10章 配列や辞書の繰り返し処理を理解しよう

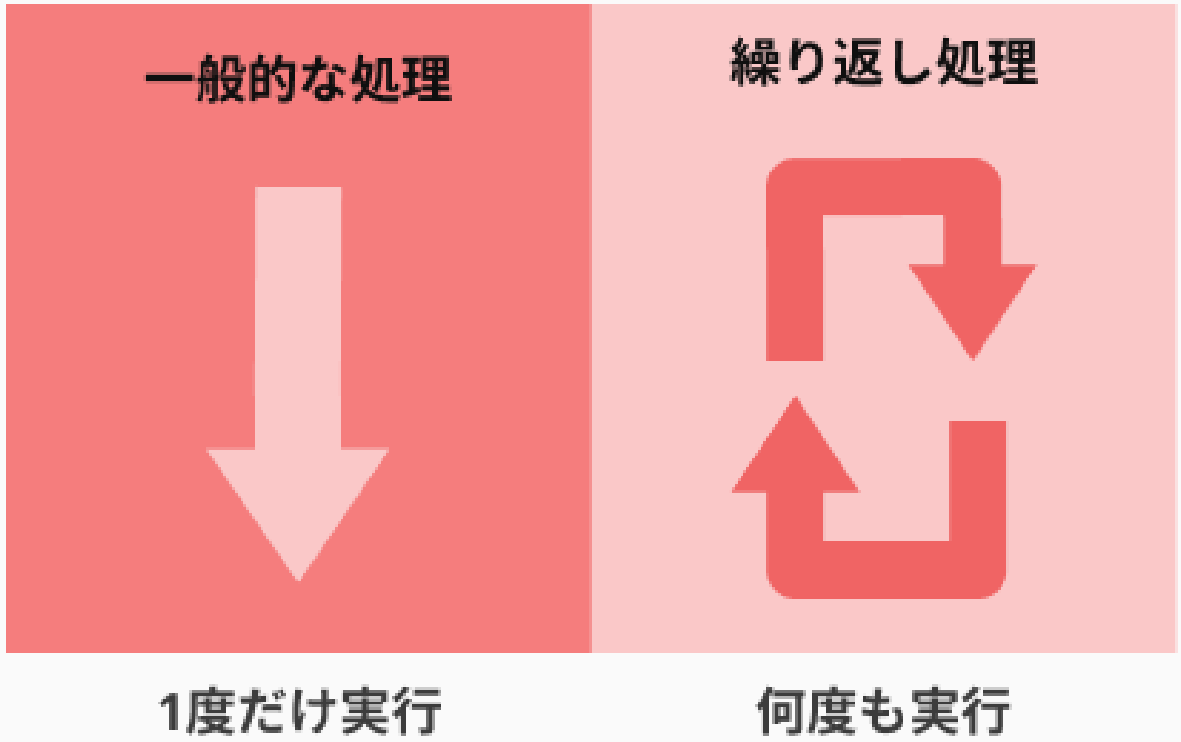
10章 配列や辞書の繰り返し処理を理解しよう

配列や辞書（連想配列）を使った繰り返し処理の書き方を学びます

本章の目標

- 配列や辞書（連想配列）を使った繰り返し処理の書き方を知り、実際にコードを書いてみること

繰り返し処理とは



10章 for文とwhile文の使い分け

それぞれの使うケースを確認しましょう

構文	使うケース	例
for文	繰り返す回数があらかじめわかっている場合。または配列や辞書に対して繰り返し処理を行いたい場合	1〜10までの数字を表示、配列の要素を順番に取り出す
while文	繰り返す回数があらかじめわからない場合	サイコロで6の目が出るまで繰り返す

本章では配列や辞書を利用した繰り返し処理について学びます

配列や辞書も反復可能オブジェクトなので、for文と組み合わせて使えます

- 配列や辞書の各要素を順番に取り出す
- 繰り返し処理の回数＝配列や辞書の要素数となる

配列の要素を1つずつ順番に取り出して処理できます

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

for user_name in user_names:
    print(user_name)
```

- user_name：取り出した要素を代入するループ変数名
- user_names：要素を取り出す配列（反復可能オブジェクト）名

10章 配列を使ったfor文の動作イメージ

ループ変数には配列から取り出した要素が順番に代入されます

周回	ループ変数の値
1周目	user_name ← '侍太郎'
2周目	user_name ← '侍一郎'
3周目	user_name ← '侍二郎'
4周目	user_name ← '侍三郎'
5周目	user_name ← '侍四郎'

繰り返し処理の中でこの変数を使えば、配列の各要素に対してさまざまな処理ができます

配列の要素を順番に出力するコードを書いてみましょう

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

# 配列user_namesの要素を1つずつ順番に出力する
for user_name in user_names:
    print(user_name)
```

10章 配列を使ってfor文を書いてみよう

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

# 配列user_namesの要素を1つずつ順番に出力する
for user_name in user_names:
  print(user_name)
```

```
侍太郎
侍一郎
侍二郎
侍三郎
侍四郎
```

辞書（連想配列）に対してもfor文で繰り返し処理ができます

```
personal_data = {"name":"侍太郎", "age":36, "gender":"男性"}
```

この辞書のキーと値を順番に出力するには？

items()関数を使ってキーと値をセットで取り出します

```
personal_data = {"name": "侍太郎", "age": 36, "gender": "男性"}

for key, value in personal_data.items():
    print(f"{key}は{value}です。")
```

- personal_data：要素を取り出す辞書名
- .items()：辞書からキーと値をセットで取り出す関数
- key：取り出したキーを代入するループ変数名
- value：取り出した値を代入するループ変数名

10章 辞書を使ったfor文の動作イメージ

ループ変数にはキーと値が順番に代入されます

周回	keyの値	valueの値
1周目	"name"	"侍太郎"
2周目	"age"	36
3周目	"gender"	"男性"

繰り返し処理の中でこれらの変数を使えば、辞書の各キー、各値に対してさまざまな処理ができます

10章 値だけを取り出す場合

values()関数を使うとキーを省略して値だけを取り出せます

```
personal_data = {"name": "侍太郎", "age": 36, "gender": "男性"}

for value in personal_data.values():
    print(value)
```

- .values()：辞書から値だけを取り出す関数

辞書のキーと値を順番に出力するコードを書いてみましょう

```
personal_data = {"name": "侍太郎", "age": 36, "gender": "男性"}

# 連想配列personal_dataのキーと値を1つずつ順番に出力する
for key, value in personal_data.items():
    print(f"{key}は{value}です。")

# 連想配列personal_dataの値を1つずつ順番に出力する
for value in personal_data.values():
    print(value)
```

10章 辞書を使ってfor文を書いてみよう

```
personal_data = {"name": "侍太郎", "age": 36, "gender": "男性"}

# 連想配列personal_dataのキーと値を1つずつ順番に出力する
for key, value in personal_data.items():
  print(f"{key}は{value}です。")

# 連想配列personal_dataの値を1つずつ順番に出力する
for value in personal_data.values():
  print(value)
```

```
nameは侍太郎です。
ageは36です。
genderは男性です。
侍太郎
36
男性
```

10章 補足：配列のインデックスを取り出す

enumerate関数を使うと値だけでなくインデックスも取り出せます

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

# 配列user_namesのインデックスと値を1つずつ順番に出力する
for index, value in enumerate(user_names):
    print(f"{index} : {value}")
```

- enumerate関数：配列からインデックスと値をセットで取り出す関数
- インデックス＝配列の各要素に対して「0」から順番に振られる番号

10章 補足：配列のインデックスを取り出す

0秒

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

# 配列user_namesのインデックスと値を1つずつ順番に出力する
for index, value in enumerate(user_names):
  print(f"{index} : {value}")
```

```
0 : 侍太郎
1 : 侍一郎
2 : 侍二郎
3 : 侍三郎
4 : 侍四郎
```

10章 配列・辞書でのbreak文

配列や辞書を利用したfor文でもbreak文が使えます

- 繰り返し処理の途中で強制終了し、ループから抜け出す
- 一般的に条件分岐のif文と組み合わせて使う

10章 break文の使用例

配列から特定の値を見つけたら処理を終了する

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

# 検索したいユーザー名を代入する変数
target = "侍二郎"

for user_name in user_names:
    print(user_name)

    # 変数user_nameと変数targetの値が一致すれば、break文で強制終了
    if (user_name == target):
        print(f"{target}さんが見つかったので、繰り返し処理を強制終了します。")
        break
```

10章 break文の使用例

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

# 検索したいユーザー名を代入する変数
target = "侍二郎"

for user_name in user_names:
  print(user_name)

  # 変数user_nameと変数targetの値が一致すれば、break文で繰り返し処理を強制終了する
  if (user_name == target):
    print(f"{target}さんが見つかったので、繰り返し処理を強制終了します。")
    break
```

```
侍太郎
侍一郎
侍二郎
侍二郎さんが見つかったので、繰り返し処理を強制終了します。
```

10章 配列・辞書でのcontinue文

配列や辞書を利用したfor文でもcontinue文が使えます

- 繰り返し処理の途中で中断し、次のループに進む
- 一般的に条件分岐のif文と組み合わせて使う

10章 continue文の使用例

特定の条件を満たさない要素をスキップする

```
score = {
    "国語": 80,
    "数学": 55,
    "理科": 70,
    "社会": 85,
    "英語": 60
}

print("合格した科目は以下のとおりです。")

for key, value in score.items():
    # 点数が70より小さければ、continue文で次のループに進む
    if (value < 70):
        continue
    print(f"{key} : {value}点")
```

10章 continue文の使用例

```
score = {
    "国語": 80,
    "数学": 55,
    "理科": 70,
    "社会": 85,
    "英語": 60
}

print("合格した科目は以下のとおりです。")

for key, value in score.items():
  # 変数valueの値(点数)が70より小さければ、キー(科目名)と値(点数)を出力せずにcontinue文で次のループに進む
  if (value < 70):
    continue

  print(f"{key} : {value}点")
```

```
合格した科目は以下のとおりです。
国語 : 80点
理科 : 70点
社会 : 85点
```

10章 まとめ

本章では以下の内容を学習しました

配列・辞書の繰り返し処理

- for文を使えば、配列や辞書に対して繰り返し処理を行うことができる
- 「繰り返し処理の回数＝配列や辞書の要素数」となる

辞書の便利な関数

- items()：キーと値をセットで取り出す
- values()：値だけを取り出す

10章 まとめ（続き）

本章では以下の内容を学習しました

enumerate関数

- 配列のインデックスと値をセットで取り出せる

break文とcontinue文

- 配列や辞書を利用したfor文でも使える
- break文：繰り返し処理の途中で強制終了し、ループから抜け出す
- continue文：繰り返し処理の途中で中断し、次のループに進む
- 一般的に条件分岐のif文と組み合わせて使う