

4.제 어 문

contents

- ▶ 입출력 함수
- ▶ 조건문(If)문
- ▶ while 문
- ▶ for문
- ▶ collection 모듈

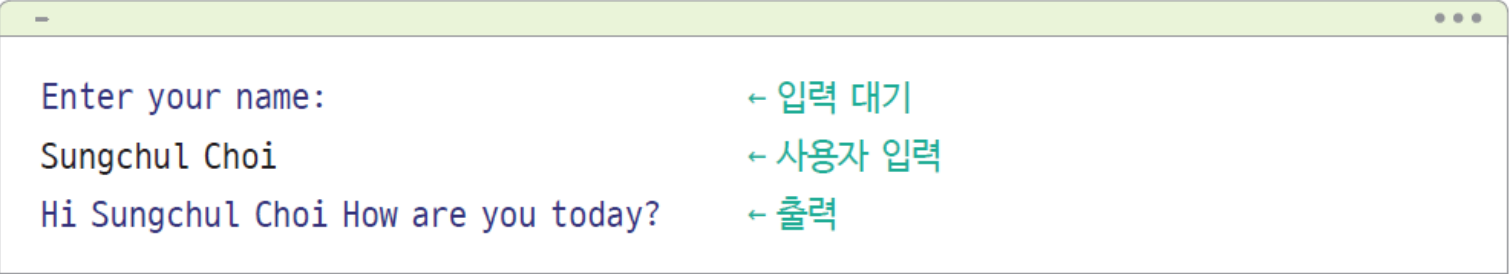


1. 입출력 함수

▶ 표준 입력 함수: input() 함수

- ▶ input() 함수: 표준 입력 함수로, 사용자가 문자열을 콘솔 창에 입력할 수 있게 해 준다.

```
1 print("Enter your name:")  
2 somebody = input()          # 콘솔 창에서 입력한 값을 somebody에 저장  
3 print("Hi", somebody, "How are you today?")
```



```
Enter your name:                ← 입력 대기  
Sungchul Choi                  ← 사용자 입력  
Hi Sungchul Choi How are you today? ← 출력
```

1. 입출력 함수

▶ 표준 출력 함수: print() 함수

- ▶ print() 함수: 표준 출력 함수로, 결과를 화면에 출력하는 함수이다.

```
>>> print("Hello World!", "Hello Again!!!")    # 콤마 사용  
Hello World! Hello Again!!!                    # 실행 시 두 문장이 연결되어 출력
```

```
1 temperature = float(input("온도를 입력하세요: ")) # 입력 시 바로 형 변환  
2 print(temperature)
```



```
-  
온도를 입력하세요: 103    ← 입력 대기 및 사용자 입력  
103.0                     ← 출력
```

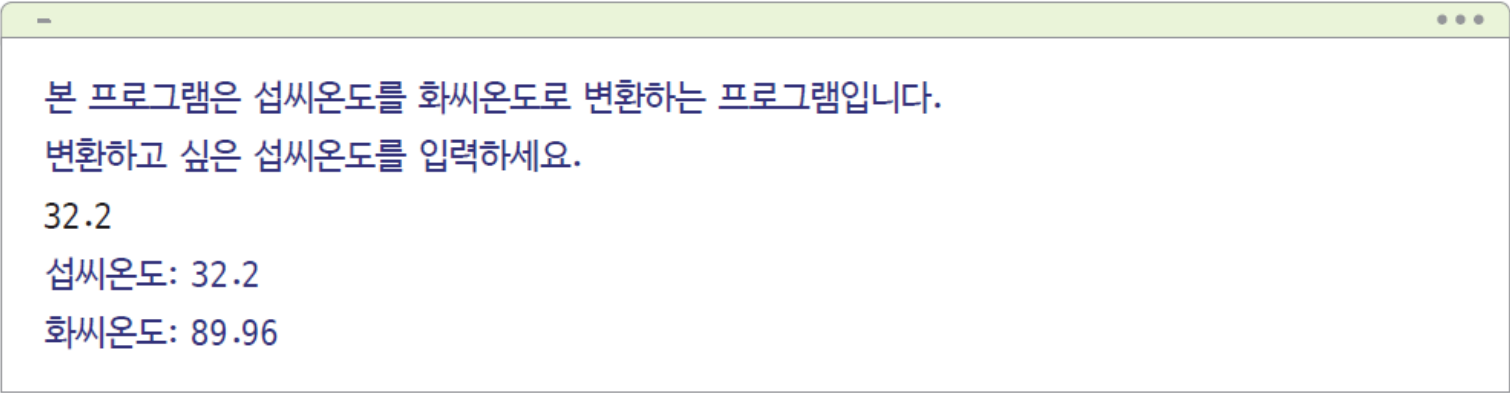
1. 입출력 함수

▶ 실습 내용

- ▶ input() 함수, print() 함수, 간단한 사칙연산을 이용하여 화씨온도 변환기(Fahrenheit temperature converter) 프로그램 작성

$$\text{화씨온도(°F)} = (\text{섭씨온도(°C)} * 1.8) + 32$$

▶ 실행결과



본 프로그램은 섭씨온도를 화씨온도로 변환하는 프로그램입니다.
변환하고 싶은 섭씨온도를 입력하세요.
32.2
섭씨온도: 32.2
화씨온도: 89.96

1. 입출력 함수

▶ 문제 해결

▶ 화씨온도 변환기 코드

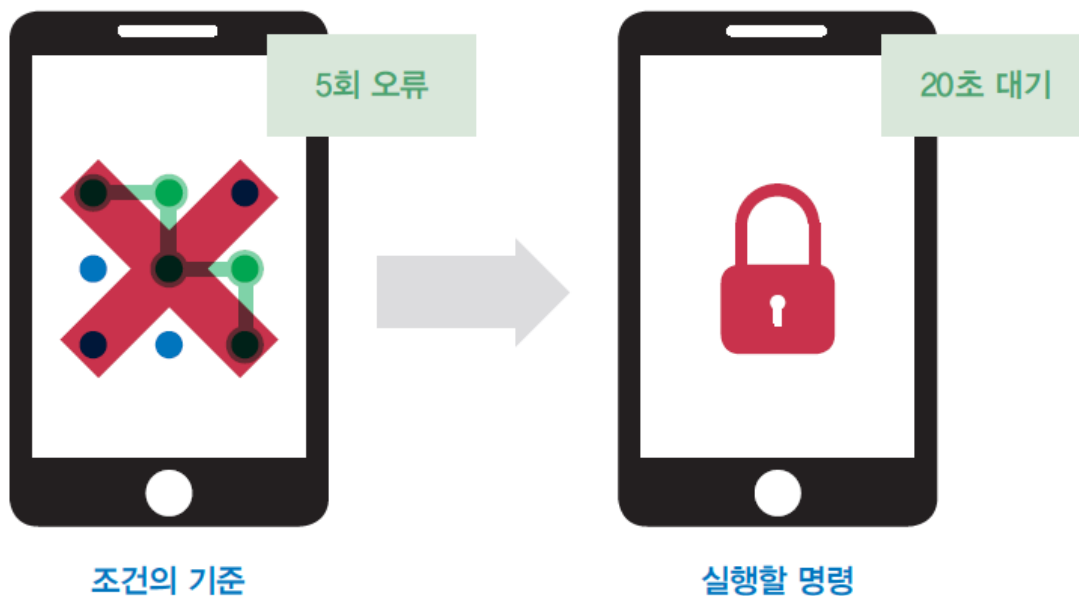
```
1 print("본 프로그램은 섭씨온도를 화씨온도로 변환하는 프로그램입니다.")
2 print("변환하고 싶은 섭씨온도를 입력하세요.")
3
4 celsius = input()
5 fahrenheit = (float(celsius) * 1.8 ) + 32
6
7 print("섭씨온도:", celsius)
8 print("화씨온도:", fahrenheit)
```



2. 조건(if)문

▶ 조건문

- ▶ 조건문(conditional statement): 조건에 따라 특정 동작을 하도록 하는 프로그래밍 명령어.
- ▶ 파이썬에서 조건문을 사용하기 위해 [if, if ~ else, if ~elif ~else]등의 명령 키워드를 사용. (switch 없음)
- ▶ 스마트폰 잠금 해제 패턴이 5회 틀리면, 20초 동안 대기 상태로 만들어라.



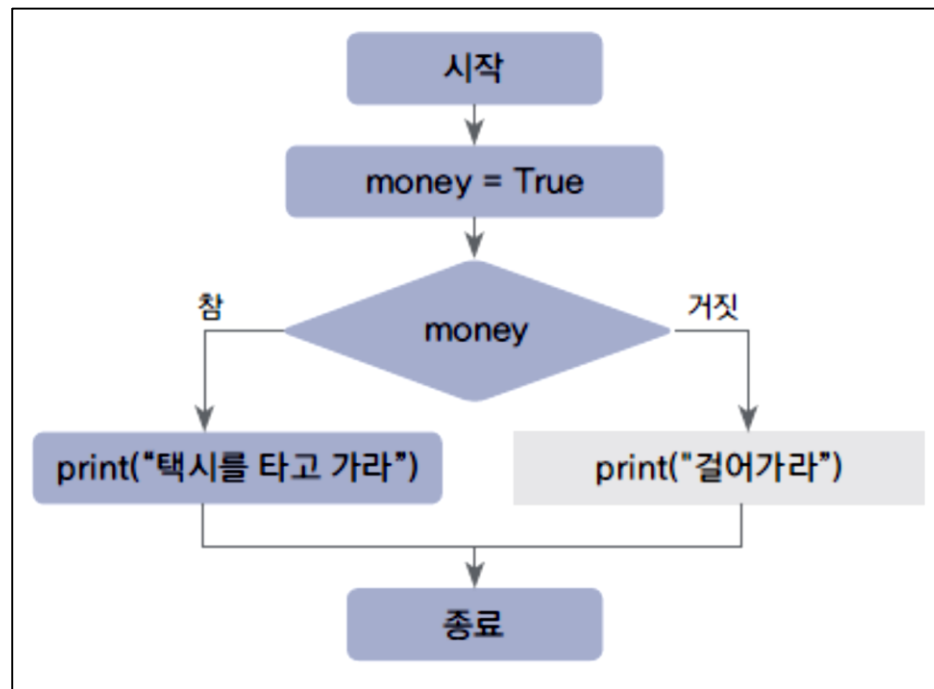
2. 조건(if)문

▶ if 문은 왜 필요할까?

- ▶ 주어진 조건을 판단한 후 그 상황에 맞게 처리해야 할 경우

'돈이 있으면 택시를 타고 가고, 돈이 없으면 걸어간다.'

```
>>> money = True
>>> if money:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
```



2. 조건(if)문

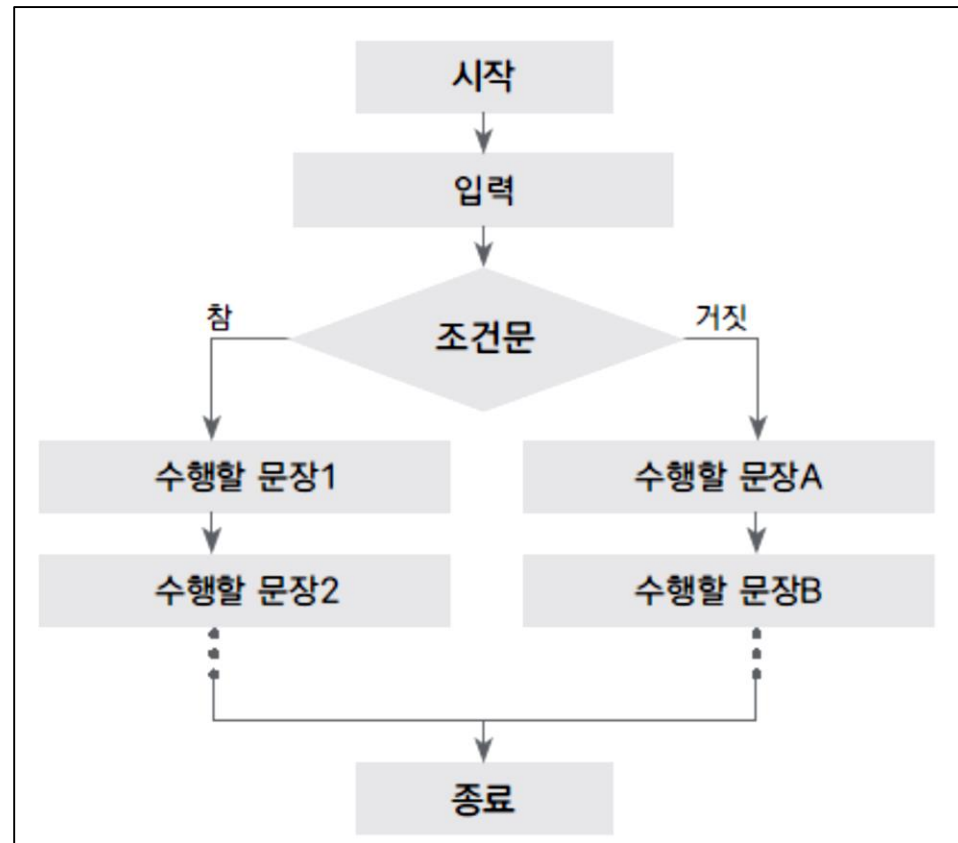
▶ 조건문 구조

▶ if 구문

if <조건>:	# if를 쓰고 조건 삽입 후 ':' 입력
<수행 명령 1-1>	# 들여쓰기 후, 수행 명령 입력
<수행 명령 1-2>	# 같은 조건에서 실행일 경우 들여쓰기 유지

▶ if ~ else 구분

if <조건>:	# if를 쓰고 조건 삽입 후 ':' 입력
<수행 명령 1-1>	# 들여쓰기 후, 수행 명령 입력
<수행 명령 1-2>	# 같은 조건에서 실행일 경우 들여쓰기 유지
else:	# 조건이 불일치할 경우 수행할 명령
<수행 명령 2-1>	# 조건 불일치 시 수행할 명령 입력
<수행 명령 2-2>	# 조건 불일치 시 수행할 명령 들여쓰기 유지



2. 조건(if)문

▶ 조건문

▶ if ~ elif ~ else 문

```
if <조건1> :  
    수행 명령어 1  
elif <조건2>:  
    수행 명령어 2-1  
...  
elif <조건n-1> :  
    수행 명령어 (n-1)  
else :  
    수행 명령어 n
```



2. 조건(if)문

▶ 조건의 판단 : 비교 연산자

비교 연산자	비교 상태	설명
$x < y$	~보다 작음	x가 y보다 작은지 검사
$x > y$	~보다 큼	x가 y보다 큰지 검사
$x == y$	같음	x와 y의 값이 같은지 검사
$x \text{ is } y$	같음(메모리 주소)	x와 y의 메모리 주소가 같은지 검사
$x != y$	같지 않음	x와 y의 값이 같지 않은지 검사
$x \text{ is not } y$	같지 않음(메모리 주소)	x와 y의 메모리 주소가 같지 않은지 검사
$x \geq y$	크거나 같음	x가 y보다 크거나 같은지 검사
$x \leq y$	작거나 같음	x가 y보다 작거나 같은지 검사



2. 조건(if)문

▶ 조건 판단 논리 연산자

연산자	설명	예시
and	두 값이 모두 참일 경우 True, 그렇지 않을 경우 False	$(7 > 5)$ and $(10 > 5)$ 는 True $(7 > 5)$ and $(10 < 5)$ 는 False
or	두 값 중 하나만 참일 경우 True, 두 값 모두 거짓일 경우 False	$(7 < 5)$ or $(10 > 5)$ 는 True $(7 < 5)$ or $(10 < 5)$ 는 False
not	값을 역으로 반환하여 판단	not $(7 < 5)$ 는 True not $(7 > 5)$ 는 False



2. 조건(if)문

▶ 들여쓰기 방법 알아보기

- ▶ if 문을 만들 때는 if 조건문 바로 다음 문장부터 모든 문장에 들여쓰기(indentation)

if 조건문:

수행할_문장1

수행할_문장2

수행할_문장3

- ▶ 들여쓰기를 무시하는 경우 오류 발생

if 조건문:

수행할_문장1

수행할_문장2

수행할_문장3

들여쓰기를 하지 않았으니
오류가 발생할 거야!



if 조건문:

수행할_문장1

수행할_문장2

수행할_문장3

2. 조건(if)문

▶ 조건문이란 무엇인가?

- ▶ if 조건문에서 '조건문'이란 참과 거짓을 판단하는 문장

```
>>> money = True
>>> if money:
```

▶ 비교 연산자

비교 연산자	설명
$x < y$	x가 y보다 작다.
$x > y$	x가 y보다 크다.
$x == y$	x와 y가 같다.
$x != y$	x와 y가 같지 않다.
$x >= y$	x가 y보다 크거나 같다.
$x <= y$	x가 y보다 작거나 같다.

```
>>> x = 3
>>> y = 2
>>> x > y ← 3 > 2
True
```

```
>>> x < y ← 3 < 2
False
```

```
>>> x == y ← 3 == 2
False
```

```
>>> x != y ← 3 != 2
True
```

2. 조건(if)문

▶ 조건문이란 무엇인가?

▶ 비교 연산자

- ▶ if 조건문에 비교 연산자를 사용하는 예시

만약 3000원 이상의 돈을 가지고 있으면 택시를 타고 가고, 그렇지 않으면 걸어가라.

```
>>> money = 2000 ← 2,000원을 가지고 있다고 설정
>>> if money >= 3000:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
걸어가라
```



2. 조건(if)문

▶ 조건문이란 무엇인가?

▶ and, or, not

연산자	설명
x or y	x와 y 둘 중 하나만 참이어도 참이다.
x and y	x와 y 모두 참이어야 참이다.
not x	x가 거짓이면 참이다.

▶ or 연산자의 사용법

돈이 3000원 이상 있거나 카드가 있다면 택시를 타고 가고, 그렇지 않으면 걸어가라.

```
>>> money = 2000  ← 2,000원을 가지고 있다고 설정
>>> card = True   ← 카드를 가지고 있다고 설정
>>> if money >= 3000 or card:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
```


2. 조건(if)문

▶ 조건문이란 무엇인가?

▶ and, or, not

연산자	설명
x or y	x와 y 둘 중 하나만 참이어도 참이다.
x and y	x와 y 모두 참이어야 참이다.
not x	x가 거짓이면 참이다.

▶ or 연산자의 사용법

돈이 3000원 이상 있거나 카드가 있다면 택시를 타고 가고, 그렇지 않으면 걸어가라.

```
>>> money = 2000  ← 2,000원을 가지고 있다고 설정
>>> card = True   ← 카드를 가지고 있다고 설정
>>> if money >= 3000 or card:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
```

2. 조건(if)문

▶ 조건문이란 무엇인가?

▶ in, not in

in	not in
x in 리스트	x not in 리스트
x in 튜플	x not in 튜플
x in 문자열	x not in 문자열

```
>>> 1 in [1, 2, 3] ← 1이 [1, 2, 3] 안에 있는가?
```

```
True
```

```
>>> 1 not in [1, 2, 3] ← 1이 [1, 2, 3] 안에 없는가?
```

```
False
```

```
>>> 'a' in ('a', 'b', 'c')
```

```
True
```

```
>>> 'j' not in 'python'
```

```
True
```

2. 조건(if)문

▶ 다양한 조건을 판단하는 elif

- ▶ if와 else만으로는 조건 판단에 어려움이 있음

주머니에 돈이 있으면 택시를 타고,
주머니에 돈은 없지만 카드가 있으면 택시를 타고,
돈도 없고 카드도 없으면 걸어 가라.

▶ 조건 판단하는 부분

- ▶ 주머니에 돈이 있는지 판단
- ▶ 주머니에 돈이 없으면,
주머니에 카드가 있는지 판단

```
>>> pocket = ['paper', 'cellphone'] ← 주머니 안에 종이, 휴대폰이 있다.  
>>> card = True ← 카드를 가지고 있다.  
>>> if 'money' in pocket:  
...     print("택시를 타고 가라")  
... else:  
...     if card:  
...         print("택시를 타고 가라")  
...     else:  
...         print("걸어가라")  
...  
택시를 타고 가라
```

- if와 else만으로는 이해하기 어렵고 산만한 느낌

2. 조건(if)문

▶ 다양한 조건을 판단하는 elif

▶ elif를 사용한다면?

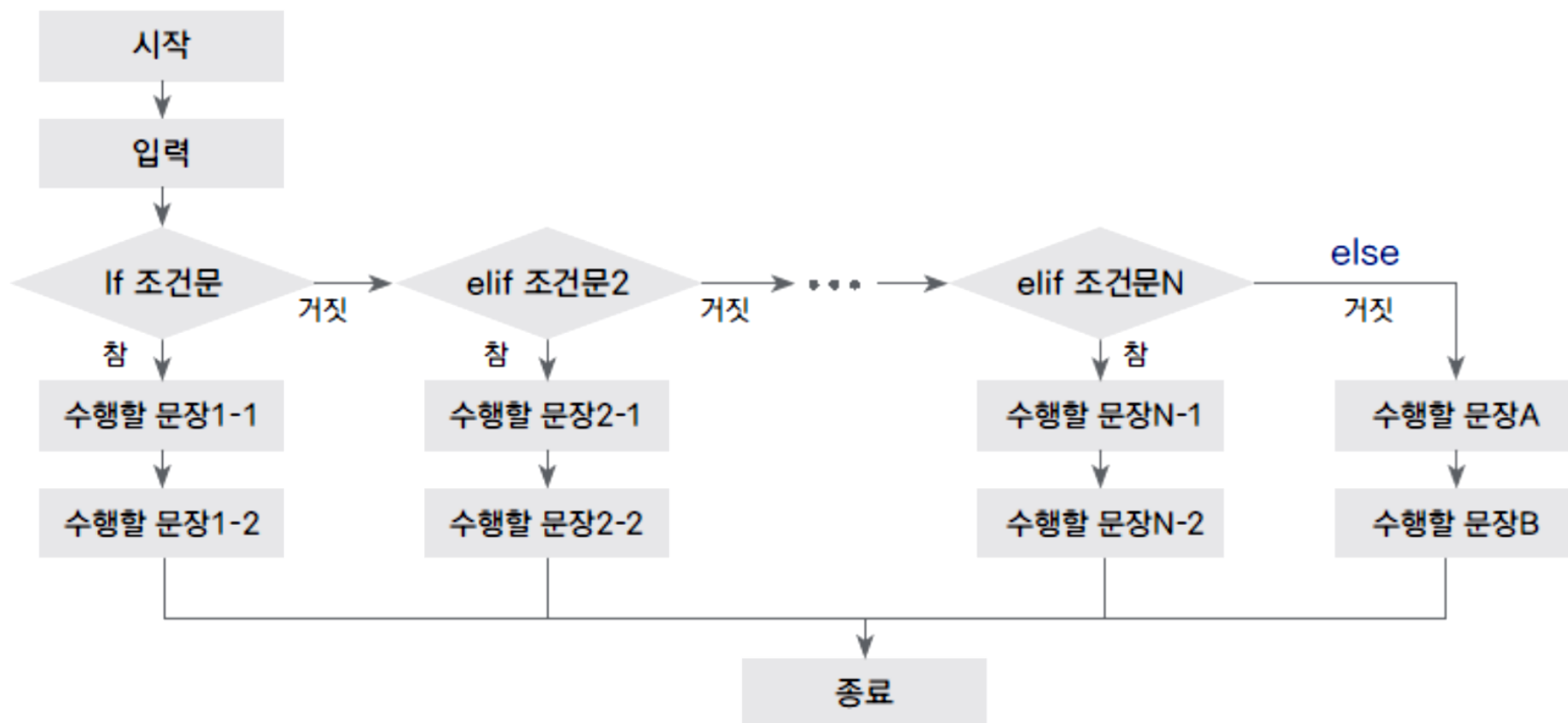
```
>>> pocket = ['paper', 'cellphone']
>>> card = True
>>> if 'money' in pocket: ← 주머니에 돈이 있으면
...     print("택시를 타고 가라")
... elif card: ← 주머니에 돈이 없고 카드가 있으면
...     print("택시를 타고 가라")
... else: ← 주머니에 돈이 없고 카드도 없으면
...     print("걸어가라")
...
택시를 타고 가라
```

- ▶ elif는 이전 조건문이 거짓일 때 수행됨

```
if 조건문:
    수행할_문장1
    수행할_문장2
    ...
elif 조건문:
    수행할_문장1
    수행할_문장2
    ...
elif 조건문:
    수행할_문장1
    수행할_문장2
    ...
(...생략...)
else:
    수행할_문장1
    수행할_문장2
    ...
```

2. 조건(if)문

- ▶ 다양한 조건을 판단하는 **elif**
 - ▶ elif는 개수에 제한 없이 사용 가능



2. 조건(if)문

▶ 조건부 표현식

- ▶ score가 60 이상일 경우 message에 문자열 “success”, 아닐 경우에 문자열 “failure” 대입하는 코드

```
if score >= 60:  
    message = "success"  
else:  
    message = "failure"
```

- ▶ 파이썬의 조건부 표현식(conditional expression) 사용

- ▶ 조건부 표현식

```
message = "success" if score >= 60 else "failure"
```

```
변수 = 조건문이_참인_경우의_값 if 조건문 else 조건문이_거짓인_경우의_값
```



2. 조건(if)문

▶ 조건문

▶ 응용 예제 1

- ▶ 응용 예제 : 시험 점수(score)를 입력 받아 등급(grade)을 구하라
- ▶ 점수가 90점 이상이면 등급은 A,
- ▶ 점수가 80~89 이면 등급은 B,
- ▶ 점수가 70~79 이면 등급은 C,
- ▶ 점수가 70점 미만이면 등급은 F

▶ 결과 화면

Enter your score: 98

A

← 사용자 점수 입력

← 올바른 값 출력



2. 조건(if)문

▶ 조건문

▶ 응용예제 2

- 조건문을 이용하여 '어떤 종류의 학생인지 맞는 프로그램'을 만들어 보자.
- 이 프로그램을 작성하는 규칙은 다음과 같다.

- 나이는 (2020 - 태어난 연도 + 1)로 계산
- 26세 이하 20세 이상이면 '대학생'
- 20세 미만 17세 이상이면 '고등학생'
- 17세 미만 14세 이상이면 '중학생'
- 14세 미만 8세 이상이면 '초등학생'
- 그 외의 경우는 '학생이 아닙니다.' 출력

당신이 태어난 연도를 입력하세요.

1982

학생이 아닙니다.

← 입력 대기

← 자신이 태어난 연도 입력

← 어떤 종류의 학생인지 출력

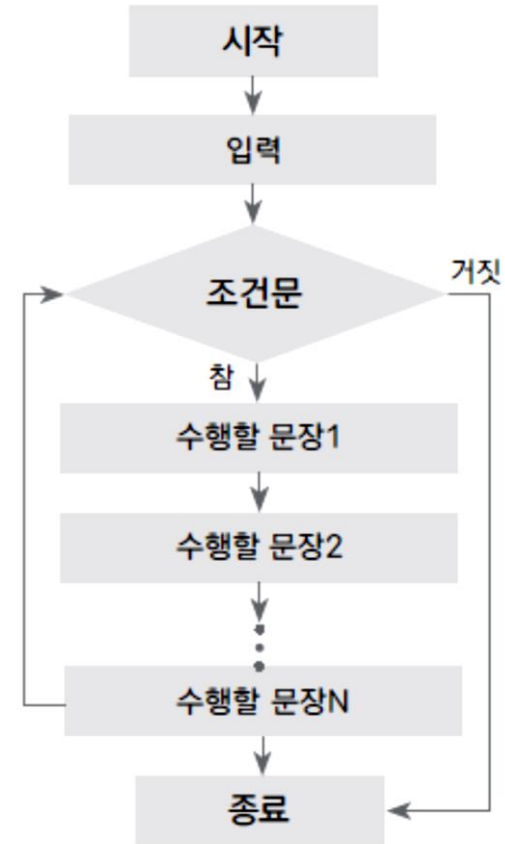
3. while 문

▶ while 문의 기본 구조

- ▶ 반복해서 문장을 수행해야 할 경우 while 문 사용
- ▶ 반복문이라고도 부름

```
while 조건문:  
    수행할_문장1  
    수행할_문장2  
    수행할_문장3  
    ...
```

- ▶ while 문은 조건문이 참인 동안에
while 문에 속한 문장이 반복해서 수행됨



3. while 문

▶ while 문의 기본 구조

- ▶ 예제) ‘열 번 찍어 안 넘어가는 나무 없다’는 속담 구현
 - ▶ while 문의 조건문은 `treeHit < 10`
 - ▶ `treeHit`이 10보다 작은 동안 while 문에 포함된 문장 반복 수행

```
>>> treeHit = 0  ← 나무를 찍은 횟수
>>> while treeHit < 10:  ← 나무를 찍은 횟수가 10보다 작은 동안 반복
...     treeHit = treeHit + 1  ← 나무를 찍은 횟수 1씩 증가
...     print("나무를 %d번 찍었습니다." % treeHit)
...     if treeHit == 10:  ← 나무를 열 번 찍으면
...         print("나무 넘어갑니다.")
...
나무를 1번 찍었습니다.
나무를 2번 찍었습니다.
```

```
나무를 3번 찍었습니다.
나무를 4번 찍었습니다.
나무를 5번 찍었습니다.
나무를 6번 찍었습니다.
나무를 7번 찍었습니다.
나무를 8번 찍었습니다.
나무를 9번 찍었습니다.
나무를 10번 찍었습니다.
나무 넘어갑니다.
```



3. while 문

▶ while 문의 기본 구조

▶ 예제) ‘열 번 찍어 안 넘어가는 나무 없다’는 속담 구현

▶ while 문이 반복되는 과정

treeHit	조건문	조건 판단	수행하는 문장	while 문
0	$0 < 10$	참	나무를 1번 찍었습니다.	반복
1	$1 < 10$	참	나무를 2번 찍었습니다.	반복
2	$2 < 10$	참	나무를 3번 찍었습니다.	반복
3	$3 < 10$	참	나무를 4번 찍었습니다.	반복
4	$4 < 10$	참	나무를 5번 찍었습니다.	반복
5	$5 < 10$	참	나무를 6번 찍었습니다.	반복
6	$6 < 10$	참	나무를 7번 찍었습니다.	반복
7	$7 < 10$	참	나무를 8번 찍었습니다.	반복
8	$8 < 10$	참	나무를 9번 찍었습니다.	반복
9	$9 < 10$	참	나무를 10번 찍었습니다. 나무 넘어갑니다.	반복
10	$10 < 10$	거짓		종료



3. while 문

▶ while 문 만들기

- ▶ 예제) 여러 가지 선택지 중 하나를 선택해 입력받기

```
>>> prompt = ""  
... 1. Add  
... 2. Del  
... 3. List  
... 4. Quit  
...  
... Enter number: ""
```

- ▶ number 변수에 0 대입하기



```
>>> number = 0  ← 번호를 입력받을 변수  
>>> while number != 4:  ← 입력받은 번호가 4가 아닌 동안 반복  
...     print(prompt)  
...     number = int(input())  
...  
1. Add  
2. Del  
3. List  
4. Quit  
  
Enter number:
```

변수 prompt 출력

3. while 문

▶ while 문 만들기

- ▶ 예제) 여러 가지 선택지 중 하나를 선택해 입력받기

- ▶ number가 4가 아닌 동안 prompt를 출력

Enter number:

1 ← 1 입력

1. Add

2. Del

3. List

4. Quit

4를 입력하지 않으면 계속 prompt의 값 출력

- 사용자가 4를 입력하면 조건문이 거짓이 되어 while 문을 빠져나감

Enter number:

4 ← 4 입력

>>> ← while 문 종료

3. while 문

▶ while 문 강제로 빠져나가기

- ▶ 강제로 while 문을 빠져나가야 할 때 break 문 사용



```
>>> coffee = 10  ← 자판기에 커피가 10개 있다.
>>> money = 300  ← 자판기에 넣을 돈은 300원이다.
>>> while money:
...     print("돈을 받았으니 커피를 줍니다.")
...     coffee = coffee - 1  ← while문을 한 번 돌 때마다 커피가 1개씩 줄어든다.
...     print("남은 커피의 양은 %d개입니다." % coffee)
...     if coffee == 0:
...         print("커피가 다 떨어졌습니다. 판매를 중지합니다.")
...         break
... 
```

- money가 300으로 고정되어 있어, while문의 조건문은 항상 참 → 무한 루프
- break 문 호출 시 while 문 종료

3. while 문

- ▶ while 문의 맨 처음으로 돌아가기

- ▶ while 문을 빠져나가지 않고 while 문의 맨 처음(조건문)으로 다시 돌아가야 할 때 사용

- ▶ 1부터 10까지의 숫자 중 홀수만 출력하는 예시

- ▶ 조건문이 참이 되는 경우 → a가 짝수
 - ▶ continue 문장 수행 시 while 문의 맨 처음, 즉 조건문 $a < 10$ 으로 돌아감
 - ▶ 따라서 a가 짝수이면 print(a)는 수행되지 않음

```
>>> a = 0
>>> while a < 10:
...     a = a + 1
...     if a % 2 == 0: continue ← a를 2로 나누었을 때 나머지가 0이면 맨 처음으로 돌아간다.
...     print(a)
...
1
3
5
7
9
```

3. while 문

▶ 무한 루프

- ▶ 무한히 반복한다는 뜻의 무한 루프(endless loop)
- ▶ 파이썬에서의 무한 루프는 while 문으로 구현
 - ▶ while 문의 조건문이 True이므로 항상 참
→ while 문 안에 있는 문장들은 무한 수행

```
while True:  
    수행할_문장1  
    수행할_문장2  
    ...
```

▶ 무한 루프 예

```
>>> while True:  
...     print("Ctrl+C를 눌러야 while 문을 빠져나갈 수 있습니다.")  
...  
Ctrl+C를 눌러야 while 문을 빠져나갈 수 있습니다.  
Ctrl+C를 눌러야 while 문을 빠져나갈 수 있습니다.  
Ctrl+C를 눌러야 while 문을 빠져나갈 수 있습니다.  
(...생략...)
```


4. for문

▶ 반복문(for)

- ▶ 특정 횟수 만큼 명령을 반복 수행
- ▶ 반복 범위를 지정하여 반복을 수행.

```
for i in range(start, stop, step):  
    print(i)
```

```
for i in range(1, 10, 1):  
    print(i)
```

```
for i in collection:  
    print(i)
```

```
a=[1,2,3,4,5]  
for i in a:  
    print(i)
```



4. for문

▶ for 문의 기본 구조

```
for 변수 in 리스트(또는 튜플, 문자열):  
    수행할_문장1  
    수행할_문장2  
    ...
```

- ▶ 리스트나 튜플, 문자열의 첫 번째 요소부터 마지막 요소까지 차례로 변수에 대입되어 '수행할 문장1', '수행할 문장2' 등이 수행됨



4. for문

▶ 예제를 통해 for 문 이해하기

1. 전형적인 for 문

```
>>> test_list = ['one', 'two', 'three']
>>> for i in test_list: ← one, two, three를 순서대로 i에 대입
...     print(i)
...
one
two
three
```

2. 다양한 for 문의 사용

```
>>> a = [(1, 2), (3, 4), (5, 6)]
>>> for (first, last) in a:
...     print(first + last)
...
3 ← first: 1, last: 2
7 ← first: 3, last: 4
11 ← first: 5, last: 6
```



4. for문

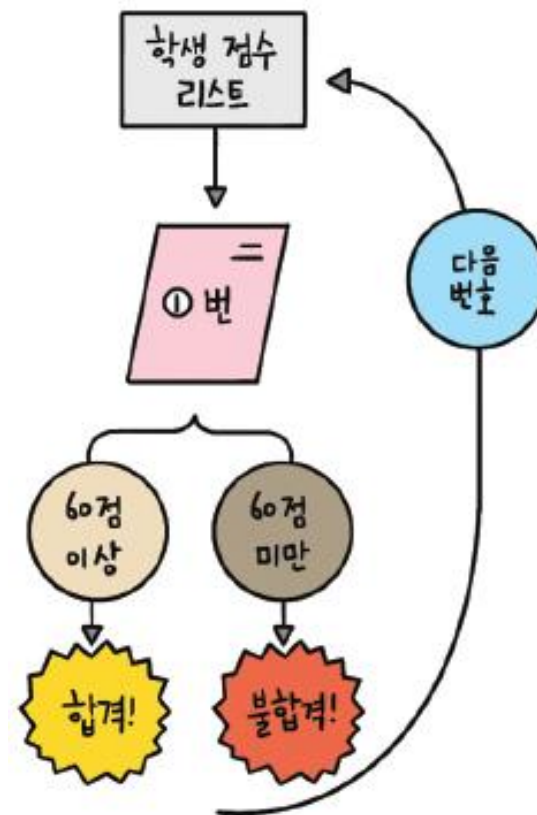
▶ 예제를 통해 for 문 이해하기

▶ for 문의 응용

총 5명의 학생이 시험을 보았는데 시험 점수가 60점 이상이면 합격이고 그렇지 않으면 불합격이다. 합격인지, 불합격인지 결과를 보여 주시오.

```
marks = [90, 25, 67, 45, 80]           # 학생들의 시험 점수 리스트

number = 0                             # 학생에게 붙여 줄 번호
for mark in marks:                      # 90, 25, 67, 45, 80을 순서대로 mark에 대입
    number = number + 1
    if mark >= 60:
        print("%d번 학생은 합격입니다." % number)
    else:
        print("%d번 학생은 불합격입니다." % number)
```



4. for문

▶ for 문과 continue 문

- ▶ for 문 안의 문장을 수행하는 도중 continue 문을 만나면 for 문의 처음으로 돌아감
- ▶ 60점 이상인 사람에게는 축하 메시지를 보내고 나머지 사람에게는 아무런 메시지도 전하지 않는 프로그램

```
marks = [90, 25, 67, 45, 80]

number = 0
for mark in marks:
    number = number + 1
    if mark < 60:
        continue
    print("%d번 학생 축하합니다. 합격입니다. " % number)
```



4. for문

▶ for 문과 함께 자주 사용하는 range 함수

- ▶ 숫자 리스트를 자동으로 만들어주는 함수

```
>>> a = range(10)
>>> a
range(0, 10) ← 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
```

- ▶ range(10)은 0부터 10 미만의 숫자를 포함하는 range 객체를 만들어 준다.

- ▶ range(a, b)
 - ▶ a: 시작 숫자
 - ▶ b: 끝 숫자 (반환 범위에 포함되지 않음)

```
>>> a = range(1, 11)
>>> a
range(1, 11) ← 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```



4. for문

- ▶ **for 문과 함께 자주 사용하는 range 함수**

- ▶ range 함수의 예시

- ▶ for와 range 함수를 사용하여 1부터 10까지 더하기

```
>>> add = 0
>>> for i in range(1, 11):
...     add = add + i
...
>>> print(add)
55
```



4. for문

▶ for 문과 함께 자주 사용하는 range 함수

▶ for와 range를 이용한 구구단

▶ ①번 for문

- 2부터 9까지의 숫자(range(2, 10))가 차례로 i에 대입됨

▶ ②번 for문

- 1부터 9까지의 숫자(range(1, 10))가 차례로 j에 대입됨
- print(i*j) 수행

```
>>> for i in range(2, 10):      ← ①번 for 문
...     for j in range(1, 10): ← ②번 for 문
...         print(i*j, end=" ")
...     print('')
...
2 4 6 8 10 12 14 16 18
3 6 9 12 15 18 21 24 27
4 8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```


4. for문

▶ for 문과 함께 자주 사용하는 range 함수

▶ for와 range를 이용한 구구단

i가 2일 때

i	j	i*j
2	1	2
	2	4
	3	8
	4	8
	5	10
	6	12
	7	14
	8	16
	9	18
②번 for 문 종료		

i가 3일 때

i	j	i*j
3	1	3
	2	6
	3	9
	4	12
	5	15
	6	18
	7	21
	8	24
	9	27
②번 for 문 종료		

i가 4일 때

i	j	i*j
4	1	4
	2	8
	3	12
	4	16
	5	20
	6	24
	7	28
	8	32
	9	36
②번 for 문 종료		

...

i가 9일 때

i	j	i*j
9	1	9
	2	18
	3	27
	4	36
	5	45
	6	54
	7	63
	8	72
	9	81
전체 for 문 종료		

4. for문

▶ 리스트 컴프리헨션(list comprehension) 사용하기

▶ 리스트 안에 for 문 포함하기

▶ 예제

- ▶ a 리스트의 각 항목에 3을 곱한 결과를 result 리스트에 담기

```
>>> a = [1, 2, 3, 4]
>>> result = []
>>> for num in a:
...     result.append(num*3)
...
>>> print(result)
[3, 6, 9, 12]
```

■ 리스트 컴프리헨션을 사용하도록 수정

```
>>> a = [1, 2, 3, 4]
>>> result = [num*3 for num in a]
>>> print(result)
[3, 6, 9, 12]
```



4. for문

▶ 리스트 컴프리헨션(list comprehension) 사용하기

- ▶ 리스트 안에 for 문 포함하기
- ▶ 예제
 - ▶ 리스트 컴프리헨션 안에 'if 조건' 사용 가능
 - ▶ [1, 2, 3, 4] 중에서 짝수에만 3을 곱하여 담도록 수정

```
>>> a = [1, 2, 3, 4]
>>> result = [num*3 for num in a if num%2 == 0]
>>> print(result)
[6, 12]
```



4. for문

▶ 리스트 컴프리헨션(list comprehension) 사용하기

▶ 리스트 컴프리헨션 문법

- ▶ 'if 조건문' 부분은 생략 가능

```
[표현식 for 항목 in 반복_가능_객체 if 조건문]
```

- ▶ for 문 여러 개 사용 가능

```
[표현식 for 항목1 in 반복_가능_객체1 if 조건문1  
    for 항목2 in 반복_가능_객체2 if 조건문2  
    ...  
    for 항목n in 반복_가능_객체n if 조건문n]
```



4. for문

▶ 리스트 컴프리헨션(list comprehension) 사용하기

▶ 구구단의 모든 결과를 리스트로 담은 리스트 컴프리헨션 사용 예제

```
>>> result = [x*y for x in range(2, 10)
...           for y in range(1, 10)]
>>> print(result)
[2, 4, 6, 8, 10, 12, 14, 16, 18, 3, 6, 9, 12, 15, 18, 21, 24, 27, 4, 8, 12, 16,
20, 24, 28, 32, 36, 5, 10, 15, 20, 25, 30, 35, 40, 45, 6, 12, 18, 24, 30, 36, 42
, 48, 54, 7, 14, 21, 28, 35, 42, 49, 56, 63, 8, 16, 24, 32, 40, 48, 56, 64, 72,
9, 18, 27, 36, 45, 54, 63, 72, 81]
```



예제 5-1: 조건문과 반복문

▶ LAB : 음료 자판기 만들기

- ▶ 조건문과 반복문을 사용하여 음료 자판기 프로그램 작성

[조건]

1. 자판기는 반복 동작한다.(while)
2. 오렌지 주스, 커피, 콜라를 각각 100원, 200원, 300원에 판매한다. (if ~ elif ~else)
3. 구매자에게 동전 액수와 주문번호(1. 오렌지 주스, 2. 커피, 3. 콜라)를 입력 받는다.
4. 입력 받은 액수보다 선택 메뉴의 가격이 비싸면 '잔액이 부족합니다'라고 메시지를 출력하고 처음부터 다시 입력을 받습니다.
5. 자판기는 주문과 동시에 잔액을 알려 줌



예제 5-2:

학생	A	B	C	D	E
국어 점수	49	80	20	100	80
수학 점수	43	60	85	30	90
영어 점수	49	82	48	50	100

```
[47.0, 74.0, 51.0, 60.0, 90.0]
```



5. Collection 모듈

▶ OrderedDict

- ▶ OrderedDict 모듈은 이름 그대로 순서를 가진 딕셔너리 객체이다. 딕셔너리 파일을 저장하면 키는 저장 순서와 상관없이 저장된다

```
1 d = {}
2 d['x'] = 100
3 d['l'] = 500
4 d['y'] = 200
5 d['z'] = 300
6
7 for k, v in d.items():
8     print(k, v)
```

```
x 100
l 500
y 200
z 300
```



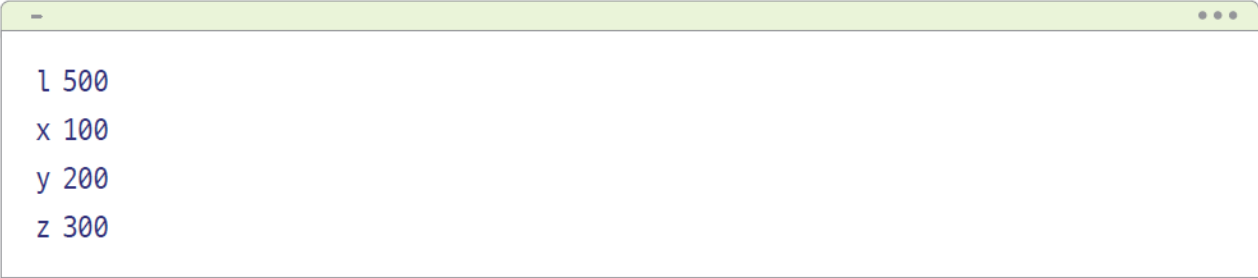
```
1 from collections import OrderedDict    # OrderedDict 모듈 선언
2
3 d = OrderedDict()
4 d['x'] = 100
5 d['y'] = 200
6 d['z'] = 300
7 d['l'] = 500
8
9 for k, v in d.items():
10     print(k, v)
```

```
x 100
y 200
z 300
l 500
```


5. Collection 모듈

▶ OrderedDict

```
1 def sort_by_key(t):
2     return t[0]
3
4 from collections import OrderedDict      # OrderedDict 모듈 선언
5
6 d = dict()
7 d['x'] = 100
8 d['y'] = 200
9 d['z'] = 300
10 d['l'] = 500
11
12 for k, v in OrderedDict(sorted(d.items(), key=sort_by_key)).items():
13     print(k, v)
```



```
l 500
x 100
y 200
z 300
```

5. Collection 모듈

▶ defaultdict

- ▶ defaultdict 모듈은 딕셔너리의 변수를 생성할 때 키에 기본 값을 지정하는 방법

```
1 d = dict()
2 print(d["first"])
```



```
Traceback (most recent call last):
  File "defaultdict1.py", line 2, in <module>
    print(d["first"])
KeyError: 'first'
```

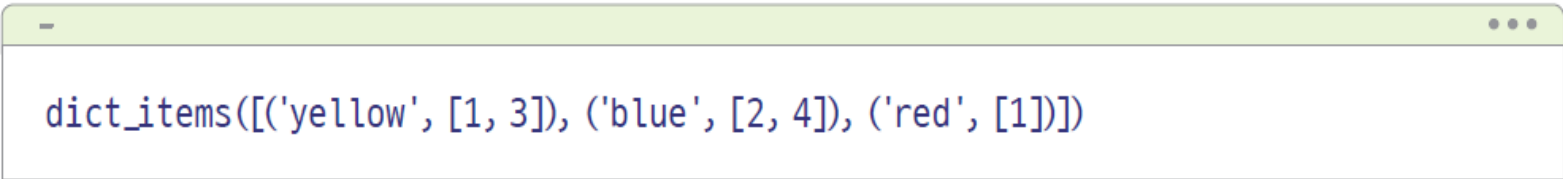
```
1 from collections import defaultdict
2
3 d = defaultdict(lambda: 0)           # Default 값을 0으로 설정
4 print(d["first"])
```

```
0
```

5. Collection 모듈

▶ defaultdict

```
1 from collections import defaultdict
2
3 s = [('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]
4 d = defaultdict(list)
5 for k, v in s:
6     d[k].append(v)
7
8 print(d.items())
9 [('blue', [2, 4]), ('red', [1]), ('yellow', [1, 3])]
```



```
dict_items([('yellow', [1, 3]), ('blue', [2, 4]), ('red', [1])])
```

5. Collection 모듈

▶ Counter 모듈

- ▶ Counter 모듈은 시퀀스 자료형의 데이터 요소 개수를 딕셔너리 형태로 반환하는 자료구조
- ▶ 즉, 리스트나 문자열과 같은 시퀀스 자료형 안의 요소 중 값이 같은 것이 몇 개 있는지 반환

```
>>> from collections import Counter
>>>
>>> text = list("gallahad")
>>> text
['g', 'a', 'l', 'l', 'a', 'h', 'a', 'd']
>>> c = Counter(text)
>>> c
Counter({'a': 3, 'l': 2, 'g': 1, 'h': 1, 'd': 1})
>>> c["a"]
3
```



5. Collection 모듈

▶ Counter 모듈

▶ 단어 카운트 예

```
>>> text = """A press release is the quickest and easiest way to get free
publicity. If well written, a press release can result in multiple published
articles about your firm and its products. And that can mean new prospects
contacting you asking you to sell to them. ...""".lower().split()
>>> Counter(text)
Counter({'and': 3, 'to': 3, 'can': 2, 'press': 2, 'release': 2, 'you': 2, 'a': 2, 'sell': 1,
'about': 1, 'free': 1, 'firm': 1, 'quickest': 1, 'products.': 1, 'written.': 1, 'them.': 1,
'...': 1, 'articles': 1, 'published': 1, 'mean': 1, 'that': 1, 'prospects': 1, 'its': 1,
'multiple': 1, 'if': 1, 'easiest': 1, 'publicity.': 1, 'way': 1, 'new': 1, 'result': 1,
'the': 1, 'your': 1, 'well': 1, 'is': 1, 'asking': 1, 'in': 1, 'contacting': 1, 'get': 1})
```



5. Collection 모듈

▶ Counter 모듈

▶ 딕셔너리로 카운트 생성

```
>>> from collections import Counter
>>>
>>> c = Counter({'red': 4, 'blue': 2})
>>> print(c)
Counter({'red': 4, 'blue': 2})
>>> print(list(c.elements()))
['red', 'red', 'red', 'red', 'blue', 'blue']
```



5. Collection 모듈

▶ Counter 모듈

▶ Counter 연산(사칙 연산, 비트 연산)

```
>>> from collections import Counter
>>>
>>> c = Counter(a = 4, b = 2, c = 0, d = -2)
>>> d = Counter(a = 1, b = 2, c = 3, d = 4)
>>> c.subtract(d)                                     # c - d
>>> c
Counter({'a': 3, 'b': 0, 'c': -3, 'd': -6})
```

```
>>> from collections import Counter
>>>
>>> c = Counter(a = 4, b = 2, c = 0, d = -2)
>>> d = Counter(a = 1, b = 2, c = 3, d = 4)
>>> print(c + d)
Counter({'a': 5, 'b': 4, 'c': 3, 'd': 2})
>>> print(c & d)
Counter({'b': 2, 'a': 1})
>>> print(c | d)
Counter({'a': 4, 'd': 4, 'c': 3, 'b': 2})
```



06. Lab: 텍스트 마이닝 프로그램

▶ 실습 내용

- ▶ 딕셔너리와 Collections 모듈을 이용하여 텍스트 마이닝 프로그램을 만들어 보자.

```
A press release is the quickest and easiest way to get free publicity. If well
written, a press release can result in multiple published articles about your
firm and its products. And that can mean new prospects contacting you asking
you to sell to them. ...
```

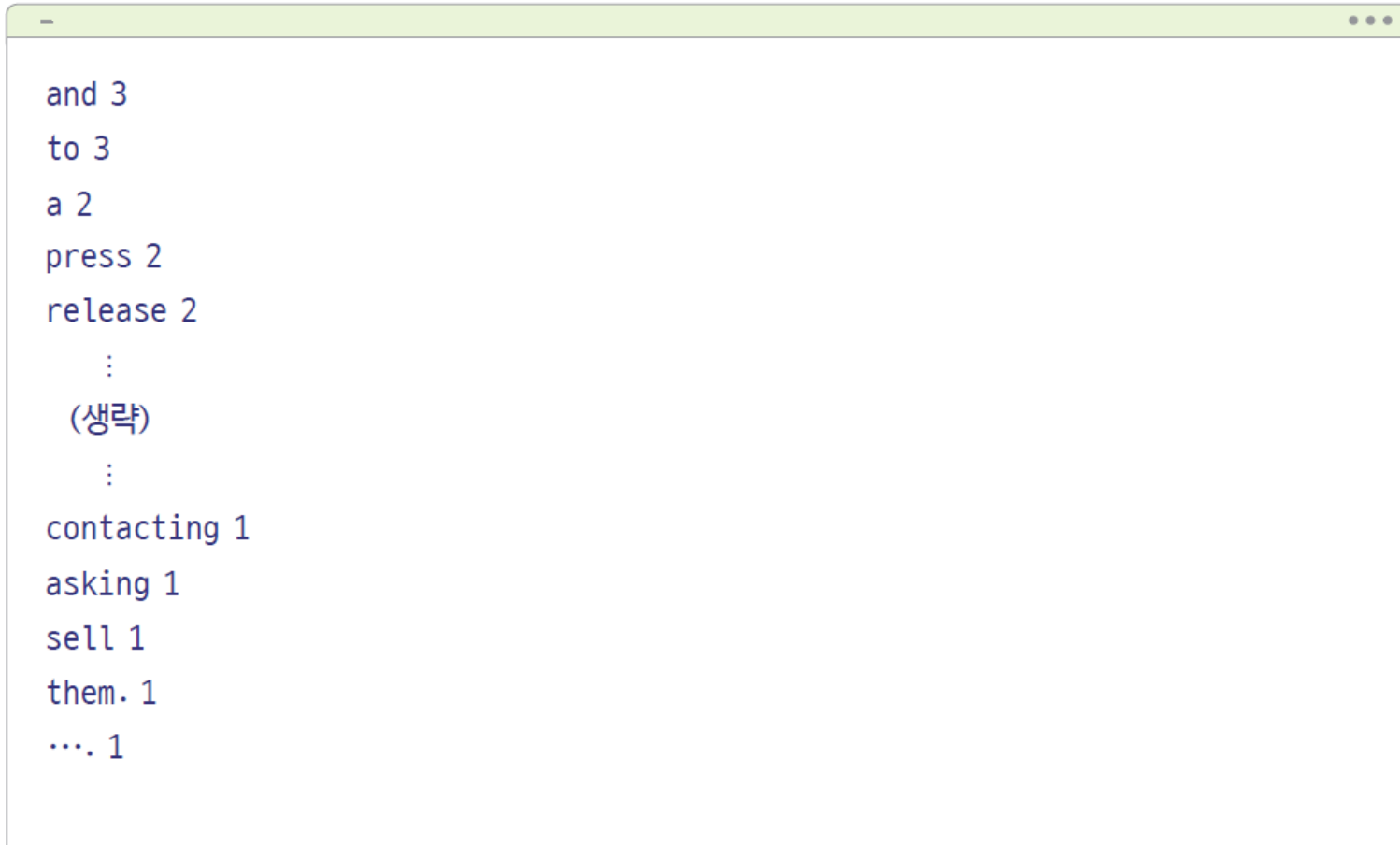
- ▶ 이 프로그램을 작성하는 규칙은 다음과 같다.

- 문장의 단어 개수를 파악하는 코드를 작성한다.
- defaultdict 모듈을 사용한다.
- 단어의 출현 횟수를 기준으로 정렬된 결과를 보여 주기 위해 OrderedDict 모듈을 사용한다.



06. Lab: 텍스트 마이닝 프로그램

▶ 실행 결과



```
and 3
to 3
a 2
press 2
release 2
:
(생략)
:
contacting 1
asking 1
sell 1
them. 1
... 1
```



06. Lab: 텍스트 마이닝 프로그램

▶ 문제 해결

```
1 text = """A press release is the quickest and easiest way to get free
  publicity. If well written, a press release can result in multiple
  published articles about your firm and its products. And that can mean new
  prospects contacting you asking you to sell to them. ...""".lower().split()
2
3 from collections import defaultdict
4
5 word_count = defaultdict(lambda: 0)          # Default 값을 0으로 설정
6 for word in text:
7     word_count[word] += 1
8
9 from collections import OrderedDict
10 for i, v in OrderedDict(sorted(word_count.items(), key=lambda t: t[1],
    reverse=True)).items():
11     print(i, v)
```

