[파일 처리]

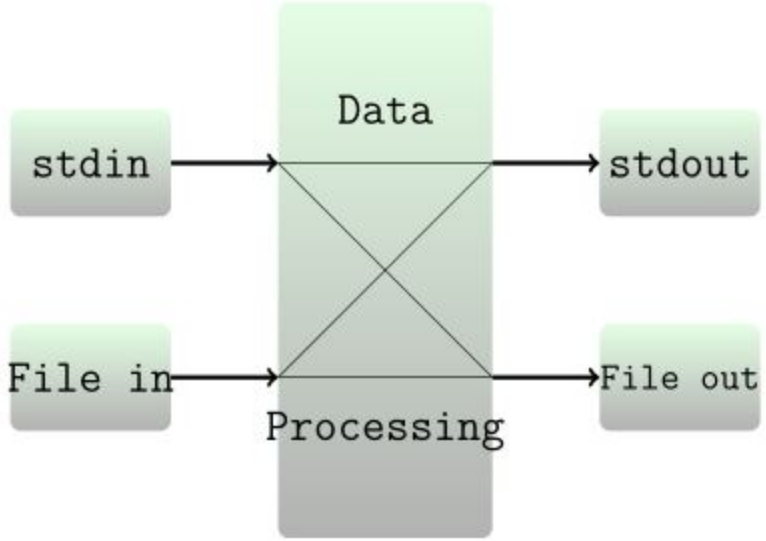
- 파일 생성하기
- 파일에 내용 쓰기
- 파일을 읽는 여러 가지 방법
- 파일에 새로운 내용 추가하기
- with ~ as 문과 함께 사용하기

표준 입출력과 파일 입출력

- 지금까지는 프로그램을 실행할 때 표준입력(키보드)에서 입력을 받고 실행결과는 표준출력(모니터 화면)으로 보여줌
- 입력해야 할 내용이 너무 많거나, 또는 얼마 안되어도 프로그램을 실행할 때마다 매번 입력하는 것은 귀찮은 일
- 또는 프로그램 실행의 결과가 너무 길게 출력되어 모니터의 한 화면을 넘어가게 되거나, 나중에 실행 결과를 다시 확인해 보고 싶을 때에는 내용을 파일에 먼저 저장하고 싶을 것
- 이런 경우 파일 입출력을 사용하면 좋음

입출력 데이터의 흐름

- 지금까지 사용해왔던 표준입력(키보드)과 표준출력(모니터화면) 대신
- 파일로부터 입력을 받거나, 처리 결과를 파일에 기록할 수 있다
- 설명한 데이터 처리 과정을 아래 그림에 표시 (입출력 2개씩 총 4개의 흐름이 있음)



파일 입출력 처리의 3 단계 절차

1. open() 함수로 파일이름과 처리목적(읽기 또는 쓰기)를 지정해 사용하려는 파일을 연다.
2. 입출력에 필요한 읽기/쓰기 함수를 사용해 원하는 처리를 실행
3. 파일의 사용이 끝나면, close() 함수를 호출해 사용하던 파일을 닫음

파일의 열기와 모드

파일을 열 때 사용하는 open() 함수는 다음과 같이 사용

```
fn = open("filename", "mode")
```

- fn은 프로그램 안에서 파일을 가리키는 파일 변수의 이름이다. 이 이름을 이용해 특정 파일로부터 읽거나 쓸 수 있다.
- filename은 읽거나 쓸 대상 파일의 이름(필요할 경우 드라이브와 디렉토리 경로를 포함 가능)
  - 현재 위치를 기준으로 한 상대 경로의 지정도 가능.
  - 경로없이 파일 이름만 주어질 경우 현재 디렉토리에서 파일을 찾는다.
  - 파일명의 대소문자 구분 여부는 운영체제에 따라 다르기 때문에 그에 맞추어 사용함
- mode는 어떤 목적으로 파일을 열 것인지를 정의하는 `모드(mode)`를 지정해야 함(다음 참조)

여러 가지 파일 모드

- 파일의 여러 사용 모드

파일 모드	모드 이름	설명
r	파일 읽기	해당 파일이 존재해야 한다. 만약 파일이 없을 경우 에러
w	파일 쓰기	파일이 이미 있으면 삭제되고, 없으면 새 파일을 만든다.
a	추가모드	내용이 있는 경우 파일의 맨 끝에 덧붙여 쓴다. 없으면 새 파일 생성
r+	읽기쓰기 모드	파일이 없으면 에러 발생

파일을 쓰기 모드로 열기

- 다음과 같이 임의의 파일을 쓰기 모드로 열기 실행.

  단 현재 폴더에 같은 이름의 파일이 존재하면 지워지니 주의해서 실행 요망!

```
# newfile.py
f = open("newfile.txt", 'w')
f.close()
```

- 0 바이트 크기의 파일이 생성되었는 지 반드시 확인

파일 경로와 슬래시(/)

- 파이썬 코드에서 파일 경로를 표시할 때 "C:/my/newfile.txt"처럼 슬래시(/)를 사용
- 만약 역슬래시(\)를 사용한다면
  - "C:\\my\\newfile.txt"처럼 역슬래시를 2개 사용하거나
  - r"C:\my\newfile.txt"와 같이 문자열 앞에 r 문자(raw string)를 덧붙여 사용

쓰기 함수

- 2종류의 쓰기 함수가 있다
- write() 함수는 인수로 문자열만 사용할 수 있지만,
- writelines() 함수는 문자열이나 리스트를 인수로 사용할 수 있다.
- write() 쓰기 함수의 사용: 다음을 실행 후 생성된 파일의 내용 확인

```
# newfile2.py
f = open("newfile2.txt", 'w')
for i in range(1,11):
    data = 'line %d입니다\n' %i
    f.write(data)
f.close()
```

읽기 함수

- 파일에서 데이터를 읽을 때 3 종류의 읽기 함수 : read(), readline(), readlines() 함수
- read() 함수는 다음 예처럼 읽을 때
  매개변수가 없으면 파일의 전체 내용을 읽는다.
  n개의 글자를 읽어 들이고 싶으면 read(n)처럼 매개변수를 준다. 다음 프로그램은 read(7)과 같이 7개의 문자를 읽는 예이다.
- readlines() 함수 : 파일의 전체 라인을 읽고 싶을 경우 사용
  - 이 함수는 파일의 각 줄을 읽어 리스트에 저장하는 데, 파일의 한 줄이 리스트 안의 한 항목이 된다. 각 항목의 끝에는 줄바꿈기호인 \n이 있다.
- readline() 함수(s가 없음에 주의) : 파일에서 한 줄씩 읽을 필요가 있을 때 이용
  파일에 몇 줄의 데이터가 있는 지 알 수 없기 때문에 반복문을 이용해 한 줄씩 읽는 데, rstrip() 함수를 이용해 줄의 끝에 있는 줄바꿈기호(\n)를 없앨 수도 있다.

with ~ as 사용

- open() 함수를 with ~ as문과 같이 사용하면 close() 함수를 호출하지 않아도 파일이 항상 닫힘
- with ~ as문을 사용하는 방법

```
with open(파일이름) as 파일객체:
    파일 읽기 또는 쓰기 함수 사용
    필요한 다른 동작
    # 파일 닫기 불필요
```

- 사용 예제

```
with open('testFile.txt', 'r') as file:
    str = file.read()
    print(str)
    # file.close() # with ~ as를 사용하면 이 문장 불필요
```

텍스트 파일과 이진파일의 차이

- 지금까지 사용한 파일은 텍스트 파일이고, 이 파일의 내용은 문서편집기(notepad) 등과 같은 텍스트 에디터 또는 print() 함수로 내용을 확인할 수 있다.
- 컴퓨터가 다루는 파일 중에는 이미지, 음악 파일 등은 텍스트 파일이 아니고 이진파일(binary file)이다.
- 이진파일을 확인하려면 별도의 프로그램(이미지 편집기, 음악 플레이어 프로그램)이 필요

이진파일의 처리

- 이진파일을 사용하려면 파일 모드의 뒤에 `"b"`를 추가
- 이진파일의 복사 예제

```
inFp, outFp = None, None
inStr = ""
inFp = open("./test.jpg", "rb")
outFp = open("./testCopied.jpg", "wb")

while True :
    inStr = inFp.read(1)
    if not inStr :
        break
    outFp.write(inStr)
inFp.close()
outFp.close()
print("image file copied.")
```

파일의 접근 방법

- 순차접근(sequential access) 방법 :
  - 지금까지의 파일 처리 방법
  - 파일의 처음부터 순서대로 데이터를 읽거나 쓰는
- 임의접근(random file access)방법 :
  - 간혹 파일의 특정 위치로 가서 데이터를 읽거나 써야 할 필요가 있을 때 사용하는 방법
  - 추가로 필요한 함수가 있음

파일의 임의 접근 함수

- tell() 함수 :
  파일의 맨 처음부터의 위치를 바이트 수(이진모드)나 글자수(텍스트 모드)로 알 수 있는 함수
- seek(offset, from) 함수:
  파일 내에서의 위치를 바꾸고 싶을 때 사용. 위치는 기준점 from에 offset을 더한다.
  - from 값이 0이면 파일의 처음부터
  - 1이면 파일의 현재위치부터
  - 2이면 파일의 맨 끝부터
  - from 값이 생략되면 기본값이 0이어서 파일의 맨 처음부터로 간주. 단 텍스트 파일에서는 파일 맨 처음부터 찾는 것만 허용됨