 서버(메일을 받는 서버) 포트 번호
- 계정 정보 (ID, 패스워드)

네트워크 7계층 중 표현 계층에 있는 MIM라이브러리를 사용하여 SMTP, POP3을 활용한다.

라이브러리
- email.smtplib : 기본 라이브러리로 제공되며 따로 설치하지 않아도 되다.
라이브러리 내부
- MIME : 전자우편을 위한 인터넷 표준 포맷
- MIMEText, MIMEMultipart : SMTP가 사용하는 양식에 맞춰서 내용을 작성해주는 클래스

```
# -*- coding:utf-8 -*-
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import smtplib
SMTP_SERVER ='smtp.gmail.com'
SMTP_PORT =465
SMTP_USER ='newdk3025@gmail.com'
SMTP_PASSWORD = (password)
def send_mail(name, addr):
        msg = MIMEMultipart()
        # 메일 전송을 위한 양식 작성
        msg['From'] = SMTP_USER
        msg['To'] = addr
        msg['Subject'] ='A mail for'+ name
        contents = name +'Test mail'
        # msg['text'] = contents
        text = MIMEText(_text = contents, _charset ='utf-8')
        msg.attach(text)
        # 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())
        print("\n 메일이 전송되었습니다 \n")
        smtp.close()
```

8.2. openpyxl로 Exel 파일 활용하기
openpyxl 라이브러리 설치

```
>>> pip install openpyxl
```

사람과 프로그램(openpyxl 기능)이 Exel 파일을 다루는 방법을 서로 비교하여 보자.

사람	프로그램
데이터가 들어있는 파일 실행	데이터가 들어있는 파일명으로 클래스 변수 생성
데이터가 들어있는 시트로 이동	클래스 변수에서 시트 이름을 활용하여 시트 이동
데이터가 있는 위치의 데이터를 활용	데이터가 있는 위치의 데이터를 활용

8.2.1. openpyxl 사용 예시

```
from openpyxl import load_workbook
#load_workbook 함수를 이용하여 엑셀 클래스 변수 생성
wb = load_workbook('Sheet1.xlsx')
# 활성화된 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])
```

```
from openpyxl import Workbook
wb = Workbook()
sheet = wb.active
sheet['A1'] ='Number'
sheet['B1'] ='Name'
sheet['A2'] =1
sheet['B2'] ='AAA'
sheet['A2'] =2
sheet['B2'] ='BBB'
#Sheet2.xlsx라는 파일이 있으면 덮어쓰기 됨
#없으면 새로운 엑셀파일 생성
wb.save('Sheet2.xlsx')
```

```
from openpyxl import load_workbook
#load_workbook 함수를 이용하여 엑셀 클래스 변수 생성
wb = load_workbook('Sheet1.xlsx')
# 활성화된 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])
```

```
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(1 +65)*3])
        wb.save('Sheet3.xlsx')
# 저장된 Exel 파일 내용 출력
rows = sheet['1:11']
for row in rows:
        print(row[0].value, row[1].value)
# 셀 번호를 일일이 확인할 수 없으므로 셀이 입력되어 있는 구간을 알아서 인식하도록 하면 좋음
for row in sheet.iter_rows():
        print(row[0].value, row[1].value)
```

```
# 앞에서 배웠던 메일링과 openpyxl을 함께 사용하는 예제.
# 이메일 주소 목록을 이메일로 보내는 프로그램
from mail_func import Email
from openpyxl import load_workbook
wb = load_workbook('Email_list.xlsx')
sheet = wb.active
```

```
e = Email()
for row in sheet.iter_rows():
            name=row[0].value
            add=row[1].value
            e.send_mail(name,addr)
```

8.3. pyautogui 라이브러리
- Python의 그래픽 라이브러리.
- 매뉴얼이 상세하지 않다.
코드에 적용

```
import pyautogui
```

8.3.1. 함수
8.3.1.1. size : 화면의 크기를 반환

```
scrWD, scrHE = pyautogui.size()
print(scrWD, scrHE)
```

8.3.1.2. moveTo : 원하는 위치(절대좌표)로 커서를 이동한다. 마우스의 위치를 확인하고 싶다면 ctrl 옵션을 설정한다.

```
pyautogui.moveTo(scrWD, scrHE)
pyautogui.moveTo(scrWD /2, scrHE /2)
```

8.3.1.3. moveRel : 원하는 위치(상대좌표)로 커서를 이동한다. 이동하지 않으려면 None값으로 인자를 설정한다.

```
for i in range(1, 10):
            pyautogui.moveRel(150, None)
            pyautogui.moveRel(None, -150)
            pyautogui.moveRel(-150, None)
            pyautogui.moveRel(None, 150)
```

8.3.1.4. click : 클릭하는 함수
- 옵션을 통해 클릭 횟수와 어떤 버튼을 클릭한 것인지 지정 가능하다.
> x, y : 마우스 위치 이동
> clicks : 마우스 클릭 횟수
> interval : 클릭 간격 조정 (ex: second, once 등)
> button : 마우스 버튼 위치 선택(left/right)

```
pyautogui.click(x = scrWD /3, y = scrHE /2, button ='left')
```

8.3.1.5. typewrite : 키를 입력하는 함수
- 영어만 입력할 수 있다.

```
pyautogui.typewrite('cls')
```

8.3.1.6. press : 특수키를 입력할 때 사용하는 함수

```
pyautogui.press('enter')
```

8.3.1.7. locteCenterOnScreen : 그림과 일치하는 위치의 좌표 반환 함수
- 그림 파일의 확장자를 png로 설정해야 한다.

```
lx, ly = pyautogui.locateCenterOnScreen('test.png')
pyautogui.moveTo(lx, ly)
pyautogui.click(clicks =1, button ='left')
```

9. 웹 크롤링

웹사이트 또한 문서이고, 웹브라우저는 그러한 문서를 해석한 후 보여주는 장치이다. 웹브라우저 별로 내부에서 사용하는 웹 드라이버가 존재한다.

웹 드라이버의 역할
- 웹 문서 분석 후 이를 활용하여 화면을 구성한다.
- 웹 문서에 이벤트를 전달하고 결과값을 받는다.
- 웹 드라이버가 제공하는 방법으로 서로 주고받아야 한다.

웹 드라이버를 직접 다루는 것은 브라우저를 만드는 것과 동일하다. Python 라이브러리를 통해 웹 문서에 있는 내용을 가져오는 웹 크롤러를 만들어보자.

Selenium
- 일종의 서버 프로그램
- 라이브러리로 제공된다.
- 다양한 브라우저의 웹 드라이버를 컨트롤할 수 있다.

9.1. 웹 자동화
9.1.1. Selenium 설치

```
>>> python -m pip install --upgrade pip
>>> pip install selenium
```

9.1.2. 웹 페이지 분석
- 가져오고자 하는 정보의 html 태그, class 이름 등을 상세하게 살펴본다.

9.1.3. 코딩

try ~ except ~ finally 구문
- try 안의 코드를 수행하다가 에러가 발생하면 발생한 시점 이후의 코드는 수행하지 않고 except로 넘어가서 코드를 수행한다. finally는 에러 여부와 상관없이 무조건 수행된다.

> 예제 1: 학교 급식 목록 크롤링

```
#크롬 브라우저를 띄우기 위해 selenium 웹 드라이버를 가져옴
from selenium import webdriver
#크롬 드라이버로 크롬 브라우저를 실행
driver = webdriver.Chrome('chromedriver')
try:
        #네이버 뉴스 페이지로 이동
        driver.get('http://dsm2015.cafe24.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('meal-content-wrapper')

        # 위 div_id 안에 li태그로 구분되어 있는 정보를 가져와 리스트로 저장
        news_list = title_id.find_elements_by_tag_name('div')
        # 가져온 태그들에 대해 반복문을 수행하면서 각각의 문자열을 출력
        for news in news_list:
                print(ews.text)
except Exception as e:
        print(e)
finally:
        #브라우저 종료
        driver.quit()
```

> 예제2 : NAVER에서 대덕소프트웨어마이스터고등학교를 검색하여 나오는 블로그와 뉴스 게시글의 제목과 URL

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
driver = webdriver.Chrome('chromedriver')
try:
        driver.get('http://www.naver.com')
        print(driver.title)
        elem = driver.find_element_by_id('query')
        elem.clear()
        #clear()를 해주는 이유는 간혹 포털마다 검색어가 이미 입력되어 있는 경우가 있기 때문
        elem.send_keys('대덕소프트웨어마이스터고등학교')
        elem.send_keys(Keys.RETURN)
        blogs = driver.find_element_by_class_name('_blogBase')
        blogs_list = blogs.find_elements_by_tag_name('li')
        #blogs_list의 자료형은 list가 된다.
        news = driver.find_element_by_class_name('news')
        news_list = news.find_elements_by_xpath('./ul/li')
        print("\n"+"======블로그 검색 결과======"+"\n")
        for post in blogs_list:
                #print(post.text)
                #print('-'*20)
                post_title = post.find_element_by_class_name('sh_blog_title')
                #print(post_title.text)
                print(post_title.get_attribute('title'))
                print(post_title.get_attribute('href') +"\n")
        print("\n"+"======뉴스 검색 결과======"+"\n")
        for page in news_list:
                news_title = page.find_element_by_class_name('_sp_each_title')
                print(news_title.text)
                print(news_title.get_attribute('href') +"\n")
except Exception as e:
        print(e)
finally:
        driver.close()
```