데이터 처리 프로그래밍

Data Processing Programming


03
문자열

목차

1. 문자열 개념

문자열String

- 문자, 단어 등으로 구성된 문자들의 집합
- 시퀀스 자료형sequence data type

s = “String”

S	t	r	i	n	g

```
>>> s = "String"
>>> s[0]
'S'
>>> s[1]
't'
>>> s[2]
'r'
>>> s[3]
'i'
>>> s[4]
'n'
>>> s[5]
'g'
```

문자열 생성

- 문자열은 작은따옴표(') 또는 큰 따옴표(")로 표현

```
>>> 'Hello'
'Hello'
>>> "파이썬은 재미있다."
'파이썬은 재미있다.'
>>> '''파이썬은 심플하다'''
'파이썬은 심플하다'
>>> """파이썬은 문자열 처리가 뛰어나다"""
'파이썬은 문자열 처리가 뛰어나다'
```

따옴표가 있는 문자열 생성

- 문자열에 작은따옴표가 있을 경우, 큰따옴표로 둘러싸서 표현
- 문자열에 큰따옴표가 있을 경우, 작은따옴표로 둘러싸서 표현
- 이스케이프 코드 \를 이용하여 작은따옴표(')와 큰따옴표(")를 문자열에 포함

```
>>> string = "Python's built-in string classes"
>>> string
"Python's built-in string classes"
>>> string = '다들 "파이썬은 매우 쉽다."라고 말한다.'
>>> string
'다들 "파이썬은 매우 쉽다."라고 말한다.'
>>> string = 'Python\'s built-in string classes'
>>> string
"Python's built-in string classes"
>>> string = "다들 \"파이썬은 매우 쉽다.\"라고 말한다."
>>> string
'다들 "파이썬은 매우 쉽다."라고 말한다.'
```

이스케이프 문자

- 특수 문자
- 문자 앞에 백슬래시(\)를 사용
- 문자열에서 특수 문자를 표현하는데 사용

이스케이프 문자	이름
₩₩	역슬래시backslash
₩'	작은따옴표single quote
₩"	큰따옴표double quote
₩a	벨bell
₩b	백스페이스backspace
₩f	폼피드formfeed
₩n	라인피드linefeed
₩r	캐리지리턴carriage return
₩t	수평 탭tab
₩v	수직 탭tab
₩ooo	8진수 문자octal value
₩xhh	16진수 문자hex value

여러 줄이 있는 문자열 생성

- 이스케이프 문자(\n)를 이용하여 여러 줄이 있는 문자열 생성

```
>>> text = "동해물과 백두산이 마르고 닳도록\n하느님이 보우하사 우리나라 만세.\n무궁화 삼천리 화려강산\n대한 사람, 대한으로 길이 보전하세."
>>> print(text)
동해물과 백두산이 마르고 닳도록
하느님이 보우하사 우리나라 만세.
무궁화 삼천리 화려강산
대한 사람, 대한으로 길이 보전하세.
```

- 작은 따옴표 3개 또는 큰 따옴표 3개를 이용하여 여러 줄이 있는 문자열 생성

```
>>> text = '''동해물과 백두산이 마르고 닳도록
... 하느님이 보우하사 우리나라 만세.
... 무궁화 삼천리 화려강산
... 대한 사람, 대한으로 길이 보전하세.'''
>>> print(text)
동해물과 백두산이 마르고 닳도록
하느님이 보우하사 우리나라 만세.
무궁화 삼천리 화려강산
대한 사람, 대한으로 길이 보전하세.
```

```
>>> text = """동해물과 백두산이 마르고 닳도록
... 하느님이 보우하사 우리나라 만세.
... 무궁화 삼천리 화려강산
... 대한 사람, 대한으로 길이 보전하세."""
>>> print(text)
동해물과 백두산이 마르고 닳도록
하느님이 보우하사 우리나라 만세.
무궁화 삼천리 화려강산
대한 사람, 대한으로 길이 보전하세.
```

2. 문자열 연산

문자열 더하기

- + 연산자를 사용하여 문자열 연결

```
>>> s1 = "동해물과 "
>>> s2 = "백두산이"
>>> s1 + s2
'동해물과 백두산이'
```

문자열 곱하기

- * 연산자를 사용하여 문자열 반복

```
>>> s = "String "
>>> s * 3
'String String String '
```

문자열 길이

- 문자열 길이를 구하는 len() 함수

```
>>> s = "String "
>>> len(s)
7
>>> len(s * 3)
21
```

문자열 인덱싱indexing

- 문자열은 리스트처럼 문자 하나하나가 상대적인 주소offset를 가짐
- 이 주소를 사용해 할당된 값을 가져오는 인덱싱을 사용

s = “String”

S	t	r	i	n	g
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

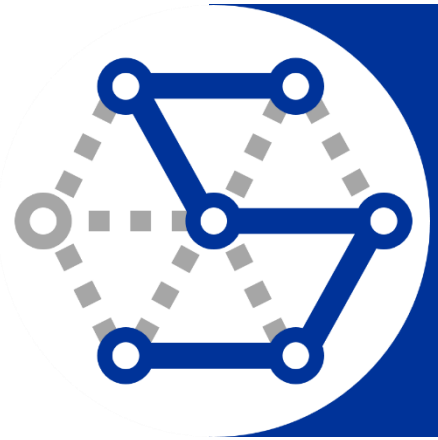
```
>>> s = "String"
>>> s[0]
'S'
>>> s[1]
't'
>>> s[2]
'r'
>>> s[3]
'i'
>>> s[4]
'n'
>>> s[5]
'g'
```

```
>>> s[-6]
'S'
>>> s[-5]
't'
>>> s[-4]
'r'
>>> s[-3]
'i'
>>> s[-2]
'n'
>>> s[-1]
'g'
```

문자열 슬라이싱slicing

- 문자열의 주소를 이용하여 문자열을 조각(부분)을 추출

```
>>> s = "SuanLab - Suan Computer Laboratory"
>>> s
'SuanLab - Suan Computer Laboratory'
>>> s[0:7]
'SuanLab'
>>> s[:7]
'SuanLab'
>>> s[10:]
'Suan Computer Laboratory'
>>> s[-10:]
'Laboratory'
>>> s[-19:-11]
'Computer'
```



3. 문자열 함수

문자열 함수

함수	설명
capitalize()	첫 문자를 대문자로하고, 나머지 문자를 소문자로 하는 문자열 반환
casefold()	모든 대소문자 구분을 제거
center(width [, fillchar])	길이 너비만큼 중앙정렬된 문자열 반환
count(sub [, start[, end]])	[start, end] 범위에서 부분 문자열 sub의 중복되지 않은 수를 반환

```
>>> s = "string"
>>> s = s.capitalize()
>>> s
'String'
>>> s = s.casefold()
>>> s
'string'
>>> s.center(10)
'  string  '
>>> s.count('s')
1
```

문자열 함수

함수	설명
find(sub [, start [, end]])	[start, end]에서 부분 문자열 sub가 문자열의 가장 작은 인덱스를 반환. sub가 발견되지 않는 경우는 -1 반환
rfind(sub [, start [, end]])	[start, end]에서 부분 문자열 sub가 문자열의 가장 작은 큰 인덱스를 반환. sub가 발견되지 않는 경우는 -1 반환
index(sub [, start [, end]])	find()과 유사하지만 부분 문자열 sub가 없으면 ValueError 발생
rindex(sub [, start [, end]])	find()과 유사하지만 부분 문자열 sub가 없으면 ValueError 발생

```
>>> s.find('t')
1
>>> s.find('z')
-1
>>> s.index('t')
1
>>> s.index('z')
Traceback (most recent call last):
  File "<input>", line 1, in <module>
ValueError: substring not found
```

문자열 함수

함수	설명
isalnum()	문자열의 모든 문자가 영숫자로 1개 이상 있으면 True, 아니면 False 반환
isalpha()	문자열의 모든 문자가 영문자로 1개 이상 있으면 True, 아니면 False 반환
isascii()	문자열의 모든 문자가 ASCII이거나 비어있으면 True, 아니면 False 반환

```
>>> s.isalnum()
True
>>> "한글".isalnum()
True
>>> "!@#".isalnum()
False
>>> s.isalpha()
True
>>> "한글".isalpha()
True
>>> "!@#".isalpha()
False
>>> s.isascii()
True
>>> "한글".isascii()
False
>>> "!@#".isascii()
True
```

문자열 함수

함수	설명
isdecimal()	문자열의 모든 문자가 10진수 문자이며 1개 이상 있을 때 True, 그렇지 않으면 False 반환
isdigit()	문자열의 모든 문자가 숫자이며 1개 이상 있을 때 True, 그렇지 않으면 False 반환
isidentifier()	문자열이 유효한 식별자인 경우 True 반환
islower()	문자열의 모든 문자가 소문자이며 1개 이상 있을 때 True, 그렇지 않으면 False 반환
isnumeric()	문자열의 모든 문자가 수치형이며 1개 이상 있을 때 True, 그렇지 않으면 False 반환

```
>>> s = "12345"
>>> s.isdecimal()
True
>>> "1.23".isdecimal()
False
>>> s.isdigit()
True
>>> "1.23".isdigit()
False
>>> s.isidentifier()
False
>>> "True".isidentifier()
True
>>> s = "String"
>>> s.islower()
False
>>> "string".islower()
True
>>> s = "12345"
>>> s.isnumeric()
True
>>> "1.23".isnumeric()
False
```

문자열 함수

함수	설명
isspace()	문자열 내에 공백 문자가 있고, 문자가 1개 이상 있으면 True, 그렇지 않으면 False
istitle()	문자열이 제목이 있는 문자열에 문자가 1개 이상 있으면 True, 그렇지 않으면 False
isupper()	문자열의 문자가 모두 대문자에 문자가 1개 이상 있으면 True, 그렇지 않으면 False
join(iterable)	iterable에 있는 문자열에 연결된 문자열을 반환
ljust(width [, fillchar])	너미만큼의 문자열에서 왼쪽 정렬된 문자열을 반환
rjust(width [, fillchar])	너미만큼의 문자열에서 오른쪽 정렬된 문자열을 반환
lower()	모든 대소문자가 소문자로 변환된 문자열을 반환

```
>>> " ".isspace()
True
>>> "  string ".isspace()
False
>>> "String".istitle()
True
>>> "STRING".istitle()
False
>>> "STRING".isupper()
True
>>> s = "String"
>>> " ".join(s)
'S t r i n g'
>>> "_".join(s)
'S_t_r_i_n_g'
>>> s.ljust(10)
'String    '
>>> s.lower()
'string'
```

문자열 함수

함수	설명
strip([chars])	문자열 양쪽에 지정된 chars 또는 공백을 제거한 문자열을 반환
lstrip([chars])	문자열 왼쪽에 지정된 chars 또는 공백을 제거한 문자열을 반환
rstrip([chars])	문자열 오른쪽에 지정된 chars 또는 공백을 제거한 문자열을 반환
partition(sep)	문자열에서 첫번째 sep를 기준으로 분할하여 3개의 튜플을 반환
rpartition(sep)	문자열에서 마지막 sep를 기준으로 분할하여 3개의 튜플을 반환
replace(old, new[,count])	문자열의 모든 old를 new로 교체한 문자열을 반환

```
>>> "  String  ".strip()
'String'
>>> "  String  ".lstrip()
'String  '
>>> "  String  ".rstrip()
'  String'
>>> "String".partition('t')
('S', 't', 'ring')
>>> "String".replace('Str', 'R')
'Ring'
```

문자열 함수

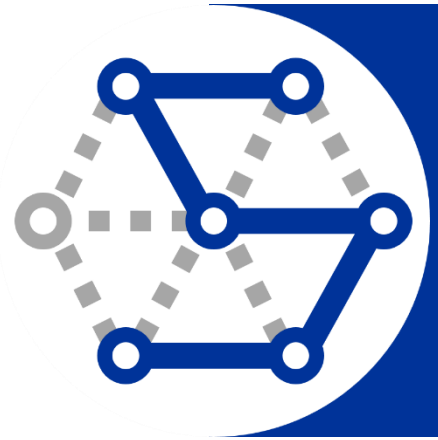
함수	설명
split(sep=None, maxsplit=1)	sep를 구분자 문자열로 사용하여 문자열의 단어 목록을 반환
rsplit(sep=None, maxsplit=1)	sep를 구분자 문자열로 사용하여 문자열의 단어 목록을 반환
splitlines([keepends])	문자열에서 라인 단위로 구분하여 리스트를 반환
startswith(prefix [, start[, end]])	[start, end] 범위에서 지정한 prefix로 끝나면 True, 아니면 False 반환
endswith(suffix [, start[, end]])	[start, end] 범위에서 지정한 suffix로 끝나면 True, 아니면 False 반환

```
>>> "1 2 3".split()
['1', '2', '3']
>>> "123\n123\n123".splitlines()
['123', '123', '123']
>>> "String".startswith('S')
True
>>> "String".endswith('g')
True
```

문자열 함수

함수	설명
swapcase()	문자열에서 소문자를 대문자로 대문자를 소문자로 변환한 문자열 반환
title()	문자열에서 첫 글자만 대문자이고 나머지는 소문자인 문자열 반환
upper()	문자열에서 모든 문자를 대문자로 변환한 문자열을 반환
zfill(width)	너비 만큼의 문자열에서 비어있는 부분에 '0'이 채워진 문자열 반환

```
>>> "String".swapcase()
'sTRING'
>>> "sTRING".title()
'String'
>>> "String".upper()
'STRING'
>>> "123".zfill(8)
'00000123'
```



4. 문자열 서식

문자열 포맷팅formatting

- 문자열 내에서 서식에 맞추어 특정 값을 삽입 또는 변경

```
>>> "나는 자전거가 %d개 있다." % 2
'나는 자전거가 2개 있다.'
>>> "나는 자전거가 %s 있다." % "두개"
'나는 자전거가 두개 있다.'
>>> "나는 %s에서 %s으로 가고 있다." % ("서울", "춘천")
'나는 서울에서 춘천으로 가고 있다.'
>>> number = 10
>>> loc_from = "서울"
>>> loc_to = "춘천"
>>> "자전거 %d대가 %s에서 %s으로 가고 있다." % (number, loc_from, loc_to)
'자전거 10대가 서울에서 춘천으로 가고 있다.'
```

문자열 포맷 코드

- 문자열 포멧팅에서 사용할 수 있는 다양한 포맷 코드

코드	설명
%s	문자열String
%c	문자Character
%d	정수Integer
%f	부동소수floating-point
%o	8진수
%x	16진수
%%	문자 '%'

정렬, 공백, 소수점 포맷

- 포맷 문자 앞에 숫자는 길이를 의미
- -는 왼쪽 정렬을 의미
- 소수점 '.' 뒤에 숫자는 소숫점 이하 개수를 의미

```
>>> "%8s" % "Hello"
'   Hello'
>>> "%-8sPython" % "Hello"
'Hello   Python'
>>> "%0.2f" % 3.1415926535897
'3.14'
>>> "%8.2f" % 3.1415926535897
'    3.14'
>>> "%-8.2f" % 3.1415926535897
'3.14    '
```

문자열 format 함수

- 문자열 format 함수의 인덱스를 이용한 고급 포맷팅

```
>>> "나는 자전거가 {0}개 있다.".format(2)
'나는 자전거가 2개 있다.'
>>> "나는 자전거가 {0} 있다.".format("두개")
'나는 자전거가 두개 있다.'
>>> "나는 {0}에서 {1}으로 가고 있다.".format("서울", "춘천")
'나는 서울에서 춘천으로 가고 있다.'
>>> number = 10
>>> loc_from = "서울"
>>> loc_to = "춘천"
>>> "자전거 {0}대가 {1}에서 {2}으로 가고 있다.".format(number, loc_from, loc_to)
'자전거 10대가 서울에서 춘천으로 가고 있다.'
```

문자열 format 함수

- 문자열 format 함수의 인덱스와 이름을 이용한 포맷팅

```
>>> "나는 자전거가 {number}개 있다.".format(number=2)
'나는 자전거가 2개 있다.'
>>> number = 10
>>> loc_from = "서울"
>>> loc_to = "춘천"
>>> "나는 {loc_from}에서 {loc_to}으로 가고 있다.".format(loc_from=loc_from, loc_to=loc_to)
'나는 서울에서 춘천으로 가고 있다.'
>>> number = 10
>>> "자전거 {number}가 {loc_from}에서 {loc_to}으로 가고 있다.".format(number=number, loc_from=loc_from, loc_to=loc_to)
'자전거 10가 서울에서 춘천으로 가고 있다.'
>>> "자전거 {0}가 {loc_from}에서 {loc_to}으로 가고 있다.".format(10, loc_from=loc_from, loc_to=loc_to)
'자전거 10가 서울에서 춘천으로 가고 있다.'
```

문자열 format 함수

- 문자열 format 함수의 정렬, 공백, 소수점

포맷	설명
:8	길이 8
:<	왼쪽 정렬
:>	오른쪽 정렬
:^	가운데 정렬
:~	'~'로 공백 채우기 (정렬 문자 <, >, ^ 앞에 넣은 문자로 공백 채우기)
:0.2f	소수점을 2자리까지 표현
:8.4f	길이 8, 소수점 2자리
{{	'{' 문자 표현
}}	'}' 문자 표현

```
>>> "{0:8}".format("Hello")
'Hello   '
>>> "{0:<8}".format("Hello")
'Hello   '
>>> "{0:>8}".format("Hello")
'   Hello'
>>> "{0:^8}".format("Hello")
' Hello  '
>>> "{0:~^8}".format("Hello")
'~Hello~~'
>>> "{0:-^8}".format("Hello")
'-Hello--'
>>> "{0:0.2f}".format(3.1415926535897)
'3.14'
>>> "{0:8.4f}".format(3.1415926535897)
'  3.1416'
>>> "{{ 중괄호 }}".format()
'{ 중괄호 }'
```

f 문자열 포맷팅

- f 문자열 포맷팅을 이용한 변수 값 참조

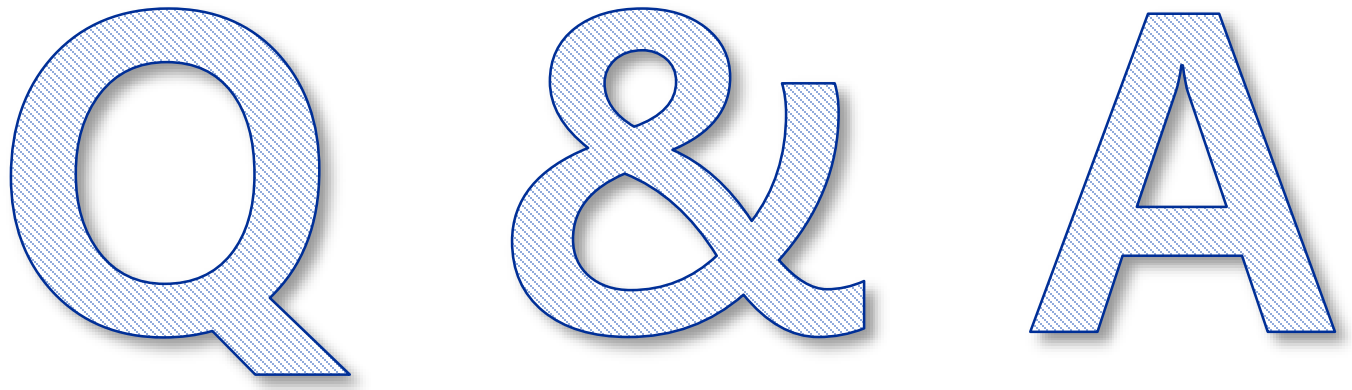
```
>>> number = 10
>>> f'나는 자전거가 {number}개 있다.'
'나는 자전거가 10개 있다.'
>>> number = 2
>>> f'나는 자전거가 {number}개 있다.'
'나는 자전거가 2개 있다.'
>>> f'나는 자전거가 {number}개 있었다. 지금은 한대를 팔아서 {number-1}개 있다.'
'나는 자전거가 2개 있었다. 지금은 한대를 팔아서 1개 있다.'
>>> loc_from = "서울"
>>> loc_to = "춘천"
>>> f'나는 {loc_from}에서 {loc_to}으로 가고 있다.'
'나는 서울에서 춘천으로 가고 있다.'
```

f 문자열 포맷팅

- f 문자열 포맷팅의 정렬, 공백, 소수점

포맷	설명
:8	길이 8
:<	왼쪽 정렬
:>	오른쪽 정렬
:^	가운데 정렬
:~	'~'로 공백 채우기 (정렬 문자 <, >, ^ 앞에 넣은 문자로 공백 채우기)
:0.2f	소수점을 2자리까지 표현
:8.4f	길이 8, 소수점 2자리
{{	'{' 문자 표현
}}	'}' 문자 표현

```
>>> f'{"Hello":8}'
'Hello   '
>>> f'{"Hello":<8}'
'Hello   '
>>> f'{"Hello":>8}'
'   Hello'
>>> f'{"Hello":^8}'
' Hello  '
>>> f'{"Hello":~^8}'
'~Hello~~'
>>> f'{"Hello":-^8}'
'-Hello--'
>>> f'{3.1415926535897:0.2f}'
'3.14'
>>> f'{3.1415926535897:8.4f}'
'  3.1416'
>>> f'{{ 중괄호 }}'
'{ 중괄호 }'
```


Q & A