핵심 정리

반복문과 들여쓰기

▼ **표 24-1** 리스트 메서드

메서드	설명
append(값) append(리스트)	리스트 끝에 값을 추가, 리스트[len(리스트):] = [값]과 같음 리스트를 넣으면 리스트 안에 리스트가 들어감
extend(리스트)	리스트 끝에 다른 리스트 연결, 리스트[len(리스트):] = [값, 값]과 같음
insert(인덱스, 값)	리스트의 특정 인덱스에 값을 추가
pop() pop(인덱스)	리스트의 마지막 값을 삭제한 뒤 삭제한 값을 반환 인덱스를 지정하면 해당 인덱스의 값을 삭제한 뒤 삭제한 값을 반환, del 리스트[인덱스]와 같음
remove(값)	리스트에서 특정 값을 삭제
index(값)	리스트에서 특정 값의 인덱스를 구함
count(값)	리스트에서 특정 값의 개수를 구함
reverse()	리스트에서 값의 순서를 반대로 뒤집음
sort() sort(reverse=True)	리스트의 값을 작은 순서대로 정렬(오름차순) reverse=True는 큰 순서대로 정렬(내림차순)
clear()	리스트의 모든 값을 삭제, del a[:]와 같음
copy()	리스트를 복사하여 새 리스트 생성

튜플은 값의 정보를 구하는 index, count 메서드만 사용할 수 있습니다.

인덱스로 범위를 지정하여 리스트 조작하기

리스트는 메서드를 사용하지 않고, 인덱스로 범위를 지정하여 조작할 수 있습니다.

```
리스트[len(리스트):] = [값]        # 리스트 끝에 값이 한 개 들어있는 리스트 추가
                                  # 리스트.append(값)과 같음
리스트[len(리스트):] = [값, 값]    # 리스트 끝에 다른 리스트 연결
                                  # 리스트.extend([값, 값])과 같음
del 리스트[인덱스]    # 특정 인덱스의 값 삭제, 리스트.pop(인덱스)와 같음
del 리스트[:]   # 시작 인덱스와 끝 인덱스를 생략하여 리스트의 모든 값을 삭제, 리스트.clear()와 같음
```

리스트(튜플)와 반복문

for 변수 in 뒤에 리스트(튜플)를 지정하면 반복하면서 모든 요소를 꺼내옵니다. 특히 enumerate(리스트)를 지정하면 인덱스와 요소를 동시에 꺼내올 수 있습니다.

```
for 변수 in 리스트:    # 반복하면서 요소를 꺼내옴
    반복할 코드

for 인덱스, 요소 in enumerate(리스트):    # 반복하면서 인덱스와 요소를 꺼내옴
    반복할 코드

for 인덱스 in range(len(리스트)):    # 리스트의 길이로 반복
    리스트[인덱스]                   # 인덱스로 요소에 접근

while 인덱스 < len(리스트):    # 리스트의 길이로 반복
    리스트[인덱스]             # 인덱스로 요소에 접근
    인덱스 += 1
```

min, max, sum 함수

min은 리스트(튜플)에서 가장 작은 값, max는 가장 큰 값, sum은 요소의 합계를 구합니다.

리스트(튜플) 표현식

리스트(튜플) 표현식은 리스트 안에 식, for 반복문, if 조건문 등을 지정하여 리스트(튜플)를 생성합니다.

# 리스트 표현식 	
[식 for 변수 in 리스트]
[i for i in range(10)]
list(식 for 변수 in 리스트)
 	
# 튜플 표현식 	
tuple(식 for 변수 in 리스트 if 조건식)
 	
# if 조건문 사용 	
[식 for 변수 in 리스트 if 조건식]
[i for i in range(10) if i % 2 == 0]
list(식 for 변수 in 리스트 if 조건식)
	# for와 if를 여러 번 사용
	[식 for 변수1 in 리스트1 if 조건식1
	for 변수2 in 리스트2 if 조건식2
	...
	for 변수n in 리스트n if 조건식n]
	[i * j for j in range(2, 10) for i in range(1, 10)]
	
	list(식 for 변수1 in 리스트1 if 조건식1
	for 변수2 in 리스트2 if 조건식2
	...
	for 변수n in 리스트n if 조건식n]

리스트(튜플)에 map 함수 사용

map은 리스트(튜플)의 요소를 지정된 함수로 처리해주는 함수입니다.

```
리스트 = list(map(함수, 리스트))
a = list(map(int, a))
튜플 = tuple(map(함수, 튜플))

변수1, 변수2 = list(map(함수, 리스트))    # 언패킹 사용
a, b = list(map(str, range(2)))

변수1, 변수2 = map(함수, 리스트)          # 언패킹 사용
a, b = map(int, input().split())
```

2차원 리스트

2차원 리스트는 가로×세로의 평면 구조로 이루어져 있습니다. 2차원 리스트는 리스트 안에 리스트를 넣어서 만들 수 있으며 안쪽의 각 리스트는 , (콤마)로 구분해줍니다. 2차원 리스트의 요소에 접근하거나 할당할 때는 리스트에 [ ](대괄호)를 두 번 사용하며 [ ] 안에 세로 인덱스와 가로 인덱스를 지정해줍니다. 일반적으로 2차원 공간은 가로×세로로 표기하지만 리스트로 만들 때는 세로×가로로 표기합니다.

```
리스트 = [[값, 값], [값, 값], [값, 값]]    # 2차원 리스트 만들기

리스트[세로인덱스][가로인덱스]         # 2차원 리스트의 요소에 접근
리스트[세로인덱스][가로인덱스] = 값     # 2차원 리스트의 요소에 값 저장

리스트 = [(값, 값), (값, 값), (값, 값)]    # 리스트 안에 튜플을 넣음
튜플 = ([값, 값], [값, 값], [값, 값])      # 튜플 안에 리스트를 넣음
튜플 = ((값, 값), (값, 값), (값, 값))      # 튜플 안에 튜플을 넣음
```

3차원 리스트

3차원 리스트는 높이×세로×가로 형태로 이루어져 있습니다. 3차원 공간은 가로×세로×높이로 표기하지만 리스트로 만들 때는 높이×세로×가로로 표기합니다.

```
리스트 = [[[값, 값], [값, 값]], [[값, 값], [값, 값]], [[값, 값], [값, 값]]]    # 3차원 리스트 만들기

리스트[높이인덱스][세로인덱스][가로인덱스]         # 3차원 리스트의 요소에 접근
리스트[높이인덱스][세로인덱스][가로인덱스] = 값     # 3차원 리스트의 요소에 값 저장
```

문자열 메서드

▼ **표 24-2** 문자열 메서드

메서드	설명
replace('바꿀문자열', '새문자열')	문자열 안의 문자열을 다른 문자열로 바꿈
translate(테이블)	문자열 안의 문자를 다른 문자로 바꿈, str.maketrans('바꿀문자', '새문자')로 변환 테이블을 만들어야 함
split() split('기준문자열')	공백을 기준으로 문자열을 분리하여 리스트로 만듦 기준 문자열을 지정하면 기준 문자열로 문자열을 분리
join(리스트)	구분자 문자열과 문자열 리스트(튜플)의 요소를 연결하여 문자열로 만듦
upper() lower()	upper는 문자열의 문자를 대문자로 바꾸고, lower는 소문자로 바꿈
lstrip(), rstrip(), strip() lstrip('삭제할문자들') rstrip('삭제할문자들') strip('삭제할문자들')	lstrip은 문자열에서 왼쪽 공백을 삭제, rstrip은 오른쪽 공백을 삭제, strip은 양쪽 공백을 삭제, 삭제할 문자들을 지정하면 해당 문자들을 삭제
ljust(길이), rjust(길이), center(길이)	문자열을 지정된 길이로 만든 뒤 왼쪽(ljust), 오른쪽(rjust), 가운데(center)로 정렬하며 남는 공간은 공백으로 채움
zfill()	지정된 길이에 맞춰서 문자열의 왼쪽에 0을 채움
find('찾을문자열') rfind('찾을문자열')	find는 왼쪽에서부터, rfind는 오른쪽에서부터 특정 문자열을 찾아서 인덱스 반환, 문자열이 없으면 -1을 반환
index('찾을문자열') rindex('찾을문자열')	index는 왼쪽에서부터, rindex는 오른쪽에서부터 특정 문자열을 찾아서 인덱스를 반환, 문자열이 없으면 에러 발생
count('문자열')	현재 문자열에서 특정 문자열이 몇 번 나오는지 알아냄

문자열 서식 지정자

문자열은 서식 지정자를 조합하여 문자열을 만들 수 있습니다. 서식 지정자는 `%`로 시작하며 자료형을 뜻하는 문자가 붙습니다. 서식 지정자를 사용한 뒤 `%` 다음에 문자열을 지정해주면 이 문자열이 서식 지정자에 들어갑니다. 서식 지정자가 여러 개 일 때는 값 여러 개를 튜플로 만들어서 지정해줍니다.

- `%s`: 문자열
- `%d`: 정수
- `%f`: 실수

```
'%서식지정자' % 값                 # 서식 지정자 한 개 사용
'I am %s.' % 'maria'               # 'I am maria.'

'%서식지정자1, %서식지정자2' % (값1, 값2)    # 서식 지정자 여러 개 사용
'Today is %d %s.' % (3, 'April')            # 'Today is 3 April.'
```

소수점 이하 자릿수를 지정하고 싶다면 `f` 앞에 `.`(점)과 자릿수를 지정합니다.

```
'%.자릿수f' % 숫자    # 소수점 이하 자릿수 지정하기
'%.3f' % 2.3          # '2.300'
```

`%`뒤에 숫자를 붙이면 문자열을 지정된 길이로 만든 뒤 오른쪽으로 정렬하고 남는 공간을 공백으로 채웁니다. 길이를 음수로 지정하면 왼쪽으로 정렬합니다.

```
%길이s     # 문자열을 지정된 길이로 만든 뒤 오른쪽으로 정렬하고 남는 공간을 공백으로 채움
'%10s' % 'python'     # '    python'

%-길이s    # 문자열을 지정된 길이로 만든 뒤 왼쪽으로 정렬하고 남는 공간을 공백으로 채움
'%-10s' % 'python'    # 'python    '
```

`%`와 `d`사이에 0과 숫자 개수를 넣으면 자릿수에 맞춰서 앞에 0이 들어갑니다.

```
'%0개수d' % 숫자    # 자릿수에 맞춰서 0이 들어감
'%03d' % 1          # '001'

'%0개수.자릿수f' % 숫자    # 실수의 소수점 이하 자릿수 지정
'%08.2f' % 3.6             # '00003.60'
```

문자열 포매팅

문자열 포매팅을 사용할 때는 `{ }`(중괄호) 안에 인덱스를 지정하고, `format`에는 `{ }` 부분에 들어갈 값을 지정해줍니다.

```
'{0}'.format(값)                      # 값을 한 개 넣음
'{0} {1}'.format(값1, 값2)            # 값을 두 개 넣음
'{0} {0} {1} {1}'.format(값1, 값2)    # 같은 인덱스에는 같은 값이 들어감
'{} {} {}'.format(값1, 값2, 값3)      # 인덱스를 생략하면 format에 지정한 순서대로 값이 들어감
'{name1} {name2}'.format(name1=값1, name2=값2)    # { }에 이름을 지정
```

파이썬 3.6부터는 변수에 값을 넣고 `{ }` 안에 변수 이름을 지정하면 됩니다. 이때는 문자열 앞에 `f`를 붙입니다.

```
변수1, 변수2 = 값1, 값2
f'{변수1} {변수2}'
```

문자열 포매팅에서 `<`은 문자열을 지정된 길이로 만든 뒤 왼쪽으로 정렬하고 남는 공간을 공백으로 채웁니다. `>`은 오른쪽으로 정렬합니다.

```
'{인덱스:<길이}'.format(값)  # 문자열을 지정된 길이로 만든 뒤 왼쪽 정렬, 남는 공간을 공백으로 채움
'{인덱스:>길이}'.format(값)  # 문자열을 지정된 길이로 만든 뒤 오른쪽 정렬, 남는 공간을 공백으로 채움
```

문자열 포매팅에서 인덱스나 이름 뒤에 `:`(콜론)을 붙이고 0과 숫자 개수를 지정하면 자릿수에 맞춰서 0이 들어갑니다.

```
'{인덱스:0개수d'}'.format(숫자)          # 자릿수에 맞춰서 0이 들어감
'{인덱스:0개수.자릿수f'}.format(숫자)    # 실수의 소수점 이하 자릿수 지정
```

문자열 포매팅은 채우기, 정렬, 길이, 자릿수, 자료형을 조합하여 사용할 수 있습니다.

```
'{인덱스:[[채우기]정렬][길이][.자릿수][자료형]}'

'{0:0<10}'.format(15)      # '1500000000': 길이 10, 왼쪽으로 정렬하고 남는 공간은 0으로 채움
'{0:0>10.2f}'.format(15)   # '0000015.00': 길이 10, 오른쪽으로 정렬하고 소수점 이하 자릿수는 2자리

'{0: >10}'.format(15)    # '        15': 남는 공간을 공백으로 채움
'{0:>10}'.format(15)     # '        15': 채우기 부분을 생략하면 공백이 들어감
'{0:x>10}'.format(15)    # 'xxxxxxxx15': 남는 공간을 문자 x로 채움
```

서식 지정자 자료형

▼ 표 24-3 서식 지정자 자료형

자료형	설명
s	문자열
b	2진수
c	문자
d	10진 정수
o	8진 정수, 예) '%o' % 8은 '10'
x	16진 정수, 0~9, a~f, 예) '%x' % 254는 'fe'
X	16진 정수, 0~9, A~F, 예) '%X' % 254는 'FE'
e	실수 지수 표기법, 예) '%e' % 2.3은 '2.300000e+00'
E	실수 지수 표기법, 예) '%E' % 2.3은 '2.300000E+00'
f	실수 소수점 표기
F	실수 소수점 표기, f와 같음, nan은 NAN, inf는 INF로 표시(nan은 숫자가 아니라는 뜻, inf는 무한대)
g	실수 일반 형식, 예) '%g' % 2.3e-10은 '2.3e-10'
G	실수 일반 형식, 예) '%G' % 2.3e-10은 '2.3E-10'
%	% 문자 표시