<[파이썬 데이터 탐험의 시작] 제1장: 똑똑한 이름표, 판다스 시리즈!>

선생님: (엑셀 파일 아이콘과 데이터가 적힌 표를 보여주며) 얘들아, 안녕! 우리가 수많은 데이터를 정리하고 분석하려면 어떻게 해야 할까? 마치 엑셀처럼, 표 형태로 데이터를 다룰 수 있다면 정말 편하겠지? 파이썬에는 바로 이 '슈퍼 엑셀'과 같은 역할을 하는 아주 강력한 라이브러리, **판다스(Pandas)**가 있단다!

코딩새싹 🌱: 판다스요? 대나무 먹는 판다처럼 데이터를 맛있게 먹어치우나요? 🐼

선생님: 하하! 아주 재미있는 상상이야! 판다스는 복잡한 데이터를 아주 쉽게 조작하고 분석할 수 있게 도와주는, 데이터 과학자들의 필수 도구란다. 그리고 이 판다스의 가장 기본이 되는 첫 번째 보물 상자가 바로 **시리즈(Series)**야!

<1. 리스트의 진화! 이름표를 가진 시리즈(Series)>

선생님: 시리즈는 우리가 잘 아는 리스트와 비슷하지만, 각 데이터 값마다 '이름표(인덱스, index)'를 붙일 수 있는 아주 똑똑한 1차원 데이터 보관함이야. 엑셀 시트의 한 열(세로줄)을 떠올리면 쉽지.

```
import pandas as pd # 판다스 라이브러리를 pd라는 별명으로 불러오자!

# 값과 이름표(인덱스)를 함께 지정해서 시리즈 만들기!
s = pd.Series([1, 2, 3, 4], index=["a", "b", "c", "d"])
print("--- 이름표가 붙은 시리즈 ---")
print(s)
```

궁금이 💡: 와! 정말 리스트 `[1, 2, 3, 4]`에 `a, b, c, d`라는 이름표가 붙어서 나왔어요! 그럼 순서 번호 대신 저 이름표로 값을 꺼낼 수 있나요?

선생님: 정확해! 그게 바로 시리즈의 가장 큰 장점 중 하나지!

```
# 'a' 이름표가 붙은 칸의 값을 꺼내줘!
print("\n'a' 인덱스의 값:", s["a"])
```

<2. 실전! 데이터 정리 마법 - 도수분포표 만들기>

선생님: 자, 우리 반 학생 24명의 몸무게를 무작위로 뽑았다고 상상해보자. 이 데이터만 봐서는 한눈에 파악하기 어렵지? 그래서 "20~25kg 사이는 몇 명, 25~30kg 사이는 몇 명" 이렇게 구간별로 학생 수를 세어서 표로 만드는 게 좋겠지? 이걸 **도수분포표**라고 해!

코딩새싹 🌱: 네! 그렇게 정리하면 어떤 몸무게의 학생들이 가장 많은지 바로 알 수 있겠네요!

선생님: 이걸 판다스 시리즈로 아주 멋지게 만들어 볼 거야!

```
import numpy as np # 무작위 숫자와 빠른 계산을 위해 넘파이 소환!

# 1. 임의의 몸무게 데이터 생성 (20~50kg 사이 24명)
weight = np.random.randint(20, 51, 24)
print("\n--- 무작위 몸무게 데이터 (24명) ---")
print(weight)
```

```
# 2. 각 구간의 학생 수를 셀 '계수기' 준비 (처음엔 모두 0명)
# [20~25, 25~30, 30~35, 35~40, 40~45, 45~50] 총 6개의 구간
hist = np.zeros(6) # 0이 6개 들어있는 넘파이 배열!

# 3. for 반복문으로 학생들의 몸무게를 하나씩 확인하며 계수기 숫자 올리기!
for i in weight:
  if 20 <= i < 25:
    hist[0] += 1
  elif 25 <= i < 30:
    hist[1] += 1
  elif 30 <= i < 35:
    hist[2] += 1
  elif 35 <= i < 40:
    hist[3] += 1
  elif 40 <= i < 45:
    hist[4] += 1
  else: # 45~50
    hist[5] += 1

# 4. 결과(hist)에 이름표(bins)를 붙여 판다스 시리즈로 완성!
bins = ['20~25kg', '25~30kg', '30~35kg', '35~40kg', '40~45kg', '45~50kg'] # 구간 이름표
s_weight = pd.Series(hist, index=bins, dtype=int) # 정수 타입으로!
print("\n--- 몸무게 도수분포표 (시리즈) ---")
print(s_weight)
```

궁금이 💡: 와! if-elif-else문으로 각 몸무게가 어떤 구간에 속하는지 확인해서 hist 배열의 숫자를 하나씩 올리고, 마지막에 그 결과랑 이름표를 pd.Series에 쏙 넣어주니까 정말 보기 좋은 표가 뚝딱 만들어졌어요!

선생님: 그렇지? 수학 성적 도수분포표도 똑같은 원리로 만들 수 있단다! (사용자가 제공한 수학 성적 코드 예시를 보여주며)

〈3. 이름표가 핵심! 시리즈의 연산 마법〉

선생님: 판다스 시리즈의 연산은 아주 똑똑한 규칙이 있어. 바로 **이름표(인덱스)를 기준으로 계산**한다는 거야! 예를 들어, 두 가게의 일부 상품 판매량을 시리즈로 만들었다고 해보자.

```
data1 = {'사과': 300, '바나나': 100, '오렌지': 200} # 가게1 판매량
s1 = pd.Series(data1)

data2 = {'사과': 20, '포도': 30, '오렌지': 100} # 가게2 판매량
s2 = pd.Series(data2)
```

```
print("\n--- 가게1 판매량 (s1) ---")
print(s1)
print("\n--- 가게2 판매량 (s2) ---")
print(s2)

# 두 가게의 판매량을 합치면?
total_sales = s1 + s2
print("\n--- 두 가게 판매량 합계 ---")
print(total_sales)
```

코딩새싹 🌱: 앗! '사과'랑 '오렌지'는 두 가게에 모두 있으니까 합쳐졌는데, '바나나'랑 '포도'는 한쪽 가게에만 있어서 NaN이라고 나왔어요! NaN이 뭐예요?

선생님: NaN은 'Not a Number', 즉 '숫자가 아님(데이터 없음)'이라는 뜻이야. 판다스는 이렇게 이름표(인덱스)가 같은 것끼리만 계산하고, 짝이 없는 이름표는 계산할 수 없다고 NaN으로 알려준단다. 순서가 아니라 이름표를 따라다니는 아주 똑똑한 방식이지!

<4. 데이터 찾기 종합! (딕셔너리 + 리스트)>

선생님: 시리즈는 마치 딕셔너리와 리스트가 합쳐진 것 같아서, 데이터를 찾는 방법도 다양해.

```
data1 = {'a': 300, 'b': 100, 'd': 200}
s1 = pd.Series(data1)
print("\n--- s1 시리즈로 데이터 찾기 ---")
print(s1)

# 방법 1: 딕셔너리처럼 이름표(인덱스)로 찾기
print("s1['a'] ->", s1['a'])

# 방법 2: 리스트처럼 순서 번호로 찾기 (0부터 시작)
print("s1[1] ->", s1[1]) # 두 번째 항목인 100이 나오지!

# 방법 3: 리스트처럼 슬라이싱하기
print("s1[:2] ->\n", s1[:2]) # 맨 앞부터 2개!

# 이름표(키)가 있는지 확인하기
print("\n'f'라는 이름표가 s1에 있나요?", 'f' in s1) # False

# 값(value)이 있는지 확인하기
print("300이라는 값이 s1에 있나요?", 300 in s1.values()) # .values()를 꼭 써줘야 해! True
```

궁금이 💡: in을 쓸 때, 그냥 s1에 쓰면 이름표(인덱스)에서 찾고, .values()를 붙여주면 값에서 찾는 거군요!

딕셔너리랑 규칙이 비슷하네요!

선생님: (흐뭇하게) 아주 잘했어! 오늘 우리는 판다스의 기본 벽돌인 '시리즈'에 대해 배웠어. 이름표(인덱스)를 붙여 데이터를 똑똑하게 관리하고, 이름표를 기준으로 계산하며, 도수분포표 같은 유용한 데이터 정리까지 해봤지! 이 시리즈가 모이면 다음 시간에 배울 '데이터프레임'이라는 거대한 표가 된단다! 오늘 탐험도 정말 훌륭했어! 🌟