23장

진급 프로젝트:친구 분석

23장 진급 프로젝트:친구 분석

23. 프로젝트 문제

» 친구 이름과 전화번호가 들어 있는 파일에서 입력 데이터를 정해진 형식에 맞게 읽는 프로그램을작성하라. 작성한 프로그램은 어떤 방법으로든 친구 정보를 메모리에 저장하고 분석해야 한다. 예를 들어 전화번호의 지역 번호를 사용해 친구들이 사는 지역을 표시할 수도 있고, 친구들이 사는 주(state)가 전부 몇 개인지 셀 수도 있다.

23.1 파일 읽기

23.1.1 파일 형식

» 파일의 각 줄에 있는 정보를 변수에 넣는 read_file이라는 함수를 만들자.

» read_file은 사용자가 다음 형식에 맞춰 파일에 정보를 입력했다고 가정한다.
각 줄마다 친구 정보가 들어간다.

```
친구 1 이름
친구 1 전화번호
친구 2 이름
친구 2 전화번호
〈계속〉
```

- 이름과 전화번호가 서로 다른 줄에 있다는 사실에 유의하라. 이는 작성하는 프로그램이 각 줄의 마지막을 뜻하는 새줄 문자(\n)를 인식해야 한다는 뜻이다.

23.1.1 파일 형식

파이썬은 새줄 문자를 쉽게 처리할 수 있다.

- 파일 형식이 이렇다는 사실을 알면 파일을 한 줄씩 읽을 수 있다.
- 첫 번째 줄부터 시작해 두 줄 간격으로 정보를 튜플에 저장한다.
- 마찬가지로 두 번째 줄부터 시작해 두 줄 간격으로 정보를 다른 튜플에 저장한다.
- 결과적으로 두 튜플에는 다음과 같은 내용이 들어간다.

```
(친구 1 이름, 친구 2 이름, <계속>)
(친구 1 전화번호, 친구 2 전화번호, <계속>)
```

- 두 튜플의 0번 인덱스 위치에 있는 정보는 모두 친구 1에 대한 정보이며, 1번 인덱스 위치에 있는 정보는 모두 친구 2에 대한 정보다. 이 같은 패턴이 맨 마지막 친구까지 반복된다.
- 이 파일을 처리하려면 모든 줄을 하나씩 처리해야 한다.
- 이 내용을 보고 각 줄을 처리하는 루프가 필요하다고 생각해야 한다. 루프는 파일의 각 줄을 문자열로 읽어 줄 것이다.

23.1.2 새줄 문자

» 새줄 문자는 \n으로 표현한다.

```
print("no newline")
```

```
no newline
```

```
print("yes newline₩n")
```

```
yes newline
```

```

```

» 출력된 내용과 프롬프트 사이가 한 줄 더 떨어져 있다. 한 줄이 떨어진 이유는 역슬래시(\)다음에 영어 알파벳 n을 넣은 특수한 문자 조합이 파이썬에게는 줄을 바꿔야 한다는 뜻이기 때문이다.

23.1.3 새줄 문자 제거하기

- 파일에서 한 줄을 읽어오면 그 줄에는 눈에 보이는 문자 외에 새줄 문자가 딸려온다.
- 이 특수 문자(새줄 문자)를 제외한 나머지 문자들만 저장해야 하므로, 튜플에 정보를 저장하기 전에 특수 문자를 제거해야 한다.
- 파일에서 읽어온 각 줄은 문자열이다. 따라서 새줄 문자를 제거하는 가장 쉬운 방법은 \n을 빈 문자열 ""으로 바꾸는 것이다. 이렇게 하면 새줄 문자를 제거한 결과를 얻을 수 있다

```
word = "bird\n" ---- 새줄 문자가 들어 있는 문자열을 저장한 변수를 만듦
print(word) ---- 출력하면 빈 줄이 한 줄 더 보임
word = word.replace("\n", "") ---- 새줄 문자를 빈 문자열로 대치하고 그 결과를 같은 변수에 저장
print(word) ---- 출력하면 빈 줄이 보이지 않음
```

코드 23-1 새줄 문자 제거하기

23.1.4 정보를 저장하기 위해 튜플 사용하기

» 다음 단계는 각 데이터를 변수에 저장하는 것이다. 많은 데이터가 모인 컬렉션이 필요하므로 한 튜플에 친구 이름을 모두 넣고, 다른 튜플에 전화번호를 모두 넣자.

» 한 줄을 읽어올 때마다 새 정보를 튜플에 추가한다. 튜플에 원소를 추가하면 이미 튜플에 들어 있던 원소들 맨 뒤에 새로 추가한 원소가 덧붙여진 튜플을 얻는다.

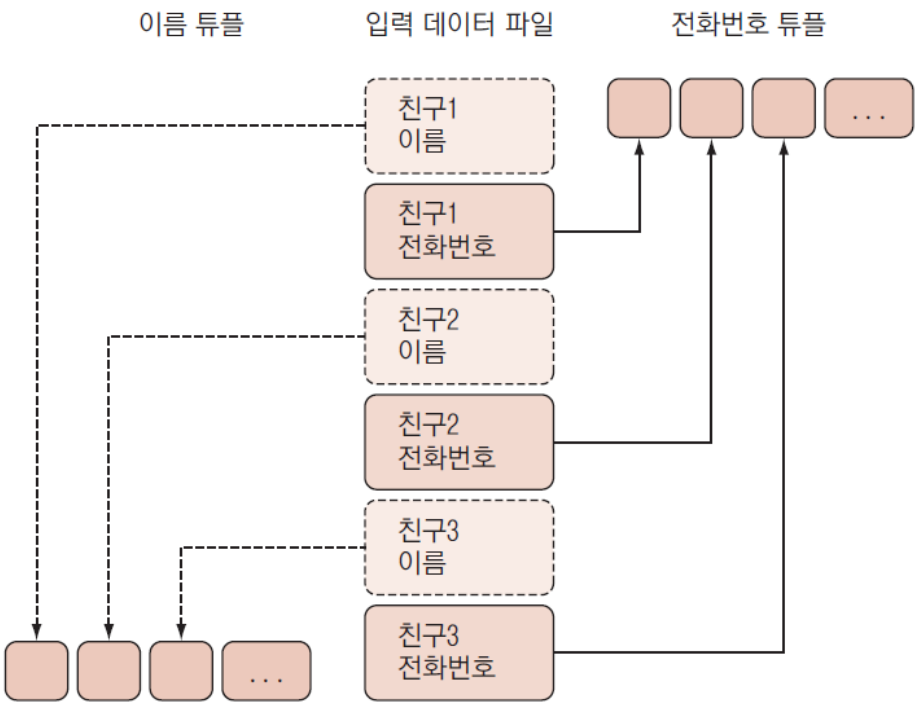

그림 23-1
입력 데이터는 한 사람에 대한 정보가 두 줄로 되어 있다. 첫 번째 줄은 친구 이름, 두 번째 줄은 친구 전화번호, 세 번째 줄은 다른 친구의 이름, 네 번째 줄은 그 친구의 전화번호 등의 패턴이다.

첫 번째 줄부터 시작해서 한 줄 건너 하나씩 친구 이름을 튜플에 저장한다. 두 번째 줄부터 시작해서 한 줄 건너 하나씩 친구 전화번호를 다른 튜플에 저장한다

23.1.5 반환할 값

» 작성해야 할 함수는 파일을 읽고, 정보를 정리한 후, 정리된 정보를 돌려줘야 한다.
튜플이 두 개이므로 다음과 같이 튜플의 튜플을 반환한다.

```
((친구1 이름, 친구2 이름, ... ), (친구1 전화, 친구2 전화, ...))
  -------------------------    -------------------------
        첫 번째 튜플                  두 번째 튜플
```

» 튜플의 튜플을 반환하는 이유는 함수가 오직 하나의 값만 반환할 수 있기 때문이다.

23.1.5 반환할 값


```
def read_file(file):
    """
    file, 파일 객체
    첫 번째 줄부터 시작해서 매 두 번째 줄을 튜플에 넣는다
    두 번째 줄부터 시작해서 매 두 번째 줄을 다른 튜플에 넣는다
    Returns 두 튜플로 이뤄진 튜플
    """
    first_every_2 = ()
    second_every_2 = ()
    line_count = 0

    for line in file:
        stripped_line = line.replace("\n", "")
        if line_count%2 == 0:
            first_every_2 += (stripped_line,)
        elif line_count%2 == 1:
            second_every_2 += (stripped_line,)
        line_count += 1
    return (first_every_2, second_every_2)
```

독스트링

이름과 전화번호를 저장할 빈 튜플

줄 번호를 저장할 변수

파일의 모든 줄에 대해 루프 수행

새줄 문자 없앰

홀수 번째에 있는 줄

이름을 튜플에 추가

짝수 번째에 있는 줄

전화번호를 튜플에 추가

줄 번호를 1증가 시킴

튜플의 튜플 반환

friends.txt* listings_ch

```
Ana
801-456-789
Ben
609 4567890
Cory
(206)-345-2619
Danny
6095648765
```

코드 23-2
파일에서 이름과 전화번호 읽기

23.2 사용자 입력 정리하기

23.2 사용자 입력 정리하기

» 전화번호 형식을 특별히 지정하지 않았기 때문에 사용자들이 전화번호를 어떤 형식으로 입력했을 지 알 수 없다.

- 중간에 대시(-)나 괄호나 공백이 들어 있을 수도 있고, 숫자가 아닌 이상한 다른 문자가 들어 있지 말라는 법도 없다.
- 따라서 전화번호 정보를 분석하기 전에 먼저 모든 전화번호를 일정한 형식에 맞춰 정리해야 한다.
- 이 과정에서 특수 문자를 모두 없애고, 숫자들을 한꺼번에 연결해야 한다.

```
def sanitize(some_tuple):
    """
    phones, 문자열이 들어 있는 튜플
    튜플에 들어 있는 문자열에서 모든 대시, 공백, 괄호를 없앤다
    Returns 대시, 공백, 괄호를 없앤 문자열로 이뤄진 새 튜플
    """
    clean_string = () ---- 불필요한 문자를 빈 문자열로 바꿈
    for st in some_tuple:
        st = st.replace(" ", "")
        st = st.replace("-", "") ---
        st = st.replace("(", "")    |--- 빈 튜플
        st = st.replace(")", "") ---
        clean_string += (st,) ---- 정리한 전화번호를 튜플에 추가
    return clean_string ---- 튜플을 반환
```

코드 23-3 전화번호에서 대시, 공백, 괄호 없애기

23.3 지금까지 만든 코드 테스트 및 디버깅하기

23.3.2 이름과 전화번호가 들어 있는 텍스트 파일 만들기

- 파일을 처리할 때는 '파일 객체'를 만들어야 한다.

- 스파이더에서 새 파일을 만들자. Read_file 함수가 원하는 형식에 맞춰 데이터를 몇 줄 입력하자. 먼저 이름을 넣고, 다음 줄에 전화번호를 넣어라. 그 다음 줄에 다시 다른 이름을 넣고, 다음 줄에 다른 전화번호를 넣어라.

```
짱구
000 123-4567
짱구 엄마
(890) 098-7654
짱구 아빠
321-098-0000
```

- 이제 이 파일을 friends.txt 또는 원하는 다른 이름으로 저장하라. 저장한 파일이 작성한 파이썬 프로그램과 같은 디렉터리에 있어야 한다. 이 파일은 단순한 텍스트 파일로 작성한 프로그램에서 이 파일을 읽을 것이다

23.3.3 파일을 열어서 읽기

» 파일 이름을 사용해 파일을 열면(open) 파일 객체를 만들 수 있다.

» 따라서 읽고 싶은 파일 이름을 인자로 하여 open이라는 함수를 호출한다.

- 파일은 여러분이 작성한 .py 프로그램 파일과 같은 위치에 있어야 한다.

```
friends_file = open('friends.txt') ---- 파일 열기
(names, phones) = read_file(friends_file) ---- 함수 호출

print(names)
print(phones) ---- 함수가 반환한 내용을 사용자에게 표시

clean_phones = sanitize(phones) ---- 함수가 잘 작동하는지 테스트
print(clean_phones) ---- 함수가 반환한 내용을 사용자에게 표시

friends_file.close() ---- 파일 닫기
```

코드 23-4 파일에서 이름과 전화번호 읽기

23.3.3 파일을 열어서 읽기

코드 23-4는 파일을 열고, 23.1절과 23.2절에서 만든 함수를 호출해서 얻은 결과가 여러분이 원하는 결과와 같은지 검증하는 방법을 보여준다.

- open 함수는 friends.txt라는 이름의 파일을 연다.
- open은 파일 객체를 만들어내며, read_file()은 그 파일 객체를 인자로 받는다.
- read_file()은 튜플의 튜플을 반환한다. 이 반환 값을 이름과 전화번호를 담는 두 튜플로 나눠 저장한 다음, 각각을출력해서 제대로 이름과 번화번호가 분리됐는지 살펴본다.
- sanitize 함수가 작동하는지 보려면 전화번호가 들어 있는 튜플을 인자로 넘기면 된다. 여기서는함수가 반환한 결과를 출력해서 원하는 대로 전화번호의 특수 문자들이 정리되었는지 살펴본다.

23.4 함수 재사용하기

23.4 함수 재사용하기

» 파일에서 데이터를 읽는 함수를 이미 작성했다. 그 함수를 사용해 이름과 전화번호를 두 튜플에 나눠 담았다. 파일에 들어 있는 데이터의 구조가 같다면 같은 함수를 다른 데이터 집합에 사용해도 된다.

23.4 함수 재사용하기

» 사용자에게 전화번호를 받을 것이므로, 전화의 지역 코드와 그 코드에 대응하는 주 이름을 연결해주는 파일이 있다고 가정하자. 이 파일은 이름과 전화번호가 담겨있는 파일과 형식이 같다.

```
지역 코드1
주 이름1
지역 코드 2
주 이름2
<계속>
```

» 이 정보가 map_areacodes_states.txt라는 텍스트 파일에 들어 있고, 이 파일의 앞부분을 예로 든다면 다음과 같을 것이다

```
201
New Jersey
202
Washington D.C.
203
Connecticut
204
<계속>
```

» 이 파일이 있으면 read_data라는 같은 함수를 사용해 지역 코드와 주 이름을 연결할 수 있다.

```
map_file = open('map_areacodes_states.txt')
(areacodes, places) = read_file(map_file)
```

23.5 정보 분석하기

23.5.1 명세

» 친구들 이름, 전화번호, 지역 코드, 지역 코드에 대응하는 각 주 이름이 들어간 네 가지 튜플을 입력으로 받는 analyze_friends라는 함수를 작성하라.

» analyze_friends 함수는 정보를 표시하기만 하고, 값을 반환하지는 않는다.
다음과 같은 친구 정보를 받았다면

```
Ana
801-456-789
Ben
609 4567890
Cory
(206)-345-2619
Danny
6095648765
```

» 다음과 같이 출력한다.

```
You have 4 friends!
They live in ('Utah', 'New Jersey', 'Washington')
```

- 친구는 4명이지만 두 명이 같은 주에 살고 있으므로 주는 3개만 표시된다.

23.5.2 도우미 함수

» 정보를 분석하는 작업이 복잡해서 몇 가지 도우미 함수를 작성해야 한다. 도우미 함수(helper function)는 다른 함수가 작업을 수행할 수 있도록 돕는 함수를 말한다.

» get_unique_area_codes 함수

- 이 함수는 아무 매개변수도 받지 않고, 지역 코드를 중복 없이 반환한다. 이때 지역 코드 간의 특별한 순서도 없다. 이렇게 중복이 없는 경우 지역 코드들이 유일하다(unique)고 한다.

```
def get_unique_area_codes():
    """
    Returns phones에 있는 지역 코드 목록(중복된 경우 1개만 포함)
    """
    area_codes = () ---- 유일한 지역 코드를 저장하기 위한 튜플
    for ph in phones: ---- 모든 지역 코드를 처리. phones 변수는 analyze_friends의 매개변수임
        if ph[0:3] not in area_codes: ---- 지역 코드가 이미 들어 있지 않은지 검사
            area_codes += (ph[0:3],) ---- 유일한 지역 코드를 담는 튜플에 지역 코드가
                                         하나만 들어 있는 튜플을 덧붙임
    return area_codes
```

코드 23-5
중복 없이 지역 코드를 모으는 도우미 함수

23.5.2.2 지역 코드에서 주 이름 얻기

get_states 함수는 지역 코드가 들어 있는 튜플을 받아서 각 지역 코드에 해당하는 주 이름이 들어 있는 튜플을 반환한다.

- 이 함수도 analyze_friends에 내포된 함수이므로 analyze_friends 함수의 모든 매개변수를 사용할 수 있다.

```
def get_states(some_areacodes):
    """
    some_areacodes, 지역 코드들이 들어 있는 튜플
    Returns 지역 코드에 해당하는 주 이름이 들어 있는 튜플
    """
    states = ()
    for ac in some_areacodes:
        if ac not in all_areacodes: -------- 사용자가 잘못된 지역번호를 입력함. all_areacodes 변수는
            states += ("BAD AREACODE",)      analyze_friends의 매개변수임
        else:
            index = all_areacodes.index(ac) ---- 튜플에서 지역 코드의 위치를 찾음
            states += (all_places[index],) ---- 지역 코드의 위치를 활용해 주 이름을 가져옴
    return states
```

코드 23-6
중복을 제거한 지역 코드로 이뤄진 튜플에서 각 주 이름을 찾는 도우미 함수

23.5.2.2 지역 코드에서 주 이름 얻기

» analyze_friends에서 사용할 도우미 함수는 이렇게 둘 뿐이다. 두 도우미 함수를 사용하면 analyze_friends 함수를 더 간단하고 읽기 좋게 만들 수 있다.

```
def analyze_friends(names, phones, all_areacodes, all_places):
    """
    names, 친구 이름이 들어 있는 튜플
    phones, 친구 전화번호(특수 문자는 없앰)가 들어 있는 튜플
    all_areacodes, 지역 코드가 들어 있는 튜플
    all_places, 지역 코드에 대한 주 이름이 들어 있는 튜플
    친구가 몇 명인지와 친구들의 전화번호가 속한 주 목록을 표시한다
    (이때 같은 주는 한 번만 표시한다)
    """

    def get_unique_area_codes():
        """
        Returns phones에 있는 지역 코드 목록(중복된 경우 1개만 포함)
        """
        area_codes = ()
        for ph in phones:
            if ph[0:3] not in area_codes:
                area_codes += (ph[0:3],)
        return area_codes
```

23.5.2.2 지역 코드에서 주 이름 얻기


```
def get_states(some_areacodes):
    """
    some_areacodes, 지역 코드가 들어 있는 튜플
    Returns 지역 코드에 해당하는 주 이름이 들어 있는 튜플
    """
    states = ()
    for ac in some_areacodes:
        if ac not in all_areacodes:
            states += ("BAD AREACODE",)
        else:
            index = all_areacodes.index(ac)
            states += (all_places[index],)
    return states

num_friends = len(names) ---- 친구 인원 수
unique_area_codes = get_unique_area_codes() ---- 유일한 지역 코드만 남김
unique_states = get_states(unique_area_codes) ---- 유일한 지역 코드에 해당하는 주 이름을 가져옴
    print("You have", num_friends, "friends!") ---- 친구 수를 출력
    print("They live in", unique_states) ---- 유일한 주 이름을 찍음
            ------- 반환할 값이 없음 -------
```

코드 23-7
analyze_friends
함수의 본문

23.5.2.2 지역 코드에서 주 이름 얻기

» 이 프로그램의 마지막 단계는 코드 23-8과 같이 두 파일을 읽고, 데이터를 분석하는 함수를 호출한 다음, 파일을 닫는 것이다.

```
friends_file = open('friends.txt') ---- 프로그램과 같은 디렉터리에 있는 파일을 엶
(names, phones) = read_file(friends_file) ---- 두 데이터 집합을 읽을 때 같은 함수를 활용

areacodes_file = open('map_areacodes_states.txt') ---- 프로그램과 같은 디렉터리에 있는 파일을 엶
(areacodes, states) = read_file(areacodes_file) ---- 두 데이터 집합을 읽을 때 같은 함수를 활용

clean_phones = sanitize(phones) ---- 전화번호 데이터를 정규화(정해진 형식에 따라 정리)
analyze_friends(names, clean_phones, areacodes, states) ---- 대부분의 작업을 수행하는 함수를 호출

friends_file.close()
areacodes_file.close()  ---- 파일을 닫음
```

코드 23-8
파일을 읽고, 내용을 분석하고, 파일을 닫는 명령

23.6 요약

23.6 요약

- 파이썬에서 파일을 열어서 그 내용을 처리(각 줄을 문자열로 읽기)할 수 있다.
- 코드를 잘 조직화시킬 때는 함수가 유용하다. 한 번 함수를 작성하면 그 함수를 여러 다른 입력에 대해 몇 번이고 사용할 수 있다.
- 함수를 자주 테스트해야 한다. 함수를 작성하고 바로 테스트하라. 몇 가지 함수를 작성한 뒤에는 함수들이 서로 잘 연동해 동작하는지 확인해야 한다.
- 함수 안에 다른 함수를 내포할 수 있다. 프로그램 전체의 목표를 달성할 때는 필요하지 않지만 특정 작업을 완료하기 위해 꼭 필요한 작업이 있다면, 그 작업을 내포 함수로 포함시키면 좋다.