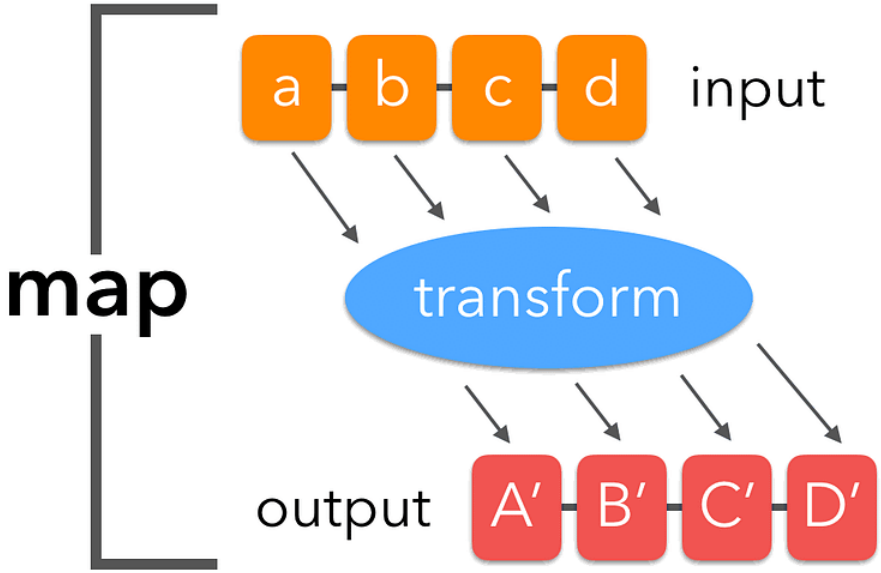
AI Experts
Who Lead
The Future

1

매핑(mapping) 개요와 활용

매핑 개요

- **매핑(mapping)**
  - 반복적인 개체의 각 항목에 변환 함수를 적용해 새로운 반복가능 개체로 변환해야 할 때 유용
  - 내장 함수 map() 사용
- **내장 함수 map()**
  - 반복 구문을 사용하지 않고 반복가능(iterable) 컨테이너(container)의 모든 항목(item)을 특정 함수의 인자로 처리할 수 있는 함수



모든 항목에 10 더하기

- 함수 addsome

```python
def addsome(value):
    return value + 10

lst = [10, 23, 4, 7]
result = map(addsome, lst)
list(result)
```

✓ 0.0s Python

```
[20, 33, 14, 17]
```

항목에서 콤마 제거

- 함수 myreplace()

```python
def myreplace(s):
    return s.replace(',', '')

lst = ['34,000', '1,999', '2,786', '1,000,000']
result = map(myreplace, lst)
list(result)
```

Python

```
['34000', '1999', '2786', '1000000']
```

모든 항목을 정수로 변환

- 내장 함수 int() 활용

```python
lst = ['3', '4', '10']
result = map(int, lst)
list(result)
```

```
[3, 4, 10]
```

모든 항목을 콤마를 제거하고 정수로 변환

- **map()을 2 번 사용**

```python
def myreplace(s):
    return s.replace(',', '')

lst = ['34,000', '1,999', '2,786', '1,000,000']
result = map(int, map(myreplace, lst))
list(result)
```

✓ 0.0s Python

```
[34000, 1999, 2786, 1000000]
```

다양한 map() 활용


```
int('101', 2)
```

✓ 0.0s Python

```
5
```

```
list(map(int, [3, 5, 11]))
```

✓ 0.0s Python

```
[3, 5, 11]
```

```
list(map(int, ['3', '5', 11]))
```

✓ 0.0s Python

```
[3, 5, 11]
```

```
list(map(int, ['10', '17', '1f'], [2, 8, 16]))
```

✓ 0.0s Python

```
[2, 15, 31]
```

익명 함수 사용

- 람다 함수

```
(lambda x, y: x + y)(30, 4)
```

✓ 0.0s Python

```
34
```

```
# def myreplace(s):
#     return s.replace(',', '')

lst = ['34,000', '1,999', '2,786', '1,000,000']
result = map(int, map(lambda s: s.replace(',', ''), lst))
list(result)
```

✓ 0.0s Python

```
[34000, 1999, 2786, 1000000]
```