python #1

▼ 문제

다음은 직원(Employee)들의 정보를 담은 리스트입니다. 이 데이터를 활용하여 아래 조건을 만족하는 필터링 로직을 작성하세요.

```
employees = [
    {"name": "Alice", "department": "Engineering", "age": 30, "salary": 85000},
    {"name": "Bob", "department": "Marketing", "age": 25, "salary": 60000},
    {"name": "Charlie", "department": "Engineering", "age": 35, "salary": 95000},
    {"name": "David", "department": "HR", "age": 45, "salary": 70000},
    {"name": "Eve", "department": "Engineering", "age": 28, "salary": 78000}
]

1) 부서가 "Engineering"이고 salary >= 80000인 직원들의 이름만 리스트로 출력하세요.
2) 30세 이상인 직원의 이름과 부서를 튜플 (name, department) 형태로 리스트로 출력하세요.
3) 급여 기준으로 직원 리스트를 salary 내림차순으로 정렬하고, 상위 3명의 이름과 급여를 출력하세요.
4) 모든 부서별 평균 급여를 출력하는 코드를 작성해보세요.
```

▼ 코드 - 문제 1,2,3

```
# 데이터
employees = [
    {"name": "Alice", "department": "Engineering", "age": 30, "salary": 85000},
    {"name": "Bob", "department": "Marketing", "age": 25, "salary": 60000},
    {"name": "Charlie", "department": "Engineering", "age": 35, "salary": 95000},
    {"name": "David", "department": "HR", "age": 45, "salary": 70000},
    {"name": "Eve", "department": "Engineering", "age": 28, "salary": 78000}
]
```

```
# 문제 1: 부서가 "Engineering"이고 salary >= 80000인 직원들의 이름만 리스트로 출력
eng_high_salary = [emp["name"] for emp in employees if emp["department"] == "Engineering" and emp
["salary"] >= 80000]
print("문제 1:", eng_high_salary)
```

```
문제 1: ['Alice', 'Charlie']
```

- for emp in employees if emp["department"] == "Engineering" and emp["salary"] >= 80000
  - department가 "Engineering"이고 salary가 80000 이상인 직원
- [emp["name"]]
  - 이름을 리스트로 출력

```
# 문제 2: 30세 이상인 직원의 이름과 부서를 튜플 (name, department) 형태로 리스트로 출력
over_30 = [(emp["name"], emp["department"]) for emp in employees if emp["age"] >= 30]
print("문제 2:", over_30)
```

```
문제 2: [('Alice', 'Engineering'), ('Charlie', 'Engineering'), ('David', 'HR')
```

- for emp in employees if emp["age"] >= 30
  - age가 30세 이상인 직원
- [(emp["name"], emp["department"])]
  - 이름과 부서를 튜플 형태로 리스트로 출력

```
# 문제 3: 급여 기준으로 직원 리스트를 salary 내림차순 정렬 후, 상위 3명의 이름과 급여 출력
sorted_by_salary = sorted(employees, key=lambda x: x["salary"], reverse=True)
print("문제 3:")
for emp in sorted_by_salary[:3]:
    print(emp["name"], emp["salary"])
```

```
문제 3:
Charlie 95000
Alice 85000
Eve 78000
```

- sorted(employees, key=lambda x: x["salary"]
  - salary 기준
- reverse=True
  - 내림차순 정렬
- for emp in sorted_by_salary[:3]
  - 상위 3명 추출
- emp["name"], emp["salary"]
  - 이름과 급여 출력

▼ 코드 - 문제 4

```
# 문제 4: 모든 부서별 평균 급여 출력
from collections import defaultdict

dept_salaries = defaultdict(list) # 부서별 급여 딕셔너리
for emp in employees:
    dept_salaries[emp["department"]].append(emp["salary"]) # 부서를 key로 하고 급여를 value(리스트)에 추가

print("문제 4:")
for dept, salaries in dept_salaries.items():
    avg_salary = sum(salaries) / len(salaries) # 부서별 평균 급여 계산
    print(dept, avg_salary)
```

- dept_salaries = defaultdict(list)
  - key: 부서
  - value: 급여 list
- for emp in employees: dept_salaries[emp["department"]].append(emp["salary"])
  - dept_salaries의 department를 key로 하는 리스트에 salary를 추가
  - dept_salaries에 해당 department가 key에 없으면 자동으로 빈 리스트를 만든 후 추가 (defaultdict)
- for dept, salaries in dept_salaries.items()
  - dept_salaries의 각 key(department)와 value 리스트(salaries)에 대해

- avg_salary = sum(salaries) / len(salaries)
    - 해당 부서의 급여 합계를 급여 개수로 나눈 부서별 평균급여 출력