5章 配列（リスト・タプル・セット）を理解しよう

5章 配列（リスト・タプル・セット）を理解しよう

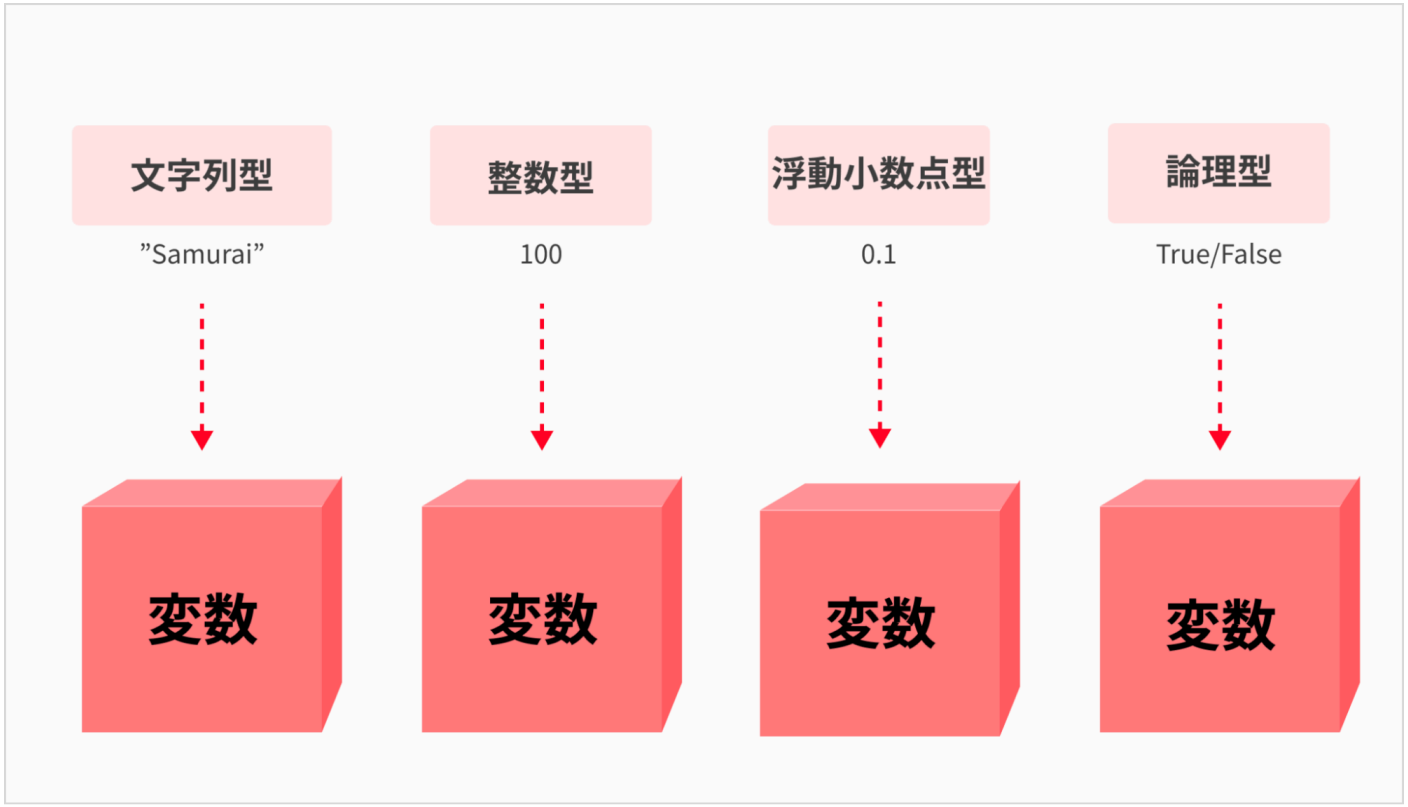
配列の概要を知り、実際に使ってみます

本章の目標

- 配列とは何か、概要をつかむこと
- 配列の作り方、使い方を知ること
- 実際に配列を使ってみること

5章 なぜ配列が必要なのか

これまでは、変数に1つのデータしか入れていませんでした



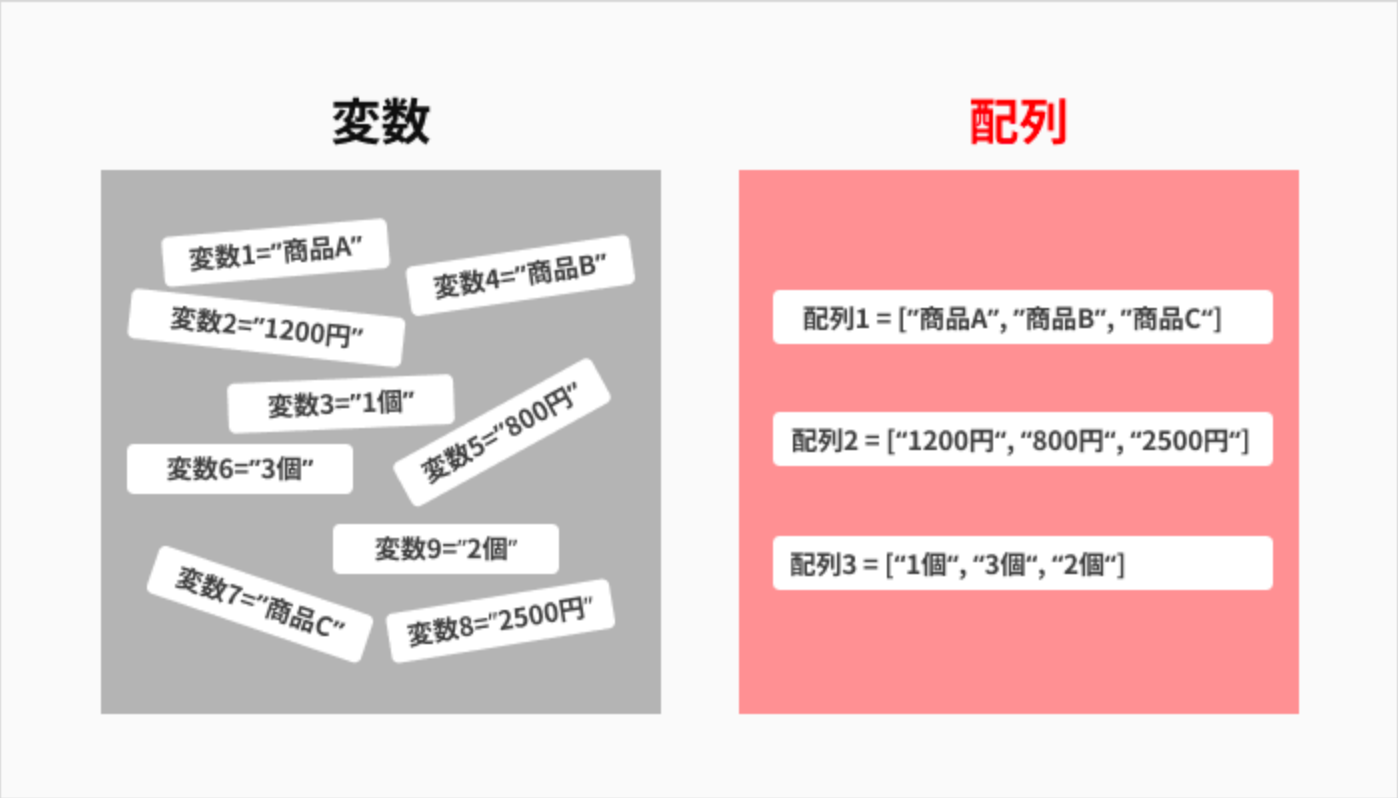
5章 複数のデータをまとめて管理したいケース

Amazonのようなショッピングサイトで買い物をするシーンをイメージしてください

「商品名」「価格」「数量」が保管されている

5章 変数と配列の違い

複数のデータは、1つのまとまったデータとして管理したほうが楽です

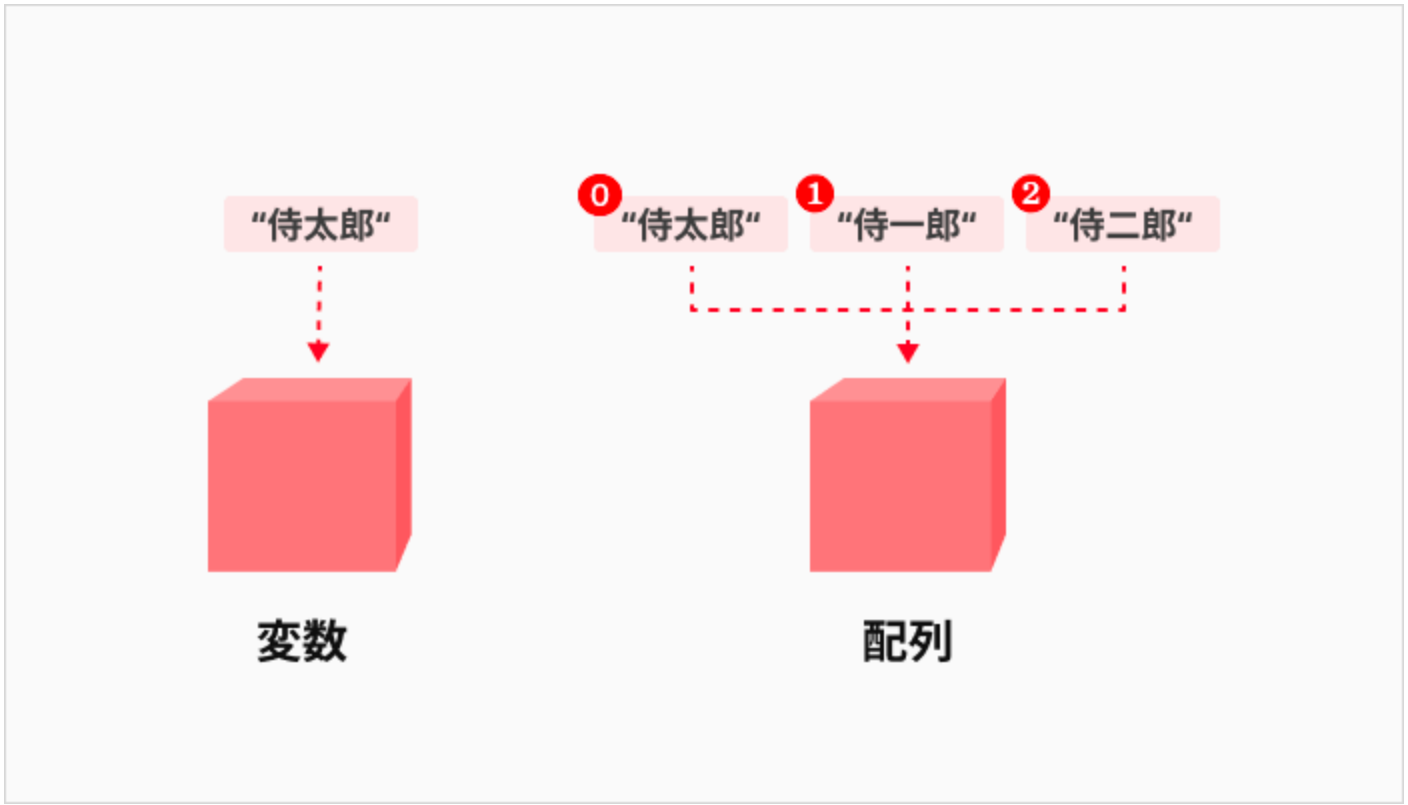


5章 配列とは

配列とは簡単にいえば、複数のデータのまとまりのことです

- 配列を使うことで、複数のデータを1つにまとめられる
- 配列に入れる1つひとつのデータを要素と呼ぶ

5章 配列のイメージ



5章 Pythonの配列の種類

Pythonの配列には、以下の4種類が存在します

種類	説明
リスト（list）	要素の追加・変更・削除が可能
タプル（tuple）	要素の追加・変更・削除が不可
セット（set）	順番と重複の概念を持たない
ディクショナリ（dictionary）	キーと値のペアで管理（7章で学習）

5章 リストの作り方

リスト（list）とは、要素の追加・変更・削除が行える配列のことです

```
リスト名 = [要素1, 要素2, 要素3, ...]
```

- 大かっこ [ ] ではさみ、その中に要素をカンマ（,）区切りで入れる
- 配列名は複数形にするのが一般的

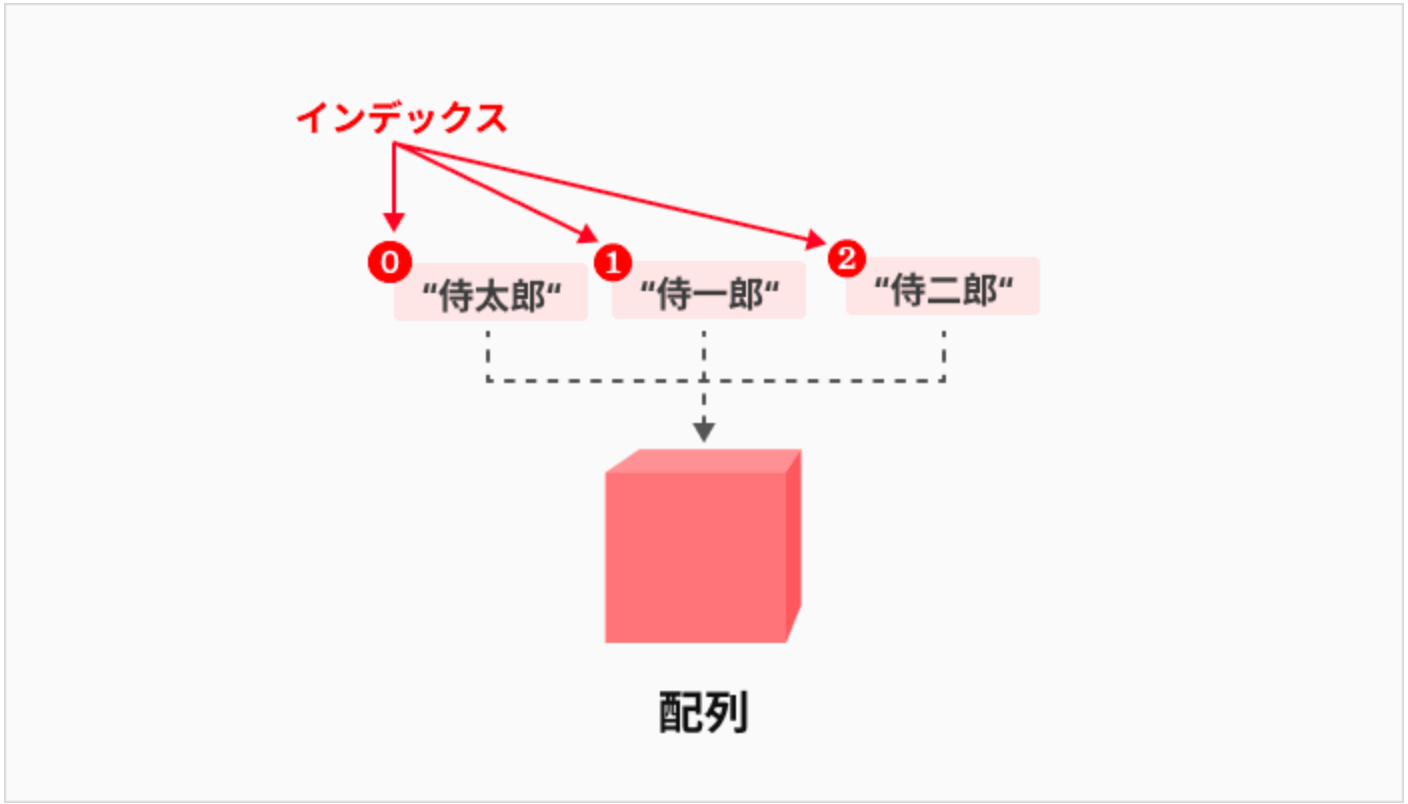
5章 リストの作成例

リストを作成してみましょう

```
user_names = ["侍太郎", "侍一郎", "侍二郎", "侍三郎", "侍四郎"]
user_ages  = [36, 33, 29, 25, 22]
```

- 文字列のリスト、数値のリストなど、様々なデータを格納できる

5章 インデックスとは



5章 リストの要素を取り出す方法

リスト名[インデックス] で要素を取り出せます

```
user_names = ["侍太郎", "侍一郎", "侍二郎", "侍三郎", "侍四郎"]

# 2番目の要素である「侍一郎」を取り出して表示する
print(user_names[1])
```

- インデックスは「0」から始まる点に注意
- 2番目の要素を取り出すにはインデックス「1」を指定

5章 リストの要素を更新する方法

リスト名[インデックス] = 新しい値 で要素を更新できます

```
user_names = ["侍太郎", "侍一郎", "侍二郎", "侍三郎", "侍四郎"]

# 2番目の要素を更新する
user_names[1] = "侍花子"
```

- 単独の変数と同様に、新しい値を再代入すればOK

5章 リストの要素を追加・削除する方法

メソッドを使って要素を追加・削除できます

メソッド	説明
append	要素を末尾に追加
pop	指定したインデックスの要素を削除

```
リスト名.append(追加する要素の値)
リスト名.pop(削除する要素のインデックス)
```

5章 リストの追加・削除の例

appendとpopを使ってみましょう

```
user_names = ["侍太郎", "侍一郎", "侍二郎", "侍三郎", "侍四郎"]

# 6番目の要素を追加する
user_names.append("侍五郎")

# 3番目の要素を削除する（0始まりのため2を指定）
user_names.pop(2)
```

5章 リストの追加・削除の結果

```
user_names = ["侍太郎", "侍一郎", "侍二郎", "侍三郎", "侍四郎"]

# 6番目の要素を追加する
user_names.append("侍五郎")
print(user_names)

# 3番目の要素を削除する
user_names.pop(2) # 0始まりのため、3番目のインデックスは2
print(user_names)
```

```
['侍太郎', '侍一郎', '侍二郎', '侍三郎', '侍四郎', '侍五郎']
['侍太郎', '侍一郎', '侍三郎', '侍四郎', '侍五郎']
```

Visual Studio Codeで以下のコードを実行してみましょう

```
user_names = ["侍太郎", "侍一郎", "侍二郎", "侍三郎", "侍四郎"]

print(user_names[1]) # 2番目の要素だけを表示

user_names[1] = "侍花子" # 2番目の要素を更新
print(user_names)

user_names.append("侍五郎") # 6番目の要素を追加
print(user_names)

user_names.pop(2) # 3番目の要素を削除
print(user_names)
```

5章 リストを使ってみよう：実行結果

```
user_names = ["侍太郎", "侍一郎", "侍二郎", "侍三郎", "侍四郎"]

print(user_names[1]) # 2番目の要素だけを表示

user_names[1] = "侍花子" # 2番目の要素を更新
print(user_names)

user_names.append("侍五郎") # 6番目の要素を追加
print(user_names)

user_names.pop(2) # 3番目の要素を削除（0始まりのため2を指定）
print(user_names)
```

```
侍一郎
['侍太郎', '侍花子', '侍二郎', '侍三郎', '侍四郎']
['侍太郎', '侍花子', '侍二郎', '侍三郎', '侍四郎', '侍五郎']
['侍太郎', '侍花子', '侍三郎', '侍四郎', '侍五郎']
```

5章 タプルの作り方

タプル（tuple）とは、要素の追加・変更・削除が行えない配列のことです

```
タプル名 =（要素1, 要素2, 要素3, ...）
```

- 小かっこ ( ) で要素をはさむ
- 値の書き換えができないため、不変のデータを管理するのに適している

5章 タプルの用途

タプルは不変のデータを管理するのに適しています

適している例

- ユーザーID
- 国名コード
- 固定の設定値

適していない例

- 更新日時
- 在庫数
- ユーザーの入力値

5章 タプルの要素を取り出す方法

タプル名[インデックス] で要素を取り出せます

```
country_names = ("日本", "アメリカ", "イギリス", "フランス")

# 3番目の要素である「イギリス」を取り出す
print(country_names[2])
```

- リストと同様に、インデックスは「0」から始まる

5章 タプルの値は変更できない

タプルのデータを書き換えようとするとエラーになります

```
country_names = ("日本", "アメリカ", "イギリス", "フランス")

# 3番目の要素を書き換える
country_names[2] = "ドイツ"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-15-2570c19ce039> in <module>
      2
      3 # 3番目の要素を書き換える
----> 4 country_names[2] = "ドイツ"

TypeError: 'tuple' object does not support item assignment
```

SEARCH STACK OVERFLOW

Visual Studio Codeで以下のコードを実行してみましょう

```
country_names = ("日本", "アメリカ", "イギリス", "フランス")

# 3番目の要素を取り出す
print(country_names[2])

# すべての要素を取り出す
print(country_names)
```

5章 タプルを使ってみよう：実行結果

```
country_names = ("日本", "アメリカ", "イギリス", "フランス")

# 3番目の要素を取り出す
print(country_names[2])

# すべての要素を取り出す
print(country_names)
```

```
イギリス
('日本', 'アメリカ', 'イギリス', 'フランス')
```

5章 セットの作り方

セット（set）とは、「順番」と「重複」の概念を持たない配列のことです

```
セット名 = {要素1, 要素2, 要素3, ...}
```

- 中かっこ { } で要素をはさむ
- セットは「集合」とも呼ばれる

5章 セットの特徴❶ 順番の概念を持たない

セットでは、各データにどのような順番が与えられるかわかりません

5章 セットの特徴❷ 重複の概念を持たない

セットでは同じ値は1つしか存在しません

5章 セットの要素を取り出す方法

セットではインデックスでの取り出しはできません

```
# 1〜10までに含まれる素数のセット
prime_numbers = {2, 3, 5, 7}

# セット全体を表示
print(prime_numbers)

# セットの3番目の要素を表示 ⇒ できないためエラー
print(prime_numbers[2])
```

```
{2, 3, 5, 7}
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-ae90ad45b565> in <module>
      6
      7 # セットの3番目の要素を表示 ⇒ できないためエラー
----> 8 print(prime_numbers[2])

TypeError: 'set' object is not subscriptable
```

SEARCH STACK OVERFLOW

5章 セットの要素を追加・削除する方法

メソッドを使って要素を追加・削除できます

メソッド	説明
add	要素を追加
remove	指定した値の要素を削除

```
セット名.add(追加する要素の値)
セット名.remove(削除する要素の値)
```

5章 セットの追加・削除の例

addとremoveを使ってみましょう

```
prime_numbers = {2, 3, 5, 7}

prime_numbers.add(11)    # 11をセットに追加
prime_numbers.remove(3)  # セットから3を削除
```

5章 セットの追加・削除の結果

0秒

```
# 1～10までに含まれる素数のセット
prime_numbers = {2, 3, 5, 7}

# セット全体を表示
print(prime_numbers)

prime_numbers.add(11) # 11をセットに追加
print(prime_numbers)

prime_numbers.remove(3) # セットから3を削除
print(prime_numbers)
```

```
{2, 3, 5, 7}
{2, 3, 5, 7, 11}
{2, 5, 7, 11}
```

Visual Studio Codeで以下のコードを実行してみましょう

```
country_names = {"アメリカ", "イギリス", "日本", "フランス"}

print(country_names) # セット全体を表示

country_names.add("ドイツ") # 「ドイツ」を追加
print(country_names)

country_names.remove("イギリス") # 「イギリス」を削除
print(country_names)
```

5章 セットを使ってみよう：実行結果

```
country_names = {"アメリカ", "イギリス", "日本", "フランス"}

# セット全体を表示
print(country_names)

country_names.add("ドイツ") # 「ドイツ」をセットに追加
print(country_names)

country_names.remove("イギリス") # セットから「イギリス」を削除
print(country_names)
```

```
{'日本', 'イギリス', 'フランス', 'アメリカ'}
{'イギリス', 'フランス', '日本', 'ドイツ', 'アメリカ'}
{'フランス', '日本', 'ドイツ', 'アメリカ'}
```

5章 リスト・タプル・セットの比較

3種類の配列の違いをまとめます

特徴	リスト	タプル	セット
記号	[ ]	( )	{ }
追加・変更・削除	○	×	追加・削除のみ
順番	あり	あり	なし
重複	あり	あり	なし

5章 まとめ

本章では以下の内容を学習しました

配列とは

- 複数のデータのまとまりのこと
- 配列に入っているそれぞれの値を「要素」という
- 各要素には「0」から順番にインデックスが振られている

配列の種類

- リスト、タプル、セット、ディクショナリの4種類

5章 まとめ（続き）

本章では以下の内容を学習しました

リスト

- 要素の追加・変更・削除が可能、[ ] で作成

タプル

- 要素の追加・変更・削除が不可、( ) で作成

セット

- 順番・重複の概念を持たない、{ } で作成