6章 連想配列（ディクショナリ）を理解しよう

6章 連想配列（ディクショナリ）を理解しよう

連想配列の概要を学び、実際に使ってみます

本章の目標

- 連想配列（ディクショナリ）とは何かを理解する
- ディクショナリの基本操作（作成・取得・追加・変更・削除）を習得する

6章 連想配列（ディクショナリ）とは

インデックスの代わりに「キー」を使う配列

通常の配列（リスト）

- 番号（インデックス）で要素を管理
- 例：`list[0]`、`list[1]`

連想配列（ディクショナリ）

- 「キー」という名前で要素を管理
- 例：`dict["name"]`、`dict["age"]`

6章 ディクショナリの作成

{ } と キー: 値 の形式で作成します

```
# ディクショナリの作成
person = {"name": "田中", "age": 25, "city": "東京"}
```

構文

```
変数名 = {
    "キー1": 値1,
    "キー2": 値2,
    "キー3": 値3
}
```

ポイント

- `{ }` で囲む
- キーと値を `:` でつなぐ
- 各ペアは `,` で区切る

ディクショナリ[キー] で値を取得します

```
person = {"name": "田中", "age": 25, "city": "東京"}
print(person["name"])  # 田中
print(person["age"])   # 25
```

リストとの違い

リスト	ディクショナリ
`list[0]`	`dict["キー"]`
番号でアクセス	キーでアクセス

ディクショナリ[キー] = 値 で追加・変更できます

新しいキーの場合 → 追加

```
person = {"name": "田中"}
person["age"] = 25
# {"name": "田中", "age": 25}
```

既存のキーの場合 → 変更

```
person = {"name": "田中", "age": 25}
person["age"] = 30
# {"name": "田中", "age": 30}
```

pop() メソッドで要素を削除できます

```
person = {"name": "田中", "age": 25, "city": "東京"}
person.pop("city")
print(person)  # {"name": "田中", "age": 25}
```

構文

```
ディクショナリ.pop("キー")
```

- 指定したキーの要素が削除される
- リストの `pop()` と似ているが、キーを指定する点が異なる

6章 ディクショナリの操作まとめ

基本操作を整理しましょう

操作	書き方	例
作成	`{キー: 値}`	`{"name": "田中"}`
取得	`辞書[キー]`	`person["name"]`
追加	`辞書[新キー] = 値`	`person["age"] = 25`
変更	`辞書[既存キー] = 値`	`person["age"] = 30`
削除	`辞書.pop(キー)`	`person.pop("age")`

6章 まとめ

本章では以下の内容を学習しました

連想配列（ディクショナリ）とは

- インデックス（番号）の代わりに「キー」で要素を管理する配列
- キーと値のペアでデータを格納する
- `{ }` と `キー: 値` の形式で作成する

6章 まとめ（続き）

本章では以下の内容を学習しました

値の取得

- `ディクショナリ["キー"]` で値を取得

要素の追加・変更

- `ディクショナリ["キー"] = 値` で追加または変更

要素の削除

- `ディクショナリ.pop("キー")` で削除