

문자열 자료형의 메소드와 포맷팅



학습목표

- 문자열 자료형의 특징을 알고, 문자열에 각종 연산을 적용할 수 있다.
- <mark>문자열 메소드의 종류와 사용법을</mark> 알고, 기본적인 코딩을 할 수 있다.
- <mark>문자열 포맷팅 방법</mark>에 대해 알고, 기본적인 코딩을 할 수 있다.

학습내용

- 문자열 자료형의 특징 및 각종 연산
- 문자열 메소드
- 문자열 포맷팅

1 문자열 자료형의 특징

문자, 단어 등으로 구성된 문자들의 집합

문자열 안에 따옴표 넣기

이스케이프 문자 사용(₩)

따옴표를 다르게 사용

여러 줄의 문장 표현

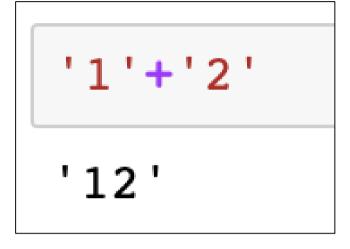
이스케이프 문자 사용(₩)

따옴표를 세 개 사용





연결 연산자(+)





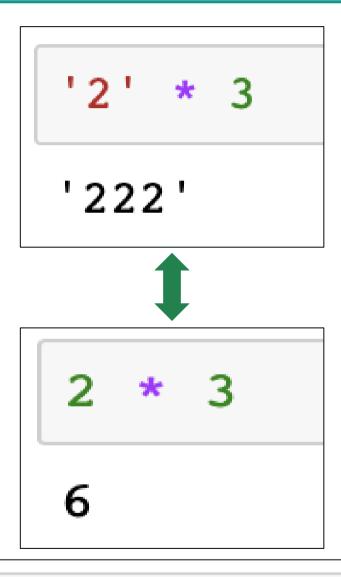
```
1 + 2
```

```
a = 'Hello! '
b = "World"
print(a+b)
Hello! World
```

2 문자열 자료형의 연산



반복 연산자(*)



'안녕하세요'*3

' 안녕하세요안녕하세요안녕하세요 '





선택 연산자(인덱싱)

문자	안	녕	하	세	요
인덱스	0	1	2	3	4

a = '안녕하세요' print(a[0])

안

문자열은 시퀀스 자료형으로 인덱스가 있고, 인덱스로 값의 접근이 가능

요





범위 선택 연산자(슬라이싱)

문자	안	녕	하	세	요
인덱스	0	1	2	3	4

a = '안녕하세요' print(a[1:3]) 녕하

a = '안녕하세요'
print(a[0:5:2])
안하요

변수[시작(이상):끝(미만):스텝]

2 문자열 자료형의 연산



각종 연산자의 활용

```
#역순 정렬
a = '안녕하세요'
print(a[::-1])
요세하녕안
```

```
# 술라이상, 연결
a = '12345'
b = '34567'
c = a[0:3] + b[1:]
print(c)
```

```
#소텔 활용
a = '1,2,3,4,5,6,7,8,9'
print(a[::2])
123456789
```

☑ 문자열 메소드의 종류와 활용법

파이썬의 문자열 자료형은 변경이 불가능하기 때문에 직접 문자열을 수정하는 방식이 아닌 변경된 문자를 반환하는 방식으로 메소드를 사용

```
a = 'abcdefghi'
print(a.upper())
print(a)
```

ABCDEFGHI abcdefghi

☑ 문자열 메소드의 종류와 활용법

upper()

문자열을 대문자로 변경

a = 'abcdefghi' print(a.upper())

ABCDEFGHI

lower()

문자열을 소문자로 변경

a = 'abcdefghi' print(a.lower())

abcdefghi

☑ 문자열 메소드의 종류와 활용법

swapcase()

대문자는 소문자로, 소문자는 대문자로 변경

```
a = 'abCDefgHi'
print(a.swapcase())
```

ABcdEFGh I

capitalize()

첫 문자만 대문자로 변경

```
a = 'abcd abcd'
print(a.capitalize())
```

Abcd abcd

☑ 문자열 메소드의 종류와 활용법

title()

각 단어의 첫 문자를 대문자로 변경

```
a = 'abcd abcd'
print(a.title())
Abcd Abcd
```

strip()

문자열 양쪽 끝을 잘라냄(기본은 공백)

```
a = ' abcd efg '
print(a)
print(a.strip())

abcd efg
abcd efg
```

☑ 문자열 메소드의 종류와 활용법

istrip()

문자열 왼쪽 끝을 잘라냄

```
a = ' abcd efg '
print(a)
print(a.lstrip())

abcd efg
abcd efg
```

rstrip()

문자열 오른쪽 끝을 잘라냄

```
a = ' abcd efg '
print(a)
print(a.rstrip())

abcd efg
abcd efg
```

☑ 문자열 메소드의 종류와 활용법

split()

문자열을 구분자로 분리해 리스트로 반화

```
a = 'abcde'
print(a.split('c'))
['ab', 'de']
```

splitlines()

문자열을 라인 단위로 분리해 리스트로 반환

```
a = """abcd
efge"""
print(a.splitlines())
['abcd', 'efge']
```

☑ 문자열 메소드의 종류와 활용법

index()

해당 문자열의 인덱스를 반환 (없는 경우 에러)

```
a = 'abcdef'
print(a.index('e'))
4
```

📨 문자열 메소드의 종류와 활용법

find()

해당 문자열의 인덱스를 반환 (없는 경우 -1)

```
a = 'abcdef'
print(a.find('e'))
4
```

```
a = 'abcdef'
print(a.find('g'))
-1
```



☑ 문자열 메소드의 종류와 활용법



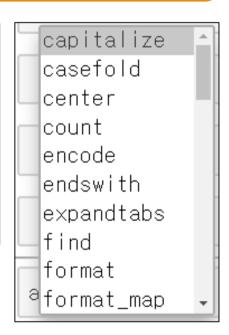
그 외 문자열 메소드

count, startswith, isdigit, isnumeric, isalpha



- ① dir(str) 명령으로 확인 가능
- ② 문자열 정의 후 . + Tab 키로 확인 가능

```
print(dir(str))
['__add__', '__class__', '__cd
'__getitem__', '__getnewargs_
_mod__', '__mul__', '__ne__',
_', '__str__', '__subclasshook
'format_map', 'index', 'isalnu
```







변수를 활용하여 문자열을 생성할 때 사용

- 중괄호 안에 format()의 내용 삽입
 - ➡ C언어의 printf문과 비슷

```
a = "{}, World!".format("Hello")
print(a)
```

Hello, World!



☑ 문자열 포맷팅의 활용



여러 개를 사용하려면, 순서대로 값을 입력

```
a = "{} {} ".format("Hello",", ","World!")
print(a)
```

Hello, World!



중괄호에는 문자열이 아닌 다른 자료형이 올 수

```
print(a)
```

2, 2, (1, 2), [1, 2]



☑ 문자열 포맷팅의 활용



중괄호에 순서를 적어 포맷팅 출력 순서를 변경 가능

```
a = "{1} {2} {0} ".format("Hello",", ","World!")
print(a)
```

, World!Hello



중괄호에 식별자를 적어 출력 가능

```
a = "Today is {today}, Tomorrow is {tomorrow}".format(today="Monday",tomorrow="Tuesday")
print(a)
```

Today is Monday, Tomorrow is Tuesday





자리 표기 가능



- ① f로 타입을 실수로 변경
- ②:0.숫자로 소수점 자리 수 변경
- ③ :숫자로 공백 출력 가능

Run! 프로그래밍

Mission 1

문자열의 특징, 메소드 등을 활용하여 특정 문자 뽑아내기

s = """Today is December 31, 2020.

Tomorrow is January 1, 2021."""

정답

#인덱스를 찾아 슬라이싱 연산 print(s.find('3')) print(s.find('1')) print(s[18:20])

#split 함수를 사용해 분리 s = s.splitlines()[0] s = s.split(',')[0] print(s.split(' ')[3])

Run! 프로그래밍

Mission 2

문자열 포맷팅 활용

```
print("{} X {:f} = {:2}".format(2,2,4))

print("{} X {} = {:2}".format(2,3,6))

print("{} X {} = {:2}".format(2,4,8))

print("{} X {} = {:2}".format([2],5,10))

print("{} X {} = {:2}".format(2,6,12))

print("{} X {} = {:2}".format(2,7,"십사"))

print("{} X {} = {:2}".format(2,8,16))

print("{} X {} = {:2}".format(2,8,16))

print("{} X {:0.1f} = {:2.1f}".format(2,9,18))
```

학습정리

1. 문자열 자료형의 특징 및 각종 연산

문자열 자료형이란?	• 문자, 단어 등으로 구성된 문자들의 집합
문자열 자료형의 특징	・ 큰따옴표(")와 작은따옴표(') 모두 사용 가능 ・ 내장 함수 str()을 활용해 문자열 자료형으로 변경 가능
문자열 자료형의 연산	• 인덱싱, 슬라이싱, 각종 연산을 수행할 수 있음

2. 문자열 메소드

문자열	• 파이썬의 문자열 자료형은 변경이 불가능하기 때문에 직접
메소드의	문자열을 수정하는 방식이 아닌 변경된 문자를 반환하는 방식으로
활용	메소드를 사용
문자열	• upper(), lower(), swapcase(), capitalize(), title(), strip(),
메소드의	lstrip(), rstrip(), replace('전','후'), join(), split(), splitlines(),
종류	index(), find()

학습정리

3. 문자열 포맷팅

문자열 포맷팅의 활용

- 변수를 활용하여 문자열을 생성할 때 사용
- 여러 개를 사용하려면, 순서대로 값을 입력
- 중괄호에는 문자열이 아닌 다른 자료형이 올 수 있음
- 중괄호에 순서를 적어 포맷팅 출력 순서를 변경 가능
- 중괄호에 식별자를 적어 출력 가능
- 자리 표기 가능