04

String & Boolean

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.

Comparison Operators

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Boolean

- 참, 거짓을 나타내는 자료형
- 모든 객체는 Boolean 값이 존재

```
>>> 1 > 2, 1 < 2
(False, True)
>>> bool('hack')
True                          # Nonempty means True

>>> bool()
False                        # Empty means False
>>> bool('')
False
```

Boolean

- 비교 연산자를 활용하여 Boolean 값 확인 가능

```
>>> 1 == 1
True
>>> 1 == '1'
False
>>> 'a' != 'b'
True
```

String comparisons

- 왼쪽부터 한 글자씩 차례로 비교
- 각 문자는 유니코드 값(ord) 기준으로 비교
- 다른 문자가 처음 나오는 위치에서 크고 작음이 결정
- 그 외에는 더 긴 문자열의 크기가 큰 것으로 결정

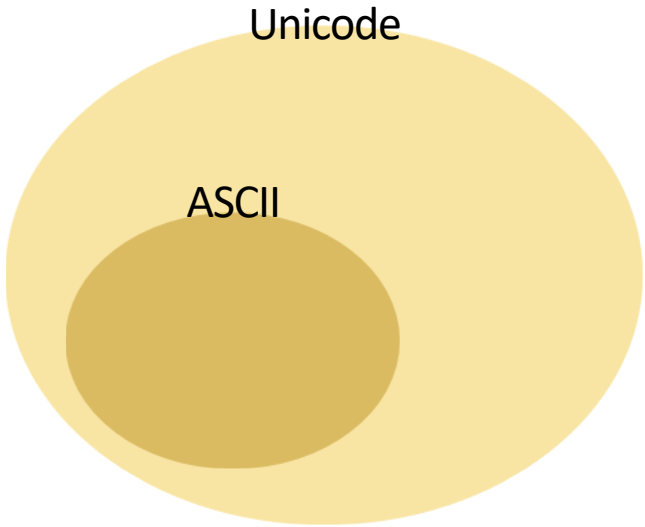
```
>>> 'hack' == 'hack', 'hact' > 'hack', 'hacker' > 'hack'
(True, True, True)
```

Unicode(ord)

ASCII TABLE

American Standard Code for Information Interchange

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]




```
>>> ord('A')
65
>>> ord('가')
44032
```

Immutable

- 문자열은 변경 불가능(immutable) -> 인덱스를 이용한 수정 불가
- 슬라이싱과 +를 이용해 새로운 문자열 생성

```
>>> S = 'text'
>>> S[0] = 'n'
TypeError: 'str' object does not support item assignment

>>> S = ‘n’ + S[1:]
>>> S
‘next’
```

Immutable

```
>>> S = 'text'
>>> S = S + 'ual!'          # To change a string, make a new one
>>> S
'textual!'
>>> S = S[:4] + ' processing' + S[-1]
>>> S
'text processing!'
```

replace()

- replace(old, new, n)는 문자열에서 특정 부분을 다른 문자열로 교체
- n을 지정하면 앞에서부터 최대 n번까지만 교체

```
>>> S = 'text'
>>> S = S.replace('ex', 'hough')
>>> S
'thought'

>>> '--@--@--@--'.replace('@', 'PY', 2)
'--PY--PY--@--'
```

find()

- find()는 문자열 안에서 특정 부분 문자열이 처음 나타나는 위치(인덱스)를 반환
- 찾는 문자열이 없으면 -1을 반환

```
>>> S = 'hack'
>>> S.find('ac')    # Call the find method to look for 'ac' in string S
1
>>> S.find('k')
3
>>> S.find(‘b’)
-1
```

(not) in

- in 연산자는 문자열 안에 특정 문자가 포함되어 있는지 확인할 때 사용 → 결과는 bool
- not in은 반대로 포함되어 있지 않은 경우에 True 반환

```
>>> 'a' in 'apple'
True
>>> 'z' not in 'banana'
True
```

join(), count()

- `'구분자'.join(문자열)`는 문자열의 각 요소 사이에 구분자가 삽입됨

```
>>> a = ','
>>> a.join('abcd')
'a,b,c,d'
```

- count(sub)는 문자열 안에 부분 문자열이 몇 번 등장하는지를 반환
- 대소문자를 구분하며, 겹치지 않는 횟수만 계산됨

```
>>> 'banana'.count('a')
3
>>> 'bAnAna'.count('a')
1
```

split,() strip()

- split()은 문자열을 지정한 구분자를 기준으로 나누어 리스트로 반환
- 구분자를 생략하면 공백 기준으로 나뉘며, 연속된 공백은 하나로 처리됨

```
>>> 'a:b:c'.split(':')
['a', 'b', 'c']
>>> 'Python is fun'.split()
['Python', 'is', 'fun']
```

- strip()은 문자열 양쪽 끝의 공백이나 특정 문자를 제거
- lstrip()은 왼쪽만, rstrip()은 오른쪽만 제거

```
>>> '   hello   '.strip()
'hello'
>>> '--hello--'.strip('-')
'hello'
```

capitalize(), lower(), upper()

- capitalize()는 문자열의 첫 글자만 대문자로, 나머지는 소문자로 변환

```
>>> 'hello world'.capitalize()
'Hello world'
>>> 'PYTHON'.capitalize()
'Python'
```

- lower()는 문자열의 모든 문자를 소문자로 변환
- upper()는 문자열의 모든 문자를 대문자로 변환

```
>>> 'HELLO'.lower()
'hello'
>>> 'Python3'.lower()
'python3'
```

String Methods

- 그 외에도 다양한 문자열과 관련된 함수들을 확인 가능

  https://docs.python.org/3/library/stdtypes.html#string-methods

S.capitalize() S.ljust(width [, fill]) S.casefold() S.lower() S.center(width [, fill]) S.lstrip([chars]) S.count(sub [, start [, end]]) S.maketrans(x [, y [, z]]) S.encode([encoding [, errors]]) S.partition(sep) S.endswith(suffix [, start [, end]]) S.removeprefix(prefix) S.expandtabs([tabsize]) S.removesuffix(suffix) S.find(sub [, start [, end]]) S.replace(old, new [, count]) S.format(fmtstr, *args, **kwargs) S.rfind(sub [, start [, end]]) S.format_map(mapping) S.rindex(sub [, start [, end]]) S.index(sub [, start [, end]]) S.rjust(width [, fill]) S.isalnum() S.rpartition(sep) S.isalpha() S.rsplit([sep [, maxsplit]]) S.isascii() S.rstrip([chars]) S.isdecimal() S.split([sep [, maxsplit]]) S.isdigit() S.splitlines([keepends]) S.isidentifier() S.startswith(prefix [, start [, end]]) S.islower() S.strip([chars]) S.isnumeric() S.swapcase() S.isprintable() S.title() S.isspace() S.translate(map) S.istitle() S.upper() S.isupper() S.zfill(width) S.join(iterable) ...

String Formatting

- 문자열 내 특정한 값을 바꿔야 할 때 사용

  “현재 온도는 18도입니다.”
  “현재 온도는 20도입니다.”

- 바로 대입하는 경우

```
>>> "I eat %d apples." %3
'I eat 3 apples.'

>>> "I eat %s apples." %"five"
'I eat five apples.'
```

String Formatting

- 변수를 사용하는 경우, 변수의 자료형 명시

```
>>> number = 3
>>> "I eat %d apples." %number
'I eat 3 apples.'
```

- 2개 이상의 값 넣기

```
>>> number = 10
>>> day = "three"
>>> "I ate %d apples. so I was sick for %s days." % (number, day)
'I ate 10 apples. so I was sick for three days.'
```

String Formatting

- 문자열 포맷 코드

코드	설명
%s	문자열(string)
%c	문자 1개(character)
%d	정수(Integer)
%f	부동 소수(floating-point)
%o	8진수
%x	16진수
%%	Literal % (문자 ‘ %’ 자체)

String Formatting

- 문자열 앞에 f, 변수 앞뒤로 대괄호{}를 이용한 방법

```
>>> number = 3
>>> f"I eat {number} apples"
'I eat 3 apples'
```

```
>>> number = 10
>>> day = "three"
>>> f"I ate {number} apples. So I was sick for {day} days."
'I ate 10 apples. So I was sick for three days.'
```

4주차 과제

- 04_String_Boolean.ipynb