

기초 PYTHON 프로그래밍

4. 문자열 자료형

1. 문자열 생성하기
2. 문자열 인덱싱 / 슬라이싱
3. 문자열 연결 / 반복
4. 문자열 길이 / 포함 관계
5. 문자열 메소드 사용하기

1. 문자열 생성하기

◆ 문자열 객체는 따옴표를 이용하여 생성한다.

- 홑따옴표 (' ... ')
- 쌍따옴표 (" ... ")
- 홑따옴표 세 개 (''' ... ''')
- 쌍따옴표 세 개 (""" ... """)

```
>>> s = 'Python is great!'
>>> s = "Python is great!"
>>> s = '''Python is great!'''
>>> s = """Python is great!"""
```

```
>>> print(s)
Python is great!
```

1. 문자열 생성하기

◆ 역슬래쉬(\)를 이용하여 긴 문자열 생성하기

- 홑따옴표 또는 쌍따옴표 한 쌍을 사용할 때 역슬래쉬(\)는 문자열이 연결됨을 의미한다.

```
>>> sentence = 'Python is the \nmost popular programming \nlanguage in these days.'
```

```
>>> print(sentence)
```

```
Python is the most popular programming language in these days.
```

1. 문자열 생성하기

◆ 홀따옴표와 쌍따옴표를 같이 사용하기

- **say 'hello' to mom.**

```
>>> a = 'say 'hello' to mom'
SyntaxError: invalid syntax

>>> b = "say 'hello' to mom"
>>> c = '''say 'hello' to mom'''
>>> d = """"say 'hello' to mom""""
>>> print(d)
say 'hello' to mom
```

```
>>> s = 'say "hello" to mom'
>>> s = '''say "hello" to mom'''
>>> s = """"say "hello" to mom""""
>>> print(s)
say "hello" to mom
```

1. 문자열 생성하기

◆ 홀따옴표 또는 쌍따옴표 세 개 사용하기

```
>>> s = '''say 'hello' to "mom"'''
```

```
>>> print(s)
```

```
say 'hello' to "mom"
```

```
>>> s = """say "hello" to 'mom'"""
```

```
>>> print(s)
```

```
say "hello" to 'mom'
```

1. 문자열 생성하기

◆ 홀따옴표 또는 쌍따옴표 세 개 사용하기

```
# letter to Alice

print("""Dear Alice,

How are you?
I am busy to study programming this vacation.

Say hello to your parents.

Sincerely,
Bob""")
```

```
Dear Alice,

How are you?
I am busy to study programming this vacation.

Say hello to your parents.

Sincerely,
Bob
```

2. 문자열 인덱싱 / 슬라이싱

◆ 문자열 객체의 특징

- **immutable**하다.
- 순서가 있는 자료형으로 인덱싱을 이용할 수 있다.

```
>>> greeting = 'hello world'
```

인덱스	0	1	2	3	4	5	6	7	8	9	10
greeting	h	e	l	l	o		w	o	r	l	d
음수 인덱스	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

2. 문자열 인덱싱 / 슬라이싱

◆ 문자열 인덱싱

```
>>> greeting = 'hello world'
```

인덱스	0	1	2	3	4	5	6	7	8	9	10
greeting	h	e	l	l	o		w	o	r	l	d
	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> print(greeting[7])
```

o

```
>>> print(greeting[0])
```

h

```
>>> greeting[0] = 'H'    # TypeError 발생
```

문자열 객체는 **immutable**하기 때문에 생성된 후에 수정할 수 없다.

2. 문자열 인덱싱 / 슬라이싱

- ◆ 문자열 슬라이싱 - 범위를 이용하여 문자열 일부분에 접근한다

```
>>> lang = 'python programming'
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
lang	p	y	t	h	o	n		p	r	o	g	r	a	m	m	i	n	g
	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- `lang[a]` - 인덱스 `a`의 문자
- `lang[a:b]` - 인덱스 `a`부터 `b-1`까지의 문자열 ($a < b$)
- `lang[a:b:c]`
 - $a < b, c > 0$ - `a`부터 `b-1`까지 `c` 간격의 문자열
 - $a > b, c < 0$ - `a`부터 `b+1`까지 `c` 간격의 문자열

`lang[:]` - lang 문자열 전체
`lang[::-1]` - lang 문자열 전체

2. 문자열 인덱싱 / 슬라이싱

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
lang	p	y	t	h	o	n		p	r	o	g	r	a	m	m	i	n	g
	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

lang[3:10]	# 'hon pro'
lang[:5]	# 'pytho' (lang[0:5]와 같다)
lang[10:]	# 'gramming' (10부터 끝까지)
lang[5:-8]	# 'n pro' (양수, 음수 섞어서 사용 가능)
lang[10:5]	# 빈 문자열
lang[-10:-13]	# 빈 문자열
lang[2:10:3]	# 'tnr'
lang[:10:3]	# 'ph o' (lang[0:10:3]와 같다)
lang[10::2]	# 'gamn' (10부터 끝까지 2칸씩)
lang[12:3:-2]	# 'agr o' (12에서 4까지 2칸씩 앞으로)

3. 문자열 연결 / 반복

◆ 문자열 연결하기 (+)

```
a = 'hello'  
b = 'world'
```

```
c = a + b  
print(c)  
d = a + ' ' + b  
print(d)
```

```
helloworld  
hello world
```

```
score = 95
```

```
x = 'I got ' + score + ' in the exam.' # 에러
```

```
x = 'I got ' + str(score) + ' in the exam.'
```

```
print(x)
```

```
I got 95 in the exam.
```

3. 문자열 연결 / 반복

◆ 문자열 반복하기 (*)

```
>>> a = 'hello'
```

```
>>> a * 3    # 문자열을 3회 반복한다
```

```
'hellohellohello'
```

4. 문자열 길이 / 포함 관계

◆ 문자열 길이 - len() 내장 함수

```
>>> subject = 'programming'
>>> len(subject)
11
```

◆ 문자열 포함 - in, not in 연산자

```
>>> 'r' in subject
True
>>> 'gram' in subject
True
>>> 'abcd' not in subject
True
```

5. 문자열 메소드 이용하기

◆ 문자열 객체에서 사용할 수 있는 메소드

- 메소드는 객체에 어떤 일을 처리하도록 하는 코드이다.
- 문자열.메소드() 형태로 사용한다.

```
>>> dir(str)
['__add__', '__class__', ... '__subclasshook__', 'capitalize', 'casefold',
'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format',
'format_map', 'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit',
'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle',
'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace',
'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines',
'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

5. 문자열 메소드 이용하기

메소드	설명
upper()	문자열을 모두 대문자로 바꾼다. <pre>>>> name = 'steve jobs' >>> name.upper() # 'steve jobs'.upper() 'STEVE JOBS'</pre>
lower()	문자열을 모두 소문자로 바꾼다. <pre>>>> school = 'SOGANG University' >>> school.lower() 'sogang university'</pre>
isupper() islower()	isupper() 메소드는 문자열이 모두 대문자이면 True를 반환한다. islower() 메소드는 문자열이 모두 소문자이면 True를 반환한다. <pre>>>> a = 'hello'; b = 'WORLD' >>> a.islower() True >>> b.isupper() True</pre>

5. 문자열 메소드 이용하기

메소드	설명
capitalize()	첫 문자를 대문자로 바꾼다. <pre>>>> y = 'steve jobs made toy story' >>> y.capitalize() 'Steve jobs made toy story'</pre>
title()	문자열에서 각 단어의 첫 문자들을 대문자로 바꾼다. <pre>>>> y.title() 'Steve Jobs Made Toy Story'</pre>
istitle()	<pre>>>> x = 'I Am Learning Programming.' >>> x.istitle() True >>> y = 'I am learning Programming.' >>> y.istitle() False</pre>

5. 문자열 메소드 이용하기

메소드	설명
count()	<p>부분 문자열을 센다.</p> <pre>>>> state = 'mississippi' >>> state.count('s') 4 >>> state.count('ssi') 2 >>> state.count('s', 5) # [5:] 으로 잘라서 count한 결과 2 >>> state.count('s', 1, 5) # [1:5] 범위 2</pre>
find()	<p>부분 문자열 찾아서 첫 인덱스 알려 준다.</p> <pre>>>> state.find('s') 2 >>> state.find('i', 5) # [5:] 범위에서 'i'를 찾는다 7</pre>

5. 문자열 메소드 이용하기

메소드	설명
join(리스트)	<p>문자열.join(리스트) - 리스트에 있는 자료들을 문자열로 연결한다.</p> <pre>>>> friends = ['alice', 'bob', 'cindy'] >>> dash = '-' >>> dash.join(friends) # '-'.join(friends) 'alice-bob-cindy'</pre>
split()	<p>문자열을 스페이스 기준으로 잘라서 리스트에 저장한다.</p> <pre>>>> lists = 'alice bob cindy david' >>> lists.split() # 스페이스로 분리 ['alice', 'bob', 'cindy', 'david'] >>> date = '2016/12/25' >>> date.split('/') # '/'로 분리 ['2016', '12', '25']</pre>