

06

Data Types

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

Dictionary

- 키(key)와 값(value)의 쌍으로 이루어진 자료형
- 중괄호 {} 사용

```
person = {'name': 'Tom', 'age': 20}
```

딕셔너리의 특징

- 순서 존재 (Python 3.7+)
- 키는 중복 불가, 값은 중복 가능
- 키는 변경 불가능한 값만 가능 (str, int, tuple 등)

Dictionary

- 접근: dict['key']
- 수정: dict['key'] = new_value
- 존재하지 않는 키는 오류 발생

```
>>> P = {'name': 'Alice', 'age': 20, 'job': 'student'}
>>> P['name']
'Alice'
>>> P['age'] = 24
>>> P['age']
24
>>> P['height']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'height'
```

Dictionary

```
>>> P = {'name': {'first': 'Alice', 'last': 'Kim'}, 'age': 20, 'job': ['student', 'mech']}
>>> P['name']
{'first': 'Alice', 'last': 'Kim'}
>>> P['name']['last']
'Kim'
>>> P['job']
['student', 'mech']
>>> P['job'][-1]
'mech'
>>> P['job'].append('robotics')
>>> P
{'name': {'first': 'Alice', 'last': 'Kim'}, 'age': 20, 'job': ['student', 'mech', 'robotics']}
```

Dictionary

- `dict.keys()`: 딕셔너리 `dict`의 모든 키 반환
- `dict.values()`: `dict`의 모든 값 반환
- `dict.items()`: `dict`의 키-값 쌍(튜플) 반환

```
>>> P = {'name': 'Alice', 'age': 24, 'job': 'student'}
>>> P.keys()
dict_keys(['name', 'age', 'job'])
>>> P.values()
dict_values(['Alice', 24, 'student'])
>>> P.items()
dict_items([('name', 'Alice'), ('age', 24), ('job', 'student')])
```

Dictionary

- 'key' in dict → 키 존재 여부 확인
- 'value' in dict.values()로 값 존재 여부 확인 가능

```
>>> 'age' in P
```

```
True
```

```
>>> 'height' not in P
```

```
True
```

```
>>> 'Alice' in P.values()
```

```
True
```

get()

- dict.get('key', default=None)
- 키가 없을 경우 None 또는 기본값 반환

```
>>> P = {'name': 'Alice', 'age': 24, 'job': 'student'}  
>>> P.get('name')  
'Alice'  
>>> print(P.get('height'))  
None  
>>> P.get('height', 160)  
160
```


Eliminating elements from Dictionary

- `del dict['key']`: 지정한 키와 해당 값을 딕셔너리에서 삭제
- `dict.pop('key')`: 키에 해당하는 값을 반환하고 삭제
- `dict.clear()`: 딕셔너리의 모든 항목을 제거

```
>>> P= {'name': {'first': 'Alice', 'last': 'Kim'}, 'age': 20, 'job': ['student', 'mech', 'robotics']}
>>> del P['job']
>>> P
{'name': {'first': 'Alice', 'last': 'Kim'}, 'age': 20}
>>> P.pop('age')          # P= {'name': {'first': 'Alice', 'last': 'Kim'}}
20
>>> P.clear()             # P= {}
```

Set

- 집합(set) 자료형
- 중괄호 {} 또는 set()으로 생성

```
s = {"apple", "banana", "cherry"}
```

집합의 특징

- 중복 자동 제거
- 순서 없음 (unordered) → 인덱싱 불가능
- 변경 가능 (mutable), 하지만 집합 안에 리스트나 딕셔너리는 못 넣음

Set

- 중괄호 {}로 생성하거나 set() 함수 사용
- 빈 집합은 set()으로만 생성 가능 ({}: 딕셔너리)

```
>>> s1 = set([1, 2, 3])
>>> s1
{1, 2, 3}
>>> s2 = set("Hello")
>>> s2
{'H', 'l', 'e', 'o'}
```

Set

- `set.add(value)`: 집합에 value 추가
- `value in set`: 해당 value가 set에 포함되어 있는지 확인

```
>>> s = {"apple", "banana", "cherry"}
>>> len(s)
3
>>> "banana" in s
True
>>> "banana" not in s
False
>>> s.add("orange")
>>> s
{'orange', 'banana', 'apple', 'cherry'}
```

Eliminating elements from Set

- `set.remove(value)`: 지정한 값을 집합에서 제거 (없으면 오류)
- `set.pop()`: 집합에서 임의의 값을 꺼내고 제거 (unordered)
- `set.clear()`: 집합의 모든 요소를 제거



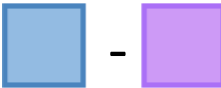
```
>>> s = {"apple", "banana", "cherry", "orange"}
>>> s.remove("banana")      # s= {'orange', 'apple', 'cherry'}
>>> s.pop()                 # s= {'apple', 'cherry'}
'orange'
>>> s.clear()
>>> s
set()
```

Indexing

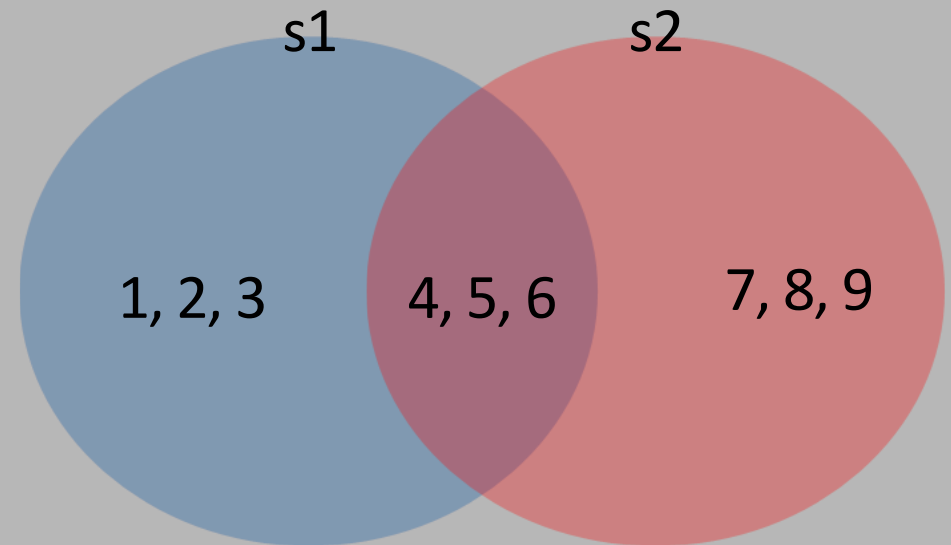
- 인덱스로 접근 불가
- 리스트나 튜플로 변환하여 접근 가능

```
>>> s1 = set([1, 2, 3])
>>> s1[0]
TypeError: 'set' object is not subscriptable
>>> l1 = list(s1)
>>> l1[0]
1
>>> t1 = tuple(s1)
>>> t1[0]
1
```

Set

- & (intersection): 교집합 
- | (union): 합집합 
- - (difference): 차집합 

```
>>> s1 = set([1, 2, 3, 4, 5, 6])
>>> s2 = set([4, 5, 6, 7, 8, 9])
>>> s1 & s2
{4, 5, 6}
>>> s1 | s2
{1, 2, 3, 4, 5, 6, 7, 8, 9}
>>> s1 - s2
{1, 2, 3}
```



Summary

리스트 (List)

- 순서가 있으며, 변경 가능
- 중복된 값 허용

튜플 (Tuple)

- 순서가 있으며, 변경 불가능
- 중복된 값 허용

집합 (Set)

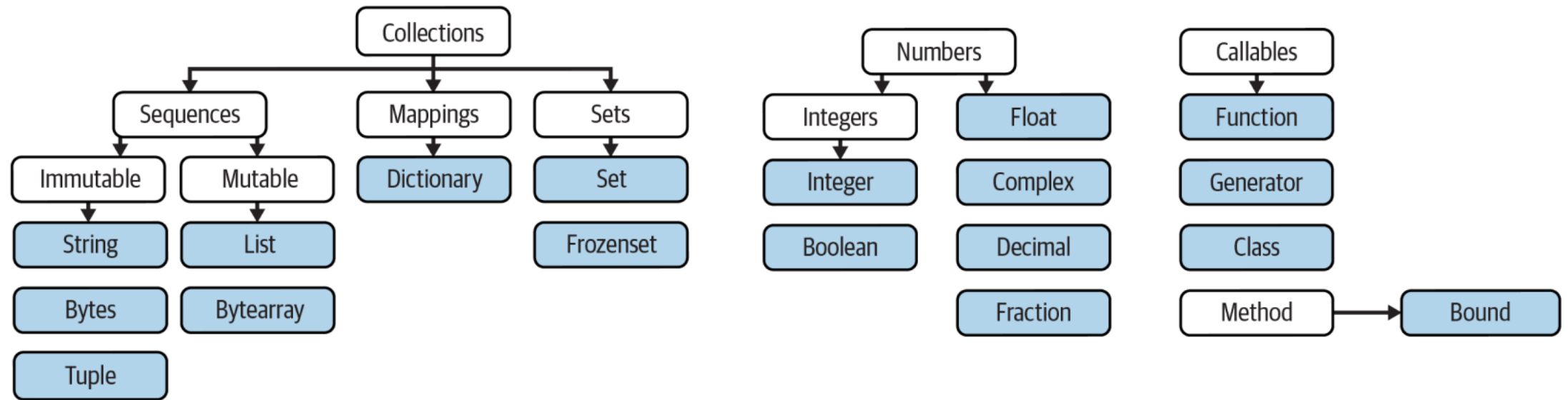
- 순서가 없고, 변경 및 인덱싱 불가
- 중복된 값 허용하지 않음

딕셔너리 (Dictionary)

- 순서가 있으며, 변경 가능한 키-값 쌍
- 중복된 값 허용하지 않음

Summary

Python's major built-in object type



6주차 과제

- 06_Data_types.ipynb