<제12장. 데이터 탐험가 출동! 판다스를 만나다!>

선생님: (탐정 모자를 쓰고 돋보기를 들며) 얘들아, 안녕! 오늘은 우리가 데이터라는 광활한 바다를 항해하는 데 필요한 아주 강력한 도구, **판다스(Pandas)**를 소개하려고 해! 판다스는 복잡하고 많은 양의 데이터를 마치 게임 캐릭터 다루듯이 쉽고 재미있게 분석하고 정리할 수 있게 도와주는 마법 같은 라이브러리란다! 📊✨

코딩새싹 🌱: 판다스요? 혹시 대나무 먹는 그 귀여운 동물 판다랑 관련이 있나요? 🐼

선생님: 하하! 이름은 비슷하지만, 이 판다스는 데이터를 아주 잘 다루는 똑똑한 친구란다! 판다스에는 크게 두 가지 중요한 데이터 구조가 있는데, 바로 '시리즈(Series)'와 '데이터프레임(DataFrame)'이야. 먼저 시리즈부터 만나볼까?

<1. 판다스의 첫 번째 마법 도구: 시리즈(Series) - 한 줄 데이터의 달인!>

선생님: 시리즈(Series)는 마치 이름표가 붙어있는 한 줄짜리 목록과 같아. 엑셀 시트의 한 열(세로줄)을 떠올리면 쉽지. 각 데이터 값마다 고유한 이름표, 즉 **인덱스(index)**를 붙여줄 수 있단다. 자, 요일별 기온을 시리즈로 만들어볼까?

```
import pandas as pd # 판다스 라이브러리를 pd라는 이름으로 불러올게!


# 리스트를 이용해 Series 생성
# Series는 하나의 컬럼(열)처럼 1차원 데이터를 담는 구조입니다.
temperatures = pd.Series([72, 75, 78, 80, 83, 85, 88],
                         index=['월요일', '화요일', '수요일', '목요일', '금요일', '토요일', '일요일'])


print("요일별 기온 시리즈:")
print(temperatures)
```

궁금이 💡: `pd.Series()`로 만드네요! 첫 번째 리스트 `[72, 75, ...]`는 값들이고, `index=[...]`는 그 값들의 이름표군요!

선생님: 정확해! 시리즈는 값(values)과 그 값에 해당하는 인덱스(index)로 이루어져 있지. 한번 확인해볼까?

```
print("\n시리즈의 인덱스:", temperatures.index)
print("시리즈의 값들:", temperatures.values)
```

코딩새싹 🌱: 와! 인덱스는 요일 이름들이고, 값들은 숫자 기온들이네요! 그럼 특정 요일의 기온을 알고 싶으면

어떻게 해요? 예를 들어 월요일 기온이요!

선생님: 아주 좋은 질문이야! 시리즈에서는 인덱스 이름표를 사용해서 값을 바로 꺼낼 수 있단다. 두 가지 방법이 있지!

```
print("\n월요일 기온 (점 찍고 이름표):", temperatures.월요일)
print("월요일 기온 (대괄호 안에 이름표):", temperatures["월요일"])
```

궁금이 💡: 오! 점(.)을 찍고 인덱스 이름을 쓰거나, 대괄호 [] 안에 따옴표랑 같이 인덱스 이름을 넣으면 되는군요!

선생님: 그렇지! 자, 그럼 여기서 **<시리즈 퀴즈 1번!>** temperatures 시리즈에서 금요일의 기온을 조회하려면 어떻게 해야 할까?

코딩새싹 🌱: 음… temperatures['금요일'] 이렇게 하면 될 것 같아요!

선생님: 정답! 확인해볼까?

```
# 퀴즈 1: 금요일의 기온 조회
print("\n퀴즈1: 금요일의 기온:", temperatures['금요일'])
```

선생님: 아주 잘했어! 이제 **<시리즈 퀴즈 2번!>** 지금 기온은 화씨인데, 이걸 전부 섭씨로 바꾸고 싶어. 화씨(F)에서 섭씨(C)로 바꾸는 공식은 C = (F - 32) * 5/9 란다. 이 시리즈 전체에 이 공식을 적용해볼 수 있을까?

궁금이 💡: 시리즈 전체에 한 번에 계산이 되나요? 마치 숫자 하나 계산하듯이요?

선생님: 판다스 시리즈의 마법 중 하나지! 시리즈에 직접 사칙연산을 하면 각 값에 똑같은 연산이 한꺼번에 적용된단다!

```
# 퀴즈 2: 화씨에서 섭씨로 변환
temperatures_celsius = (temperatures - 32) * 5/9
print("\n퀴즈2: 섭씨로 변환된 기온:")
print(temperatures_celsius)
```

코딩새싹 🌱: 와! 정말 모든 기온이 한 번에 섭씨로 바뀌었어요! 판다스, 정말 똑똑한데요!

선생님: 마지막 **<시리즈 퀴즈 3번!>** 만약 다음 주 월요일 기온이 90도로 예측된다면, 이 temperatures 시리즈에 "다음 주 월요일"이라는 새 이름표와 90이라는 값을 추가하려면 어떻게 할까?

궁금이 💡: 딕셔너리에 새 값 추가하듯이 하면 되지 않을까요? `temperatures['다음 주 월요일'] = 90` 이렇게요?

선생님: 빙고! 시리즈도 딕셔너리처럼 새로운 인덱스로 값을 할당하면 데이터가 추가된단다!

```
# 퀴즈 3: 새로운 데이터 추가
temperatures['다음 주 월요일'] = 90
print("\n퀴즈3: '다음 주 월요일' 기온 추가 후:")
print(temperatures)
```

<2. 판다스의 두 번째 마법 도구: 데이터프레임(DataFrame) - 모든 것을 담는 표!>

선생님: 시리즈가 한 줄짜리 목록이었다면, **데이터프레임(DataFrame)**은 여러 줄과 여러 칸으로 이루어진 표 전체라고 생각하면 돼! 엑셀 시트 그 자체를 떠올리면 된단다. 기온 정보뿐만 아니라 습도 정보도 함께 담아볼까?

```
# 두 개의 컬럼을 가진 DataFrame 생성
# DataFrame은 여러 개의 컬럼(열)을 담을 수 있는 2차원 구조입니다.
weather_data = pd.DataFrame({
    '기온': [72, 75, 78, 80, 83, 85, 88], # '기온'이라는 이름의 열
    '습도': [65, 60, 58, 57, 55, 54, 52]  # '습도'라는 이름의 열
}, index=['월요일', '화요일', '수요일', '목요일', '금요일', '토요일', '일요일']) # 행 이름표는 그대로 요

print("\n요일별 날씨 데이터프레임:")
print(weather_data)
```

코딩새싹 🌱: 와! 정말 표처럼 생겼어요! '기온' 열이랑 '습도' 열이 있고, 각 줄에는 요일 이름표가 붙어있네요!

선생님: 그렇지? 데이터프레임은 여러 개의 시리즈가 모여서 만들어진 거라고 생각할 수도 있어. 자, 그럼 **<데이터프레임 퀴즈 1번!>** 이 `weather_data`에서 화요일과 목요일의 '기온'만 쏙 뽑아오려면 어떻게 해야 할까? 이럴 땐 `.loc` 라는 특별한 도구를 사용한단다!

궁금이 💡: `.loc`요? 위치(location)를 찾는 건가요? 어떻게 쓰는 거죠?

선생님: 맞아! `.loc` 다음 대괄호 `[]` 안에, 앞에는 가져오고 싶은 행 이름표(들), 뒤에는 가져오고 싶은 열 이름표(들)를 적어주면 돼.

```
# 퀴즈 1: 화요일과 목요일의 기온 조회
# .loc[ [행_이름표_리스트], 열_이름표 ]
print("\n퀴즈1: 화요일과 목요일의 기온:")
print(weather_data.loc[['화요일', '목요일'], '기온'])
```

코딩새싹 🌱: 아하! ['화요일', '목요일'] 이렇게 리스트로 행들을 지정하고, '기온'이라고 열을 딱 지정하니까 정말 그것만 쏙 나왔어요!

선생님: 다음은 **<데이터프레임 퀴즈 2번!>** 아까 시리즈에서 했던 것처럼, weather_data에 있는 모든 '기온' 값을 섭씨로 변환해서 "섭씨 기온"이라는 새로운 열로 추가해볼까?

궁금이 💡: 데이터프레임의 한 열은 시리즈랑 비슷하다고 하셨으니까… '기온' 열을 꺼내서 계산하고, 그걸 새로운 열 이름으로 넣어주면 될까요? weather_data['섭씨 기온'] = (weather_data['기온'] - 32) * 5/9 이렇게요?

선생님: 정확해! 아주 논리적인 추론이야!

```
# 퀴즈 2: 화씨에서 섭씨로 변환하여 새로운 컬럼에 추가
weather_data['섭씨 기온'] = (weather_data['기온'] - 32) * 5/9
print("\n퀴즈2: '섭씨 기온' 컬럼 추가 후:")
print(weather_data)
```

코딩새싹 🌱: 와! "섭씨 기온"이라는 새 열이 생겼어요! 정말 편리한데요!

선생님: 마지막 **<데이터프레임 퀴즈 3번!>** 이 표에 있는 평균 기온과 평균 습도를 각각 계산해서 보여주려면 어떻게 할까? 판다스에는 평균을 구해주는 .mean()이라는 착한 기능이 있단다.

궁금이 💡: 그럼 각 열을 선택하고 .mean()을 붙이면 될까요? weather_data['기온'].mean() 이렇게요?

선생님: 바로 그거야! 각 열이 시리즈니까, 시리즈에 쓸 수 있는 기능을 똑같이 쓸 수 있지!

```
# 퀴즈 3: 평균 기온과 평균 습도 계산
mean_temperature = weather_data['기온'].mean()
mean_humidity = weather_data['습도'].mean()

print("\n퀴즈3: 평균 기온과 습도")
print("평균 기온 (화씨):", mean_temperature)
```

```
print("평균 습도:", mean_humidity)
```

<제13장. 데이터의 고향, CSV 파일을 만나다!>

선생님: (CSV라고 적힌 종이 파일을 보여주며) 자, 우리가 이렇게 판다스로 멋지게 정리한 데이터나, 다른 프로그램에서 만들어진 표 형태의 데이터는 보통 **CSV 파일**이라는 형식으로 많이 저장하고 주고받는단다. CSV는 'Comma Separated Values'의 약자로, 쉼표(,)로 값들이 구분된 파일이라는 뜻이야. 마치 우리가 손으로 쓴 목록 같은 거지.

코딩새싹 🌱: 쉼표로 값을 나눈 파일이요? 메모장 같은 데서 열어볼 수 있나요?

선생님: 물론! 아주 간단한 형태라서 어떤 편집기에서도 열어볼 수 있어. 예를 들어 "weather.csv" 파일이 이렇게 생겼다고 상상해보자.

(선생님은 "weather.csv" 파일 내용을 상상하며 설명한다)

```
날짜,온도,습도  <-- 이것이 첫 줄, 헤더(머리말)라고 부르지!
2024-05-20,18,70
2024-05-21,22,65
2024-05-22,20,68
2024-05-23,24,60
2024-05-24,17,72
2024-05-25,19,75
```

<1. CSV 파일 읽기 첫 번째 방법: 파이썬 기본 기능 활용하기 (하지만 조금 불편할 수도!)>

선생님: 파이썬의 기본 파일 읽기 기능으로도 CSV 파일을 읽을 수는 있어. `readlines()`를 쓰면 각 줄이 리스트의 항목으로 들어오지.

```
# "weather.csv" 파일이 있다고 가정하고, 기본 기능으로 읽어보기
# (실제 실행을 위해서는 "weather.csv" 파일이 같은 폴더에 위와 같은 내용으로 저장되어 있어야 해요!)
try:
    with open("weather.csv", "r", encoding="utf-8") as file: # 한글이 있을 수 있으니 encoding!
      data_lines = file.readlines() # 모든 줄을 리스트로 읽어옴
      print("\n--- weather.csv 파일 readlines()로 읽기 (일부) ---")
      if len(data_lines) > 5: # 여섯 번째 줄이 있다면
```

```
        print("여섯 번째 줄의 첫 번째 글자:", data_lines[5][0]) # data_lines[5]는 여섯 번째 줄 문자열, [0
    else:
        print("파일에 여섯 줄 이상이 없어요.")
except FileNotFoundError:
    print("\n앗! weather.csv 파일을 찾을 수 없어요. 예시를 위해 넘어갈게요!")
```

궁금이 💡: data_lines[5][0] 이렇게 하면 여섯 번째 줄의 첫 번째 글자만 나오는군요! 그런데 만약 쉼표로 구분된 각 값을 꺼내려면… 우리가 직접 쉼표를 기준으로 글자를 잘라야 하나요? 복잡할 것 같아요!

선생님: 맞아, 궁금아! 그렇게 하려면 조금 번거로울 수 있어. 그래서 파이썬에는 CSV 파일을 더 똑똑하게 다룰 수 있는 특별한 도구가 준비되어 있단다!

<2. CSV 파일 읽기 두 번째 방법: csv 모듈 전문가에게 맡기기!>

선생님: 바로 csv 라는 이름의 모듈이야! 이 친구는 CSV 파일을 줄별로, 그리고 각 줄 안에서 쉼표로 구분된 값들별로 깔끔하게 나누어준단다.

```
import csv # csv 모듈을 불러오자!

# (마찬가지로 "weather.csv" 파일이 필요해요!)
try:
    with open("weather.csv", newline='', encoding="utf-8") as file: # newline='' 옵션은 CSV 파일을 다룰 때
        csv_reader = csv.reader(file) # csv 전문가에게 파일을 넘겨주면,

        header = next(csv_reader) # next() 함수는 첫 번째 줄(헤더)을 읽고 다음 줄로 넘어가게 해줘.
        print(f"\n--- weather.csv 파일 csv.reader로 읽기 (헤더: {header}) ---")

        print("온도가 20도 이상인 날짜들:")
        for row in csv_reader: # csv_reader는 한 줄씩 목록(리스트) 형태로 데이터를 돌려줘.
            # row는 ['2024-05-20', '18', '70'] 이런 식의 리스트가 돼!
            if int(row[1]) >= 20: # 온도는 두 번째 값(인덱스 1)이고, 문자열이니 숫자로 바꿔서 비교!
                print(row[0]) # 날짜는 첫 번째 값(인덱스 0)
except FileNotFoundError:
```

```
    print("\n앗! weather.csv 파일을 찾을 수 없어요. 예시를 위해 넘어갈게요!")
except IndexError:
    print("\n앗! CSV 파일의 형식이 예상과 달라요. 각 줄에 충분한 데이터가 있는지 확인해주세요.")
except ValueError:
    print("\n앗! CSV 파일의 숫자 데이터 형식이 잘못되었어요. 온도가 숫자로 잘 적혀있는지 확인해주세요.")
```

코딩새싹 🌱: 와! `csv.reader()`를 쓰니까 `for row in csv_reader:` 할 때 `row`가 알아서 쉼표 기준으로 나뉜 리스트가 되네요! `row[0]`은 날짜, `row[1]`은 온도! 정말 편해요! `next(csv_reader)`는 왜 쓴 거예요?

선생님: CSV 파일의 첫 줄은 보통 각 열이 무엇을 의미하는지 알려주는 '헤더(머리말)'가 있거든. `next()` 함수는 이 헤더 줄을 한 번 읽고 지나가서, 그 다음 줄부터 실제 데이터만 깔끔하게 처리할 수 있게 도와주는 거란다. 그리고 온도 값은 파일에 문자열로 저장되어 있으니, `int(row[1])`처럼 숫자로 바꿔서 비교해야 한다는 것도 중요한 포인트지!

궁금이 💡: 판다스랑 CSV 파일 다루는 법까지 배우니까 정말 데이터 전문가가 된 것 같아요! 이제 어떤 데이터가 와도 분석할 수 있을 것 같은 자신감이 생겨요!

선생님: (활짝 웃으며) 아주 멋진 자세야! 판다스와 CSV 파일 다루는 법은 데이터 과학의 첫걸음이자 가장 중요한 기술 중 하나란다. 오늘 배운 내용들을 잘 기억하고 연습한다면, 여러분은 정말 멋진 데이터 탐험가가 될 수 있을 거야! 수고 많았다, 우리 꼬마 데이터 과학자들! 🚀