기초 인공지능 프로그래밍

7장 파일 입출력

파일 입출력 (file I/O)

- 키보드 입력을 표준 입력, 모니터 출력을 표준 출력이라 함
- input( ) 함수는 키보드로부터 데이터를 입력 받는 함수
- print( ) 함수는 모니터로 결과를 출력하는 함수(옵션을 변경하여 파일에 출력 가능)
- 하지만, 입력 데이터의 양이 많아지면 손으로 직접 입력하기 어렵고, 출력 데이터의 양도 많아져 파일에 보관해야 할 필요성도 생김
- 본 강의에서는 문자열로 이루어진 파일 입출력을 다룸
- 파일 입출력은 다음과 같은 세 단계를 통하여 이루어짐
    1. 파일 열기 (File Open)
    2. 파일에서 읽거나 또는 파일에 쓰기
    3. 파일 닫기 (File Close)

파일 열기 (file open)

- 형식

```
fp_r = open("in.txt", 'r')      # read mode
fp_w = open("out.txt", 'w')     # write mode
```

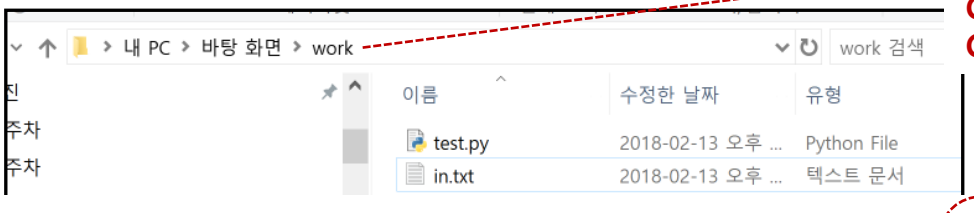
  - **fp_r**, **fp_w** : 파일 정보가 저장되어 있는 파일 객체를 가리키는 변수
  - **in.txt**, **out.txt** : 사용할 파일 이름
  - File Mode : 파일을 어떤 용도로 사용할지 결정

    'r' : 읽기 전용. 존재하지 않는 파일이면 에러 발생

    'w' : 쓰기 전용. 파일이 없으면 빈 파일을 새로 만들고, 기존 있던 파일이면 내용을 모두 지우고 파일을 쓰기 전용으로 open.

    'a' : append의 약자이며, 기존 파일에 덧붙여서 이어서 쓰기 작업을 함

- script 파일과 같은 폴더에 저장되어 있는 데이터 파일을 읽거나 쓰는 경우는 파일 이름만 명시함. 다른 폴더에 있는 데이터 파일인 경우는 전체 경로를 이름과 함께 명시해야 함



마우스 클릭
Ctrl-C (경로 복사)
Ctrl-V (script에 붙이기)

```
fp_r = open("C:WWUsersWWSOGANGWWDesktopWWworkWWin.txt", 'r')
```

W를 문자열에 표현하기 위해서 WW 로 바꾸어주어야 함

파일 읽기

- open() 함수에서 읽기 모드로 반환 받은 파일 객체를 가리키는 변수를 fp_r 이라고 가정
- **fp_r.read( )** : 파일 전체를 하나의 문자열로 읽어 들임
- **fp_r.readlines( )** : 파일을 줄 단위로 읽어 각 줄을 문자열 형식으로 저장, 이들 전체 문자열을 원소로 하는 리스트를 반환
- **fp_r.readline( )** : 한 줄을 문자열 형식으로 읽어 이를 반환
- 파일 전체를 한 줄씩 읽는 방법 (for 반복문 사용하면 편리)

```
for aLine in fp_r :          # 변수 aLine은 fp_r에서 한 줄씩 읽어 들인 문자열
    statements               # 읽어들인 데이터로 처리할 명령어들
```

- 하나의 script 파일에서 필요한 데이터 파일이 여러 개인 경우는 각각 open()을 해서 변수에 저장해야 함

파일 쓰기

- open() 함수에서 쓰기 모드로 반환 받은 파일 객체를 가리키는 변수를 fp_w 이라고 가정
- **fp_w.write(str)** : 인자로 받은 문자열 데이터 하나를 파일에 씀
  - 줄바꿈 문자가 자동으로 삽입되지 않음
  - 줄바꿈이 필요한 경우, 문자열 인자의 마지막에 줄바꿈 문자를 추가해야 함
- **print(*objects, file = fp_w)**
  - 표준 출력인 화면 대신 파일에 쓰기 위해서는 file 인자의 값을 변경해야 함
  - print() 함수는 디폴트로 줄바꿈이 발생
  - 줄바꿈을 하지 않으려면, end 인수 값 변경

```
print(*objects, file = fp_w, end = "")
# file = fp_w  : file 인수값을 기본 값인 표준 출력에서 원하는 파일 객체
                 변수명으로 변경
# end = "" : 출력시 마지막 문자열 인수값을 기본 값인 줄바꿈이 아닌
              빈문자열로 변경
```

파일 닫기

- 파일을 열었다면 마지막에 반드시 파일을 닫아야 함

```
fp_r = open("in.txt", 'r')      # read mode
fp_w = open("out.txt", 'w')     # write mode
........
fp_r.close()
fp_w.close()
```

- 파일이 열려 있는 상태에서는 파일을 지우거나 이름을 바꿀 수 없음
- Python shell을 종료하거나, restart하거나 또는 close method를 호출하여야 파일이 닫혀 파일 지우기 또는 이름 바꾸기 등을 수행할 수 있음
- 따라서, 파일 작업을 마치면 반드시 파일을 닫도록 함
- 파일에 쓰기 작업을 한 후에 닫기를 하지 않으면 쓰기 작업을 한 내용이 저장되지 않음
- open() 한 모든 데이터 파일은 반드시 close() 하여야 함

파일 입출력 예제

- 한 줄씩 읽어 화면에 출력

```
fp = open("in_data.txt", 'r')
for s in fp :                    # 한 줄씩 읽어
    print(s, end = "")           # 화면에 출력
fp.close()
```

하나의 출력 명령어 실행 후 자동 줄바꿈이 발생하지 않도록

in_data.txt(입력 파일)

```
1 3 5
7 3
```

파일에 데이터 입력할 때 첫 라인의 끝에 enter 키를 입력하기때문에 줄바꿈 문자가 데이터 파일에 저장되어 있음

화면 출력

```
1 3 5
7 3
```

print() 함수에서 자동 줄바꿈이 발생하지 않도록 했지만, 기존 데이터 파일의 첫 줄 끝에 저장되어 있는 줄바꿈 문자를 읽어와서 출력하기 때문에 줄바꿈이 발생

파일 입출력 예제

- 입력 파일의 전체 데이터를 출력 파일에 저장 하기(파일 복사)

```
fp_r = open("in_data.txt", 'r')
fp_w = open("out_data.txt", 'w')
s = fp_r.read()          # 한번에 모두 읽어 하나의 문자열로 저장
fp_w.write(s)            # 읽은 문자열 파일에 쓰기
fp_r.close(); fp_w.close()       # 두 파일 닫기
```

또는, 한 줄씩 읽어
출력 파일에 쓰기
하는 방법

in_data.txt

```
1 3 5
7 3
```

out_data.txt

```
1 3 5
7 3
```

```
fp_r = open("in_data.txt", 'r')
fp_w = open("out_data.txt", 'w')
for s in fp_r :          # 한 줄씩 읽어서 쓰기
    fp_w.write(s)        # 또는 print(s, end = "", file = fp_w)

fp_r.close(); fp_w.close()
```

파일 입출력 예제

- 입력 파일의 각 수를 제곱하여 출력 파일에 저장

```
fp_r = open("in_mul.txt", 'r')
fp_w = open("out_mul.txt", 'w')

print("The square results are", file = fp_w)

for s in fp_r :  # 한 줄씩 읽기
   num = list( int(x) for x in s.split() )   # 정수변환

   for i in range(len(num)) :
       sq = num[i] * num[i]
       print("%4d  " %sq, file = fp_w, end = '')
   fp_w.write("₩n")  # 한 줄 처리 후, 줄바꿈

fp_r.close();  fp_w.close()
```

또는 print(file = fp_w)

in_mul.txt

```
10 20 30
40 50 60 70
```

out_mul.txt

```
The square results are
 100   400   900
1600  2500  3600  4900
```