Unit 24.

Status	시작 전
담당자	
마감일	
완료일	

Unit 24. 문자열 응용하기

24.1.1 문자열 바꾸기

replace('바꿀문자열', '새문자열')은 문자열 안의 문자열을 다른 문자열로 바꾼다

```
>>> 'Hello, world!'.replace('world', 'Python')
'Hello, Python!'
```

바뀐 결과를 유지하고 싶으면 replace를 사용한뒤에 다시 변수에 할당

```
>>> s = 'Hello, world!'
>>> s = s.replace('world!', 'Python')
>>> s
'Hello, Python'
```

24.1.2 문자 바꾸기

translate는 문자열 안의 문자를 다른 문자로 바꾼다. **str.maketrans('바꿀문자', '새문자')**
로 변환 테이블을 만듭니다. 다음에 **translate(테이블)**을 사용하면 문자를 바꾼 뒤 결과를 반환한다.

```
>>> table = str.maketrans('aeiou', '12345')
>>> 'apple'.translate(table)
'1ppl2'
```

24.1.3 문자열 분리하기

split( )은 공백을 기준으로 문자열을 분리하여 리스트로 만든다.

```
>>> 'apple pear grape pineapple orange'.split()
['apple', 'pear', 'grape', 'pineapple', 'orange']
```

24.1.5 대문자를 소문자로 소문자를 대문자로

```
>>> 'python'.upper()
'PYTHON'
```

```
>>> 'PYTHON'.lower()
'python'
```

24.1.7 왼쪽 공백 오른쪽 공백 삭제하기

lstrip( )

```
>>> '   Python   '.lstrip()
'Python   '
```

rstrip( )

```
>>> '   Python   '.rstrip()
'   Python'
```

24.1.9 양쪽의 특정 문자 삭제하기

strip( )

strip('삭제할문자들')과 같이 삭제할 문자들을 문자열 형태로 넣어주면 문자열 양쪽에 있는 해당 문자를 삭제한다.

```
>>> ', python.'.strip(',.')
' python'
```

문자열 위치 찾기

find('찾을문자열') find는 왼쪽에서 부터 문자열을 찾는다

오른쪽에서 부터 문자열 찾으려면 **rfind('찾을문자열')**

```
>>> 'apple pineapple'.rfind('pl')
12
>>> 'apple pineapple'.rfind('xy')
-1
```

- **index('찾을문자열')**

문자열 개수 세기

count('문자열')은 현재 문자열에서 특정 문자열이 몇 번 나오는지 알아낸다.

24.2 문자열 서식 지정자와 포매팅 사용하기

문자열에는 서식 지정자와 포매팅이 있다

문자열 안에서 특정 부분을 원하는 값으로 바꿀 때 서식 지정자 또는 문자열 포매팅을 사용

24.2.1 서식 지정자로 사용하기

• '%s' % '문자열'

```
>>> 'I am %s.' % 'james'
'I am james.'
```

• '%d' % 숫자

```
>>> 'I am %d years old.' % 20
'I am 20 years old.'
```

24.2.3 서식 지정자로 소수점 표현하기

• '%f' % 숫자

```
>>> '%f' % 2.3
'2.300000'
```

소수점 이하 자릿수를 지정하고 싶다면 다음과 같이 f 앞에 .(점)과 자릿수를 지정해주면 된다.

```
>>> '%.2f' % 2.3
'2.30'
>>> '%.3f' % 2.3
'2.300'
```

24.2.4 서식 지정자로 문자열 정렬하기

% 뒤에 숫자를 붙이면 문자열을 지정된 길이로 만든 뒤 오른쪽으로 정렬하고 남는 공간을 공백으로 채운다. 오른쪽으로 정렬하고 남는 공간을 공백으로 채운다.

• %길이s

```
>>> '%10s' % 'python'
'    python'
```

왼쪽 정렬은 길이 왼쪽에 - 를 붙인다

• %-길이s

24.2.5 서식 지정자로 문자열 안에 값 여러 개 넣기

- 문자열 안에 값을 두 개 이상 넣으려면 %를 붙이고, 괄호 안에 값(변수)을 콤마로 구분해서 넣어주면 된다.
- **'%d %s' % (숫자, '문자열')**
- 값을 괄호로 묶지 않으면 에러가 뜬다

```
>>> 'Today is %d %s.' % (3, 'April')
'Today is 3 April.'
```

서식 지정자가 여러 개면 괄호 안의 값(변수) 개수도 서식 지정자 개수와 똑같이 맞춰주어야 한다. 서식 지정자를 서로 붙이면 결과도 붙어서 나오므로 주의

24.2.6 format 메서드 사용하기

• '{인덱스}'.format(값)

- 문자열 포매팅은 { }(중괄호) 안에 포매팅을 지정하고 format 메서드로 값을 넣는다.

```
>>> 'Hello, {0}'.format('world!')
'Hello, world!'
>>> 'Hello, {0}'.format(100)
'Hello, 100'
```

- 같은 인덱스가 지정된 { }를 여러 개 넣으면 같은 값이 여러 개 들어간다.

```
>>> '{0} {0} {1} {1}'.format('Python', 'Script')
'Python Python Script Script'
```

- 문자열 포매팅에 변수를 그대로 사용

```
>>> language = 'Python'
>>> version = 3.6
>>> f'Hello, {language} {version}'
'Hello, Python 3.6'
```

24.2.12 format 메서드로 문자열 정렬하기

인덱스 뒤에 :(콜론)을 붙이고 정렬할 방향과 길이를 지정

• '{인덱스:<길이}'.format(값)

```
>>> '{0:<10}'.format('python')
'python
```

24.2.13 숫자 개수 맞추기

%와 d 사이에 0과 숫자 개수를 넣어주면 자릿수에 맞춰서 앞에 0이 들어간다. { }를 사용할 때는 인덱스나 이름 뒤에 :(콜론)를 붙이고 03d 처럼 0과 숫자 개수를 지정하면 됩니다.

- **'%0개수d' % 숫자**
- **'{인덱스:0개수d}'.format(숫자)**

```
>>> '%03d' % 1
'001'
>>> '{0:03d}'.format(35)
'035'
```

소수점 이하 자릿수를 지정하고 싶으면 %08.2f처럼 .(점) 뒤에 자릿수를 지정해줍니다.

- **'%0개수.자릿수f' % 숫자**
- **'{인덱스:0개수.자릿수f'}.format(숫자)**

```
>>> '%08.2f' % 3.6
'00003.60'
>>> '{0:08.2f}'.format(150.37)
'00150.37'
```