[4주차] 살아 움직이는 데이터: 제어문과 함수 👮‍♂️

안녕하세요! 4주차 수업에 오신 것을 환영합니다.

지난 시간에는 파이썬의 변수와 자료구조를 이용해 데이터를 '저장'하는 법을 배웠습니다.

오늘은 여기서 한 단계 더 나아가, 저장된 데이터를 '처리'하고 '조작'하는 방법을 배우겠습니다.

조건문(if) 을 이용해 특정 조건의 데이터를 골라내고,

반복문(for) 으로 수백, 수만 개의 데이터를 한 번에 처리하며,

이 모든 과정을 **함수(def)** 라는 재사용 가능한 부품으로 만드는 방법을 알아봅니다.

🚦 모듈 1: 조건을 따라 움직이는 코드, 조건문(if)

- **조건문이란?**
  - 특정 조건이 '참(True)'일 때만 코드 블록을 실행하도록 만드는 문법입니다.
- **기본 구조**

```
if 조건식:
    # 조건식이 True일 때 실행할 코드
elif 다른_조건식:
    # 다른_조건식이 True일 때 실행할 코드
else:
    # 위의 모든 조건식이 False일 때 실행할 코드
```

- 머신러닝에서의 활용
  - ex1) "만약 평점이 9.0 이상이면 '추천 영화'로 분류해라"
  - ex2) "만약 데이터에 빈 값이 있으면, 특정 값으로 채워라"
  - 데이터 전처리나 모델의 규칙을 정의할 때 필수적으로 사용됩니다.

In [ ]:
```
# 영화 평점에 따라 추천 여부를 결정하는 예제
movie_rating = 9.1

# '만약' movie_rating이 9.0보다 크거나 같으면
if movie_rating >= 9.0:
    print("✨ 강력 추천하는 영화입니다!")
# '그렇지 않으면'
else:
    print("볼만한 영화입니다.")
```

✨ 강력 추천하는 영화입니다!

In [ ]:
```
# 영화 평점에 따라 추천 여부를 결정하는 예제
movie_rating = 8.5

# '만약' movie_rating이 9.0보다 크거나 같으면
if movie_rating >= 9.0:
    print("✨ 강력 추천하는 영화입니다!")
# '그렇지 않으면'
else:
    print("볼만한 영화입니다.")

# movie_rating 값을 8.5로 바꾸고 다시 실행해보세요!
```

볼만한 영화입니다.

In [ ]:
```
# 조금 더 세분화된 추천 시스템
movie_rating = 8.8

if movie_rating >= 9.0:
    print("✨ 강력 추천! (9.0점 이상)")
# '그렇지 않고 만약' 8.5점 이상이면
elif movie_rating >= 8.5:
    print("👍 추천! (8.5점 이상)")
# '그 외 모든 경우'
else:
    print("🤔 보통. (8.5점 미만)")
```

👍 추천! (8.5점 이상)

🔁 모듈 2: 모든 데이터를 한 번에! 반복문(for)

- **반복문이란?**
  - 리스트나 딕셔너리 같은 자료구조에 담긴 모든 데이터에 대해,
  - **동일한 작업을 순서대로 반복** 실행하도록 만드는 문법입니다.
- **기본 구조**

```
for 아이템을_담을_변수 in 데이터_리스트:
    # 리스트의 각 아이템에 대해 실행할 코드
```

- 머신러닝에서의 활용
  - 수백만 개의 이미지 데이터를 하나씩 불러와 처리하거나,
  - 전체 데이터의 평균/합계를 구하는 등 대규모 데이터 처리에 필수적입니다.

In [ ]:
```
# 3주차에 만들었던 영화 데이터셋
my_movie_dataset = [
    { "title": "인셉션",
     "director": "크리스토퍼 놀란",
      "rating": 9.29 },
    { "title": "기생충",
     "director": "봉준호",
      "rating": 9.07 },
    { "title": "어바웃 타임",
     "director": "리차드 커티스",
      "rating": 9.27 }
]

# 데이터셋(리스트)에서 영화 정보(딕셔너리)를 하나씩 꺼내 movie라는 변수에 담아 반복
print("--- 모든 영화 제목 출력 ---")
for movie in my_movie_dataset:
    # 각 영화(딕셔너리)에서 'title' 키의 값을 꺼내 출력
    print(movie['title'])
```

```
--- 모든 영화 제목 출력 ---
인셉션
기생충
어바웃 타임
```

In [ ]:
```
# 반복문과 조건문을 함께 사용하여 '평점 9.2 이상인 영화'만 골라내기

print("\n--- 평점 9.2 이상인 추천 영화 ---")
for movie in my_movie_dataset:
    # '만약' 현재 영화의 평점이 9.2보다 크거나 같으면
    if movie['rating'] >= 9.2:
        # 해당 영화의 제목과 평점을 출력
        print(f"제목: {movie['title']}, 평점: {movie['rating']}")
```

```
--- 평점 9.2 이상인 추천 영화 ---
제목: 인셉션, 평점: 9.29
제목: 어바웃 타임, 평점: 9.27
```

⚙️ 모듈 3: 나만의 코드 부품, 함수(def)

- **함수란?**
  - 특정 작업을 수행하는 코드 덩어리에,
  - **이름을 붙여 재사용**할 수 있도록 만든 것입니다.
- **기본 구조**

```
def 함수이름(입력값1, 입력값2, ...):
    # 실행할 코드
    return 결과값
```

- 머신러닝에서의 활용
  - 데이터 전처리 과정을 함수로 만들어 여러 데이터셋에 동일하게 적용.
  - 모델 자체를 하나의 함수로 볼 수 있습니다.

    [입력(데이터) → 처리(예측) → 출력(결과)]
  - 복잡한 코드를 기능별로 나누어 관리하기 용이해집니다.

In [ ]:
```
# 두 숫자를 더하는 간단한 함수
def add_numbers(x, y):
    """
    이 함수는 두 개의 숫자 x와 y를 입력받아,
    그 합을 계산하여 결과값으로 돌려줍니다.
    """
    result = x + y
    return result

# 함수 호출 (사용)
sum_result = add_numbers(5, 3)
print(f"5와 3의 합은? {sum_result}")

another_sum = add_numbers(100, 200)
print(f"100과 200의 합은? {another_sum}")
```

```
5와 3의 합은? 8
100과 200의 합은? 300
```

In [ ]:
```
# 모듈 2에서 했던 '평점 기반 영화 추천' 로직을 함수로 만들어 봅시다.

def recommend_movies(dataset, min_rating):
    """
    영화 데이터셋과 최소 평점을 입력받아,
    조건에 맞는 영화 제목 리스트를 결과로 돌려줍니다.
    """
    recommended_list = [] # 추천 목록을 담을 빈 리스트
    for movie in dataset:
        if movie['rating'] >= min_rating:
            recommended_list.append(movie['title']) # 리스트에 추가
    return recommended_list


# 함수를 이용해 평점 9.0 이상 영화 추천받기
my_best_movies = recommend_movies(my_movie_dataset, 9.0)
print(f"평점 9.0 이상 추천 영화: {my_best_movies}")

# 같은 함수를 재사용해서 평점 9.1 이상 영화 추천받기
my_good_movies = recommend_movies(my_movie_dataset, 9.1)
print(f"평점 9.1 이상 추천 영화: {my_good_movies}")
```

```
평점 9.0 이상 추천 영화: ['인셉션', '기생충', '어바웃 타임']
평점 9.1 이상 추천 영화: ['인셉션', '어바웃 타임']
```