<[파이썬 수학] 제12장. 똑똑한 모임, 집합처럼 생각하기!>

선생님: (서로 다른 모양의 구슬과 똑같은 모양의 구슬 여러 개를 보여주며) 얘들아, 안녕! 수학 시간에 '집합'에 대해 들어본 적 있니? 집합은 아주 특별한 규칙을 가진 모임이야. 바로 **'중복된 것은 하나로만 취급한다'**는 규칙이지! 예를 들어, "사과, 바나나, 사과"가 있다면, 이 모임의 과일 종류는 '사과'와 '바나나' 두 가지인 것처럼 말이야.

코딩새싹 🌱: 아! 그럼 똑같은 건 여러 개 있어도 하나로만 세는 거군요!

선생님: 맞아! 오늘은 파이썬의 리스트와 `for` 반복문을 이용해서 이 집합의 멋진 연산들을 직접 구현해볼 거야. 마치 우리가 직접 집합 계산기를 만드는 것과 같지!

<미션 1: 두 그룹 합치기! (합집합)>

선생님: 자, 여기 두 개의 친구 목록 리스트 a와 b가 있어. 두 그룹의 친구들을 모두 초대해서 하나의 파티 초대 명단을 만들고 싶은데, 이름이 중복되면 한 번만 적어야 해. 어떻게 하면 될까?

```
a = [1, 2, 3, 4]
b = [3, 4, 5, 6]

# 방법 1: 일단 다 합치고, 나중에 중복 검사하기!
c = a + b # 일단 두 리스트를 합치면 [1, 2, 3, 4, 3, 4, 5, 6] 이 되겠지?
d = [] # 최종 파티 명단을 적을 깨끗한 종이를 준비!

for i in c: # 합쳐진 명단을 한 명씩 보면서
  if i not in d: # 만약 그 이름이 아직 깨끗한 종이(d)에 없다면,
    d.append(i) # 종이에 이름을 적어줘!

print("최종 파티 명단:", d)
```

궁금이 💡: 오! `if i not in d:` 이 부분이 핵심이네요! `d` 리스트에 이미 이름이 있는지 확인해서, 없을 때만 추가하니까 중복이 완벽하게 제거됐어요!

선생님: 정확해! 똑같은 원리로 과일 목록도 합쳐볼 수 있겠지?

```
a = ['apple', 'melon', 'orange', 'cow']
```

```
b = ['chicken', 'pig', 'cow', 'orange']


# 똑같은 방법으로 합집합 구하기!
c = a + b
d = []
for i in c:
  if i not in d:
    d.append(i)
print("과일과 동물 합집합:", d)
```

<미션 2: 공통으로 아는 친구 찾기! (교집합)>

선생님: 이번엔 두 그룹에 **모두 포함된** 친구, 즉 공통으로 아는 친구만 찾아내려면 어떻게 해야 할까?

코딩새싹 🌱: 으음… 한쪽 그룹을 기준으로, 다른 쪽 그룹에도 이름이 있는지 물어보면 될 것 같아요!

선생님: 바로 그거야! `for`문으로 `a` 리스트를 한 명씩 보면서, 그 친구가 `b` 리스트 안에도 `in`는지 확인하면 되지!

```
a = ['apple', 'melon', 'orange', 'cow']
b = ['chicken', 'pig', 'cow', 'orange']
c = [] # 공통 친구를 적을 새 종이


for i in a: # a 목록을 한 명씩 보면서,
  if i in b: # 만약 그 친구가 b 목록에도 있다면,
    c.append(i) # 공통 친구 명단에 추가!


print("\n공통으로 포함된 것(교집합):", c)
```

궁금이 💡: `if i in b:` 이 한 줄로 공통된 친구를 찾아낼 수 있다니, `in` 연산자는 정말 유용하네요!

<미션 3: 우리 그룹에만 있는 친구 찾기! (차집합)>

선생님: 마지막 미션! `a` 그룹에는 속해 있지만, `b` 그룹에는 속하지 않은 친구들만 찾아내려면 어떻게 할까?

코딩새싹 🌱: 앗! 방금 전 교집합이랑 거의 똑같은데요? `in` 대신에 `not in`을 쓰면 되겠어요!

선생님: 정답! 우리 학생들이 이제 원리를 완벽하게 깨우쳤구나!

```
a = ['apple', 'melon', 'orange', 'cow']
b = ['chicken', 'pig', 'cow', 'orange']
c = [] # a 그룹에만 있는 친구를 적을 새 종이

for i in a: # a 목록을 한 명씩 보면서,
  if i not in b: # 만약 그 친구가 b 목록에는 없다면,
    c.append(i) # a 그룹에만 있는 친구 명단에 추가!

print("\na에만 있는 것(차집합):", c)
```

<선생님의 프로그래머 꿀팁! 🍯>

선생님: (미소 지으며) 여러분은 오늘 리스트와 `for`, `if`문이라는 기본 도구만으로 합집합, 교집합, 차집합이라는 집합의 핵심 기능들을 모두 직접 만들어냈어! 정말 대단한 일이야! 이건 마치 도끼와 톱만으로 멋진 의자를 만들어낸 것과 같지.

궁금이 💡: 와, 정말 뿌듯해요! 그런데 선생님, 혹시 파이썬에 이런 집합 연산을 더 쉽게 해주는 '전용 전동 공구' 같은 건 없나요?

선생님: (눈을 반짝이며) 역시 궁금이! 바로 그 질문을 기다리고 있었단다! 당연히 있지! 파이썬에는 '집합'을 위해 태어난 자료형, 바로 **`set`**이 있단다! `set`을 사용하면 오늘 우리가 한 모든 작업을 단 한 줄의 코드로 해치울 수 있어!

```
# 파이썬의 'set'을 사용한 초간단 집합 연산!
a = ['apple', 'melon', 'orange', 'cow']
b = ['chicken', 'pig', 'cow', 'orange']

# 1. 각 리스트를 set으로 변신! (중복은 알아서 제거됨)
set_a = set(a)
set_b = set(b)

# 2. 기호 하나로 연산 끝!
union_set = set_a | set_b     # 합집합 ( | <-- Shift + \ )
```

```
intersection_set = set_a & set_b # 교집합
difference_set = set_a - set_b # 차집합

print("\n--- 파이썬의 set 기능을 사용한 결과 ---")
print("합집합:", list(union_set)) # 다시 리스트로 바꿔서 볼 수 있어!
print("교집합:", list(intersection_set))
print("차집합:", list(difference_set))
```

코딩새싹 🌱: 우와!!! 우리가 여러 줄에 걸쳐서 짰던 코드가 기호 하나로 해결되네요! 정말 마법 같아요!

선생님: 그렇지? 하지만 오늘 여러분이 for 반복문으로 직접 그 원리를 구현해봤기 때문에, 이 set이라는 도구가 내부적으로 어떤 똑똑한 일을 하는지 더 깊이 이해할 수 있게 된 거란다. 기본 원리를 아는 것이 가장 중요하니까! 오늘 정말 어려운 개념을 스스로 탐구하고 해결해낸 여러분 모두가 자랑스럽구나!