

Day 11 - File Input/Output

파일열기:

```
f = open("파일명", "모드")
```

파일명: 열고자 하는 파일의 이름이나 경로

모드: 파일을 어떻게 열 것인지를 지정

r: 읽기 모드 (기본값)

w: 쓰기 모드 (파일이 있으면 덮어쓰기)


a: 추가 모드 (파일의 끝에 내용을 추가)

b: 바이너리 모드 (텍스트가 아닌 바이너리 데이터를 읽고/쓸 때 사용)

+: 읽기와 쓰기 모드

ex) open('filename', 'mode')

```
[2] file = open('data.txt', 'wt')
    for i in range(10):
        file.write('파일 열기 테스트: ' + str(i) + '\n')
    file.close() # 저장하고 다른사람들이 접근하기위해서 닫아주기
    print('data.txt 파일에 쓰기 완료!')
```

 data.txt 파일에 쓰기 완료!

With 문

```
# close를 굳이 안해도 with 으로 자동으로 해준다. (자동으로 열고 닫기)
with open('./data/word.txt', 'w') as f:
    while True:
        data = input('단어를 입력하세요: ')
        if data.lower() == 'quit':
            break
        f.write(data + '\n')
```

```
단어를 입력하세요: apple
단어를 입력하세요: bannana
단어를 입력하세요: dsadsad
단어를 입력하세요: dsadsad
단어를 입력하세요: quit
```

- close문을 사용하지 않아도 자동으로 닫아주고, 오류 발생시 자동으로 close가 실행된다.

```
with open('./data/word.txt', 'w') as f:
    while True:
        data = input('단어를 입력하세요: ')
        if data.lower() == 'quit':
            break
        f.write(data + '\n')
```

with 문은 `__enter__()`와 `__exit__()` 메서드를 가진 객체와 함께 사용됩니다.

- open이라는 함수는 이 두 가지의 매직 메서드를 가진다.
- 이것을 컨텍스트 매니저라고 한다.

```
myContext = MyContext()
```

```
with MyContext() as resource:
```

- With문을 사용해서 새 객체를 만들 수 있다.

```
class MyContext:
    def __enter__(self):
        print('시작합니다!')
        return "리소스"

    def __exit__(self, exc_type, exc_value, traceback):
        print('끝났습니다!')

with MyContext() as resource:
    print(f'{resource}를 사용 중입니다')
```

시작합니다!
리소스를 사용 중입니다
끝났습니다

IT 강남 F.Class님이 회의에서
파일을 공유했습니다.

- 파일뿐만 아니라 enter와 exit가 구현되는 객체는 다 사용이 가능하다.

파일 읽기:

read(): 파일의 모든 내용을 문자열로 반환
readline(): 파일의 한 줄을 문자열로 반환
readlines(): 파일의 모든 줄을 리스트로 반환