03

String

Python built-in objects

Object type	Example literals/creation
Numbers	1234, 3.1415, 0b111, 1_234, 3+4j, Decimal, Fraction
Strings	'code', "app's", b'a\x01c', 'h\u00c4ck', 'hÄck'
Lists	[1, [2, 'three'], 4.5], list(range(10))
Tuples	(1, 'app', 4, 'U'), tuple('hack'), namedtuple
Dictionaries	{'job': 'dev', 'years': 40}, dict(hours= 10)
Sets	set('abc'), {'a', 'b', 'c'}
Files	open('docs.txt'), open(r'C:\data.bin', 'wb')
Other core objects	Booleans, types, None
Program-unit objects	Functions, modules, classes
Implementation objects	Compiled code, stack tracebacks

Mark Lutz. (2025). Learning Python, 6th Edition. n.p.: O'Reilly Media, Inc.

String

- 문자열(String)이란?
- 문자, 단어 등으로 구성된 문자들의 집합

  ‘Hello, World!’ “a” “123”

- 시작과 끝은 큰/작은따옴표로 표기 또는 따옴표 3개 연속으로 사용

  ‘Hello’ “Hello” “””Python is fun””” ’’’Python is fun’’’ ‘ ’

String

문자열의 size

```
>>> import sys
>>> sys.getsizeof("")
41
>>> sys.getsizeof("a")
42
>>> sys.getsizeof("Hello")
46
>>> sys.getsizeof("Hello, world!")
54
```

String

문자열 안에 따옴표를 포함시키고 싶을 때

“Hello, world!”

“Hello, ‘world!’ ”

‘Hello, “world!” ’

큰따옴표와 작은따옴표 혼합하여 사용

```
>>> print('Hello, 'world!' ')
    print('Hello, 'world!' ')
          ^^^^^^^^^^^^^^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

Escape Code

이스케이프 코드(Escape code)를 활용해 표현 가능

```
>>> print('Hello, \'world!\'')
Hello, 'world!'
```

코드	설명
\\	문자 \를 그대로 표현할 때 사용
\'	작은따옴표(‘)를 그대로 표현할 때 사용
\"	큰따옴표(“)를 그대로 표현할 때 사용
\n	문자열 안에서 줄을 바꿀 때 사용
\t	문자열 사이에 탭 간격을 줄 때 사용
\b	백 스페이스
\f	폼 피드(줄바꿈 문자, 현재 커서를 다음 줄로 이동)
\000	널 문자

Escape Code

문자열을 여러 줄 출력할 때 사용

```
>>> lines = "Python\nis\nfun"
>>> lines
'Python\nis\nfun'
>>> print(lines)
Python
is
fun
```

Multiple lines

연속된 따옴표로도 표현 가능

```
>>> lines = """Python
... is
... fun"""
>>> lines
'Python\nis\nfun'
>>> print(lines)
Python
is
fun
```

Operators

- len(): 문자열의 길이(문자 개수) 반환
- +(concatenation): 두 문자열을 결합
- *(repetition): 문자열을 반복하여 새로운 문자열 생성

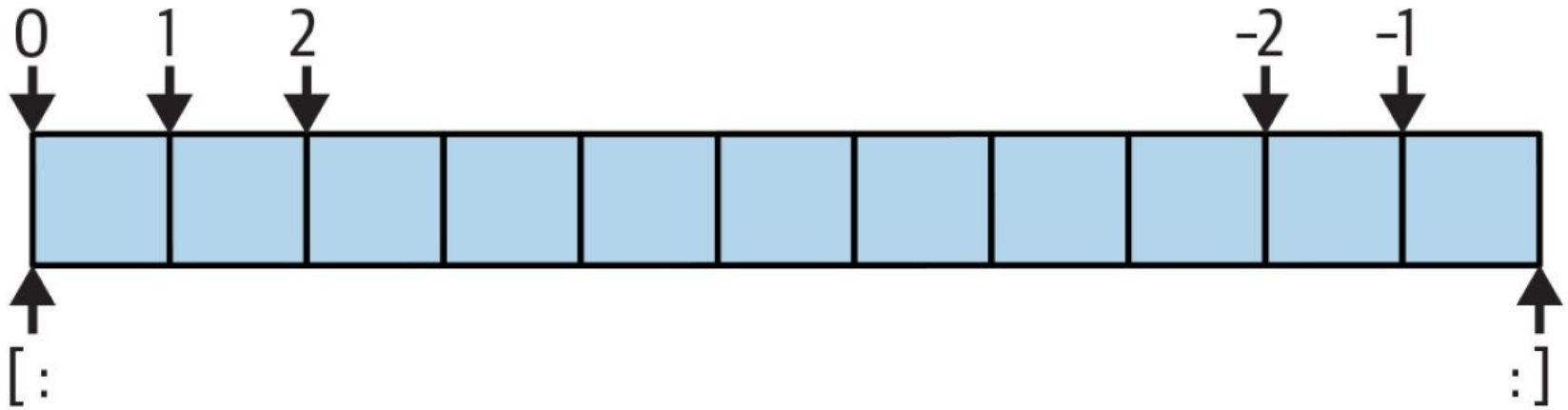
```
>>> len('abc')
3
>>> 'abc' + 'def'
'abcdef'
>>> 'Py!' * 4              # 'Py!' + 'Py!' + 'Py!' + 'Py!'
'Py!Py!Py!Py!'
```

Operators

```
>>> S
'Code'
>>> S + 'xyz'          # Concatenation
'Codexyz'
>>> S                  # S is unchanged
'Code'
>>> S * 8              # Repetition
'CodeCodeCodeCodeCodeCodeCodeCode'
```

Indexing

```
>>> S = 'code' >>> S[0], S[-2]        # Indexing from front or end ('c', 'd')
>>> S[1:3], S[1:], S[:-1]             # Slicing: extract a section ('od', 'ode', 'cod')
```



Indexing

```
>>> S
'Code'
>>> S[-1]                # The last item in S
'e'
>>> S[len(S)-1]          # Negative indexing, the hard way
'e'
```

Slicing

```
>>> S = 'code'
>>> S[99]
IndexError: string index out of range
>>> S[1:99]
'ode'
```

Slicing

- 슬라이싱은 [시작:끝:간격] 형태로 사용 → 간격(step)을 주면 일정 간격으로 문자 선택
- S[::2]처럼 생략도 가능하며, 전체 문자열에서 짝수 인덱스 문자만 추출 가능

```
>>> S = 'abcdefghijklmnop'
>>> S[1:10:2]                    # Skipping items
'bdfhj'
>>> S[::2]
'acegikmo'
```

Slicing

```
>>> S = 'hello'
>>> S[::−1]                    # Reversing items
'olleh'
>>> S = 'abcedfg'
>>> S[5:1:−1]                  # Bounds roles differ
er
'fdec'
```

Type Conversion

```
>>> type('123')
<class 'str'>
>>> int('123')
123
>>> type(int('123'))
<class 'int'>
>>> float('123')
123.0
>>> type(float('123'))
<class 'float'>
```

Type Conversion

```
>>> int('a')
Traceback (most recent call last):
  int('a')
  ~~~^^^^^
ValueError: invalid literal for int() with base 10: 'a'
>>> float('a')
Traceback (most recent call last):
  float('a')
  ~~~~~^^^^^
ValueError: could not convert string to float: 'a'
```

Type Conversion

```
>>> '62' + 1
Traceback (most recent call last):
   '62' + 1
   ~~~~~^~~
TypeError: can only concatenate str (not "int") to str

>>> int('62'), str(62)        # Convert from/to string (62, '62')
>>> int('62') + 1
63
```

print()

```
>>> help(print)        # 파이썬의 내장 함수
Help on built-in function print in module builtins:

print(*args, sep=' ', end='\n', file=None, flush=False)
    Prints the values to a stream, or to sys.stdout by default.

    sep
      string inserted between values, default a space.
    end
      string appended after the last value, default a newline.
    file
      a file-like object (stream); defaults to the current sys.stdout.
    flush
      whether to forcibly flush the stream.
```

*args

```
print("Python","is","fun")
print("Hello, world!")
```

```
Python is fun
Hello, world!
```

■ : sep = ' '(space)

■ : end = '\n'(newline)

```
print("Python","is","fun",sep=',',end=' ')
print("Hello, world!")
```

```
Python,is,fun Hello, world!
```

■ : sep = ','(comma)

■ : end = ' '(space)

3주차 과제

- 03_String.ipynb