

Python 자료구조의 확장: 튜플, 딕셔너리, 집합

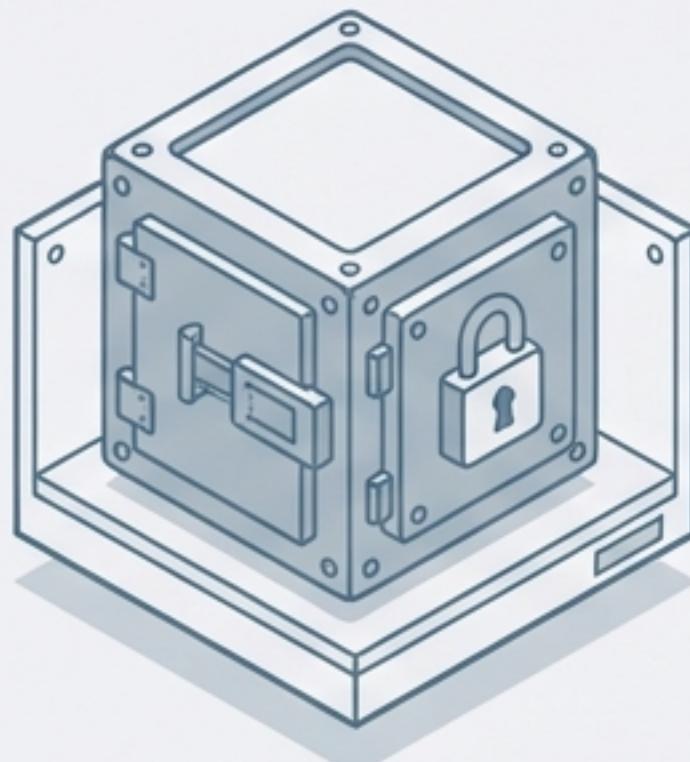
AI 활용 프로그래밍 Week 7

데이터를 더 안전하게 보관하고, 더 빠르게 검색하며,
증복을 제거하는 기술. 단순한 리스트(List)를 넘어
상황에 맞는 최적의 도구를 선택하는 방법을
학습합니다.



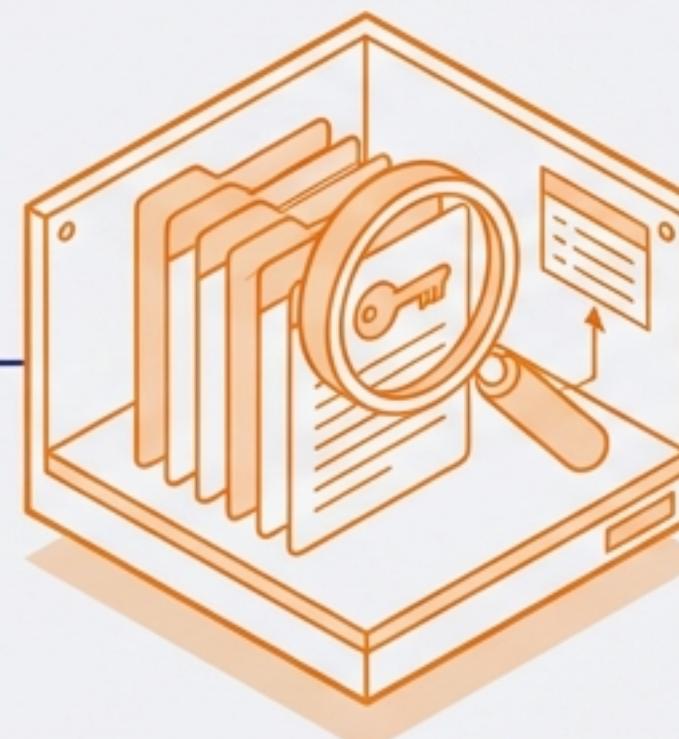
오늘의 학습 여정 (Agendas)

1. Tuple (튜플)



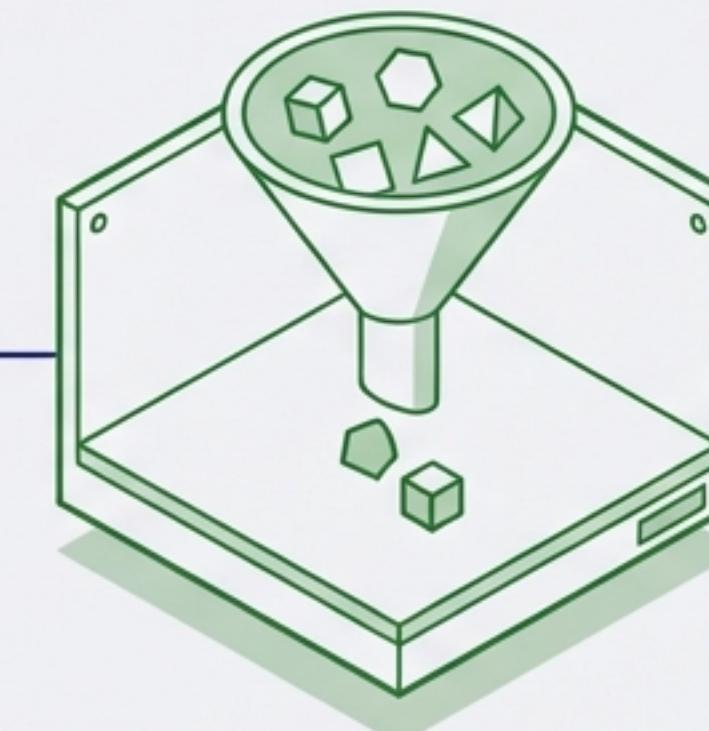
수정 불가능한 안전한 저장소

2. Dictionary (딕셔너리)



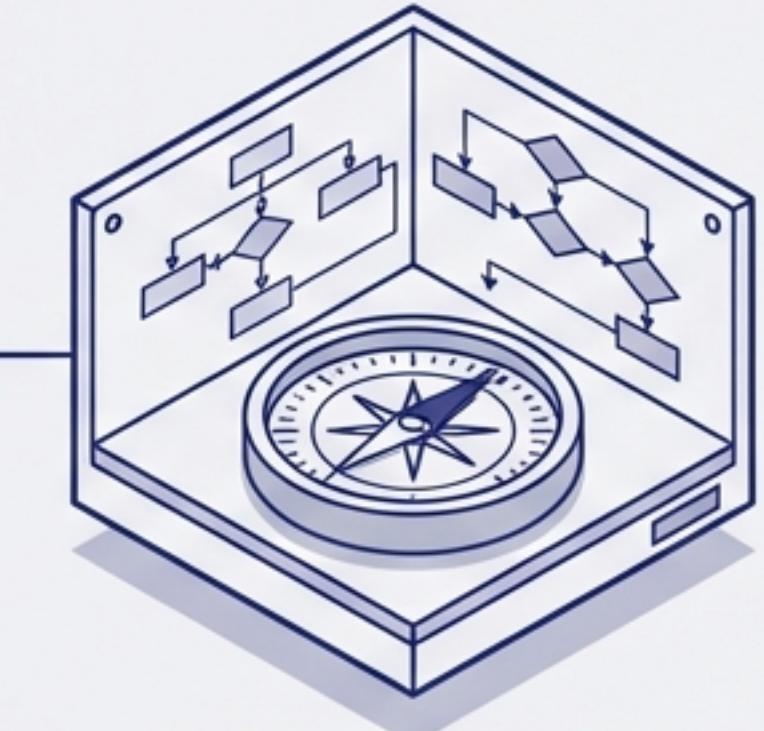
키(Key)를 이용한 검색 엔진

3. Set (집합)



중복을 허용하지 않는 필터

4. Selection Guide



최적의 자료구조 선택 전략

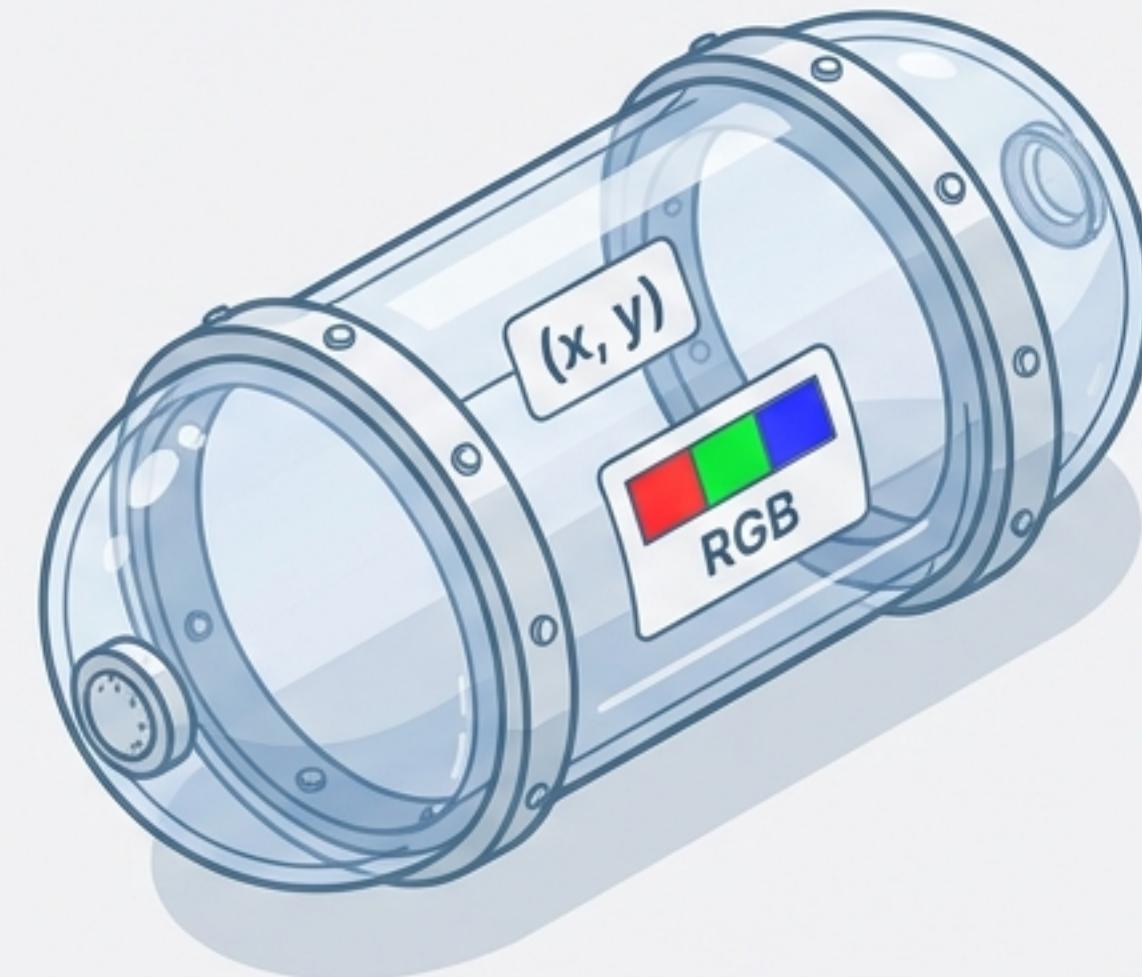
- 1) 튜플(tuple) 기초
- 2) 딕셔너리(dict) 기초
- 3) 집합(set) 기초
- 4) 자료구조 선택 가이드
- 5) With AI 실습 + 퀴즈/과제

튜플 (Tuple): 변하면 안 되는 데이터의 보관

List (리스트)

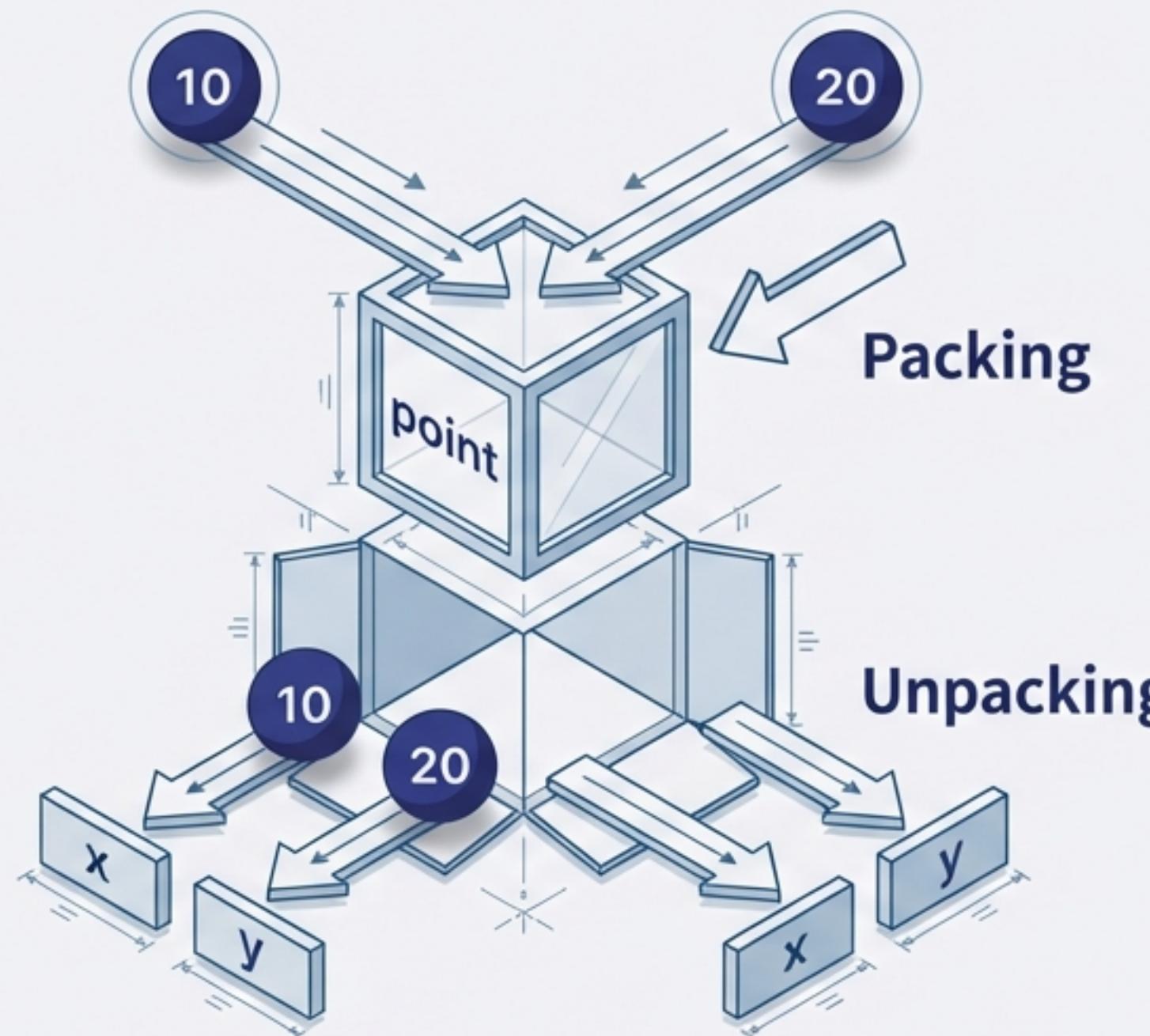


Tuple (튜플)



- 리스트와 유사하게 여러 값을 순서대로 저장하지만, 수정할 수 없습니다 (Immutable).
- **의미:** 프로그램 실행 중 실수로 값이 변경되는 것을 방지합니다.
- **활용 예시:** 좌표 (x, y) , RGB 색상 값, (이름, 학번) 등 고정된 데이터 묶음.

튜플의 핵심 문법: Packing & Unpacking

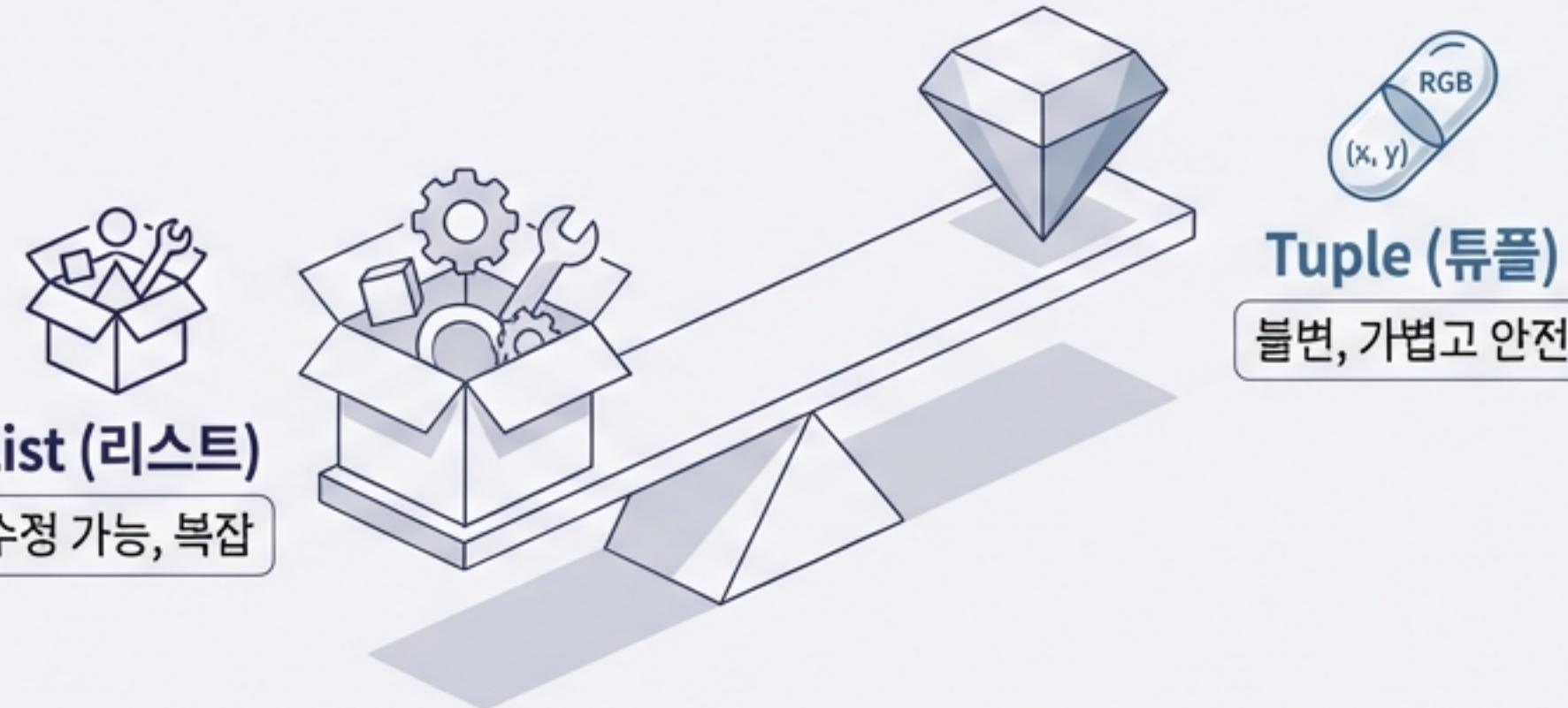


```
# Packing & Unpacking
point = (10, 20)
x, y = point
print(x, y) # 10 20

# Swap (값 교환)
a, b = 1, 2
a, b = b, a # Pythonic Swap!
```

괄호 () 없이도 패킹이 가능하며, 함수에서 여러 값을 한 번에 반환할 때 필수적으로 사용됩니다.

선택의 기준: 리스트 vs 튜플



리스트 (List)	수정, 추가, 삭제가 빈번하게 필요할 때 사용.
튜플 (Tuple)	상수처럼 고정된 값의 묶음으로 사용할 때. (더 가볍고 안전함)

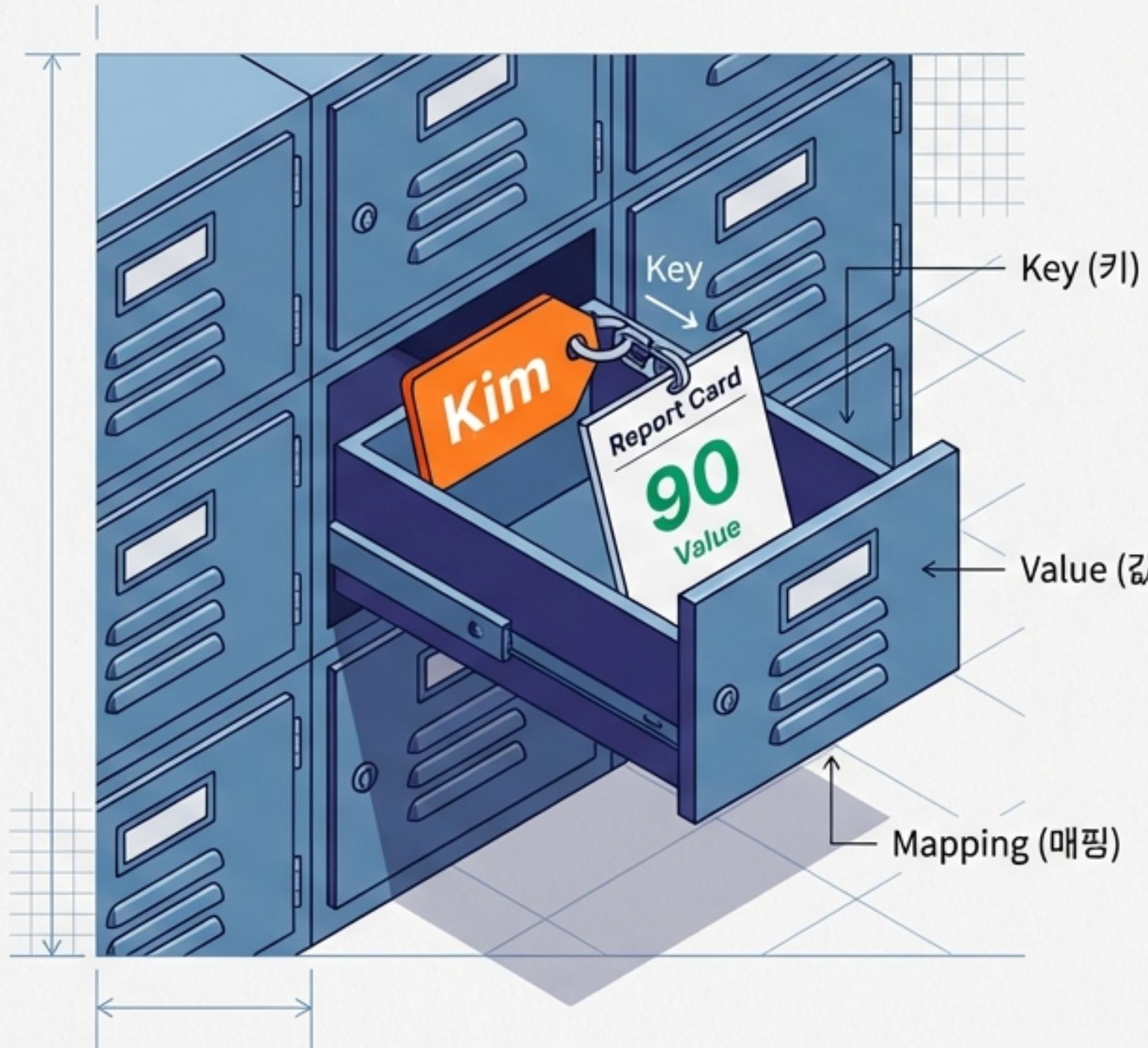
전략적 질문: 데이터가 프로그램 실행 중에 변경되어야 하는가?



Yes → List No → Tuple



딕셔너리 (Dictionary): 검색을 위한 맵핑(Mapping)



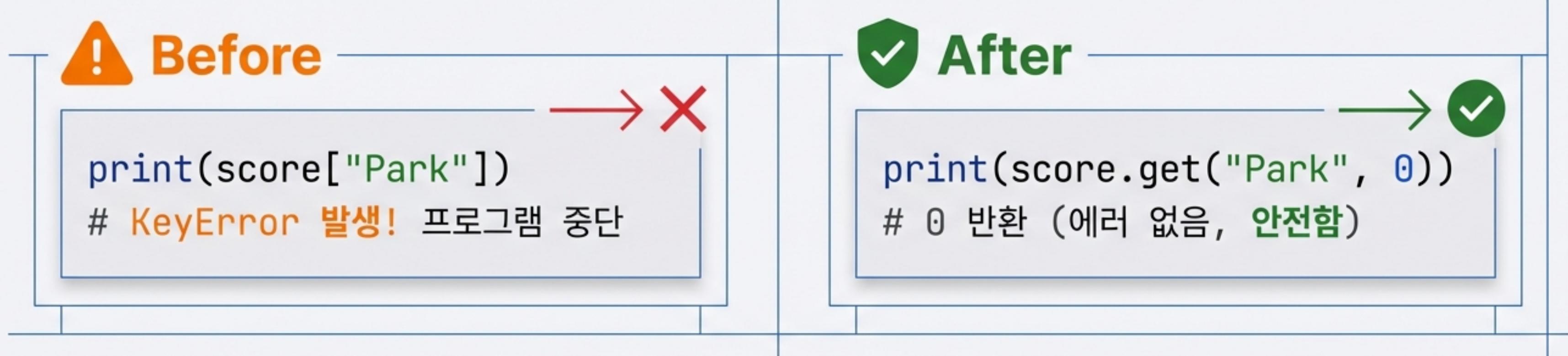
구조: Key-Value (키-값) 쌍으로 이루어진 자료구조.

특징: 순서보다 **검색 속도**가 중요할 때 사용합니다. 키(Key)를 알면 값(Value)을 즉시 찾을 수 있습니다.

현실 예시: 학번 → 이름, 상품코드 → 가격, 단어 → 빈도수.

```
score = {"Kim": 90, "Lee": 80}
```

안전한 접근을 위한 .get() 메서드

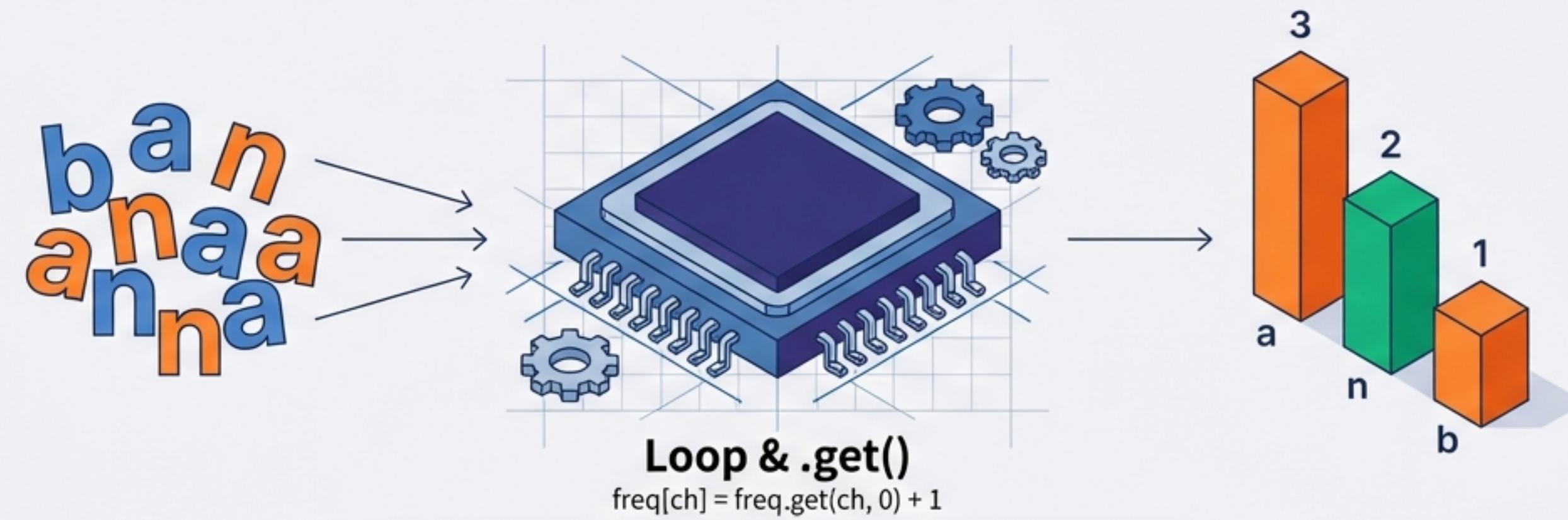


문제: 존재하지 않는 키(Key)에 접근하면 치명적인 에러(KeyError)가 발생합니다.

해결: .get() 메서드를 사용하면 키가 없을 때 반환할 기본값(Default)을 지정할 수 있습니다.

Tip: 방어적 프로그래밍을 위해 직접 접근보다 get() 활용을 권장합니다.

실전 패턴: 데이터 빈도 분석



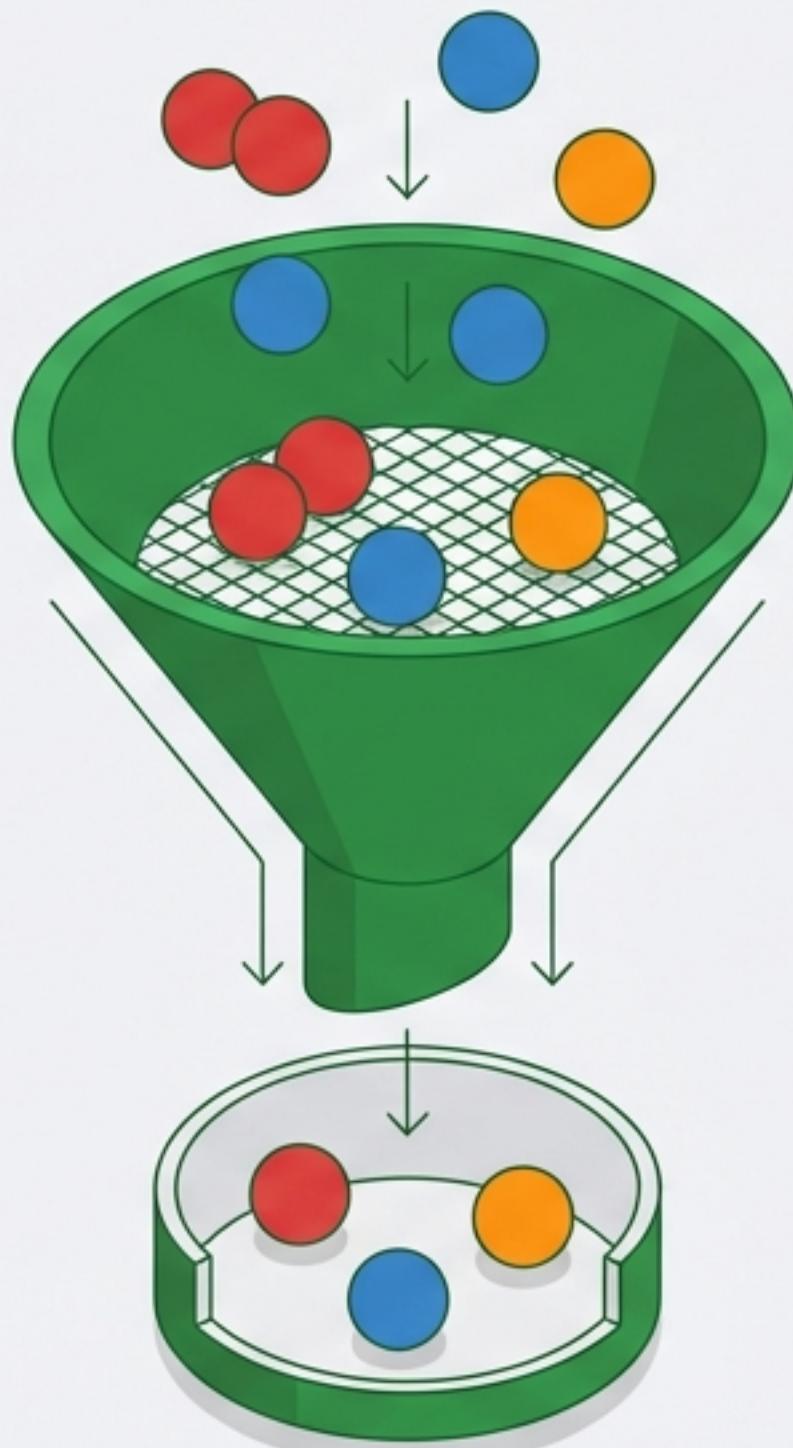
```
text = "banana"
freq = {}

for ch in text:
    # 키가 없으면 0을 가져오고 1을 더함
    freq[ch] = freq.get(ch, 0) + 1

print(freq) # {'b': 1, 'a': 3, 'n': 2}
```

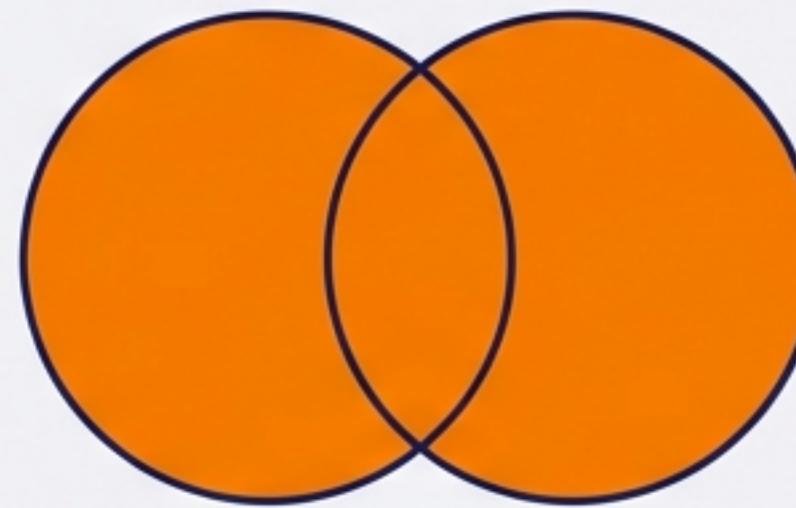
텍스트 분석이나 로그 분석에서 가장 빈번하게 사용되는 패턴입니다.

집합 (Set): 중복을 허용하지 않는 필터

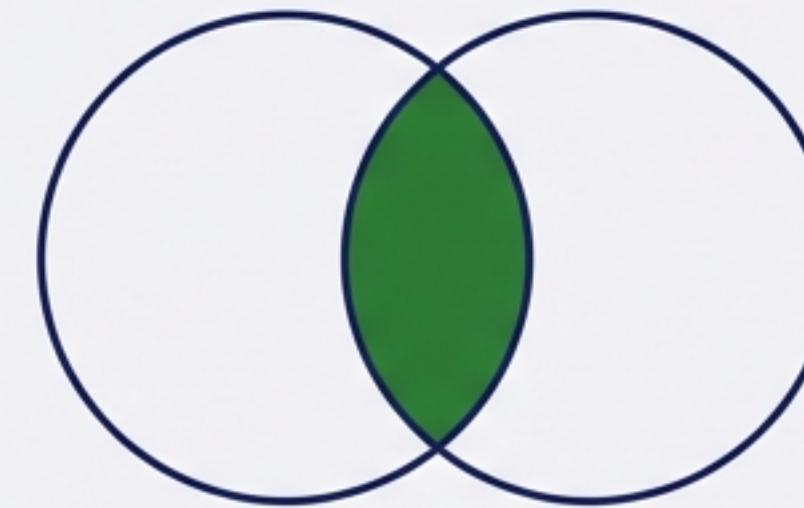


- 특징 1: 중복된 데이터가 들어오면 자동으로 제거됩니다.
- 특징 2: 저장된 순서가 없습니다 (Unordered). 인덱싱 불가.
- 활용 : 리스트의 중복 제거 (`list(set(data))`) 혹은 고유값(Unique Value) 확인.

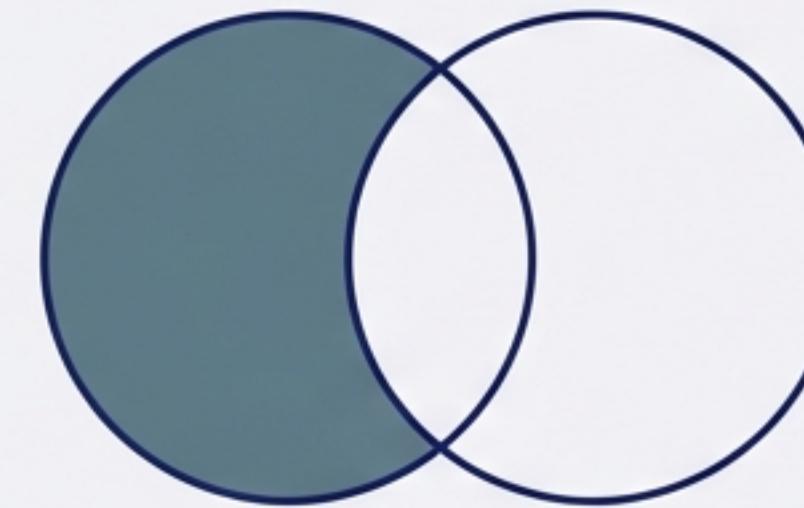
집합 연산과 데이터 비교



합집합 (|)



교집합 (&)



차집합 (-)

복잡한 조건문(for/if) 없이 수학적 기호만으로 데이터를 직관적으로 비교하고 필터링합니다.

```
print(A | B) # Union  
print(A & B) # Intersection  
print(A - B) # Difference
```

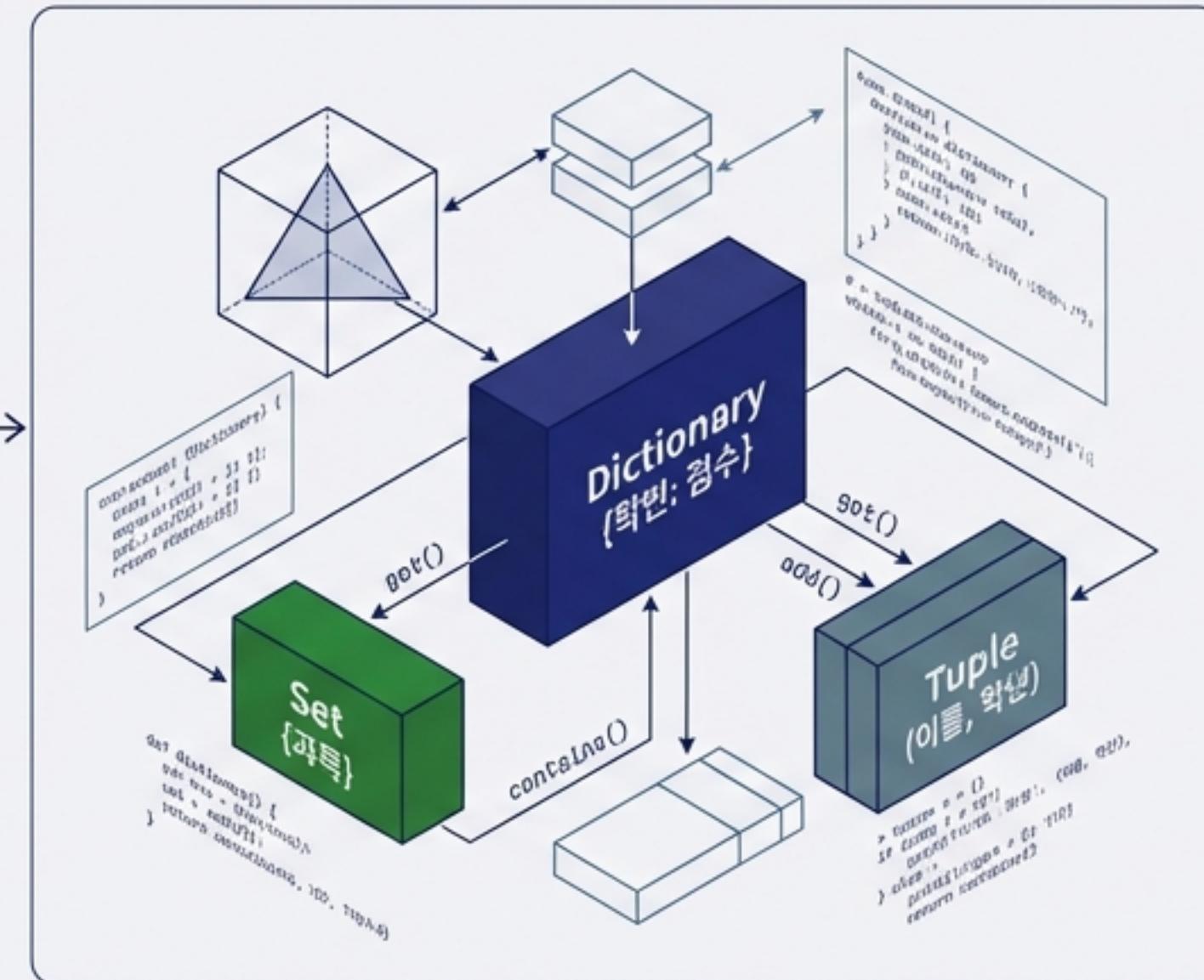
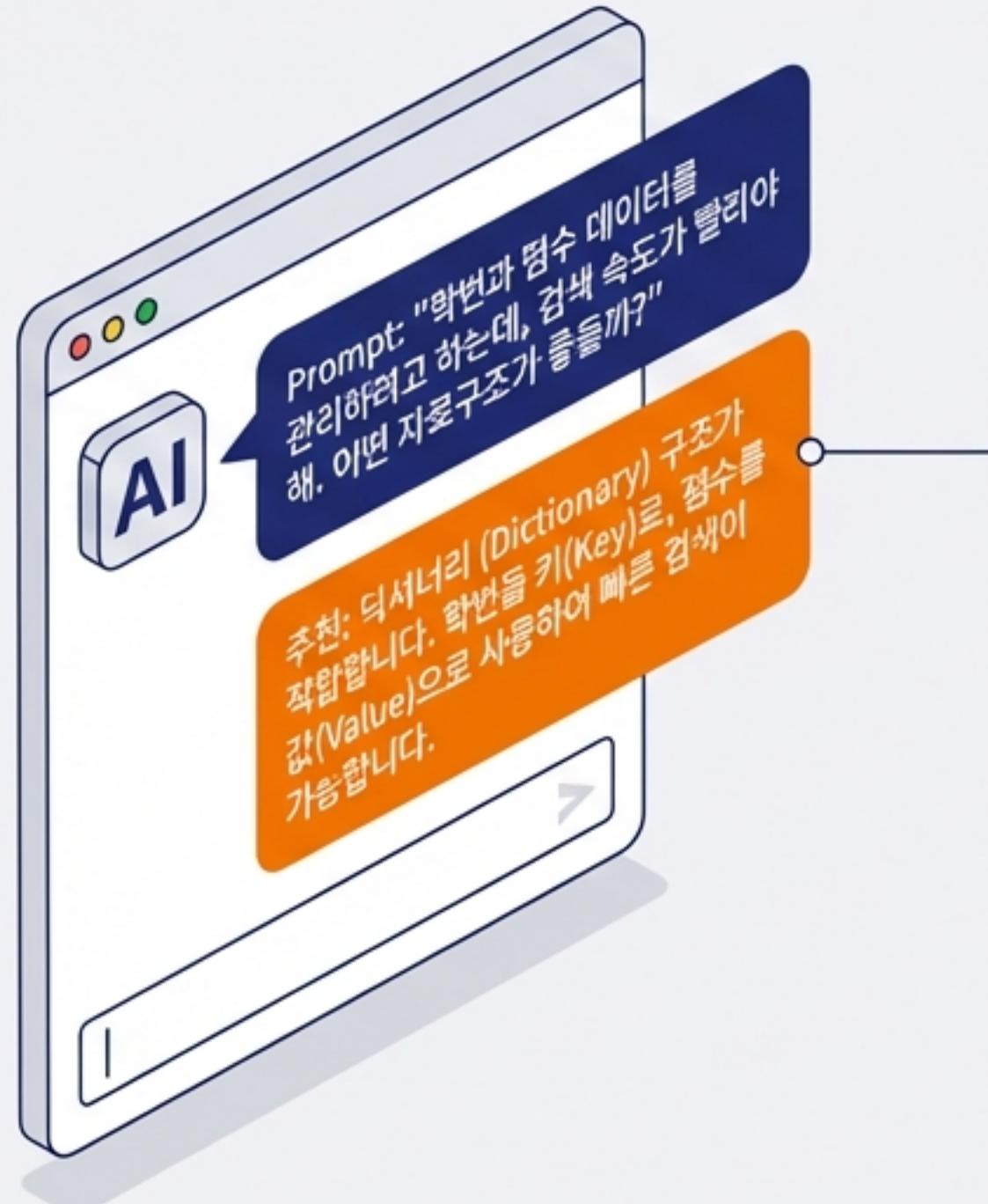
한눈에 보는 자료구조 비교 (Cheat Sheet)

자료구조 (Structure)	순서 (Order)	중복 (Duplicates)	수정 (Mutable)	핵심 용도 (Use Case)
List []	O	O	O	범용 데이터 저장
Tuple ()	O	O	X	안전한 상수 묶음
Set {}	X	X	O	중복 제거, 집합 연산
Dict {k:v}	X (3.7+ O)	Key X	O	키 기반 빠른 검색

자료구조 선택 가이드 (Selection Guide)



With AI: LLM을 활용한 자료구조 학습



AI에게 문제 상황을 설명하면 최적의 자료구조를 추천받을 수 있습니다.

활용:
복잡한 딕셔너리 구조나
집합 연산 코드를 AI와 함께
작성하고 최적화해 보세요.

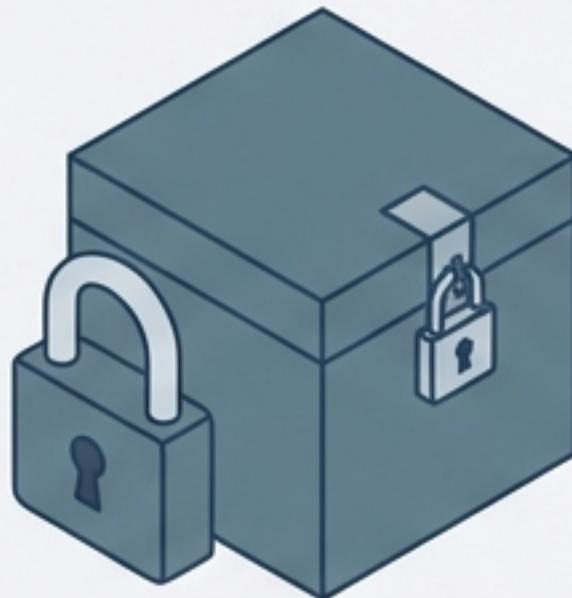


학습 목표: LLM을 활용해 자료구조 기반 프로그램을 설계·구현·개선할 수 있다.

핵심 요약 (Key Takeaways)

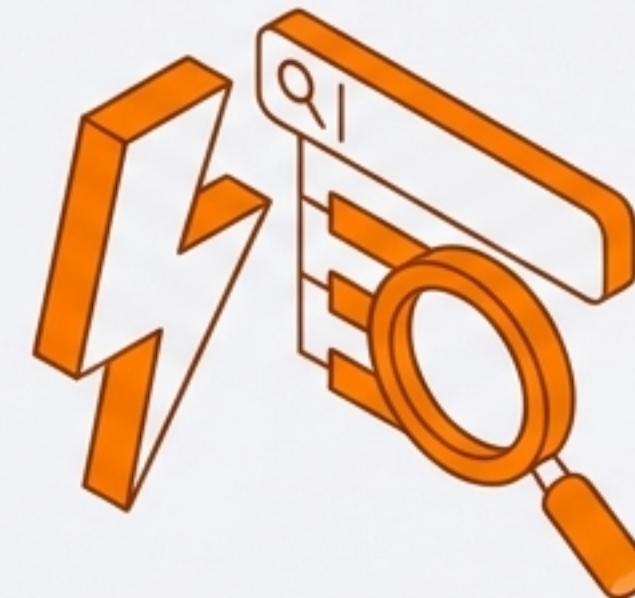
안전성 (Safety)

변하면 안 될 때는
튜플로 잠그세요.



효율성 (Efficiency)

빠른 검색이 필요할 때는
딕셔너리로 맵핑하세요.



고유성 (Uniqueness)

중복을 없애고 싶을 때는
집합으로 걸러내세요.



적절한 자료구조의 선택이 프로그램의 성능과 코드의 품질을 결정합니다.

Q & A

Week 7 Complete

다음 시간에는 '데이터 전처리' 과정을 통해
오늘 배운 자료구조들을 실무 데이터에 적용해 봅니다.

