주요 데이터 분석 - 딕셔너리

딕셔너리(Dictionary)

- 딕셔너리는 키-값 쌍으로 데이터를 저장
- 이는 JSON 형식과 유사하며, 데이터를 효율적으로 저장하고 검색할 때 유용
- API 공공데이터 반환값에서 많이 사용

https://www.criminalip.io/ko/developer/api/get-asset-search

특징

- 키(Key)와 값(Value)의 쌍으로 데이터 저장
- 순서가 없음
- 키는 변경 불가능한 타입이어야 하며 (문자열, 숫자, 튜플), 값은 어떤 타입이든 가능
  - 리스트의 인덱스(순서)에서는 정수만 가능 ex) list[0], list[-1]과 같이 접근한 것
- 빠른 검색, 수정, 삭제를 위한 최적화된 데이터 구조

딕셔너리 생성

- 딕셔너리는 중괄호 `{}`를 사용하여 생성
- 주의사항 : Key는 고유한 값이므로 중복되는 Key 값이 생성되면 하나를 제외한

```
person = {"name": "John", "age": 30, "city": "New York"} print(person)
```

딕셔너리 키로 사용 가능한 타입

사용 가능한 타입	설명
문자열 (String)	파이썬에서 가장 일반적으로 사용되는 불변 타입
숫자 (Integer, Float)	모든 종류의 숫자 (정수, 부동소수점)는 불변
튜플 (Tuple)	튜플의 요소가 모두 불변 타입일 때만 해시 가능하고 키로 사용 가능

딕셔너리 키로 사용 불가능한 타입

사용 불가능한 타입	설명
리스트 (List)	리스트는 요소를 추가, 삭제, 변경할 수 있어 변경 가능(mutable)
딕셔너리 (Dictionary)	딕셔너리 역시 키나 값이 변경 가능
집합 (Set)	집합은 내용을 변경할 수 있어서, 딕셔너리의 키로 사용 불가

딕셔너리 value 얻기

- 딕셔너리의 값에 접근하는 두 가지 주요 방법
- 직접 인덱싱을 사용하는 방법과 `get()` 메서드를 사용하는 방법

```
person = {"name": "John", "age": 30, "city": "New York"} print(person["nam
e"])
```

```
person = {"name": "John", "age": 30, "city": "New York"} print(person.get
("name",0))
```

딕셔너리 키, 값, 아이템 접근하기

- `keys()` 메서드는 딕셔너리의 모든 키를 포함하는 dict_keys 객체 반환
- `values()` 메서드는 딕셔너리의 모든 값들을 포함하는 dict_values 객체 반환
- `items()` 메서드는 딕셔너리의 키-값 쌍을 튜플로 갖는 dict_items 객체 반환
  - 각 튜플은 하나의 키-값 쌍을 나타냄
  - 종종 반복문에서 딕셔너리의 키와 값을 동시에 접근하는 데 사용

```
my_dict = {"name": "John", "age": 30, "city": "New York"} # 모든 키만 가져오기 keys = my_dict.keys() print(keys) # 모든 값만 가져오기 values = my_dict.values() print(values) # 모든 키-값 쌍 가져오기 items = my_dict.items() print(items)
```

딕셔너리 요소 추가 및 수정

- 딕셔너리의 특정 키에 대한 값을 접근하거나 수정할 때는 키를 사용

```
my_dict = {"name": "John", "age": 30, "city": "New York"} my_dict["email"] = "john@example.com" # 새로운 키와 값 추가 my_dict["age"] = 31 # 기존 키의 값 수정 print(my_dict) #ElasticSearch 데이터베이스 POST, PUT 메소드를 이용함!!!
```

딕셔너리 요소 제거

- `del` 키워드, `pop()` 메서드, 또는 `popitem()` 메서드를 사용하여 딕셔너리에서 요소를 제거

```
# del 키워드를 사용하여 키-값 쌍 제거 del my_dict["city"] # pop() 메서드를 사용하여 키-값 쌍 제거 및 값 반환 age = my_dict.pop("age") print("Removed age:", age)
```

커피 주문 프로그램

```
menus = {"아메리카노": 4000, "카페라떼": 4500, "카푸치노": 5000}
print("====== 메뉴판 ======") for menu, price in menus.items(): print(f"
{menu}: {price}원") selected_menu = input("메뉴를 선택하세요: ") money =
int(input("금액을 입력하세요.")) price = menus.get(selected_menu,0) if
price == 0: print("메뉴가 없습니다.") else: change = money - price if
change >= 0: print(f"{selected_menu}를 선택하셨습니다. 거스름돈은 {change}원
입니다.") else: print("돈이 부족합니다.")
```

```
#while 반복 while True 사용 #메뉴를 선택한다. (여러개의 메뉴를 선택 한다.) #
구매한 메뉴를 리스트로 보관도 해야 한다. #현금을 넣는다. #구매한후에 거스름돈
을 받는다. #구매했던 리스트와 총 구매가격? 출력!!!
```

while 문 사례 (20분 쉬시고, 10분 정도 시간 드립니다.)

```
menus = {"아메리카노": 4000, "카페라떼": 4500, "카푸치노": 5000} order_list
= [] total_price = 0 print("====== 메뉴판 ======") for menu, price in
menus.items(): print(f"{menu}: {price}원") while True: selected_menu =
input("주문할 메뉴를 입력하세요.") price = menus.get(selected_menu, 0) if
price != 0: order_list.append(selected_menu) total_price = total_price +
price elif selected_menu == "q": print("주문종료") break else: print("메뉴
가 없습니다.") print(f"선택한 메뉴 : {order_list}") money = int(input("금액
을 입력하세요.")) change = money - total_price if change >= 0: print(f"총
금액은 {total_price}원입니다. 거스름돈은 {change}원입니다.") else: print("돈
이 부족합니다.")
```