ILE3-004 문자열

문자열로 변환

다른 형태의 변수를 문자열로 변경

`str()`함수를 이용하여 변경

```
>>> f = open('test.txt')
>>> a = 10
>>> f.write(10)      #에러 발생
>>> f.write(str(10)) #정상적으로 기록
```

> Class의 경우에는 `__str__()` 함수가 호출되어 내용이 문자열로 변경 됨

문자열 만들기

% 연산자

Python 2에서 주로 사용되는 방식으로 C/C++의 `sprintf()` 함수와 동일
두 개 이상의 값을 전달해 주어야 하는 경우에는 `()`로 감싸서 전달

> 유사한 형태인 정수/소수 간의 변환은 가능하나, 문자열과 같이 변환이 힘든 경우 에러 발생

> 항상 %값과 전달값의 개수는 동일하여야 함

```
>>> a = 10
>>> b = 20
>>> c = 10.123
>>> d = '합격'
>>> '결과는 %d 입니다.' % a
결과는 10 입니다.
>>> '결과는 %d %d 입니다' % (a, b)
결과는 10 20 입니다.
>>> '결과는 %d, %s입니다' % (a, d)
결과는 10, 합격입니다.
>>> '결과는 %f 입니다.' % a
결과는 10.000000 입니다.
>>> '결과는 %d 입니다.' % c
결과는 10 입니다.
>>> '결과는 %f 입니다.' % d              # %f에 문자열을 넣어 에러 발생
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must be real number, not str
>>> '결과는 %d %d 입니다' % (a, b, c)    # %항목보다 매개변수가 많아 에러 발생
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: not all arguments converted during string formatting
>>> '결과는 %d %d %f 입니다' % (a, b)    # %항목이 매개변수보다 많아 에러 발생
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
TypeError: not all arguments converted during string formatting
```

format() 함수

C/C++의 `sprintf()` 함수와 유사하나, `중괄호 {}`를 지정한 값으로 변경을 수행

기본적인 사용 방법

중괄호가 포함된 문자열에 대하여 format() 함수를 호출하고, 매개변수로 치환할 값을 전달

> 중괄호의 개수가 format() 함수의 매개변수 개수보다 적은 경우는 정상 작동 하나 많은 경우에는 에러 발생

```
>>> a = 10
>>> b = 20
>>> '결과는 {} 입니다'.format(a)
결과는 10 입니다.
>>> '결과는 {} {} 입니다'.format(a, b)
결과는 10 20 입니다.
>>> '결과는 {} 입니다'.format(a, b)
결과는 10 입니다.
>>> '결과는 {} {} 입니다'.format(a)    # 중괄호가 매개변수보다 개수가 많아 에러 발생
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: Replacement index 1 out of range for positional args tuple
```

사용할 매개변수를 지정

중괄호 내에 숫자를 입력할 경우 format() 함수의 매개변수의 해당 번째 데이터를 사용

> 중괄호 내 숫자를 입력하여 사용하는 경우에는 모든 중괄호 내부에 숫자를 입력해 주어야 정상 작동

```
>>> a = 10
>>> b = 20
>>> '결과는 {0} 입니다'.format(a, b)
결과는 10 입니다.
>>> '결과는 {1} 입니다'.format(a, b)
결과는 20 입니다.
>>> '결과는 {1} {0} 입니다'.format(a, b)
결과는 20 10 입니다.
>>> '결과는 {0} {0} 입니다'.format(a, b)
결과는 10 10 입니다.
>>> '결과는 {0} {} 입니다'.format(a, b)    # 매개변수에 대한 인덱스 정보가 없는 중괄호가 있어 에러 발생
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: cannot switch from automatic field numbering to manual field specification
```

출력 데이터의 형식 지정

C/C++의 `sprintf()`와 같이 중괄호 항목에 들어갈 형식에 대하여 지정이 가능하며 `{:x}` 형태로 `:` 뒤에 입력 받을 형식을 지정

문자열 형식

형식명	출력되는 문자열 형식	입력 받을 수 있는 형식
s	문자열	문자열
d	정수	정수, 소수

f	소수	정수, 소수
b	2진수	정수
o	8진수	정수
x	16진수 (소문자)	정수
X	16진수 (대문자)	정수
c	문자(1글자)	정수
e	10의 제곱(소문자 e)	정수, 소수
e	10의 제곱(대문자 E)	정수, 소수
%	퍼센트	정수, 소수

> 다른 형식의 값이 들어온 경우에는 에러 발생

```
>>> '결과는 {:d} 입니다.'.format(10)
결과는 10 입니다.
>>> '결과는 {:f} 입니다.'.format(10)
결과는 10.000000 입니다.
>>> '결과는 {:x} 입니다.'.format(10)
결과는 a 입니다.
>>> '결과는 {:s} 입니다.'.format(10)    # s는 정수형을 입력으로 받을 수 없음
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: Unknown format code 's' for object of type 'int'
```

출력 데이터의 포맷 지정

자리수 및 정렬, 부호 등을 지정
출력 형식 지정자와 함께 사용하여야 하며, 출력 형식자의 앞에 지정

출력 형식

형식명	출력 형식
숫자	해당 숫자칸만큼 출력 칸을 사용하고 오른쪽 정렬
0숫자	해당 숫자칸만큼 출력 칸을 사용하고 오른쪽 정렬, 빈칸은 0으로 채움
+	양수인 경우 부호(+) 출력
공백	양수인 경우 앞에 공백을 포함하여 출력
=숫자	해당 숫자칸만큼 출력 칸을 사용하고 오른쪽 정렬하고, 첫번째 위치에 부호를 표현
.숫자	소수점 출력 자리수 지정(반올림), 소수에서만 사용 가능

```
>>> '결과는 {:d} 입니다.'.format(-100)         # 기본적인 음수 출력
결과값은 -100 입니다.

>>> '결과는 {:10d} 입니다.'.format(10)         # 오른쪽 정렬 후 빈칸으로 채움
결과는         10 입니다.
>>> '결과는 {:010d} 입니다.'.format(10)        # 오른쪽 정렬 후 0으로 채움
```

```
결과는 0000000010 입니다.
>>> '결과는 {:10d} 입니다.'.format(-10)        # 오른쪽 정렬 후 0으로 채움
결과는        -10 입니다.
>>> '결과는 {:010d} 입니다.'.format(-10        # 오른쪽 정렬 후 0으로 채우나, 음수인 경우에는 부호를 맨 처음 위치에 기록
결과는 -000000010 입니다.

>>> '결과는 {:d} 입니다.'.format(-100)
결과값은 -100 입니다.
>>> '결과는 {:+d} 입니다.'.format(100)         # 양수의 경우 + 부호 출력
결과값은 +100 입니다.
>>> '결과는 {: d} 입니다.'.format(100)         # 양수의 경우 앞에 공백 출력
결과값은  100 입니다.

>>> '결과는 {:.0f} 입니다.'.format(100.1239)   # 소수점 자리 설정(0자리)
결과값은 100 입니다.
>>> '결과는 {:.3f} 입니다.'.format(100.1239)   # 소수점 자리 설정(3자리)
결과값은 100.124 입니다.

>>> '결과는 {:s} 입니다.'.format(10)    # s는 정수형을 입력으로 받을 수 없음
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: Unknown format code 's' for object of type 'int'
```

f-string

매개변수로 변수를 전달 하는 경우, 바로 문자열에 바로 변수를 사용하는 방법

문자열 앞에 `f`를 붙여서 사용

```
>>> result = '합격'
>>> point = 75
>>> f'결과는 {result}, 점수는 {point}점입니다.'
결과는 합격, 점수는 75점입니다.

>>> result = '합격'
>>> point_math = 9
>>> point_eng = 12
>>> f'결과는 {result}, 점수는 {point_math + point_eng}점입니다.'
결과는 합격, 점수는 21점입니다.
```

문자열 편집

앞 뒤 공백 제거

문자열에 대하여 아래의 함수를 이용하여 앞, 뒤 공백이 제거

함수명	기능
`strip()`	앞,뒤 공백 제거
`lstrip()`	앞 공백 제거
`rstrip()`	뒤 공백 제거

```
>>> a = '          Hello!          '
>>> b = '          월드            '
```

```
>>> print(a + b)
          Hello!                    월드
>>> print(a.strip() + b.strip())
Hello!월드
>>> print(a.rstrip() + b.lstrip())
          Hello!월드
```

문자열이 구성된 형식 확인

해당 문자열이 숫자로만 구성되어 있는지, 소문자로만 구성되어 있는지를 판단하는 함수
다른 언어의 경우 일반적으로 정규표현식을 사용하여야 함

함수명	기능
`isalpha()`	알파벳으로만 구성
`isdigit()`	숫자로만 구성
`isdecimal()`	정수 형태로만 구성
`isalnum()`	알파벳/숫자만 구성
`isspace()`	공백 문자열로만 구성
`islower()`	소문자로만 구성
`isupper()`	대문자로만 구성
`isidentifier()`	식별자(ID)로 사용 가능 여부

```
>>> pw = '1q2w3e4r'
>>> f'비밀번호 사용 가능? {True != pw.isalnum()}'
비밀번호 사용 가능? False
```

부분 문자열 생성

`대괄호와 콜론 [:]`를 이용하며, 시작 위치 및 종료 위치를 지정하여 준다

> 개수가 아니므로 유의 할 것

> 숫자를 지정하지 않으면 시작지점 or 종료지점을 자동으로 사용

```
>>> tel = '01012345678'
>>> digit_1 = tel[:3]
>>> digit_2 = tel[3:7]
>>> digit_3 = tel[7:11]
>>> f'{digit_1}-{digit_2}-{digit_3}'
010-1234-5678
```

지정한 문자열의 최초 위치 검색

앞뒤로 검색이 가능하며, 해당 문자열이 나온 최초 위치의 Index를 반환
없는 경우에는 -1을 반환

함수명	기능
`find()`	앞에서부터 검색
`rfind()`	뒤에서부터 검색

```
>>> path = 'C:\\Windows\\notepad.exe'
>>> drive_pos = path.find('\\')
>>> drive = path[:drive_pos]
>>> f'{drive_pos} {drive}'
2 C:
>>> file_pos = path.rfind('\\')
>>> file = path[file_pos+1:]
>>> f'{file_pos} {file}'
10 notepad.exe
```

문자열 존재 확인

문자열에서 해당 문자열이 존재하는지 확인하는 기능
함수를 사용하지 않고 문자열에 `in`을 사용함

```
>>> path = 'C:\\Windows\\notepad.exe'
>>> 'Windows' in path
True
>>> path.find('Windows')
3
>>> 'excel' in path
False
>>> path.find('excel')
-1
```

문자열 토큰화

일정 규격으로 나누어져 있는 문자열 분해 (ex. CSV 파일)
`split()` 함수를 사용

```
>>> row = '쿠크다스|12개입|4,800원'
>>> tokens = row.split('|')
>>> print(tokens[0])
쿠크다스
>>> print(tokens[1])
12개입
>>> print(tokens[2])
4,800원
```

문자열의 길이 확인

`len()` 함수를 이용하여 확인

```
>>> len('1234')
4
```

```
>>> len('가나다')
3
```

과제

입력받은 CSV 파일에서 아래의 조건인 row를 화면으로 출력(print 함수 사용)

- 이름에 '엄마손 파이'가 포함 (ex. '엄마손 파이 번들', '할인 엄마손 파이')
- 가격이 숫자로만 이루어져 있지 않음

Input

Column은 아래와 같음

- 이름
- 재고 개수
- 가격

```
쿠크다스,14232,3400
초코하임 화이트,122,3300
엄마손 파이 딸기맛,123,2800
쿠크다스 귀지맛,43543,2400
초코하임 초코맛,1243,3300
엄마손 파이 유통기한 지난 것,133,1800원
```

Output 예제

```
엄마손 파이 유통기한 지난 것,133,1800원
```