

Python 05

김은진



str(string) : 문자열

- 문자열(string)
 - 문자(character)들의 모임
- 문자(character)
 - 영문자: a ~ z, A ~ Z, 그 외 문자들
 - 특수문자: !@#\$. ,
 - 공백문자(whitespace character):
tab, enter(newline, 줄바꿈문자), space(공백문자)



Python의 문자열 표현

>>> "spam eggs" 쌍따옴표 사용

>>> 'spam eggs ' 홑따옴표 사용

>>> a = 'Py'"thon" 따옴표 문자열 연속해서 작성 → 하나의 문자열

>>> a

'Python'

>>> "dosen't" 홑따옴표 포함: 쌍따옴표 사용

"dosen't"

>>> '"Yes", he said' 쌍따옴표 포함: 홑따옴표 사용

"'Yes', he said"



여러줄 문자열

```
>>> multiline="Life is too short\nYou need python\n"
```

줄바꿈 문자(\n)

```
>>> multiline
```

```
'Life is too short\nYou need python\n'
```

```
>>> print(multiline)
```

```
Life is too short
```

```
You need python
```

큰따옴표 연속 3개로 줄바꿈 문자 포함

```
>>> multiline2 = """Life is too short
```

```
You need python"""
```

```
>>> print(multiline2)
```

```
Life is too short
```

```
You need python
```



탈출 문자 표현법

\문자 : Escape character(탈출문자) → 특별한 문자

| Escape code | 설명 |
|-------------|------------|
| \n | 줄바꿈(=개행) |
| \t | 수평 탭문자(8칸) |
| \\ | 문자 "\w" |
| \' | 작은따옴표(') |
| \" | 큰따옴표("") |
| \r | 캐리지 리턴 |
| \f | 폼 피드 |
| \a | 벨 소리 |
| \b | 백 스페이스 |
| \000 | 널문자 |



Raw 문자열

- 탈출문자를 모두 그대로의 문자로 취급
- 문자열 앞에 r 붙이기

```
>>> ec=r"Life is too short\nYou need python\n">>> print(ec)
Life is too short\nYou need python\n
```



문자열 연산자

```
>>> head = "Python"
```

```
>>> tail = " is fun!"
```

```
>>> head + tail
```

```
'Python is fun!'
```

덧셈 연산자
= 문자열 합치기

```
>>> a = "python"
```

```
>>> a * 2
```

```
'pythonpython'
```

곱셈 연산자
= 문자열 반복하기



문자열 관련 errors

```
>>> ('un' * 3) 'ium'
```

```
SyntaxError: invalid syntax
```

연속문자열에서
괄호와 상수는 연속 사용 불가

```
>>> prefix = 'Py'
```

```
>>> prefix 'thon'
```

```
SyntaxError: invalid syntax
```

연속문자열에서
변수와 상수는 연속 사용 불가

```
>>> prefix + 'thon'
```

```
'Python'
```

+ 연산자는 상수, 변수 혼용 사용 가능



문자열 내의 문자 indexing

```
>>> s="Life is too short"
```

L i f e i s t o o s h o r t

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16]

문자 위치 index는 0 부터 16까지(문자열길이-1) 까지

```
>>> s[3]      변수명[ index ]
```

e str 을 indexing 한 결과값 → str type : "e"

```
>>> s[11]
```

```
>>> s[7]
```



문자열 내의 문자 indexing

L i f e i s t o o s h o r t

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16]

[-17] [-16] [-15] [-14] [-13] [-12] [-11] [-10] [-9] [-8] [-7] [-6] [-5] [-4] [-3] [-2] [-1]

Minus index 값은 -17 (문자열길이)부터 -1까지

```
>>> s[-1]
```

```
>>> s[-17]
```

```
>>> s[-4]
```



문자열 slicing

>>> s="Life is too short" 문자열 s 에서 Life만 추출하기

[시작 :종료: 간격] list와 동일
Included Excluded step

>>> s[0] + s[1] + s[2] + s[3]

>>> s[0:4]

>>> s[:4]



문자열 slicing

```
>>> s="Life is too short"
```

```
>>> s[5:]      시작부터 끝까지 추출  
'is too short'
```

```
>>> s[:5]      처음부터 4까지 추출  
'Life '
```

```
>>> s[:]       시작부터 끝까지 추출  
'Life is too short'
```

```
>>> s[12:-1]   12부터 -1(=17) 전 까지 추출  
'shor'
```



str 은 수정 불가 type

- 문자열은 그 값을 수정, 변경할 수 없음

```
>>> w="20010331Rainy"    "03"을 "05"로 바꾸세요
```

```
>>> w[4:6]='05'          str은 수정불가, 항상 새로 생성해야 함
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#115>", line 1, in <module>
```

```
    w[4:6]='05'
```

```
TypeError: 'str' object does not support item assignment
```

```
>>> w= w[:4]+'05'+w[6:]
```

```
>>> w
```

```
'20010531Rainy'
```



str 함수들

```
>>> e=" Life is too short "
```

```
>>> e.count( 's' )      's' 개수
```

```
2
```

```
>>> e.find( 'is' )      'is' 의 index
```

```
6
```

```
>>> e.index( 'too' )     'too' 의 index
```

```
9
```

```
>>> e.find( 'xxx' )      'xxx'의 index : 없으면 -1
```

```
-1
```

```
>>> comma = ','
```

```
>>> comma.join( e )      모든 문자들 사이에 comma 넣어 문자열 만들기
```

```
' ,L,i,f,e, ,i,s, ,t,o,o, ,s,h,o,r,t, '
```

```
>>> e.upper( )
```

```
' LIFE IS TOO SHORT '    대문자로 문자열 만들어 결과값 반환
```

```
>>> e
```

```
' Life is too short '
```

원 문자열 e는 그대로

```
>>> e.lower( )
```

```
' life is too short '
```

소문자로 문자열 만들어 결과값 반환



str 함수들

```
>>> e.lstrip( )           왼쪽 공백 제거 문자열 반환
'Life is too short '
>>> e.rstrip( )          오른쪽 공백 제거 문자열 반환
' Life is too short'
>>> e.strip( )           양쪽 공백 제거 문자열 반환
'Life is too short'
>>> e.replace( 'short', 'long' ) 문자열 일부 바꾸어 문자열 반환
' Life is too long '
>>> e.split( )           공백 기준으로 문자열 나눈 문자열들의 list 반환
['Life', 'is', 'too', 'short']
>>> e                    원 문자열 e는 그대로
' Life is too short '
>>> t = 'a:b:c:d'
>>> t.split( ':' )       ':' 기준으로 문자열 나눈 문자열들의 list 반환
['a', 'b', 'c', 'd']
```



str 함수 - format()

서울특별시 강서구 가양1동

(금요일) 오후 4:00

얇은 안개

8°C | °F

풍속 5m/s

풍속 숫자만 바꾸기

강수확률: 0%
습도: 64%
풍속: 5m/s

기온

강수확률

바람



미국 워싱턴 DC 워싱턴

일요일
비

풍속 6m/s

21°C | °F

강수확률: 80%
습도: 76%
풍속: 6m/s

기온

강수확률

바람



str 함수 – format()

>>> "풍속 {0}ms/s".format(5) 숫자

'풍속 5ms/s'

>>> "풍속 {0}ms/s".format("five") 문자열

'풍속 fivems/s'

>>> wind = 5

>>> "풍속 {}ms/s".format(wind) 변수

'풍속 5ms/s'

>>> "{1} {0}ms/s".format(5, "풍속") 위치 지정
: index 차례대로

'풍속 5ms/s'

>>> "{title} {wind}ms/s".format(wind=5, title= "풍속")
위치 지정: 이름으로

'풍속 5ms/s'

>>> "{title} {0}ms/s".format(5, title="풍속")

'풍속 5ms/s'

위치 지정
: index 와 이름 같이 사용



str 함수 - format()

- 공간만들기

```
>>> "{0:10}".format("hello")
```

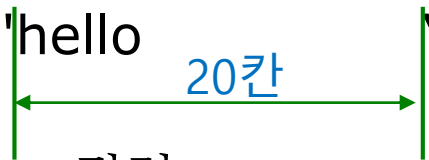
```
'hello'
```



{ index : 정렬 칸수 }
{ 이름 : 정렬 칸수 }

```
>>> "{0:20}".format("hello")
```

```
'hello'
```



- 정렬

```
>>> "{0:<10}".format("hello")
```

왼쪽 정렬

```
'hello'
```

```
>>> "{0:>10}".format("hello")
```

오른쪽 정렬

```
'    hello'
```

```
>>> "{0:^10}".format("hello")
```

가운데 정렬

```
'  hello  '
```



str 함수 – format() : 빈칸채우기

{ index : 채우기문자 정렬 칸수 }

{ 이름 : 채우기문자 정렬 칸수 }

```
>>> "{0: = <10}".format("hello")  
'hello====='
```

```
>>> "{0: *^10}".format("hello")  
'**hello***'
```

```
>>> "{0: $>10}".format("hello")  
'$$$$$hello'
```



str 함수 - format() : 숫자 서식

{ index : 칸수 콤마 }

{ 이름 : 칸수 콤마 }

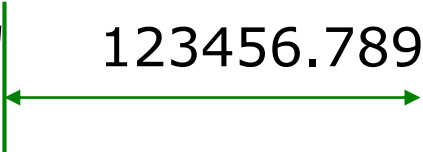
```
>>> "{0:,}".format(123456.789)
```

```
'123,456.789'
```

정수 부분에 콤마 삽입

```
>>> "{0:15}".format(123456.789)
```

```
'123456.789'
```



소수점 포함 15칸

```
>>> "{0:15,}".format(123456.789)
```

```
' 123,456.789'
```

칸수, 콤마

```
>>>
```



str 함수 - format() : 숫자 서식

```
>>> "{0:,}".format(123456.789)
```

```
'123,456.789'   정수 부분에 콤마 삽입
```

```
>>> "{0:15}".format(123456.789)
```

```
'123456.789'
|-----| 소수점 포함 15칸
```

```
>>> "{0:15,}".format(123456.789)
```

```
' 123,456.789'   칸수, 콤마
```

```
>>> "{0:15.4}".format(123456.789)
```

```
' 1.235e+05'   실수는 기본이 지수형 표현, .4는 유효숫자길이
```

```
>>> "{0:15.4f}".format(123456.789)
```

```
' 123456.7890'   f: 소수점 실수 표현
|-----|
|-----| 소수점 아래 4자리
```

| 코드 | 설명 |
|----|---------------------|
| d | 정수 (Integer) |
| f | 실수 (floating-point) |
| e | 지수 (Exponent) |



str 함수 - format() : 숫자 서식 예제

```
>>> "{0:15,.4f}".format(123456.789)
```

```
' 123,456.7890'   칸수 + 콤마 + 소수점 아래 자릿수+ 실수
```

```
>>> "{0:=>15,.4f}".format(123456.789)
```

```
'===123,456.7890'   빈칸='채우기 + 오른쪽 정렬 + .....
```

```
>>> "{{hello}}".format()
```

```
'{hello}'          문자 { 나 } 대입하기
```



str(string) : 문자열

- 문자열 : 변경 불가
- 문자열 표현법
- 탈출문자(Escape character)
- indexing (list, tuple과 동일)
- slicing (list, tuple과 동일)
- str 함수
- format() 함수 이용한 문자열 formatting

