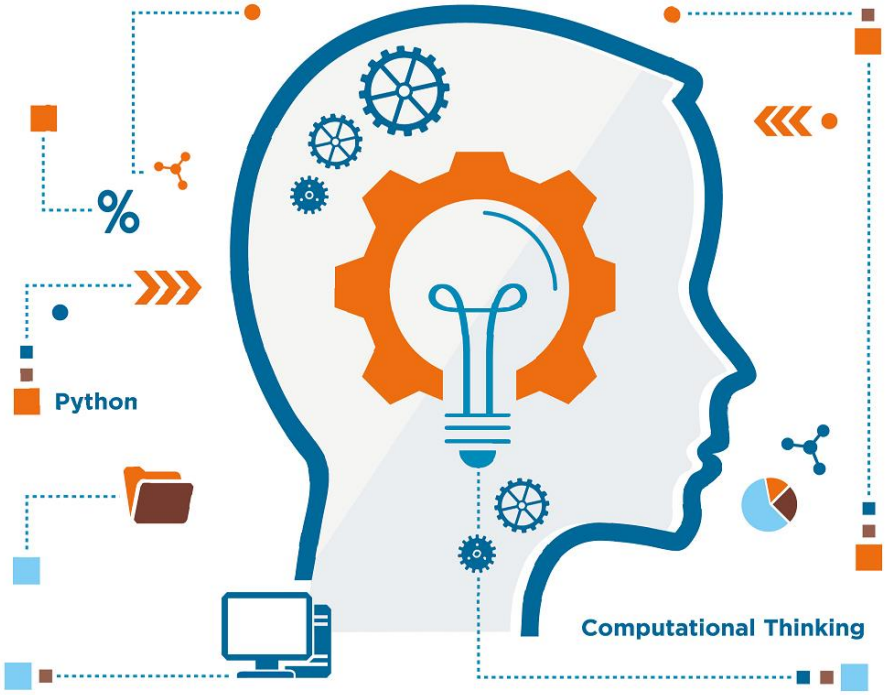


[Python]

Python으로 배우는

소프트웨어 원리

Chapter 09. 문자열과 파일

목차

http://github.com/dndxor/BCPy

01

문자열

01. 문자열

[일상 생활 속 문자열]

- 텍스트(text)라고도 불리는 문자열(string)은 메시지 전송이나 문서 편집 등 일상생활에서 많이 사용하는 데이터형이다.
- 파이썬은 문자열을 처리하기 위한 다양한 기능이 있다.

그림 9-1 일상생활에서의 문자열 사용

01. 문자열

I. 문자열의 구조

- 텍스트라고도 불리는 문자열은 메시지 전송이나 문서 편집 등 일상생활에서 많이 사용하는 데이터형
- 컴퓨터는 숫자 0과 1로 모든 데이터를 처리하지만, 사용자의 편의를 위해 전달하려는 값을 **텍스트**로 변환하여 보여줘야 함
- 문자열은 하나 이상의 **문자**가 끈에 묶인 것처럼 차례대로 **연결**되어 있는 구조
  - 문자열을 구성하는 각 **문자**는 **ASCII 코드**(8bit)로 표현된다.

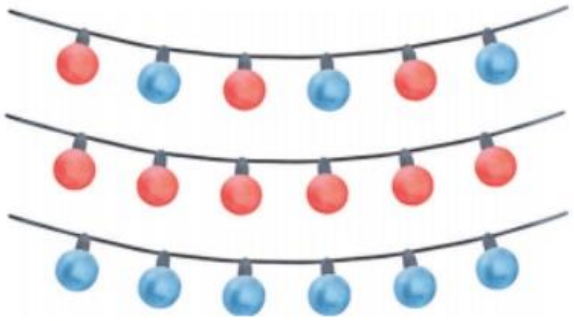
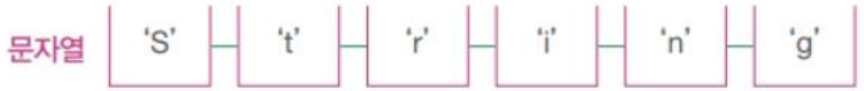


그림 9-2 문자열의 구조



01. 문자열

I. 문자열의 구조

- 문자열은 큰따옴표(“”)나 작은따옴표(‘’)를 사용해서 값을 정의할 수 있고,
- **str( )** 함수를 사용하면 숫자를 문자열로 변환
- 문자열은 리스트나 튜플처럼 대괄호 안에 [0]부터 시작하는 **인덱스**를 사용하고, 마지막 글자부터 번호를 매기는 음수 인덱스도 이용 가능

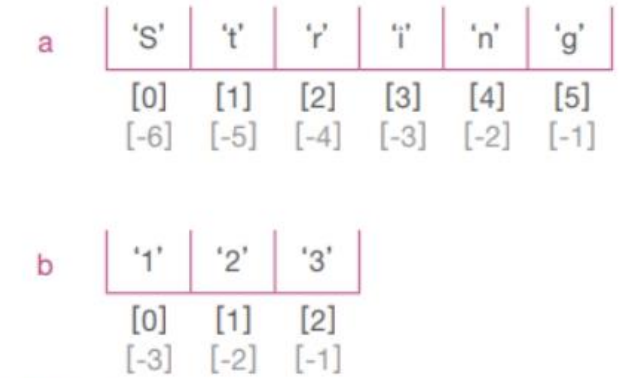


그림 9-3 문자열의 정의와 변환

```
>>> a = "String"          # 문자열 정의

>>> b = str(123)          # 숫자를 문자열로 변환
>>> b
'123'
>>> type(b)
<class 'str'>
```

✓ **TIP** 123과 ‘123’은 숫자와 문자열로 구분되는 완전히 다른 값, 문자열 ‘123’은 3개의 문자가 연결된 구조(‘1’+‘2’+‘3’)이며, 나눗셈(/, //, %)이나 뺄셈(-)과 같은 연산 불가

01. 문자열

I. 문자열의 구조

여기서 잠깐 **문자열을 리스트로 변환하기**

- 문자열을 리스트형으로 변환하면, 문자열을 구성하는 각각의 문자가 리스트의 항목이 됨
- 문자열은 리스트나 튜플, 딕셔너리처럼 항목을 반복문으로 하나씩 처리할 수 있는 이터러블 데이터형

```
>>> list('String')
['S', 't', 'r', 'i', 'n', 'g']
```

```
a = "Programming"
a
'Programming'
a[1]
'r'
for i in range(len(a)):
    print(a[i], end='')


Programming
for x in a:
    print(x, end='')


Programming
type(a)
<class 'str'>
```

01. 문자열

I. 문자열의 구조

실습 9-1 문자열 만들기

① 문자열을 만들고, 연산자를 사용해 문자열의 연결(+)과 문자열의 반복(*)을 실행

```
>>> a = 'Hello'
>>> (a + "~~") * 2
'Hello~~Hello~~'
```

② 큰따옴표(""")나 작은따옴표(''')를 3회 연속 사용하면 줄바꿈 문자('₩n')가 포함된 문자열을 만들 수 있음

```
>>> b = """Hello,              # 콤마(,) 뒤에서 Enter를 누르고,
students~"""                   # 다음 줄에서 나머지 내용을 입력
>>> b                          # 줄바꿈 문자가 포함된 문자열
'Hello,\nstudents~'
>>> len(b)                     # 줄바꿈 문자도 전체 길이에 포함
16
>>> print(b)
Hello,
students~
```

01. 문자열

II. 문자열 사용법

실습 9-2 문자열 슬라이싱과 찾기

① 문자열을 정의하고 인덱스의 범위를 설정해 **슬라이싱**

```
>>> a = "Hello, students~"
>>> a[0:5]    # 앞에서 5개 문자 추출, 'a[:5]'와 동일
'Hello'
>>> a[-5:]    # 뒤에서 5개 문자 추출
'ents~'
```

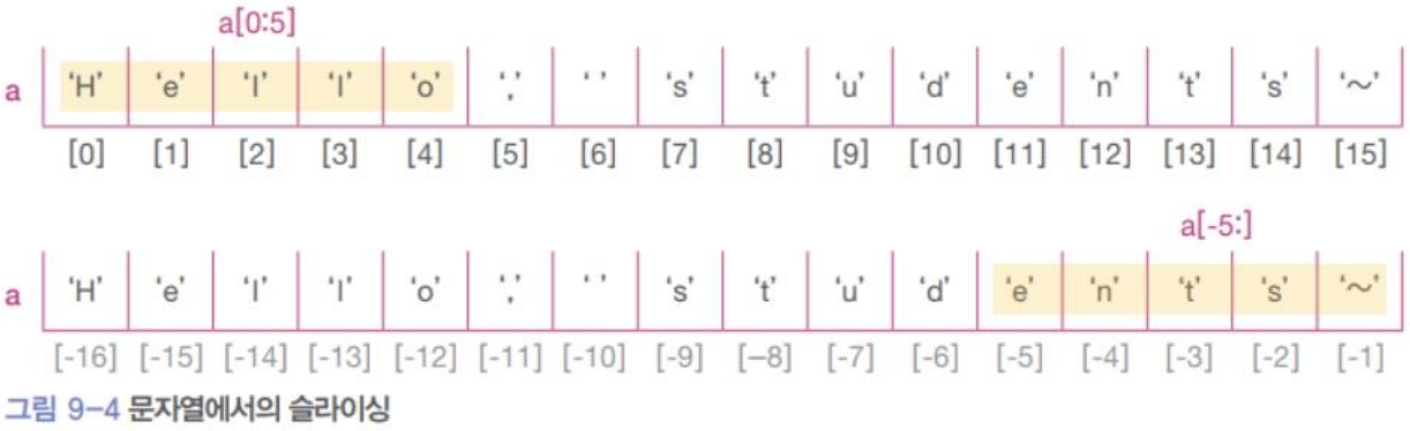


그림 9-4 문자열에서의 슬라이싱

II. 문자열 사용법

실습 9-2 문자열 슬라이싱과 찾기

② 문자열에 특정 문자가 있는지 확인(in)하거나 **index( )**와 **find( )**를 이용하여 문자의 위치를 탐색

```
>>> 'stu' in a              # 문자열 a에 'stu'가 있는지 검사
True
>>> a.index('s')            # 문자열 a에서 's'의 위치 찾기
7
>>> a.find('s')             # 문자열 a에서 's'의 위치 찾기
7
>>> a.index('xy')           # 문자열 a에서 'xy'의 위치 찾기, 없으면 오류 발생
Traceback (most recent call last):
    File "<pyshell#4>", line 1, in <module>
        a.index('xy')
ValueError: substring not found
>>> a.find('xy')                    # 문자열 a에서 'xy'의 위치 찾기, 없으면 -1 반환
-1
```

✓ **TIP** find()를 사용해서 위치를 검색하면 찾는 문자가 없을 때 오류 발생 대신 –1이 출력

01. 라이브러리 개념

I. 모듈과 라이브러리

여기서 잠깐 **이스케이프 문자**

- 문자열에 줄바꿈('₩n')이나 탭 문자('₩t')가 포함되면 print() 함수로 출력할 때 지정한 형식으로 표시
- 줄바꿈이나 탭 문자가 이스케이프 문자
- 역슬래시와 미리 정의된 문자를 조합한 형 태로 출력 형식을 지정하기 위해 사용

```
>>> c = "Hello~~\t\tHello~~"                    # 탭('\t')이 두 번 출력되는 문자열
>>> print(c)
Hello~~         Hello~~
>>> print('만나면 \"안녕~\" 이라고 한다.')      # 큰따옴표('\"')가 출력되는 문자열
만나면 "안녕~" 이라고 한다.
```

01. 문자열

II. 문자열 사용법

실습 9-3 문자열 다루기

① 문자열의 대소문자 변환과 첫 글자만 대문자로 변환하는 기능을 사용

```
>>> a = "Hello, students~"
>>> a.lower()                    # 소문자로 변환
'hello, students~'
>>> a.capitalize()               # 첫 글자만 대문자로 변환
'Hello, students~'
```

② 특정 문자를 바꾸거나, 분리 혹은 조합하는 기능을 사용

```
>>> b = a.replace(' ', '/')      # 공백(' ')을 슬래쉬('/')로 바꾸기
>>> b
'Hello,/students~'
>>> b.split('/')                 # 문자열 b를 '/'를 기준으로 분할
['Hello,', 'students~']
>>> c = '_'
>>> c.join(a)                    # 문자열 a와 언더스코어('_')를 조합
'H_e_l_l_o_,_ _s_t_u_d_e_n_t_s_~'
```

II. 문자열 사용법

[실습] split(), join()

◆ 문자열을 분리하거나 결합할 때 사용

- split() : 문자열에서 **분리 문자** 기준으로 분리하여 리스트로 만듦

  => 분리 문자 생략 시 공백을 분리 문자로 적용

- " ".join(list) : 리스트에 문자열들을 **연결 문자**로 결합하여 하나의 문자열로

```
a = "Python Programming is Good!"
words = a.split()
words
['Python', 'Programming', 'is', 'Good!']
b = ' '.join(words)
b
'Python Programming is Good!'
```

II. 문자열 사용법

실습 9-3 문자열 다루기

③ 문자열의 앞뒤에 불필요한 값이 들어있을 때 제거하거나, 문자열을 정렬하는 기능을 사용

```
>>> d = ".....I have a dream....."
>>> d.strip('.')              # 문자열의 앞뒤에 있는 '.'를 제거
'I have a dream'
>>> d.lstrip('.')             # 앞쪽(left)에 있는 '.'를 제거
'I have a dream.....'
>>> d.center(30, '_')         # 전체 30자리 중 빈칸은 '_'로 채우고 가운데 맞춤
'___.....I have a dream.....___'
```

✓ **TIP** 문자열을 분할할 때 공백(' ')을 기준으로 한다면, 'b.split()'처럼 인수를 입력하지 않아도 상관 없음

✓ **TIP** 빈칸 채우기 없이 가운데 맞춤만 하는 경우 'd.center(30)'처럼 전체 자릿수만 지정

II. 문자열 사용법

[실습] strip(), lstrip(), rstrip(), center()

◆ 문자열의 양 끝에 있는 특정 문자를 제거

- strip() : 문자열에서 **제거할 문자**를 양쪽 끝에서 제거
- lstrip(), rstrip() : 리스트에서 왼쪽 끝 기준으로 또는 오른쪽 기준으로 제거
- center(n) : 문자열을 n자리수를 확보하여 중앙에 위치

```
a = "  Programming!  "
a.strip(' ')
'Programming!'
a.rstrip(' ')
'  Programming!'
a.lstrip(' ')
'Programming!  '
a = a.strip(' ')
a
'Programming!'
a = a.strip('!')
a
'Programming'
```

```
a
'Programming'
a.center(30, '#')
'#########Programming##########'
a.center(30)
'          Programming           '
```

III. 문자열의 활용

실습 9-4 단어 빈도수 조회하기 code09-04.py

- 시를 읽고 특정 문자의 개수가 몇 번 나타나는지 찾아주는 프로그램

① 문자열 변수에 시를 저장해 두고 내용을 먼저 출력하고, 찾으려는 단어를 입력하면 빈도를 측정하여 출력

그림 9-5 단어 빈도수를 출력하는 프로그램 실행 순서

III. 문자열의 활용

실습 9-4 단어 빈도수 조회하기 code09-04.py

②

```
01 poem = """흔들리며 피는 꽃
02 도종환/
03
04 흔들리지 않고 피는 꽃이 어디 있으랴
05 이 세상 그 어떤 아름다운 꽃들도
06 다 흔들리면서 피었나니
07 흔들리면서 줄기를 곧게 세웠나니
08 흔들리지 않고 가는 사랑이 어디 있으랴
09 젖지 않고 피는 꽃이 어디 있으랴
10 이 세상 그 어떤 빛나는 꽃들도
11 다 젖으며 젖으며 피었나니
12 바람과 비에 젖으며
13 꽃잎 따뜻하게 피웠나니
14 젖지 않고 가는 삶이 어디 있으랴
15 """
16
17 print(poem[:55] + '\n...중략...\n')        # 시의 앞쪽 일부만 출력
18 word = input("검색할 단어 입력 : ")
19 print("\'{0}\'이 사용된 횟수 = {1}".format(word, poem.count(word)))
```

III. 문자열의 활용

실습 9-4 단어 빈도수 조회하기 code09-04.py

③

```
흔들리며 피는 꽃
도종환 /

흔들리지 않고 피는 꽃이 어디 있으랴
이 세상 그 어떤 아름다운 꽃들도
...중략...

검색할 단어 입력 : 꽃
'꽃'이 사용된 횟수 = 6
```

III. 문자열의 활용

[실습] '흔들리며 피는 꽃' 분석

- 문자열을 분리하거나 결합할 때 사용
  - poem 문자열이 몇 라인으로 구성되었는지 라인 수를 출력하시오.
    > 추출된 각 문자열 라인도 함께 출력하시오.
  - poem 문자열이 몇 단어로 구성되었는지 단어 수를 출력하시오.
    > 추출된 각 단어도 함께 출력하시오.

```
poem = """흔들리며 피는 꽃
도종환/

흔들리지 않고 피는 꽃이 어디 있으랴
이 세상 그 어떤 아름다운 꽃들도
다 흔들리면서 피었나니
흔들리면서 줄기를 곧게 세웠나니
흔들리지 않고 가는 사랑이 어디 있으랴
젖지 않고 피는 꽃이 어디 있으랴
이 세상 그 어떤 빛나는 꽃들도
다 젖으며 젖으며 피었나니
바람과 비에 젖으며
꽃잎 따뜻하게 피웠나니
젖지 않고 가는 삶이 어디 있으랴
"""
```

```
>> 15 Lines

흔들리며 피는 꽃
도종환/

흔들리지 않고 피는 꽃이 어디 있으랴
이 세상 그 어떤 아름다운 꽃들도
다 흔들리면서 피었나니
흔들리면서 줄기를 곧게 세웠나니
흔들리지 않고 가는 사랑이 어디 있으랴
젖지 않고 피는 꽃이 어디 있으랴
이 세상 그 어떤 빛나는 꽃들도
다 젖으며 젖으며 피었나니
바람과 비에 젖으며
꽃잎 따뜻하게 피웠나니
젖지 않고 가는 삶이 어디 있으랴

>> 57 Words

흔들리며
피는
꽃
도종환/
흔들리지
않고
```

III. 문자열의 활용

실습 9-5 기상 캐스터 만들기 code09-05.py

- 몇 가지 데이터를 입력하면 날씨 정보를 문장으로 만들어 출력

① 날씨는 맑음과 흐림으로 구분 하고, 기온(섭씨 기준)은 10도 이하면 쌀쌀함으로, 그 이외는 따뜻함으로 구분하여 날씨 안내

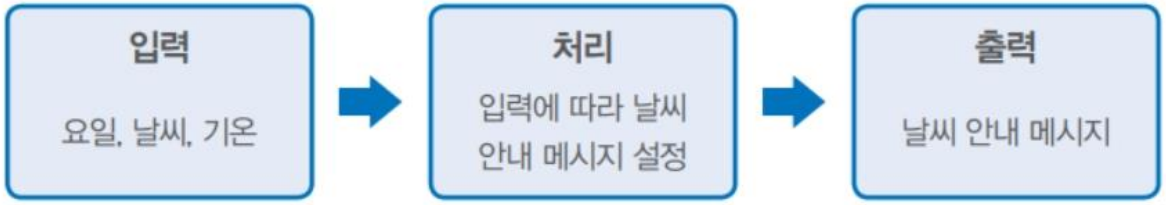


그림 9-7 기상 캐스터 만들기 실행 순서

②

```
01  weekday = input("요일 : ")
02  weather = int(input("날씨 1(맑음), 2(흐림) : "))
03  temp = int(input("기온 : "))
04  cold = ""
05
06  if weather == 1 :
07      weather = "맑습니다"
```

III. 문자열의 활용

실습 9-5 기상 캐스터 만들기 code09-05.py

②

```
08 else :
09     weather = "흐립니다"
10
11 if temp <= 10 :
12     cold = "쌀쌀한"
13 else :
14     cold = "따뜻한"
15
16 print("\n★오늘의 날씨★")
17 print("{0}요일인 오늘은 전국이 {1}. 이날 기온은 {2}도로 관측됩니다. \
18 {3} 출근길이 될 것으로 예상되므로 알맞은 옷차림을 미리 준비하시기 바랍니다. \
19 전국 대부분 지역에서 낮과 밤의 기온 차가 큰 편이므로, \
20 건강관리에 유의하시기 바랍니다. \
21 오늘 {0}요일도 편안한 하루 되시고요~".format(weekday, weather, temp, cold))
```

III. 문자열의 활용

실습 9-5 기상 캐스터 만들기 code09-05.py

③

```
요일 : 금
날씨 1(맑음), 2(흐림) : 1
기온 : 10

*오늘의 날씨*
금요일인 오늘은 전국이 맑습니다. 이날 기온은 10도로 관측됩니다. 쌀쌀한 출근길이 될 것으로 예상되므로 알맞은 옷차림을 미리 준비하시기 바랍니다. 전국 대부분 지역에서 낮과 밤의 기온 차가 큰 편이므로, 건강관리에 유의하시기 바랍니다. 오늘 금요일도 편안한 하루 되시고요~
```

```
요일 : 월
날씨 1(맑음), 2(흐림) : 2
기온 : 15

*오늘의 날씨*
월요일인 오늘은 전국이 흐립니다. 이날 기온은 15도로 관측됩니다. 따뜻한 출근길이 될 것으로 예상되므로 알맞은 옷차림을 미리 준비하시기 바랍니다. 전국 대부분 지역에서 낮과 밤의 기온 차가 큰 편이므로, 건강관리에 유의하시기 바랍니다. 오늘 월요일도 편안한 하루 되시고요~
```

III. 문자열의 활용

실습 9-6 단어 사전 만들기 code09-06.py

- 입력된 문장을 단어별로 분리하여 사전 리스트에 등록하는 과정을 무한 반복하고, 문장에 '0'이 입력되면 반복을 종료
- 프로그램을 종료하기 전에 리스트에 등록된 모든 단어를 출력

① 순서

그림 9-8 단어 사전 만들기 프로그램 실행 순서

② 사전 리스트에는 기존에 등록되지 않은 새로운 단어만 등록해야 하고, 등록된 단어를 모두 출력할 때 보기 편하게 먼저 정렬

III. 문자열의 활용

실습 9-6 단어 사전 만들기 code09-06.py

②

```
01 sentence = ""                          # 입력한 문장
02 words = []                             # 입력한 문장을 분리해 두는 단어 리스트
03 dic = []                               # 사전 리스트
04 print("문장의 단어를 등록합니다(종료는 0 입력).")
05 while True :
06     sentence = input("문장 입력 : ")
07     if sentence == '0' :
08         break
09     words = sentence.split()           # 입력한 문장을 단어 리스트에 분리해서 저장
10     for x in words :
11         if x not in dic :              # 새로운 단어인지 먼저 검사한 후 등록
12             dic.append(x)
13
14 print("*****단어 사전*****")
15 for x in sorted(dic):                  # 리스트를 정렬해서 출력
16     print(x)
```

III. 문자열의 활용

실습 9-6 단어 사전 만들기 code09-06.py

③

```
문장의 단어를 등록합니다(종료는 0 입력).
문장 입력 : i like dog
문장 입력 : you like cat
문장 입력 : she like rabbit
문장 입력 : 0
*****단어 사전*****
cat
dog
i
like
rabbit
she
you
```

02

파일

02. 파일

I. 파일의 개념

- 프로그램 실행 중에 만들어진 데이터를 종료 후에도 계속 사용하고 싶다면 파일에 저장해야 함
  - ➢ 프로그램이 실행되고 있다는 의미는 RAM과 CPU에서 실행되고 있음을 의미
  - ➢ RAM과 CPU는 휘발성 장치이므로 비휘발성 장치인 보조기억장치로 영구 저장 필요

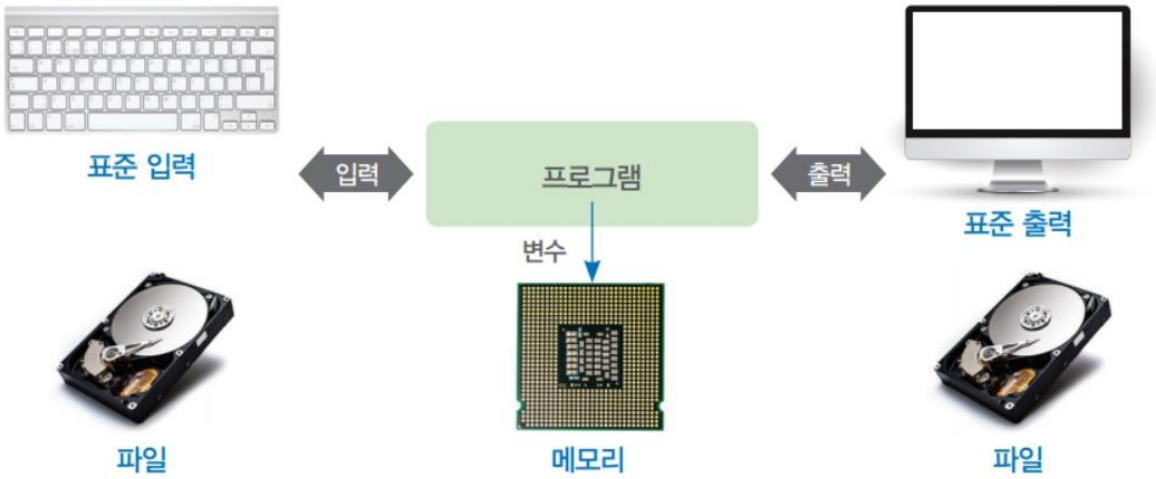


그림 9-10 표준 입출력과 파일 입출력

02. 파일

II. 파일의 사용법

- 프로그램에서 파일을 읽고 쓰려면 '열기 → 읽기/쓰기 → 닫기' 과정으로 진행
- open( ) 사용할 파일을 열거나 새롭게 생성(객체가 생성)
- read( )나 write( )로 파일을 읽거나 쓰기
- close( ) 함수로 파일을 닫으면 객체를 통한 파일 작업 종료

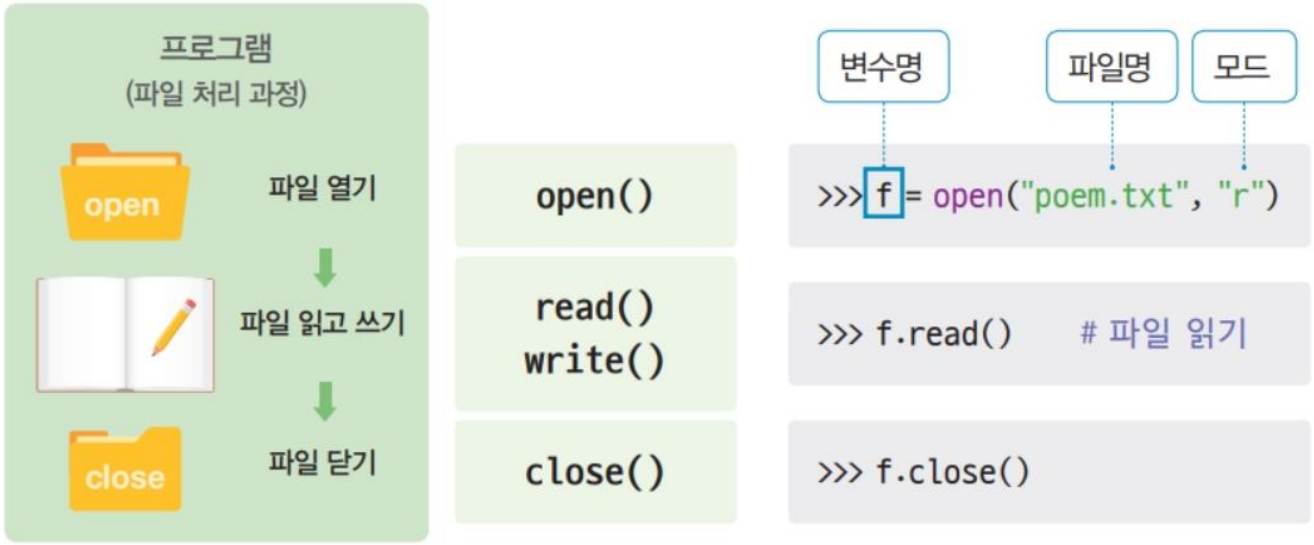


그림 9-11 파일 처리 과정

02. 파일

II. 파일의 사용법

- 파일 열기에 사용하는 open( ) 함수의 마지막 인수인 모드mode는 파일을 읽고 쓰는 방법을 지정

표 9-1 모드의 종류와 특징

모드 표기	모드 종류	기능
"r"	read	파일에 저장된 데이터 읽기
"w"	write	파일에 있던 기존 데이터를 지우고 새로운 데이터로 변경 파일이 없는 경우 새로 생성
"a"	append	파일의 끝에 데이터를 추가 파일이 없는 경우 새로 생성
"r+"	read+write	기존의 파일을 읽고 데이터를 수정
"a+"	read+append	파일을 읽고 파일 끝에 데이터를 추가 파일이 없으면 새로 생성
"t"	text	파일을 문자 단위로 처리 기본(default)값으로 정해져 있어 특별히 표시할 필요가 없음
"b"	binary	파일을 비트 단위로 처리하며 바이너리 파일을 읽거나 쓸 때 모드 표기에 추가해야 함("rb", "wb" 등)

✓ **TIP** 모드 표기에는 작은따옴표('') 사용 가능

II. 파일 사용법

[실습] 리스트로 text 파일 생성

```
poem = """흔들리며 피는 꽃
도종환/

흔들리지 않고 피는 꽃이 어디 있으랴
이 세상 그 어떤 아름다운 꽃들도
다 흔들리면서 피었나니
흔들리면서 줄기를 곧게 세웠나니
흔들리지 않고 가는 사랑이 어디 있으랴
젖지 않고 피는 꽃이 어디 있으랴
이 세상 그 어떤 빛나는 꽃들도
다 젖으며 젖으며 피었나니
바람과 비에 젖으며
꽃잎 따뜻하게 피웠나니
젖지 않고 가는 삶이 어디 있으랴
"""
```

```
plist = poem.split('₩n')          #'₩n'으로 줄 단위 추출
print(plist)
#줄 단위 쓰기
fd = open("poem.txt", "w")        #쓰기 파일 열기
for x in plist:
    fd.write(x+'₩n')              #(쓰기) 제거된 '₩n'을 복원
fd.close()                        #파일 닫기
```

```
#줄 단위 읽기
fd = open("poem.txt", "r")        #읽기 파일 열기
for x in fd.readlines():          #줄 단위로 읽기
    print(x)
fd.close()                        #파일 닫기
```

II. 파일의 사용법

실습 9-7 텍스트 파일을 읽고 화면에 출력하기

- 텍스트 파일을 읽을 때는 read( ), readline( ), readlines( ) 메소드를 사용할 수 있음

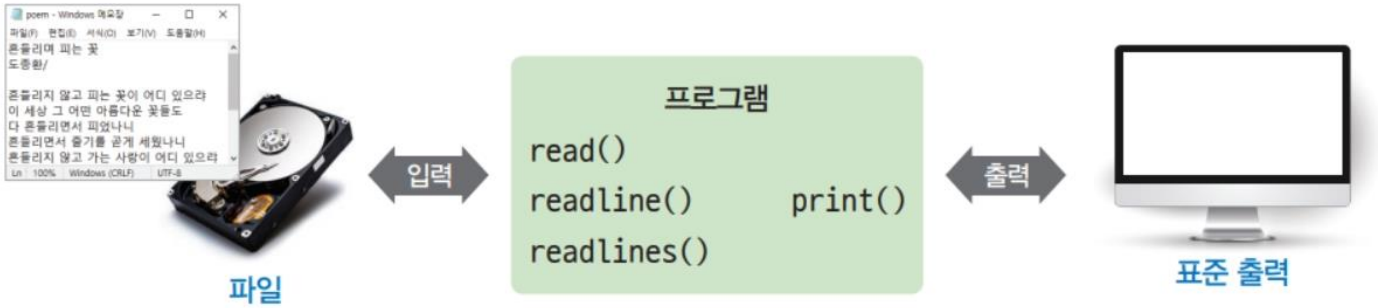


그림 9-13 텍스트 파일을 읽고 출력하는 과정

표 9-2 텍스트 파일 읽기 메소드의 종류

함수명	설명	반환값 형태
read()	처음부터 끝까지 문자 단위로 읽어서 문자열로 반환, 바이너리 파일에도 사용	'모든 내용이 들어간 문자열'
readline()	현재 위치에서 한 줄 읽어서 문자열로 반환	'한 줄 내용이 들어간 문자열\n'
readlines()	현재 위치에서 끝까지 한 줄씩 읽어서 리스트로 반환	['한 줄 문자열\n', '한 줄 문자열\n']

02. 파일

II. 파일 사용법

[실습] read(), readline(), readlines()

```
fd = open("poem.txt", "r")
x = fd.read()              # 전체를 읽어서 반환
print(x)
fd.close()
```

```
흔들리며 피는 꽃
도종환/

흔들리지 않고 피는 꽃이 어디 있으랴
이 세상 그 어떤 아름다운 꽃들도
다 흔들리면서 피었나니
흔들리면서 줄기를 곧게 세웠나니
흔들리지 않고 가는 사랑이 어디 있으랴
젖지 않고 피는 꽃이 어디 있으랴
이 세상 그 어떤 빛나는 꽃들도
다 젖으며 젖으며 피었나니
바람과 비에 젖으며
꽃잎 따뜻하게 피웠나니
젖지 않고 가는 삶이 어디 있으랴
```

```
fd = open("poem.txt", "r")
x = ' '
while x != '':
    x = fd.readline()      #한 줄만 읽어서 반환
    print(x, end='')
fd.close()
```

```
fd = open("poem.txt", "r")
for x in fd.readlines():    #전체를 한 줄씩 읽어서 반환    #fd.readlines() 대신 fd 사용 가능
    print(x, end='')
fd.close()
```

II. 파일의 사용법

실습 9-7 텍스트 파일을 읽고 화면에 출력하기

① 파일을 열고 read( ) 메소드로 'poem.txt' 파일을 처음부터 끝까지 읽기

```
>>> f = open("poem.txt", "r")
>>> print(f.read())                    # 파일을 처음부터 끝까지 읽어서 화면에 출력
흔들리며 피는 꽃
도종환 /

흔들리지 않고 피는 꽃이 어디 있으랴
이 세상 그 어떤 아름다운 꽃들도
다 흔들리면서 피었나니
... 중략 ...
꽃잎 따뜻하게 피웠나니
젖지 않고 가는 삶이 어디 있으랴
```

② 파일의 끝까지 데이터를 읽고 나면 읽을 값이 없으므로, 빈 문자열('')을 반환

```
>>> f.read()                  # 파일 끝까지 읽은 후, 또 읽으면 빈 문자열을 반환
''

>>> f.tell()                  # 현재 파일 포인터의 위치 확인(파일의 끝)
506
```

II. 파일의 사용법

여기서 잠깐 **파일의 위치 확인**

- 읽으려는 파일이 실행 중인 코드와 동일한 위치에 있는 경우 파일명("파일명")만 적으면 되지만, 그렇지 않은 경우 파일이 저장된 경로를 입력해야 함
- 파 이썬 쉘에서 "파일명"만으로 파일을 열려면 다음의 명령으로 파이썬 쉘의 현재 작업 위치를 찾아야 함

```
>>> import os                # os 모듈 불러오기
>>> os.getcwd()              # 현재 작업 위치(=코드 저장 위치) 찾기
'C:\\Python'
```

II. 파일의 사용법

실습 9-7 텍스트 파일을 읽고 화면에 출력하기

③ 파일을 처음부터 다시 읽으려면 seek( ) 메소드를 사용해 파일의 시작 위치(0)로 파일 포인터 이동

```
>>> f.seek(0)              # 파일의 처음으로 이동
0
>>> f.readline()           # 한 줄 읽기
'흔들리며 피는 꽃\n'
>>> f.readline()           # 다음 한 줄 읽기
도종환 /
```

④ 현재 위치에서 마지막 줄까지 한번에 읽는 readlines( ) 함수로 나머지 내용을 모두 읽으면 리스트로 반환

```
>>> f.readlines()
['\n', '흔들리지 않고 피는 꽃이 어디 있으랴\n', '이 세상 그 어떤 아름다운 꽃들도\n', '다 흔들리면서 피었나니\n', '흔들리면서 줄기를 곧게 세웠나니\n', '흔들리지 않고 가는 사랑이 어디 있으랴\n', '젖지 않고 피는 꽃이 어디 있으랴\n', '이 세상 그 어떤 빛나는 꽃들도\n', '다 젖으며 젖으며 피었나니\n', '바람과 비에 젖으며\n', '꽃잎 따뜻하게 피웠나니\n', '젖지 않고 가는 삶이 어디 있으랴']
```

⑤ 파일의 사용이 끝나면 close( ) 함수로 파일 닫기

02. 파일

II. 파일 사용법

[실습] 파일 포인터 사용 함수 tell(), seek()

```
memo = """Traceback (most recent call last):
  File "<pyshell#45>", line 1, in <module>
    fd.readline()
UnicodeDecodeError: 'cp949' codec can't decode byte 0
```

```
mlist = memo.split('₩n')
fd = open("memo.txt", "w")
fd.write('₩n'.join(mlist))
fd.close()
```

```
fd = open("memo.txt", "r")
fd = read()      #전체 읽기
fd.tell()        #파일 포인터 현제 위치 반환
fd.seek(0)       #파일 포인터 0으로 이동
fd.readline()  #파일 포인터 위치에서 한 줄 읽기
fd.seek(100)  #파일 포인터 100으로 이동
fd.readline()
```

```
fd.read()
'Traceback (most recent call last):₩n  File "<pyshell#45>", line
d.readline()₩nUnicodeDecodeError: ₩'cp949₩' codec can₩'t d
osition 0: illegal multibyte sequence'
fd.tell()
197
fd.seek(0)
0
fd.readline()
'Traceback (most recent call last):₩n'
fd.readline()
'  File "<pyshell#45>", line 1, in <module>₩n'
fd.tell()
80
fd.seek(100)
100
fd.readline()
"nicodeDecodeError: 'cp949' codec can't decode byte 0xc9 in
tibyte sequence"
```

II. 파일 사용법

[실습] 파일 포인터 사용 함수 tell(), seek()

```
fd = open("poem.txt", "r")
x = ' '
while x != '':
    x = fd.readline()     #한 줄만 읽어서 반환
    print(x, end='')
    tellnum = fd.tell()
    print("[%d]" %tellnum, end='')
print()
```

```
#다시 읽기
x = fd.read()              #안 읽혀짐
#처음부터 다시 읽기
fd.seek(0)                 #fd 자료의 처음으로 이동
x = fd.read()              #잘 읽혀짐
print(x)
fd.close()
```

II. 파일 사용법

[실습] text 파일 출력

❖ Ch09-textfileR.py

- 지금 작업하고 있는 파이썬 코드 파일을 읽어서 줄 번호와 함께 출력하시오.
  - 파일명은 키보드로 입력 받는다.
  - 읽을 파일의 path는 **os.getcwd()**로 얻는다.
  - 원하는 파일을 열어서 줄 번호와 함께 출력한다.

```
>출력할 파일명을 입력: Ch09-00.py
  1: import os
  2:
  3: fpath = os.getcwd()
  4: infname = input(">출력할 파일명을 입력: ")
  5: fname = fpath + '\\' + infname.strip()
  6:
  7: if infname[-3:] == '.py':
  8:     fd = open(fname, 'r', encoding='UTF-8')
  9: else:
```

```
import os
#선택한 파일 출력
fpath = os.getcwd()
infname = input(">출력할 파일명을 입력: ")
fname = fpath + '₩₩' + infname.strip()
```

```
fd = open(fname, 'r', encoding='UTF-8')  #*.py 파일의 경우 encoding='UTF-8' 사용
seq = 0
for line in fd.readlines():    #한 줄씩 인출
    seq += 1
    print("%3d: %s" %(seq, line), end='')
```

02. 파일

II. 파일 사용법

[실습] text 파일에서 문자열 찾기

❖ Ch09-textfileR.py

- 지금 작업하고 있는 파이썬 코드 파일을 읽어서 원하는 문자열을 찾아 위치를 출력하시오.
  - 찾기를 원하는 문자열 값을 입력 받는다.
  - 입력된 문자열을 파일에서 찾아서 해당 줄 내용을 출력 하고, 찾는 문자열의 위치를 "^"로 표현한다.

```
#문자열 찾기
fd.seek(0)                  #파일 포인터 처음으로 이동
instr = input("₩n>찾을 문자열 값: ")
seq = 0
for line in fd.readlines():     #전체를 줄 단위로 인출
    seq += 1
    if instr in line:     #리스트 안에서 찾는 값 검색
        print("%3d: %s" %(seq, line), end='')
        print("      "+ " "*line.index(instr)+"^")
fd.close()
```

```
>찾을 문자 값: line
 14: for line in fd:
         ^
 16:     print("%3d: %s" %(seq, line), end='')
                              ^
 22: for line in fd.readlines():
         ^
 24:     if instr in line:
                    ^
```

02. 파일

II. 파일 사용법

[실습] text 파일에서 문자열 찾은 위치 라인부터 출력하기 ❖ Ch09-textfileR01.py

- 파이썬 코드 파일을 읽어서 원하는 문자열을 찾아 그 라인부터 10줄만 출력하시오.
  - 찾기를 원하는 문자열 값을 입력 받는다.
  - 입력된 문자열을 파일에서 찾아서 해당 줄 내용을 출력 하고, 찾는 문자열의 위치를 "^"로 표현한다.
  - 첫번째 찾은 라인부터 10줄만 출력한다.

```
#문자열 찾기
fd.seek(0)      #파일 포인터 처음으로 이동
instr = input("₩n>찾을 문자 값: ")
seq = 0
while True:
    line = fd.readline()
    if line == '':
        break
    if instr in line:
        break
    else:
        seq += 1
        spos = fd.tell()   #찾기 전까지의 포인터 위치
```

```
#찾은 위치 줄부터 10줄 출력
fd.seek(spos)   #찾은 줄로 이동
lcnt = 0
while lcnt < 10:    #10
    line = fd.readline()
    if line == '':
        break
    else:
        seq += 1
        lcnt += 1
        print("%3d: %s" %(seq, line), end='')

fd.close()
```

```
>찾을 문자 값: line
 14: for line in fd:
 15:     seq += 1
 16:     print("%3d: %s" %(seq, line), end='')
 17:
 18: #문자열 찾기
 19: fd.seek(0)
 20: instr = input("\n>찾을 문자 값: ")
 21: seq = 0
 22: for line in fd.readlines():
 23:     seq += 1
```

II. 파일의 사용법

실습 9-8 텍스트 파일 저장하기

- 파일에 쓸 내용은 문자열로 직접 지정하거나 키보드로 입력받도록 하고, write( )나 writelines( ) 메소드로 파일에 기록

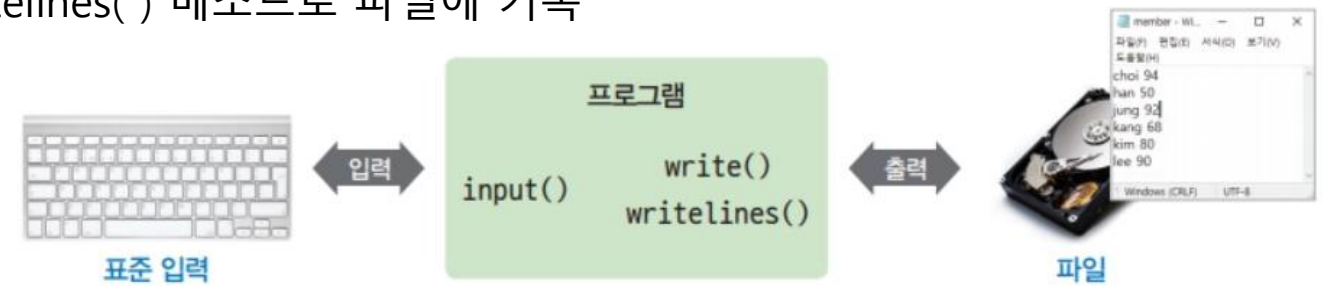


그림 9-16 텍스트 파일에 문자열 쓰기

표 9-3 문자열 쓰기 메소드

함수명	설명	사용 예	반환값
write()	문자열 쓰기, 바이너리 파일에도 사용	f.write("abcd")	문자 수
writelines()	문자열이나 리스트의 내용을 모두 쓰기	f.writelines("ab" "cd") f.writelines(['ab', 'cd'])	없음

II. 파일의 사용법

실습 9-8 텍스트 파일 저장하기

① 파일을 쓰기 모드('w')로 열기

```
>>> f = open("c:/python/member.txt", "w")
```

✓ **TIP** 'member.txt' 파일을 만들 때 'c:/python'이라는 경로가 없으면 FileNotFoundError가 발생, 사용 중인 컴퓨터의 환 경에 맞게 드라이브명('c:'나 'd:' 등)과 디렉토리명을 설정

② write( ) 함수로 지정한 문자열을 파일에 쓰면 저장한 문자 수를 반환

```
>>> f.write("choi 94\n")
8
```

줄바꿈 문자('\n')가 없으면 다음에 쓰는 내용이 같은 줄에 바로 이어서 저장되므로, 파일을 보기 좋게 표시하기 위해 추가

③ writelines( ) 함수는 하나 이상의 문자열이나 리스트를 인수로 지정하면 파일에 지정한 텍스트를 입력

```
>>> f.writelines("han 50\n","jung 92\n")
>>> f.writelines(["kang 68\n", "kim 80\n"])
```

문자열 사이에 쉼표(,)를 쓰면 오류가 발생하므로 주의

II. 파일의 사용법

실습 9-8 텍스트 파일 저장하기

④ input( ) 함수로 입력받은 텍스트를 파일에 저장

```
>>> a = input("파일에 쓸 내용 입력 : ")
파일에 쓸 내용 입력 : lee 90
>>> f.write(a)
6
```

⑤ 파일에 쓰기가 끝나면 반드시 닫기를 실행해야 파일 내용이 물리적으로 저장

```
>>> f.close()
```

II. 파일 사용법

[실습] 리스트로 text 파일 생성

```
poem = """흔들리며 피는 꽃
도종환/

흔들리지 않고 피는 꽃이 어디 있으랴
이 세상 그 어떤 아름다운 꽃들도
다 흔들리면서 피었나니
흔들리면서 줄기를 곧게 세웠나니
흔들리지 않고 가는 사랑이 어디 있으랴
젖지 않고 피는 꽃이 어디 있으랴
이 세상 그 어떤 빛나는 꽃들도
다 젖으며 젖으며 피었나니
바람과 비에 젖으며
꽃잎 따뜻하게 피웠나니
젖지 않고 가는 삶이 어디 있으랴
"""
```

```
plist = poem.split('₩n')          #'₩n'으로 줄 단위 추출
fd = open("poem.txt", "w")        #쓰기 파일 열기
#줄 단위 쓰기
for x in plist:
    fd.writelines(x+'₩n')         #(쓰기) 제거된 '₩n'을 복원
fd.close()                        #파일 닫기
```

```
plist = poem.split('₩n')          #'₩n'으로 줄 단위 추출
fd = open("poem.txt", "w")
#전체 줄 연결한 후 한번에 쓰기
fd.write('₩n'.join(plist))        #(쓰기) 리스트를 '₩n'로 연결
fd.close()
```

II. 파일의 사용법

여기서 잠깐 **파일 닫기를 생략하는 방법**

- 읽기 모드로 연 파일은 프로그램이 종료될 때 파일을 자동으로 닫아 주지만, 쓰기 모드의 파일은 닫기 과정이 꼭 필요

```
>>> with open("c:/python/member.txt", "w") as f :
        f.write("abcd")
```

```
with open("test.txt", "w") as fd:
    s = input("> ")
    fd.write(s)
    s = input("> ")
    fd.write(s)


> python Progrmming
17
> Good
4
```

```
with open("test.txt", "r") as fd:
    fd.read()


'python ProgrmmingGood'
```

02. 파일

II. 파일 사용법

[실습] text 파일 생성

❖ Ch09-textfileKeyW.py

- 키보드로 입력하는 내용을 text 파일로 생성한다.
  - 생성할 파일명은 키보드로 입력 받는다. 파일의 path는 **os.getcwd()**로 얻는다.
  - 파일에는 [성명, 성별, 나이]를 줄 바꿈으로 구분하여 기록한다.
  - 성명, 성별, 나이는 키보드로 입력을 받는다. 입력을 마치려면 입력 값 없이 Enter 키를 누름 처리

```
import os
#선택한 파일 출력
fpath = os.getcwd()
infname = input(">생성할 파일명을 입력: ")
fname = fpath + '₩₩' + infname.strip()
```

```
>생성할 파일명을 입력: members.txt
> 성명 성별 나이 (입력): Kims f 20
> 성명 성별 나이 (입력): Lees m 19
> 성명 성별 나이 (입력): Parks f 22
> 성명 성별 나이 (입력):
[['Kims', 'f', '20'], ['Lees', 'm', '19'], ['Parks', 'f', '22']]
```

```
mlist = []  #파일로 쓰기용 임시 리스트
while True:    #키보드 입력 값으로 리스트 생성
    inmember = list(input("> 성명 성별 나이 (입력): ").split())
    if len(inmember) == 0:    #입력 종료
        break
    mlist.append(inmember)
print(mlist)
```

```
#리스트로 파일 생성
fd = open(fname, 'w')
for rec in mlist:
    fd.writelines('₩t'.join(rec)+'₩n')
fd.close()
```

02. 파일

II. 파일 사용법

[실습] text 파일 읽어서 출력

❖ Ch09-textfileKeyR.py

- 생성된 text 파일을 읽어서 각 항목을 출력한다.
  - 확인할 파일명은 키보드로 입력 받는다. 파일의 path는 **os.getcwd()**로 얻는다.
  - 파일에서 각 라인을 **리스트**로 변환하여 '성명', '성별', '나이'를 추출해낸다.
  - 추출한 성명, 성별, 나이를 출력한다.

```
import os
#선택한 파일 출력
fpath = os.getcwd()
infname = input(">출력할 파일명을 입력: ")
fname = fpath + '₩₩' + infname.strip()
```

```
fd = open(fname, 'r')
for line in fd.readlines():
    rec = line.split()      #공백으로 분리 후 리스트화
    id, sex, age = rec      #리스트 언팩킹
    print("%10s : %s  %3d" %(id, sex, int(age)))
fd.close()
```

```
>확인할 파일명을 입력: members.txt
      Kims : f   20
      Lees : m   19
     Parks : f   22
```

02. 파일

II. 파일 사용법

[실습] text 파일 읽어서 출력 : with open() ~ 절 사용

- 생성된 text 파일을 읽어서 각 항목을 출력한다.
  - 확인할 파일명은 키보드로 입력 받는다. 파일의 path는 **os.getcwd()**로 얻는다.
  - 파일에서 각 라인을 **리스트**로 변환하여 '성명', '성별', '나이'를 추출해낸다.
  - 추출한 성명, 성별, 나이를 출력한다.

```
import os
#선택한 파일 출력
fpath = os.getcwd()
infname = input(">출력할 파일명을 입력: ")
fname = fpath + '₩₩' + infname.strip()
```

```
with open(fname, 'r') as fd:  #파일 열기에서 닫기 까지를 일괄 처리
        for line in fd.readlines():
            rec = line.split()       #공백으로 분리 후 리스트화
            id, sex, age = rec       #리스트 언팩킹
            print("%10s : %s  %3d" %(id, sex, int(age)))
        fd.close()
```

```
>확인할 파일명을 입력: members.txt
      Kims : f   20
      Lees : m   19
     Parks : f   22
```

III. 예외 처리

- 파일을 읽기 모드로 열 때 파일명이나 경로를 잘못 설정하면, 파일을 찾지 못하고 다음과 같이 오류가 발생

```
>>> f = open("member.txt", "r")
Traceback (most recent call last):
    File "<pyshell#98>", line 1, in <module>
        f = open("member.txt", "r")
FileNotFoundError: [Errno 2] No such file or directory: 'member.txt'
```

- 예상치 못한 상황이 발생할 때를 대비해 실행할 동작을 미리 정해 두는 것이 예외 처리
- 파이썬에서는 예외 처리 구문으로 try와 except를 사용

```
try:
    오류가 발생할 가능성이 있는 문장
except:
    오류가 발생하는 경우, 실행할 문장
```

III. 예외 처리

실습 9-9 키보드 입력에 예외 처리 적용하기

① 사용자가 정수 값을 입력하지 않으면 파이썬 오류 메시지 대신 프로그램에서 미리 정해둔 내용을 출력

```
>>> try :
        a = int(input("정수 입력 : "))
except :
        print("정수를 입력하세요.")          # 오류가 발생할 때 실행할 동작

정수 입력 : 홍길동
정수를 입력하세요.
```

사용자가 들여쓰기를 직접 지워야 함

숫자가 아닌 문자를 잘못 입력

- 예외 처리를 위해 다음과 같이 오류가 발생하지 않은 경우에도 실행할 명령을 정해 둘 수도 있음

```
try:
    오류가 발생할 가능성이 있는 문장
except:
    오류가 발생하는 경우, 실행할 문장
else:
    오류가 없을 때 실행할 문장
```

III. 예외 처리

실습 9-10 파일 열기에 예외 처리 적용하기

① 파일을 열지 못하는 경우에는 '파일 확인' 메시지를 출력하도록 하고, 파일 열기에 문제가 없으면 파일의 내용을 모두 읽어 출력

```
>>> try :
        f = open("member.txt", "r")
except :
        print("파일을 확인하세요.")
else :
        print(f.read())
        f.close()

파일을 확인하세요.
```

파일 경로가 잘못 설정된 경우

III. 예외 처리

실습 9-10 파일 열기에 예외 처리 적용하기

② 파일의 경로를 올바르게 설정하는 경우, 파일을 정상적으로 읽어서 모든 내용을 출력

```
>>> try :
        f = open("c:/python/member.txt", "r")
except :
        print("파일을 확인하세요.")
else :
        print(f.read())
        f.close()

choi 94
han 50
jung 92
kang 68
kim 80
lee 90
```

파일 경로가 맞게 설정된 경우

02. 파일

III. 예외 처리

[실습] try: ~ except: ~ else: ~

- 생성된 text 파일을 읽어서 각 항목을 출력한다.
  - 확인할 파일명은 키보드로 입력 받는다. 파일의 path는 **os.getcwd()**로 얻는다.
  - 파일에서 각 라인을 리스트로 변환하여 '성명', '성별', '나이'를 추출해낸다.
  - 추출한 성명, 성별, 나이를 출력한다.

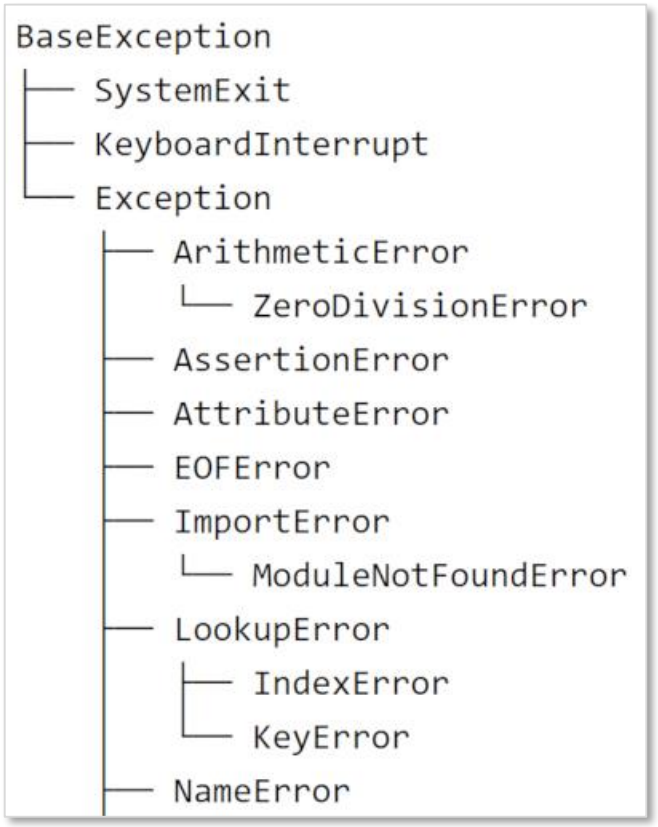
```
import os
#선택한 파일 출력
fpath = os.getcwd()
infname = input(">출력할 파일명을 입력: ")
fname = fpath + '₩₩' + infname.strip()
```

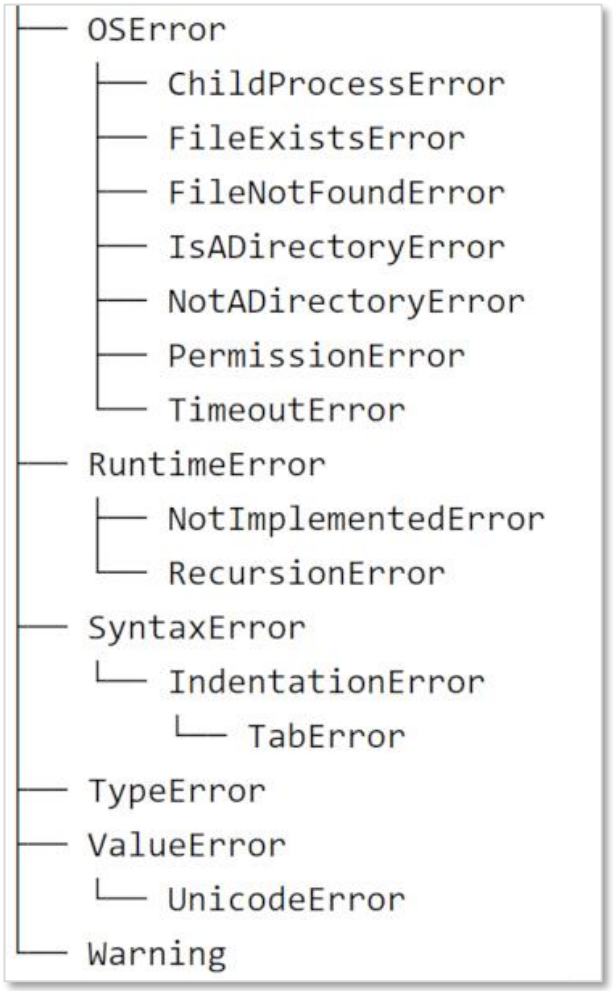
```
>확인할 파일명을 입력: members
파일이 존재하지 않습니다.
E:\Documents\강의록\Python\source\09장\members
```

```
try:
    fd = open(fname, 'r')
except:
    print(">>파일이 존재하지 않습니다.₩n%s" %fname)
else:
    for line in fd.readlines():
        rec = line.split()      #공백으로 분리 후 리스트화
        id, sex, age = rec      #리스트 언팩킹
        print("%10s : %s  %3d" %(id, sex, int(age)))
    fd.close()
```

III. 예외 처리

- **except: 예외의 종류**





02. 파일

III. 예외 처리

[실습] except: 예외의 종류 지정

❖ Ch09-textfileKeyR예외.py

- except: 지정 방법
  - except: #모든 예외 처리
  - except 예외종류: #지정된 예외만 처리 (상위 예외로 하위 예외 처리 가능)
  - except (예외종류1, 예외종류2, ...): #복수개 예외 동시 처리
  - except 예외종류 as 변수: #변수로 **표준 예외 메시지** 저장

```
try:
    a = 3 / 0                #Error 발생
    fd = open(fname, 'r')
except FileNotFoundError:
    print("파일이 존재하지 않습니다.₩n%s" %fname)
except ZeroDivisionError as ze:
    print("Error: %s" %ze)
else:
    for line in fd.readlines():
        rec = line.split()       #공백으로 분리 후 리스트화
        id, sex, age = rec       #리스트 언팩킹
        print("%10s : %s  %3d" %(id, sex, int(age)))
    fd.close()
finally:    #예외에 관계 없이 처리
    print("^^ Have a good time!")
```

```
>확인할 파일명을 입력: a
Error: division by zero
^^ Have a good time!
```

IV. 파일의 활용

❖ Ch09-MenuFileWR.py

[실습-1] 메뉴(menu) 딕셔너리를 Menu.txt 파일로 저장하기 (1)

- 메뉴(menu) 딕셔너리를 파일로 저장하는 코드를 작성하시오.
  - 파일명은 현제 코드 위치에 "Menu.txt"로 한다.
  - 파일의각 줄(line) 레코드(record)의 구성은 (대분류 메뉴명, 소분류 메뉴명, 단가) 순으로 한다.
  - 레코드의 각 데이터 항목은 '₩t'로 구분한다.

```
import os
menu = {'COFFEE': [['에스프레소', 3.0], ['아메리카노', 3.0], ['카페라떼', 4.0], ['카프치노', 4.0]],
        'LATTE': [['말차라떼', 4.0], ['초코라떼', 4.0], ['카페라떼', 4.0]],
        'TEA': [['청귤차', 4.0], ['자몽차', 4.0], ['레몬차', 4.0], ['카모마일', 4.5]],
        'ADE': [['자몽에이드', 4.5], ['레몬에이드', 4.5], ['청포도에이드', 4.5]],
        'JUICE': [['망고', 4.5], ['바나나', 4.5], ['딸기', 4.5], ['키위', 4.5]],
        'SMOOTHIE': [['청귤스무디', 4.5], ['요거트스무디', 4.5]],
        'MILK TEA': [['흑당밀크티', 4.5], ['달고나밀크티', 4.5]]}
Menu = {}  #파일로부터 만들 메뉴
```

```
########## Main ######
write_menu_file(menu)   #딕셔너리를 파일로 쓰기
########################
```

```
COFFEE  에스프레소      3.0
COFFEE  아메리카노      3.0
COFFEE  카페라떼 4.0
COFFEE  카프치노 4.0
LATTE   말차라떼 4.0
LATTE   초코라떼 4.0
LATTE   카페라떼 4.0
TEA     청귤차  4.0
TEA     자몽차  4.0
TEA     레몬차  4.0
TEA     카모마일 4.5
ADE     자몽에이드      4.5
ADE     레몬에이드      4.5
ADE     청포도에이드    4.5
JUICE   망고    4.5
JUICE   바나나  4.5
JUICE   딸기    4.5
JUICE   키위    4.5
SMOOTHIE        청귤스무디      4.5
SMOOTHIE        요거트스무디    4.5
MILK TEA흑당밀크티      4.5
MILK TEA달고나밀크티    4.5
```

02. 파일

IV. 파일의 활용

❖ Ch09-MenuFileWR.py

[실습-1] 메뉴(menu) 딕셔너리를 Menu.txt 파일로 저장하기 (2)

- 메뉴(menu) 딕셔너리를 파일로 저장하는 코드를 작성하시오.
  - 파일명은 현제 코드 위치에 "Menu.txt"로 한다.
  - 레코드의 구성은 (대분류 메뉴명, 소분류 메뉴명, 단가) 순으로 한다.
  - 레코드의 각 데이터 항목은 '₩t'로 구분한다.

```
def write_menu_file(menu):
    fpath = os.getcwd()
    fname = fpath + '₩₩' + "Menu.txt"
    fd = open(fname, 'w')
    for title, mlists in menu.items():
        for mmenu, price in mlists:
            rec = title + '₩t' + mmenu + '₩t' + str(price) + '₩n'
            fd.writelines(rec)
    fd.close()
```

```
COFFEE	에스프레소	3.0
COFFEE	아메리카노	3.0
COFFEE	카페라떼	4.0
COFFEE	카프치노	4.0
LATTE	말차라떼	4.0
LATTE	초코라떼	4.0
LATTE	카페라떼	4.0
TEA	청귤차	4.0
TEA	자몽차	4.0
TEA	레몬차	4.0
TEA	카모마일	4.5
ADE	자몽에이드	4.5
ADE	레몬에이드	4.5
ADE	청포도에이드	4.5
JUICE	망고	4.5
JUICE	바나나	4.5
JUICE	딸기	4.5
JUICE	키위	4.5
SMOOTHIE	청귤스무디	4.5
SMOOTHIE	요거트스무디	4.5
MILK TEA	흑당밀크티	4.5
MILK TEA	달고나밀크티	4.5
```

IV. 파일의 활용

[과제-1/n] Menu.txt 파일을 읽어서 Menu 딕셔너리로 만들어 출력하기

- Menu.txt 파일을 읽어서 메뉴(Menu) 딕셔너리를 생성하는 코드를 작성하시오.
  - 딕셔너리명은 Menu로 한다.
  - 데이터 항목은 이전 menu 딕셔너리와 동일하게 {키: [소분류 메뉴명, 단가], ... }로 구성한다.

```
########## Main ######
read_menu_file()          #파일 메뉴를 읽어 딕셔너리로 만들기
prt_menu(menu)        #새로운 메뉴 딕셔너리 출력
######################
```

IV. 파일의 활용

[과제-1/n] Menu.txt 파일을 읽어서 Menu 딕셔너리로 만들기 (2)

- Menu.txt 파일을 읽어서 메뉴(Menu) 딕셔너리를 생성하는 코드를 작성하시오.
  - 딕셔너리명은 Menu로 한다.
  - 데이터 항목은 이전 menu 딕셔너리와 동일하게 {키: [소분류 메뉴명, 단가], ... }로 구성한다.

```
def read_menu_file():
    global Menu
    fpath = os.getcwd()
    fname = fpath + '₩₩' + "Menu.txt"
    fd = open(fname, 'r')
    mlist = []
    line = fd.readline()     #한 줄만 읽기
    rec = line.split()       #리스트화
    title, *etc = rec        #언팩킹
    save_title = title       #변화 탐지용
    fd.seek(0)               #원위치로
    lcnt = 0
```

```
COFFEE  에스프레소      3.0
COFFEE  아메리카노      3.0
COFFEE  카페라떼 4.0
COFFEE  카프치노 4.0
LATTE   말차라떼 4.0
LATTE   초코라떼 4.0
LATTE   카페라떼 4.0
TEA     청귤차   4.0
TEA     자몽차   4.0
TEA     레몬차   4.0
TEA     카모마일 4.5
ADE     자몽에이드      4.5
```

```
[[ 소분류 메뉴 ]]
COFFEE  :  [['에스프레소', 3.0], ['아메리카노', 3.0], ['카페라떼', 4.0], ['카프치노', 4.0]]
LATTE  :  [['말차라떼', 4.0], ['초코라떼', 4.0], ['카페라떼', 4.0]]
TEA  :  [['청귤차', 4.0], ['자몽차', 4.0], ['레몬차', 4.0], ['카모마일', 4.5]]
ADE  :  [['자몽에이드', 4.5], ['레몬에이드', 4.5], ['청포도에이드', 4.5]]
JUICE  :  [['망고', 4.5], ['바나나', 4.5], ['딸기', 4.5], ['키위', 4.5]]
SMOOTHIE  :  [['청귤스무디', 4.5], ['요거트스무디', 4.5]]
MILK TEA  :  [['흑당밀크티', 4.5], ['달고나밀크티', 4.5]]
```

IV. 파일의 활용

[과제-1/11] Menu.txt 파일을 읽어서 Menu 딕셔너리로 만들기 (2)

- Menu.txt 파일을 읽어서 메뉴(Menu) 딕셔너리를 생성하는 코드를 작성하시오.
  - 딕셔너리명은 Menu로 한다.
  - 데이터 항목은 이전 menu 딕셔너리와 동일하게 {키: [소분류 메뉴명, 단가], ... }로 구성한다.

```
for line in fd.readlines():      #한 줄씩 반복 읽기
    lcnt += 1
    line = line.rstrip('₩n')
    rec = line.split('₩t')
    title, mmenu, price = rec
    if save_title != title: #title 변화 감지
        Menu[save_title] = mlist     #메뉴 딕셔너리에 추가
        save_title = title           #다음 대분류 메뉴 작업 준비
        mlist = []
    mlist.append([mmenu, float(price)])      #소분류 메뉴 리스트 구성
if lcnt > 0:          #리스트(메뉴)가 존재할 경우
    Menu[save_title] = mlist      # 마지막 항목 메뉴 딕셔너리에 추가
```

```
[[ 소분류 메뉴 ]]
COFFEE  :  [['에스프레소', 3.0], ['아메리카노', 3.0], ['카페라떼', 4.0], ['카프치노', 4.0]]
LATTE  :  [['말차라떼', 4.0], ['초코라떼', 4.0], ['카페라떼', 4.0]]
TEA  :  [['청귤차', 4.0], ['자몽차', 4.0], ['레몬차', 4.0], ['카모마일', 4.5]]
ADE  :  [['자몽에이드', 4.5], ['레몬에이드', 4.5], ['청포도에이드', 4.5]]
JUICE  :  [['망고', 4.5], ['바나나', 4.5], ['딸기', 4.5], ['키위', 4.5]]
SMOOTHIE  :  [['청귤스무디', 4.5], ['요거트스무디', 4.5]]
MILK TEA  :  [['흑당밀크티', 4.5], ['달고나밀크티', 4.5]]
```

```
TEA       자몽차    4.0
TEA       레몬차    4.0
TEA       카모마일  4.5
ADE       자몽에이드        4.5
ADE       레몬에이드        4.5
ADE       청포도에이드      4.5
JUICE     망고      4.5
JUICE     바나나    4.5
JUICE     딸기      4.5
JUICE     키위      4.5
SMOOTHIE          청귤스무디        4.5
SMOOTHIE          요거트스무디      4.5
MILK TEA 흑당밀크티         4.5
MILK TEA 달고나밀크티       4.5
```

IV. 파일의 활용

[과제-2/11] Menu.txt 파일로 menu 딕셔너리 구성 후 메뉴 출력하기

- Menu.txt 파일을 읽어서 메뉴(Menu) 딕셔너리를 생성 후에 메뉴표를 출력하시오.
  - 파일로부터 새롭게 생성할 딕셔너리명은 menu로 한다. (이전 menu 딕셔너리는 코드에서 삭제)
  - 데이터 항목은 이전 menu 딕셔너리와 동일하게 {키: [소분류 메뉴명, 단가], ... }로 구성한다.
  - 메뉴 출력 형식은 아래 그림과 같이 출력한다.
  - 메뉴 출력은 prt_menu() 함수를 작성하여 해결한다.

```
########## Main ######
read_menu_file()        #파일 메뉴를 읽어 딕셔너리로 만들기
prt_menu(menu)          #새로운 메뉴 딕셔너리 출력
#######################
```

```
[[ 메뉴 ]]
## COFFEE ##
에스프레소          3.0
아메리카노          3.0
카페라떼            4.0
카프치노            4.0

## LATTE ##
말차라떼            4.0
초코라떼            4.0
카페라떼            4.0

## TEA ##
청귤차              4.0
```

02. 파일

IV. 파일의 활용

[과제-3/11] menu 딕셔너리로부터 대분류메뉴 딕셔너리 만들기

- 대분류 메뉴 딕셔너리(menugroup)를 구성하시오. ❖ 참고: Ch08-Menu02.py
  - 이미 구성된 소분류 메뉴 딕셔너리(menu)를 참조하여 구성
  - Key는 0부터 시작하는 순서번호로 하고, 키에 대응되는 값은 대분류 메뉴명으로 구성

```
menugroup = {}  #대분류메뉴 딕셔너리


########## Main ######
read_menu_file()          #파일 메뉴를 읽어 딕셔너리로 만들기
prt_menu(menu)          #새로운 메뉴 딕셔너리 출력
new_menugroup()          #menu로부터 대분류메뉴 딕셔너리 생성
prt_menugroup(menugroup)          # 대분류메뉴 딕셔너리 출력
######################
```

```
[[ 대분류 메뉴 ]]
0 : COFFEE
1 : LATTE
2 : TEA
3 : ADE
4 : JUICE
5 : SMOOTHIE
6 : MILK TEA
```

IV. 파일의 활용

[과제-4/11] 대분류메뉴 선택

❖ 참고: Ch08-Menu03.py

- 대분류 메뉴표 출력하여주고, 그 중 하나를 선택하는 코드를 완성하시오.
  - 키보드로 대분류 메뉴의 키Key를 입력 받아 처리
  - 입력 받은 키로 소분류 메뉴 딕셔너리(menu)를 참조하여 해당 리스트를 출력
  - 선택된 대분류메뉴의 번호를 반환

```
########## Main ######
read_menu_file()          #파일 메뉴를 읽어 딕셔너리로 만들기
#prt_menu(menu)          #새로운 메뉴 딕셔너리 출력
new_menugroup()        #menu로부터 대분류메뉴 딕셔너리 생성
#prt_menugroup(menugroup)          # 대분류메뉴 딕셔너리 출력
inkey = sel_menugroup()          # 대분류메뉴 선택, 선택된 번호 반환
######################
```

```
>> [선택] 대분류 메뉴 <<
[0] COFFEE
[1] LATTE
[2] TEA
[3] ADE
[4] JUICE
[5] SMOOTHIE
[6] MILK TEA
------------------------------
>>메뉴 그룹(번호) 선택: 1
[['말차라떼', 4.0], ['초코라떼', 4.0], ['카페라떼', 4.0]]
```

IV. 파일의 활용

[과제-5/11] 소분류메뉴 선택

❖ 참고: Ch08-Menu04.py

- 선택된 대분류 메뉴의 소분류 메뉴표 출력하여주고, 그 중 하나를 선택하는 코드를 완성하시오.
  - 선택된 대분류 메뉴로 소분류 메뉴 리스트(mlist)를 출력, 이때 순서번호도 함께 출력
  - 소분류 메뉴의 순서번호를 키보드로 입력 받아 메뉴 선택, 선택된 메뉴명 출력

```
########## Main ######
read_menu_file()          #파일 메뉴를 읽어 딕셔너리로 만들기
#prt_menu(menu)          #새로운 메뉴 딕셔너리 출력
new_menugroup()         #menu로부터 대분류메뉴 딕셔너리 생성
#prt_menugroup(menugroup)          # 대분류메뉴 딕셔너리 출력
inkey = sel_menugroup()  # 대분류메뉴 선택, 선택된 번호 반환
sel_menu(inkey)              #소분류메뉴 선택, 주문 수량 처리
#######################
```

```
>> [선택] 대분류 메뉴 <<
[0] COFFEE
[1] LATTE
[2] TEA
[3] ADE
[4] JUICE
[5] SMOOTHIE
[6] MILK TEA
------------------------------
>>메뉴 그룹(번호) 선택: 1

>> [LATTE] 소분류 메뉴 <<
[0] 말차라떼              4.0
[1] 초코라떼              4.0
[2] 카페라떼              4.0
------------------------------
>>메뉴(번호) 선택: 2
> 카페라떼를 선택하셨습니다.
```

IV. 파일의 활용

[과제-6/11] 소분류메뉴 주문 수량 처리

❖ 참고: Ch08-Menu05.py

- 선택된 소분류 메뉴에 대한 주문 수량을 입력받아 주문내역 리스트(selmlist)를 구성하시오.
  - 주문 리스트(selmlist)는 [메뉴명, 단가, 주문수량, 금액]로 구성
  - 금액은 (단가*주문수량)으로 계산, 구성된 주문 리스트(selmlist)를 출력

```
########## Main ######
read_menu_file()          #파일 메뉴를 읽어 딕셔너리로 만들기
#prt_menu(menu)          #새로운 메뉴 딕셔너리 출력
new_menugroup()         #menu로부터 대분류메뉴 딕셔너리 생성
#prt_menugroup(menugroup)          # 대분류메뉴 딕셔너리 출력
inkey = sel_menugroup()  # 대분류메뉴 선택, 선택된 번호 반환
sel_menu(inkey)              #소분류메뉴 선택, 주문 수량 처리
#######################
```

```
------------------------------
>>메뉴 그룹(번호) 선택: 1

>> [LATTE] 소분류 메뉴 <<
[0] 말차라떼              4.0
[1] 초코라떼              4.0
[2] 카페라떼              4.0
------------------------------
>>메뉴(번호) 선택: 2
> 카페라떼를 선택하셨습니다.
>>몇 잔을 원하십니까? 3
> [카페라떼]를 [3]잔 선택하셨습니다.
[['카페라떼', 4.0, 3, 12000.0]]
```

02. 파일

IV. 파일의 활용

[과제-7/11] 주문 결과 출력 및 주문 계속 처리

❖ 참고: Ch08-Menu06.py

- 메뉴 선택을 계속 할 수 있도록 코드를 수정하고, 마지막에 주문 내역(selmlist)을 출력하시오.
  - 주문 종료는 키보드로 'x'나 'X' 값을 입력받아 처리, 기타 키는 주문 계속으로 처리
  - 최종 주문내역(selmlist)은 주문한 (메뉴명, 단가, 주문 수량, 금액)을 나열
  - 마지막에 주문 합계를 (총 주문 잔 수, 총 청구 금액)으로 출력

```
> [카모마일]를 [4]잔 선택하셨습니다.
>>계속 주문을 하시겠습니까? [종료: x, 계속: Enter] x
========== >> 주문 결과 << ==========
카페라떼          4000    3    12000
카모마일          4500    4    18000
-----------------------------------
주문 합계 :           [7]잔    30000원
```

```
########## Main ######
read_menu_file()          #파일 메뉴를 읽어 딕셔너리로 만들기
#prt_menu(menu)          #새로운 메뉴 딕셔너리 출력
new_menugroup()        #menu로부터 대분류메뉴 딕셔너리 생성
#prt_menugroup(menugroup)          # 대분류메뉴 딕셔너리 출력
selmlist = []     #주문내역 리스트 생성
while True:
    inkey = sel_menugroup()          # 대분류메뉴 선택, 대분류메뉴 번호 반환
    sel_menu(inkey)              #소분류메뉴 선택, 주문 수량 처리, 주문 결과 리스트 반환
    prt_result(selmlist)
    insel = input(">>계속 주문을 하시겠습니까? [종료: x, 계속: Enter] ")
    if insel == 'x' or insel == 'X':
        break
######################
```

IV. 파일의 활용

[과제-8/11] 주문 결과(selmlist) 파일(Orders_2023.txt)에 추가 (1)

- 메뉴 주문이 끝나면 주문 내역(selmlist)을 'Orders_2023.txt'에 추가하시오.
  - 'Orders.txt' 파일의 레코드는 {orderid seq menu unit qty tot} 로 구성한다.
  - orderid는 '주문일자/주문시간' 으로 구성
  - oseq는 함께 이루어진 주문의 순서번호

```
from datetime import datetime

########## Main ######
read_menu_file()          #파일 메뉴를 읽어 딕셔너리로 만들기
#prt_menu(menu)           #새로운 메뉴 딕셔너리 출력
new_menugroup()          #menu로부터 대분류메뉴 딕셔너리 생성
#prt_menugroup(menugroup)          # 대분류메뉴
selmlist = []     #주문내역 리스트 생성
while True:
    inkey = sel_menugroup()          # 대분류
    sel_menu(inkey)              #소분류메뉴 선택
    prt_result(selmlist)
    insel = input(">>계속 주문을 하시겠습니
    if insel == 'x' or insel == 'X':
        break
append_order_file(selmlist)  #주문내역 파일로 저장
######################
```

Orders

파일 편집 보기

```
2023-05-12/13:31:05.028541    01    레몬차        4.0    2    8000
2023-05-12/13:37:37.718284    01    청포도에이드  4.5    3    13500
2023-05-12/13:37:37.718284    02    자몽차        4.0    1    4000
2023-05-12/13:37:37.718284    03    카페라떼      4.0    3    12000
```

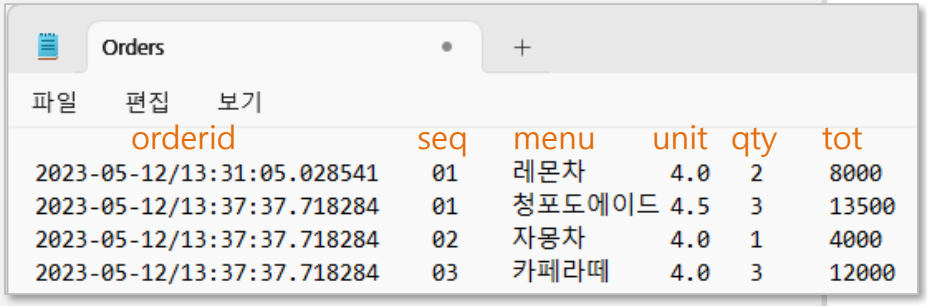
IV. 파일의 활용

[과제-8/11] 주문 결과(selmlist) 파일(Orders_2023.txt)에 추가 (2)

- 메뉴 주문이 끝나면 주문 내역(selmlist)을 'Orders_2023.txt'에 추가하시오.
  - 'Orders.txt' 파일의 레코드는 {orderid seq menu unit qty tot}로 구성한다.
  - orderid는 '주문일자/주문시간' 으로 구성
  - oseq는 함께 이루어진 주문의 순서번호

```
def append_order_file(selmlist):   #주문내역 리스트를 파일로 저장
    fpath = os.getcwd()
    fname = fpath + '₩₩' + 'Orders_2023.txt'
    fd = open(fname, 'a')  #없으면 파일 생성 후 append


    ## 완성할 부분 ##
```



02. 파일

IV. 파일의 활용

[과제-9/11] 주문 반복 함수 호출 처리로 변경

- 주문하는 반복 과정을 주문 함수 process_order() 함수호출로 바꾸시오.
  - process_order() 함수는 {대분류메뉴 번호 선택 → 메뉴 선택 → 수량 선택} 반복
  - 주문 반복 종료 시 주문 내역을 append_order_file(selmlist)를 호출하여 파일에 추가

```
########## Main ######
read_menu_file()          #파일 메뉴를 읽어 딕셔너리로 만들기
new_menugroup()       #menu로부터 대분류메뉴 딕셔너리 생성
selmlist = []     #주문내역 리스트 생성
process_order()
'''
while True:
    inkey = sel_menugro
    sel_menu(inkey)
    prt_result(selmlist)
    insel = input(">>계속
    if insel == 'x' or inse
        break
append_order_file(seln
'''

######################
```

```
def process_order():
    while True:
        inkey = sel_menugroup()
        sel_menu(inkey)
        prt_result(selmlist)
        insel = input(">>계속 주문을 하시겠습니까? [종료: x, 계속: Enter] ")
        if insel == 'x' or insel == 'X':
            break
    append_order_file(selmlist)
```

IV. 파일의 활용

[과제-10/11] 전체 주문내역 파일(Orders_2023.txt)을 읽어서 출력

- 주문내역 파일 'Orders_2023.txt' 파일을 읽어서 전체 주문 내역을 출력하시오.

```
def prt_order_file():    #(출력) 전체 주문내역 파일
    ## <완성할 부분> ##
    # 파일 열기
    #한 줄씩 출력하기

########## Main ######
read_menu_file()         #파일 메뉴를 읽어 딕셔너리로 만들기
new_menugroup()       #menu로부터 대분류메뉴 딕셔너리 생성
selmlist = []    #주문내역 리스트 생성
process_order()
prt_order_file()
#######################
```

```
============================ >> 전체 주문 내역 << ============================
['2023-05-13/14:08:27.441782', '01', '청포도에이드', '4.5', '2', '9000']
[  1] 2023-05-13/14:08:27.441782   01 청포도에이드                 4.5       2         9000
['2023-05-13/14:08:27.441782', '02', '자몽에이드', '4.5', '3', '13500']
[  2] 2023-05-13/14:08:27.441782   02 자몽에이드                   4.5       3        13500
['2023-05-13/14:14:14.885474', '01', '초코라떼', '4.0', '1', '4000']
[  3] 2023-05-13/14:14:14.885474   01 초코라떼                     4.0       1         4000
['2023-05-13/14:16:25.339349', '01', '초코라떼', '4.0', '1', '4000']
[  4] 2023-05-13/14:16:25.339349   01 초코라떼                     4.0       1         4000
['2023-05-13/14:34:50.246815', '01', '카모마일', '4.5', '3', '13500']
[  5] 2023-05-13/14:34:50.246815   01 카모마일                     4.5       3        13500
['2023-05-13/14:37:45.742627', '01', '청포도에이드', '4.5', '3', '13500']
[  6] 2023-05-13/14:37:45.742627   01 청포도에이드                 4.5       3        13500
['2023-05-13/14:37:45.742627', '02', '키위', '4.5', '3', '13500']
[  7] 2023-05-13/14:37:45.742627   02 키위                         4.5       3        13500
--------------------------------------------------------------------------------
```

IV. 파일의 활용

[과제-11/11] 전체 메뉴 구성 (1)

- 작업 선택 함수 sel_task()를 만들어서 원하는 작업을 선택할 수 있도록 하시오.
  - 작업 선택은 [1] 메뉴-대분류 [2] 메뉴-소분류 [3] 주문 [4] 주문내역 [5] 전체-주문내역 [x] 종료 로 구성

```
def sel_task():    #작업 선택 메뉴
    print("₩n>>번호로 선택: ", end='')
    for no, name in menuno:
        print("[%s] %s " %(no, name), end='    ')
    selnum = input("No?> ")
    mno = list(no for no, name in menuno)
    if selnum not in mno:
        print(">>선택 범위 초과!!")
        return -1
    else:
        return selnum

menuno = (['1', '메뉴-대분류'], ['2', '메뉴-소분류'], ['3', '주문'], ['4', '주문내역'], ['5', '전체-주문내역'],  ['x', '종료'])
```

```
[1] 메뉴-대분류  [2] 메뉴-소분류  [3] 주문  [4] 주문내역  [5] 전체-주문내역  [x] 종료
<작업 선택>  1


>> 대분류 메뉴 <<
[0] COFFEE
[1] LATTE
[2] TEA
[3] ADE
[4] JUICE
[5] SMOOTHIE
[6] MILK TEA

[1] 메뉴-대분류  [2] 메뉴-소분류  [3] 주문  [4] 주문내역  [5] 전체-주문내역  [x] 종료
<작업 선택>
```

IV. 파일의 활용

[과제-11/n] 전체 메뉴 구성 (2)

- 작업 선택 함수 sel_task()를 만들어서 원하는 작업을 선택할 수 있도록 하시오.
  - 작업 선택은 [1] 메뉴-대분류 [2] 메뉴-소분류 [3] 주문 [4] 주문내역 [5] 전체-주문내역 [x] 종료 로 구성

```
MENUGROUP, MENU, ORDER, ORDERLIST, TOTORDER, END = ('1', '2', '3', '4', '5', 'x')   #작업 선택 종류
########## Main ######
read_menu_file()        #파일 메뉴를 읽어 딕셔너리로 만들기
new_menugroup()       #menu로부터 대분류메뉴menugroup 딕셔너리 생성
while True:
    selno = sel_task()  #작업 번호 선택
    if selno == MENUGROUP:
        prt_menugroup(menugroup)
    elif selno == MENU:
        prt_menu(menu)
    elif selno == ORDER:
        selmlist = []    #주문내역 리스트 생성
        process_order()       #주문 처리
    elif selno == ORDERLIST:
        prt_result(selmlist)
    elif selno == TOTORDER:
        prt_order_file()   #파일로부터 전체 주문 내역을 읽어서 출력
    elif selno == END:
        break
    else:
        continue
######################
```

```
[1] 메뉴-대분류  [2] 메뉴-소분류  [3] 주문  [4] 주문내역  [5] 전체-주문내역  [x] 종료
<작업 선택>  1


>> 대분류 메뉴 <<
[0] COFFEE
[1] LATTE
[2] TEA
[3] ADE
[4] JUICE
[5] SMOOTHIE
[6] MILK TEA

[1] 메뉴-대분류  [2] 메뉴-소분류  [3] 주문  [4] 주문내역  [5] 전체-주문내역  [x] 종료
<작업 선택>
```

IV. 파일의 활용

실습 9-11 파일에 신규 회원 등록하기 code09-11.py

- 회원 정보는 'login.txt' 파일에 저장하고, 파일의 끝에 새로 입력하는 아이디와 비밀번호를 추가

① 아이디와 비밀번호를 입력받고, 추가 모드("a")로 파일을 열어 입력받은 데이터를 저장하고, 등록 완료를 알리는 메시지를 화면에 출력

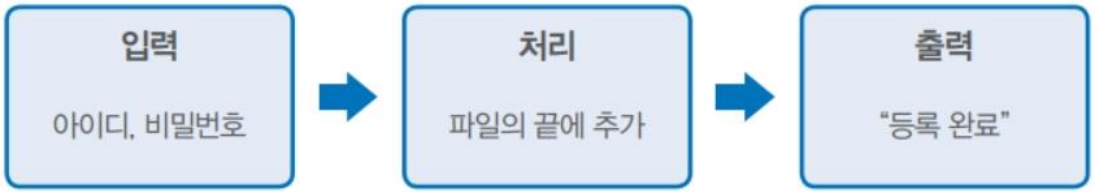


그림 9-17 신규 회원 등록 순서

② 동작 순서에 따라 프로그램을 다음과 같이 작성하고 저장

```
01  print("아이디와 비밀번호를 등록하세요.")
02  inid = input("아이디 : ")
03  inpwd = input("비밀번호 : ")
04
05  with open("c:/python/login.txt", "a") as f :       # 파일 끝에 추가("a")
```

IV. 파일의 활용

실습 9-11 파일에 신규 회원 등록하기 code09-11.py

② 동작 순서에 따라 프로그램을 다음과 같이 작성하고 저장

```
06        num = 0
07        f.writelines(inid + " " + inpwd + "\n")
08
09    print("등록되었습니다.")
```

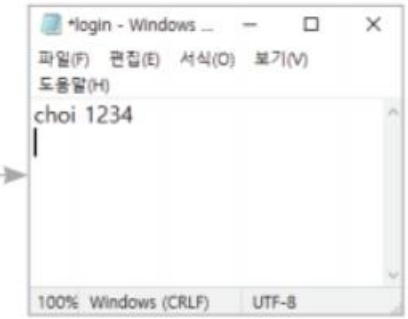
③ **파일이 없는 경우에는 파일을 먼저 생성한 후 데이터를 저장하고, 기존에 만들어 놓은 파일이 있는 경우 파일 끝에 데이터가 추가**

실행 결과

```
아이디와 비밀번호를 등록하세요.
아이디 : choi
비밀번호 : 1234
등록되었습니다.
```

첫 번째 회원 등록

파일을 만들고 데이터를 추가



IV. 파일의 활용

실습 9-11 파일에 신규 회원 등록하기 code09-11.py

③ 파일이 없는 경우에는 파일을 먼저 생성한 후 데이터를 저장하고, 기존에 만들어 놓은 파일이 있는 경우 파일 끝에 데이터가 추가

실행 결과

```
아이디와 비밀번호를 등록하세요.
아이디 : han
비밀번호 : 2222
등록되었습니다.
```

두 번째 회원 등록

그림 9-18 신규 회원 등록

IV. 파일의 활용

실습 9-12 회원 로그인 프로그램 만들기 code09-12.py

- 회원 정보는 'login.txt' 파일

① 동작 순서에 따라 프로그램을 다음과 같이 작성

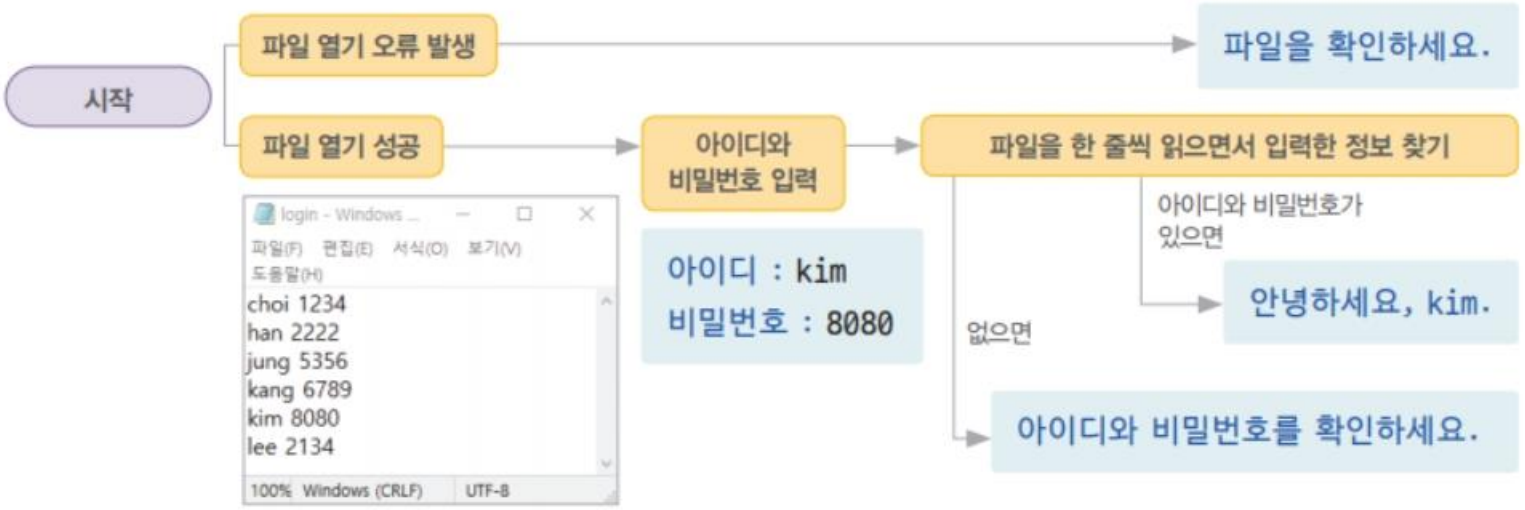


그림 9-19 회원 로그인 프로그램 실행 순서

② 파일 열기에 성공한 경우 아이디와 비밀번호를 입력을 받는다.

IV. 파일의 활용

실습 9-12 회원 로그인 프로그램 만들기 code09-12.py

② [한 줄씩 읽기]

파일을 읽을 때 for 문을 사용하면 한 줄씩 읽고 반환하는 동작을 반복하는데, 끝까지 다 읽으면 반복이 종료

첫 번째 줄을 읽었을 때 for 문의 변수 data에 저장된 문자열은 'choi 1234₩n'

```
for data in f :
```

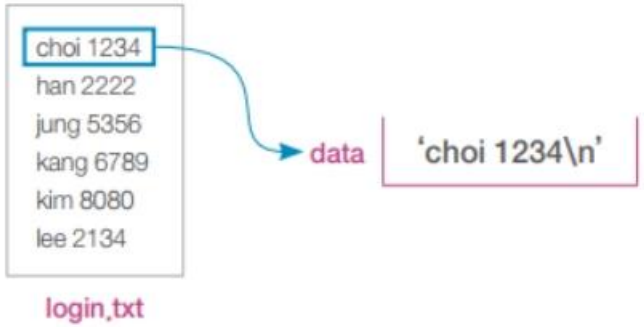


그림 9-20 첫 번째 줄 읽기

IV. 파일의 활용

실습 9-12 회원 로그인 프로그램 만들기 code09-12.py

② [**한 줄 안에 '₩n' 문자 제거**] rstrip( ) 메소드로 끝에 있는 줄바꿈 문자('₩n')를 제거하고, 아이디와 비밀번호만 data에 다시 저장

```
for data in f :
        data = data.rstrip()
```

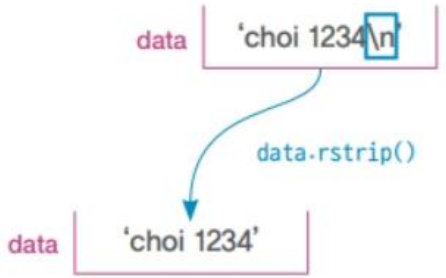


그림 9-21 줄바꿈 문자를 빼고 저장

[**데이터 항목 분리 후 리스트화**] split( )를 사용하여 문자열을 분리하고 items에 저장

```
for data in f :
        data = data.rstrip()
        items = data.split()
```

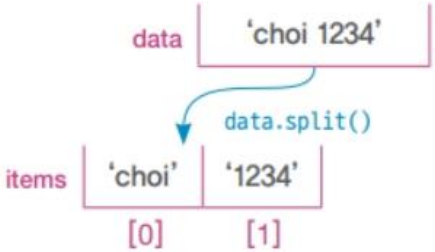


그림 9-22 아이디와 비밀번호를 분리

IV. 파일의 활용

실습 9-12 회원 로그인 프로그램 만들기 code09-12.py

③ 실행 순서와 파일 읽기 방법을 이용해 다음과 같이 전체 프로그램을 작성

```
01 try :
02     f = open("c:/python/login.txt", "r")
03 except :
04     print("파일을 확인하세요.")
05 else :
06     inid = input("아이디 입력 : ")
07     inpwd = input("비밀번호 입력 : ")
08     result = 0                    # 아이디와 비밀번호를 찾으면 1로 변경
09
10     for data in f :
11         data = data.rstrip()      # 읽은 문자열의 오른쪽에 있는 '\n'을 제거
12         items = data.split()      # 공백을 기준으로 잘라서 리스트에 저장
13
14         if items[0] == inid and items[1] == inpwd :
15             print("안녕하세요, %s." % inid)
```

IV. 파일의 활용

실습 9-12 회원 로그인 프로그램 만들기 code09-12.py

③ 실행 순서와 파일 읽기 방법을 이용해 다음과 같이 전체 프로그램을 작성

```
16              result = 1          # 일치하는 아이디와 비밀번호를 찾음
17              break
18
19      f.close()
20      if result == 0 :            # 일치하는 아이디와 비밀번호가 없음
21          print("아이디와 비밀번호를 확인하세요.")
```

④ 저장한 프로그램을 다양하게 실행시켜 결과를 확인

```
파일을 확인하세요.
```
파일이 지정 위치에 없는 경우

```
아이디 입력 : lee
비밀번호 입력 : 2134
안녕하세요, lee.
```
아이디와 비밀번호를 모두 올바르게 입력한 경우

```
아이디 입력 : lee
비밀번호 입력 : 5678
아이디와 비밀번호를 확인하세요.
```
비밀번호를 잘못 입력한 경우

IV. 파일의 활용

실습 9-13 단어 찾기와 바꾸기 code09-13.py

- 특정 단어가 파일('peom.txt')에서 사용된 횟수를 측정하고, 다른 단어로 바꾸는 프로그램
- 검색할 단어와 바꿀 단어를 각각 입력받아 실행

① 파일 내용을 먼저 읽어서 보여주고, 검색할 단어와 바꿀 단어를 입력 받음
검색할 단어가 파일에 몇 번 사용되는지 검사하고, 바꿀 단어로 교체

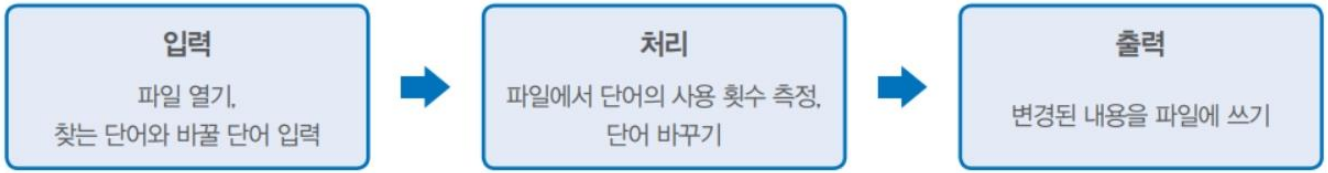


그림 9-23 단어 찾기와 바꾸기 실행 순서

IV. 파일의 활용

실습 9-13 단어 찾기와 바꾸기 code09-13.py

② 파일을 읽고 수정하기 위해서는 "r+" 모드 로 파일을 열기

파일 읽기 오류가 없는 경우에만 프로그램이 실행되도록 예외 처리 구문을 적용

```
01 try :
02     f = open("poem.txt", "r+", encoding = "utf-8")
03 except :
04     print("파일을 확인하세요.")          # 파일 읽기 오류가 발생하면 실행
05 else :
06     text = f.read()                    # 파일 읽기
07     print(text)
08
09     org = input("\n찾을 내용 : ")
10     new = input("바꿀 내용 : ")
11
12     print("%d 번을 모두 바꿉니다." % text.count(org))
13     text = text.replace(org, new)
14     f.seek(0)                          # 파일 포인터를 처음으로 이동
15     f.write(text)                      # 파일 수정
16     f.close()
```

파일의 인코딩 형식에 따라 생략해도 됨

IV. 파일의 활용

실습 9-13 단어 찾기와 바꾸기 code09-13.py

③ 저장한 프로그램을 실행

```
흔들리며 피는 꽃
도종환/

흔들리지 않고 피는 꽃이 어디 있으랴
이 세상 그 어떤 아름다운 꽃들도
다 흔들리면서 피었나니
흔들리면서 줄기를 곧게 세웠나니
흔들리지 않고 가는 사랑이 어디 있으랴
젖지 않고 피는 꽃이 어디 있으랴
이 세상 그 어떤 빛나는 꽃들도
다 젖으며 젖으며 피었나니
바람과 비에 젖으며
꽃잎 따뜻하게 피웠나니
젖지 않고 가는 삶이 어디 있으랴

찾을 내용 : 꽃
바꿀 내용 : 사람
6 번을 모두 바꿉니다.
```

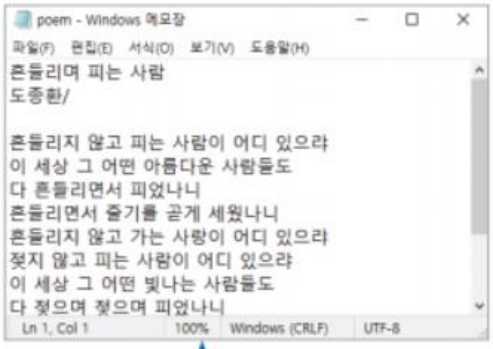


그림 9-24 변경된 내용 확인

IV. 파일의 활용

실습 9-14 파일 복사하기 code09-14.py

① 원본과 복사본 파일 이름을 입력받고, 원본 파일의 내 용을 모두 읽어 복사본에 그대로 쓰기

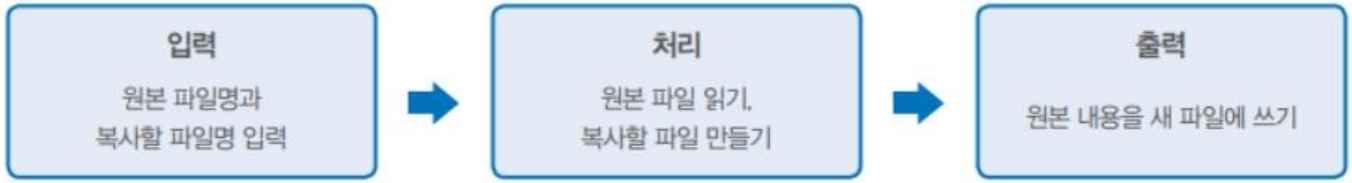


그림 9-25 파일 복사 실행 순서

② 원본 파일이 바이너리 파일인 경우에는 줄 단위로 읽을 수 없기 때문에, 전체 내용을 바이트 단위로 읽고 쓰는 "rb"와 "wb" 모드로 파일을 처리

```
orgfile = open(org, "rb")          # 바이트 단위로 파일 읽기
newfile = open(new, "wb")          # 바이트 단위로 파일 쓰기
```

③ 원본 파일이 없 는 경우에는 예외 처리 구문으로 '파일 확인' 메시지를 출력

```
01  org = input("원본 파일명 : ")
02  new = input("복사할 파일명 : ")
```

IV. 파일의 활용

실습 9-14 파일 복사하기 code09-14.py

③ 원본 파일이 없 는 경우에는 예외 처리 구문으로 '파일 확인' 메시지를 출력

```
03
04  try :
05      orgfile = open(org, "rb")              # 바이트 단위로 파일 읽기
06  except :
07      print("파일을 확인하세요.")
08  else :
09      data = orgfile.read()
10      orgfile.close()
11
12      with open(new, "wb") as newfile :      # 바이트 단위로 파일 쓰기
13          newfile.write(data)
14      print("파일이 복사되었습니다.")
```

④ 저장한 프로그램을 실행

```
원본 파일명 : photo.png
본사본 파일명 : photo(2).png
파일이 복사되었습니다.
```

바이너리 파일의 복사

```
원본 파일명 : member.txt
본사본 파일명 : me2.txt
파일이 복사되었습니다.
```

텍스트 파일의 복사

```
원본 파일명 : p.jpg
복사할 파일명 : p(2).jpg
파일을 확인하세요.
```

파일 오류로 예외 처리

Thank You !