Python
CH02: 파이썬 시작하기 II

전종준 교수

서울시립대학교 통계학과/통계데이터사이언스대학원

Contents

Dictionary

딕셔너리

딕셔너리를 배울때 생각해야할 점

- 딕셔너리를 어떻게 만들어야 하나?
- 딕셔너리에 어떻게 항목을 추가하나?
- 딕셔너리에 항목(원소)를 어떻게 확인하나?
- 딕셔너리 항목을 삭제할 수 있나?

딕셔너리란?

- 키(key)와 값(value)의 쌍으로 데이터를 저장하는 자료형
- 리스트와 달리 순서(index)를 사용하지 않고, 키를 사용하여 값에 접근
- 중괄호 {}를 사용하여 정의하며, 각 쌍은 콜론 :으로 구분

예시:

```
1 student = {"이름": "지민", "학년": 2, "점수": 87}
```

연습문제: 본인의 이름, 나이, 좋아하는 색깔을 담은 딕셔너리를 만들어라.

딕셔너리 값 조회

- 딕셔너리[키] 형태로 해당 키에 대응하는 값 조회
- 존재하지 않는 키 조회 시 KeyError 발생
- get(키) 메서드로 안전한 조회 가능 (없으면 None 반환)

예시:

```
print(student["이름"])      # "지민"
print(student.get("주소"))  # None
```

연습문제: 위에서 만든 딕셔너리에서 좋아하는 색깔을 조회해 보아라.

항목 추가 및 수정

- 새로운 키에 값을 할당하면 항목 추가
- 기존 키에 값을 재할당하면 값이 수정됨

예시:

```
student["주소"] = "서울"        # 항목 추가
student["점수"] = 92            # 값 수정
```

연습문제: 딕셔너리에 키 "취미"를 추가하고 값을 입력한 뒤, "이름"을 다른 이름으로 수정해 보아라.

항목 삭제

- `del` 딕셔너리[키]를 사용하여 특정 항목 삭제
- `clear()` 메서드를 사용하면 모든 항목 삭제

예시:

```
1 del student["주소"]
2 student.clear()
```

연습문제: 본인의 딕셔너리에서 한 항목을 삭제한 뒤 전체 딕셔너리를 출력해 보세요.

응용 예시: 물건별 가격 저장

- 상품 이름을 키로, 가격을 값으로 저장
- 키를 이용하여 가격 정보 조회 가능

예시 코드:

```
1 prices = {"사과": 1000, "바나나": 800, "수박": 12000}
2 print(prices["사과"])
```

연습문제: "딸기"를 2000원으로 추가하고, 모든 과일의 가격을 출력해 보세요.

메서드: keys(), values(), items()

- `keys()` : 모든 키를 반환
- `values()` : 모든 값을 반환
- `items()` : (키, 값) 쌍을 튜플로 반환

예시:

```
info = {"이름": "지민", "나이": 20, "학교": "동네고등학교"}

print(info.keys())
print(info.values())
print(info.items())
```

연습문제: 위 딕셔너리에서 키 목록과 값 목록을 각각 리스트로 변환해 보세요.

메서드: get(), pop()

- `get(key)` : 키에 해당하는 값을 반환, 없으면 `None` 반환
- `pop(key)` : 키에 해당하는 값을 꺼낸 뒤 해당 항목 삭제

예시:

```
1 info = {"이름": "지민", "나이": 20}
2 print(info.get("학교"))      # None
3 age = info.pop("나이")        # 20
4 print(info)                  # {"이름": "지민"}
```

연습문제: `get()`으로 존재하지 않는 키를 조회하고, `pop()`으로 키 하나를 삭제해 보세요.

메서드: update()

- `update()` : 다른 딕셔너리 또는 키-값 쌍으로 기존 딕셔너리를 갱신
- 기존 키가 있으면 값을 수정, 없으면 추가

예시:

```
1 info = {"이름": "지민", "나이": 20}
2 info.update({"나이": 21, "학교": "동네고"})
3 print(info)
```

연습문제: 본인의 딕셔너리에 취미나 이메일 정보를 `update()`로 추가해 보세요.

연습문제 1: 학생 정보 관리

- 이름, 학년, 전화번호를 포함하는 딕셔너리 작성
- 학년과 전화번호를 출력
- `get()`을 사용해 '주소' 항목 조회

조건:

- 이름: "민수", 학년: 3, 전화번호: "010-1234-5678"

예시 시작 코드:

```
student = {
    "이름": "민수",
    "학년": 3,
    "전화번호": "010-1234-5678"
}
# 여기에 코드를 추가하세요
```

연습문제 2: 점수표 수정

- 점수표 딕셔너리에 학생 한 명 추가
- 기존 학생 한 명의 점수를 수정
- 전체 점수의 합계를 계산하여 출력

기본 점수표:

```
scores = {"지민": 85, "정국": 90, "뷔": 88}
# 여기에 코드를 추가하세요
```

연습문제 3: 메서드 활용 연습

- `update()`로 값 수정
- `pop()`으로 항목 삭제
- 최종 딕셔너리를 출력

초기 딕셔너리:

```
info = {"이름": "지민", "MBTI": "INFP"}
# 여기에 코드를 추가하세요
```

Tuple & Set

튜플이란?

- 여러 개의 값을 하나의 묶음으로 저장하는 자료형
- 리스트와 유사하지만, **수정이나 삭제가 불가능**함 (불변 자료형)
- 소괄호 ()를 사용하여 생성

예시:

```
1 person = ("지민", 20, "서울")
```

연습문제: 본인의 이름, 나이, 도시를 하나의 튜플로 만들어 보세요.

튜플의 특징

- 항목 수정 및 삭제 불가능
- 인덱스를 사용하여 접근 가능
- 리스트보다 메모리 사용량이 적고 처리 속도가 빠름

예시:

```
print(person[0])    # "지민"
print(person[1:])   # (20, "서울")
```

연습문제: 앞에서 만든 튜플에서 나이와 도시만 슬라이싱으로 출력해 보세요.

튜플과 리스트의 차이점

- 리스트는 대괄호 [] 사용, 튜플은 소괄호 () 사용
- 리스트는 항목 변경 가능, 튜플은 변경 불가능
- 튜플은 읽기 전용 데이터에 적합

예시 비교:

```
lst = ["a", "b"]
tup = ("a", "b")

lst[0] = "z"      # 가능
tup[0] = "z"      # 오류 발생
```

연습문제: 튜플은 변경이 안 된다는 점을 확인하기 위해 항목을 바꾸는 코드를 작성해 보고 오류 메시지를 확인해 보세요.

튜플 언패킹

- 튜플의 각 요소를 변수에 나누어 저장 가능
- 변수 개수와 튜플 항목 수가 일치해야 함

예시:

```
name, age, city = person
print(name)
print(age)
print(city)
```

연습문제: 본인의 정보를 담은 튜플을 만들고, 언패킹하여 각 값을 출력해 보세요.

튜플 관련 함수

- `len(t)` : 튜플의 길이 반환
- `count(x)` : 항목 x가 등장한 횟수 반환
- `index(x)` : 항목 x가 처음 나타나는 인덱스 반환

예시:

```
1 numbers = (1, 2, 3, 2, 1)
2 print(len(numbers))     # 5
3 print(numbers.count(2)) # 2
4 print(numbers.index(3)) # 2
```

연습문제: 튜플에 포함된 특정 숫자가 몇 번 등장하는지 count()로 확인해 보세요.

Set

집합(set)이란?

- 중복되지 않는 값을 저장하는 자료형
- 수학의 집합과 비슷한 개념
- 순서가 없기 때문에 인덱스로 접근 불가능
- 중괄호 {} 또는 set() 함수로 생성

예시:

```
1 fruits = {"사과", "바나나", "포도"}
```

연습문제: "딸기", "사과", "딸기"를 포함하는 집합을 만들어 보세요. 출력 결과를 확인해보세요.

set의 주요 특징

- 같은 값을 여러 번 넣어도 1개만 저장됨
- 내부 요소의 순서는 유지되지 않음
- 리스트나 문자열을 set()으로 변환 가능

예시: 중복 제거

```
numbers = [1, 2, 2, 3, 3, 3]
unique = set(numbers)
print(unique)  # {1, 2, 3}
```

연습문제: 다음 리스트에서 중복을 제거하세요: `["토마토", "토마토", "양파", "상추"]`

집합에 값 추가 및 제거

- `add()` 메서드로 항목 추가
- `remove()` 또는 `discard()` 메서드로 항목 제거

예시:

```
vegetables = {"상추", "깻잎"}
vegetables.add("당근")
vegetables.remove("상추")
```

연습문제: 빈 집합을 만들고, ”양파”, ”고추”를 추가한 후 ”양파”를 삭제해 보세요.

set 연산 예시 (소개용)

- 교집합 (`&`), 합집합 (`|`), 차집합 (`-`) 연산 가능
- 실제 사용은 나중에 조건문과 함께 학습
- 지금은 `set()`이 중복을 없앤다는 개념 중심으로 이해

예시:

```
a = {"사과", "바나나"}
b = {"바나나", "수박"}
print(a & b)  # {"바나나"}
```

연습문제: 두 집합 `a = {"A", "B"}`, `b = {"B", "C"}`의 합집합을 구해보세요.

연습문제 1: 중복 제거하기

- 다음 장바구니 목록에는 중복된 항목이 포함되어 있음
- `set()`을 사용하여 중복을 제거한 후 출력할 것

시작 코드:

```
cart = ["우유", "계란", "계란", "빵", "우유"]
# 여기에 코드를 작성하세요
```

연습문제 2: 값 추가와 삭제

- 빈 집합을 만들고 과일 3개를 추가
- 그 중 1개를 삭제한 뒤 최종 결과를 출력

힌트: `set()`, `add()`, `remove()` 사용

예시 시작 코드:

```
1 fruits = set()
2 # 여기에 값을 추가하고, 하나를 삭제하세요
```

초급 넘어서기: 데이터 객체의 casting

자료형 간 형변환이란?

- 파이썬의 자료형은 `list()`, `tuple()`, `set()`, `dict()` 함수로 서로 변환 가능
- 형변환을 통해 자료를 다른 구조로 재구성할 수 있음
- 구조에 따라 변환이 가능한 경우와 불가능한 경우 존재

연습문제: `[1, 2, 3]`을 튜플로 변환한 결과를 출력해 보자.

리스트 ↔ 튜플 변환

- `tuple(리스트)` : 리스트를 튜플로 변환
- `list(튜플)` : 튜플을 리스트로 변환
- 항목의 순서와 개수는 그대로 유지됨

예시:

```
a = [1, 2, 3]
b = tuple(a)    # (1, 2, 3)

c = (4, 5, 6)
d = list(c)     # [4, 5, 6]
```

연습문제: `(7, 8, 9)`를 리스트로 변환한 뒤 출력해 보세요.

리스트/튜플 ↔ 셋(set) 변환

- `set()` 사용 시 중복 제거됨
- `list(set(...))` 또는 `tuple(set(...))` 형태로 중복 없는 시퀀스를 만들 수 있음

예시:

```
1 nums = [1, 2, 2, 3]
2 unique = set(nums)          # {1, 2, 3}
3 new_list = list(unique)     # [1, 2, 3]
```

연습문제: `["a", "b", "a", "c"]`에서 중복을 제거한 리스트를 만들어 보세요.

딕셔너리로 변환 가능한 경우

- `dict()`는 **(키, 값)** 쌍의 구조에서만 변환 가능
- 리스트나 튜플 안에 2개의 항목을 가진 튜플/리스트가 있어야 함

예시:

```
1 items = [("이름", "지민"), ("나이", 20)]
2 info = dict(items)
3 # 결과: {"이름": "지민", "나이": 20}
```

연습문제: `[["A", 90], ["B", 80]]`를 딕셔너리로 변환해 보세요.

주의사항: 변환 불가능한 경우

- `dict([1, 2, 3])` → 오류 발생 (쌍이 아님)
- `set({"a": 1, "b": 2})` → 키만 추출됨 (`{"a", "b"}`)
- 자료 구조의 내부 형태를 먼저 확인하고 변환해야 함

예시 (오류):

```
dict([1, 2, 3])       # TypeError
set({"x": 1, "y": 2})  # {"x", "y"} (키만)
```

연습문제: 딕셔너리 `{"x": 1, "y": 2}`를 리스트로 변환하면 어떤 결과가 나올지 예측해 보세요.

초급 넘어서기: 딕셔너리의 활용

딕셔너리로 테이블 구조 표현하기

- 여러 사람의 정보를 **표 형태(행렬)**로 표현할 수 있음
- **열 이름**을 키로, **각 열에 해당하는 값들**을 리스트로 저장
- 즉, 하나의 딕셔너리로 **표처럼 구조화된 데이터**를 관리할 수 있음

예시: 학생 3명의 이름, 나이, 점수를 담은 구조

```
data = {
    "이름": ["지민", "정국", "뷔"],
    "나이": [20, 21, 19],
    "점수": [88, 92, 85]
}
```

연습문제: 위 데이터에 "학교"라는 열을 추가하고, 3개의 값을 리스트로 넣어 보세요.

중첩 딕셔너리란?

- 딕셔너리 안에 또 다른 딕셔너리를 값으로 넣은 구조
- 한 사람의 여러 정보를 하나의 키에 묶어 표현할 수 있음
- 테이블 구조보다 더 유연하고 직관적인 표현 가능

예시: 학생들의 상세 정보

```
1 students = {
2     "지민": {"나이": 20, "학교": "동네고", "점수": 88},
3     "정국": {"나이": 21, "학교": "중앙고", "점수": 92}
4 }
```

연습문제: 새로운 학생 "뷔"를 추가하고, 나이 19, 학교 "서초고", 점수 85를 넣어 보세요.

중첩 딕셔너리 값 접근하기

- 바깥 딕셔너리의 키로 먼저 접근
- 안쪽 딕셔너리의 키를 다시 사용하여 원하는 값에 접근
- 딕셔너리["키1"]["키2"] 형태로 사용

예시:

```
print(students["지민"]["학교"])  # "동네고"
print(students["정국"]["점수"])  # 92
```

연습문제: 학생 "뷔"의 학교와 점수를 출력해 보세요.

중첩 딕셔너리 수정 및 추가

- 내부 딕셔너리에 접근하여 값을 수정하거나 새 항목 추가 가능
- 새로운 키-값 쌍을 내부에 직접 할당

예시:

```
1 students["지민"]["점수"] = 91
2 students["정국"]["주소"] = "서울"
```

연습문제: "뷔"의 점수를 90으로 수정하고, 주소를 "부산"으로 추가해 보세요.