Week1-5

1. 학습정리
    1. **File / Exception / Log Handling**
        1. Exception
                try :
                except 예외 :
                else :
                finally :
            1. 에러가 날 수 있는 부분에서 사용
            2. raise < 예외 타입 >(예외정보)
                1. 필요에 따라 강제로 예외 발생시킴
            3. assert
                1. 조건을 확인해 강제로 에러 발생시킴
                2. 이후에 시간이 너무 오래걸리면 미리 에러 출력하기 위해

```
def get_binary_number(decimal_number : int):
    assert isinstance(decimal_number, int)
    return bin(decimal_number)

print(get_binary_number(10.0))
```

        2. File
            1. with

```
with open("i_have_a_dream.txt", "r") as f:
    contents = f.read()
#     print(contents)
print(f)
```

                - with 블록 안에서만 f가 사용됨
                - 끝나면 바로 close
            2. read
                1. 파일내용을 모두 문자열에 저장
            3. readlines
                1. 줄단위로 구분해 리스트로 만듦
            4. readline
                1. 실행시마다 한 줄 씩 읽음

파이썬의 directory 다루기

os 모듈을 사용하여 Directory 다루기

```
import os
os.mkdir("log")
```

디렉토리가 있는지 확인하기

```
if not os.path.isdir("log"):
    os.mkdir("log")
```

2. pathlib
    1. 디렉토리를 객체로 다룰 수 있음
3. pickle
    1. 파이썬에서 쓰는 객체의 정보를 저장해서 불러와서 사용할 수 있음

Pickle

- 파이썬의 객체를 영속화(persistence)하는 built-in 객체
- 데이터, object 등 실행중 정보를 저장 ➔ 불러와서 사용
- 저장해야하는 정보, 계산 결과(모델) 등 활용이 많음

```
import pickle

f = open("list.pickle", "wb")
test = [1, 2, 3, 4, 5]
pickle.dump(test, f)
f.close()
```

```
f = open("list.pickle", "rb")
test_pickle = pickle.load(f)
print(test_pickle)
f.close()
```

3. Logging Handling
    1. logging
        1. 파이썬 기본 로그 모듈

logging 모듈

- Python의 기본 Log 관리 모듈

```
import logging

logging.debug("틀렸잖아!")
logging.info("확인해")
logging.warning("조심해!")
logging.error("에러났어!!!")
logging.critical ("망했다...")
```

2. 로깅레벨

logging level

Level	개요	예시
debug	개발시 처리 기록을 남겨야하는 로그 정보를 남김	- 다음 함수로 A 를 호출함 - 변수 A 를 무엇으로 변경함
info	처리가 진행되는 동안의 정보를 알림	- 서버가 시작되었음 - 서버가 종료됨 - 사용자 A가 프로그램에 접속함
warning	사용자가 잘못 입력한 정보나 처리는 가능하나 원래 개발시 의도치 않는 정보가 들어왔을 때 알림	- Str입력을 기대했으나, Int가 입력됨 ➔ Str casting으로 처리함 - 함수에 argument로 이차원 리스트를 기대했으나 ➔ 일차원 리스트가 들어옴, 이차원으로 변환후 처리
error	잘못된 처리로 인해 에러가 났으나, 프로그램은 동작할 수 있음을 알림	- 파일에 기록을 해야하는데 파일이 없음 --> Exception 처리후 사용자에게 알림 - 외부서비스와 연결 불가
critical	잘못된 처리로 데이터 손실이나 더이상 프로그램이 동작할 수 없음을 알림	- 잘못된 접근으로 해당 파일이 삭제됨 - 사용자의 의한 강제 종료

logging level

```
import logging

logger = logging.getLogger("main")          Logger 선언
stream_hander = logging.StreamHandler()     Logger의 output 방법 선언
logger.addHandler(stream_hander)            Logger의 output 등록
```

```
logger.setLevel(logging.DEBUG)
logger.debug("틀렸잖아!")
logger.info("확인해")
```

```
logger.setLevel(logging.CRITICAL)
logger.debug("틀렸잖아!")
logger.info("확인해")
```

4. configparser
    1. 프로그램 실행 설정을 파일에 저장
5. argparser
    1. 콘솔창에서 프로그램 실행시 세팅 정보를 저장
    2. 커맨드 라인 옵션

2. **Python data handling**
    1. CSV
        1. 필드를 쉼표(,)로 구분한 텍스트 파일
        2. csv 임포트 해서 객체 만들어서 다룸
        3. delmiter
            1. 구분자
        4. quotechar
            1. 데이터를 싸매는 문자
            2. 일반적으로 ‘
    2. 웹
        1. 정규표현식
            1. import re
            2. re.findall(정규표현식, 콘텐츠)

정규식 기본 문법 #1

문자 클래스 [ ]: [와] 사이의 문자들과 매치라는 의미

예) [abc] ← 해당 글자가 a,b,c중 하나가 있다.
"a", "before", "deep" , "dud", "sunset"

"-"를 사용 범위를 지정할 수 있음

예) [a-zA-z] - 알파벳 전체, [0-9] - 숫자 전체

정규식 기본 문법 - 메타 문자

정규식 표현을 위해 원래 의미 X, 다른 용도로 사용되는 문자

^ $ * + ? { } [ ] \ | ( )

. - 줄바꿈 문자인 ₩n를 제외한 모든 문자와 매치 a[.]b

* - 앞에 있는 글자를 반복해서 나올 수 있음

tomor*ow tomorrow tomoow tomorrrrow

+ - 앞에 있는 글자를 최소 1회 이상 반복

{m,n} - 반복 횟수를 지정 {1,} , {0,} {1,3}

203.252.101.40 [0-9]{1,3} \d{1,3}

? - 반복 횟수가 1회 01[01]?-[0-9]{4}-[0-9]{4}

| - or (0|1){3} ^ - not

2. urllib
    1. 주소로 접속해서 콘텐츠를 가져옴
3. XML
    1. 서로 다른 디바이스간에 데이터 주고 받는데 유용 (컴퓨터 <>폰)
4. JSON
    1. 파이썬 딕셔너리와 완전히 동일
    2. import json

2. 피어세션

1. 과제 코드 리뷰
2. 알고리즘 문제 리뷰
    1. https://leetcode.com/problems/remove-duplicate-letters/
    2. https://leetcode.com/problems/minimum-insertions-to-balance-a-parentheses-string/
3. 강의리뷰