<[파이썬 데이터 탐험의 시작] 제2장: 데이터의 궁전, 데이터프레임(DataFrame)을 건설하자!>

선생님: (여러 개의 과목 성적표가 모인 전체 성적표를 보여주며) 얘들아, 안녕! 지난 시간에 배운 시리즈가 한 과목의 성적표(한 줄의 데이터) 같았다면, 오늘 배울 **데이터프레임(DataFrame)**은 여러 과목, 여러 학생의 성적이 모두 담긴 이 전체 성적표와 같단다! 즉, 행과 열을 가진 2차원 테이블 형태의 데이터를 다루는 판다스의 핵심 도구이지!

코딩새싹 🌱: 와! 엑셀 시트랑 정말 똑같이 생긴 것 같아요! 그럼 데이터프레임은 어떻게 만들어요?

선생님: 데이터프레임을 만드는 방법은 여러 가지가 있어! 마치 궁전을 지을 때, 벽돌을 쌓아 올릴 수도 있고, 기둥을 먼저 세울 수도 있는 것처럼 말이야.

<1. 데이터프레임 건설하기: 다양한 재료로 궁전 짓기!>

방법 1: 딕셔너리로 기둥 세우기! (가장 일반적인 방법)
각 열(column)의 제목을 키(key)로, 그 열에 들어갈 데이터들을 리스트나 넘파이 배열로 만들어서 딕셔너리에 담아주면, 판다스가 알아서 멋진 표로 만들어준단다.

```
import numpy as np
import pandas as pd

# 24명의 무작위 몸무게와 키 데이터 생성
weights = np.random.randint(40, 90, 24)
heights = np.random.randint(150, 190, 24)

# {'기둥이름1': 데이터들, '기둥이름2': 데이터들} 형태의 딕셔너리!
my_dict = {'weight': weights, 'height': heights}
df_from_dict = pd.DataFrame(my_dict) # 이 딕셔너리로 데이터프레임 건설!

print("--- 딕셔너리로 만든 데이터프레임 (앞 5줄만) ---")
print(df_from_dict.head()) # .head()는 너무 많을 때 앞 5줄만 보여주는 유용한 기능!
```

방법 2: 넘파이 배열로 벽돌 쌓기!
이미 2차원 넘파이 배열이 있다면, 이걸로도 바로 데이터프레임을 만들 수 있어. 대신, 각 기둥(열)의 이름은 우리가 직접 `columns` 옵션으로 알려줘야 해!

```
# 24행 2열짜리 0으로 채워진 넘파이 배열을 만들고,
f_np = np.zeros((24, 2))
# 0번 열에는 몸무게, 1번 열에는 키 데이터를 채워 넣는다!
f_np[:, 0] = weights
f_np[:, 1] = heights

# 넘파이 배열 f_np로 데이터프레임을 만들고, 열 이름은 직접 지정!
df_from_np = pd.DataFrame(f_np, columns=['weight', 'height'])
```

```
print("\n--- 넘파이 배열로 만든 데이터프레임 (앞 5줄만) ---")
print(df_from_np.head())
```

궁금이 💡: 아하! 딕셔너리로 만들 땐 딕셔너리의 키가 바로 열 이름이 되고, 넘파이 배열로 만들 땐 열 이름을 따로 알려줘야 하는군요!

선생님: 맞아! 그리고 학생 이름처럼 각 줄(행)에 특별한 이름표(인덱스)를 붙여줄 수도 있단다.

```
# 인덱스(이름표)를 직접 지정해서 만들기
weights = np.array([61, 50])
heights = np.array([172, 181])
my_dict_indexed = {"weight": weights, "height": heights}
df_with_index = pd.DataFrame(my_dict_indexed, index=["김봉남", "왕새우"])
print("\n--- 이름표(인덱스)가 있는 데이터프레임 ---")
print(df_with_index)
```

코딩새싹 🌱: 와! `index=["김봉남", "왕새우"]` 하니까 행 번호 대신 이름이 나왔어요!

<2. 궁전 탐험하기: 데이터에 접근하는 여러 가지 방법!>

선생님: 자, 이제 잘 지어진 데이터 궁전을 탐험해 볼 시간이야! 원하는 데이터를 콕콕 집어내는 방법을 배워보자!

1. 기둥(열) 하나 골라보기!

```
# [도전!] 친구 5명의 영어, 수학, 과학 성적 데이터프레임 만들기!
eng_scores = np.random.randint(30, 100, 5)
math_scores = np.random.randint(30, 100, 5)
sci_scores = np.random.randint(30, 100, 5)
student_names = ["손흥민", "이강인", "김민재", "오타니", "이순신"]
score_dict = {"영어": eng_scores, "수학": math_scores, "과학": sci_scores}
df_grades = pd.DataFrame(score_dict, index=student_names)

print("\n--- 우리 반 성적표! ---")
print(df_grades)

# 방법 1: 대괄호 사용 (가장 안전하고 일반적)
print("\n영어 성적만 보기 (대괄호):")
print(df_grades['영어'])

# 방법 2: 점(.) 사용 (편리하지만, 열 이름에 띄어쓰기가 있거나, 메소드 이름과 겹치면 사용 불가!)
print("\n수학 성적만 보기 (점):")
print(df_grades.수학)
```

궁금이 💡: 열 하나만 꺼내니까 우리가 지난 시간에 배운 '시리즈'가 나오네요! 데이터프레임은 시리즈들이

모여서 만들어진 게 맞군요!

2. 한 층(행) 전체 둘러보기! (.loc 과 .iloc)

.loc[]은 **이름표(label)**로 행을 찾고, .iloc[]은 **순서 번호(integer location)**로 행을 찾는단다!

```
# .loc[] : 이름표가 '손흥민'인 행의 모든 정보
print("\n--- 손흥민 학생의 성적표 (.loc) ---")
print(df_grades.loc["손흥민"])

# .iloc[] : 0번째 순서(첫 번째)에 있는 행의 모든 정보
print("\n--- 0번째 학생의 성적표 (.iloc) ---")
print(df_grades.iloc[0])
```

3. 특정 방(하나의 값) 찾아가기!

```
# 방법 1: 열 먼저 -> 행 나중에
print("\n손흥민의 영어 성적 (방법1):", df_grades.영어["손흥민"])

# 방법 2: loc 사용 -> 행 먼저 -> 열 나중에
print("손흥민의 영어 성적 (방법2):", df_grades.loc["손흥민"]["영어"])

# 선생님의 꿀팁! 🍯 loc을 써서 한 번에 찾는 게 가장 정확하고 빨라!
print("손흥민의 영어 성적 (프로의 방법):", df_grades.loc["손흥민", "영어"])
```

<3. 궁전의 설계도 확인하기: .columns 와 .index>

선생님: 우리 데이터프레임이 어떤 기둥(열)들과 어떤 층(행)들로 이루어져 있는지 궁금하다면, 설계도를 바로 확인할 수 있어!

```
print("\n--- 데이터프레임 설계도 ---")
print("모든 과목(열 이름들):", df_grades.columns)
print("모든 학생(행 이름들):", df_grades.index)
```

코딩새싹 🌱: .columns랑 .index를 쓰니까 어떤 데이터가 있는지 한눈에 알 수 있겠네요!

<4. 또 다른 건설 방법: 리스트의 리스트(튜플)!>

선생님: 데이터가 행 단위로 묶여있을 때 더 편하게 데이터프레임을 만드는 방법도 있단다!

```
# zip으로 몸무게와 키를 짝지어서 만들기
weight = [34, 45, 76]
height = [155, 164, 175]
wh_list = list(zip(weight, height)) # [(34, 155), (45, 164), (76, 175)]
df_from_zip = pd.DataFrame(wh_list, columns=["몸무게", "키"], index=['왕새우', '새끼새우', '뚱땡이새우'])
print("\n--- zip으로 만든 데이터프레임 ---")
print(df_from_zip)
```

```
# 행 데이터가 담긴 튜플들의 리스트로 만들기
data_tuples = [("왕새우", 90, 100, 85), ("김봉남", 60, 70, 45)]
df_from_tuples = pd.DataFrame(data_tuples, columns=["이름", "국어", "영어", "수학"])
print("\n--- 튜플 리스트로 만든 데이터프레임 ---")
print(df_from_tuples)
```

궁금이 💡: 와! 데이터가 어떻게 생겼는지에 따라 만드는 방법도 여러 가지네요! 딕셔너리로 만드는 건 열 기준으로, 리스트의 리스트로 만드는 건 행 기준으로 데이터를 준비할 때 편하겠어요!

선생님: (흐뭇하게 웃으며) 바로 그거야! 오늘 우리는 판다스의 심장이라고 할 수 있는 '데이터프레임'이라는 데이터의 궁전을 짓는 다양한 방법을 배우고, 그 궁전을 자유롭게 탐험하는 방법까지 익혔어! 이제 여러분은 웬만한 표 형태의 데이터는 모두 이 데이터프레임으로 가져와 멋지게 분석할 준비가 된 거란다! 다음 시간에는 이 궁전을 더 아름답게 꾸미고, 우리가 원하는 대로 재가공하는 마법을 배워볼 거야! 오늘 정말 훌륭했어! 🏛️☒👍