8章 繰り返し処理のfor文を理解しよう

8章 繰り返し処理のfor文を理解しよう

繰り返し処理とは何か、概要を学び、実際にコードを書いてみます

本章の目標

- 繰り返し処理とは何か、概要をつかむこと
- for文の書き方を知り、実際にコードを書いてみること
- break文とcontinue文の使い方を知ること

8章 なぜ繰り返し処理が必要なのか

1〜10までの整数を順番に出力したい場合を考えてみましょう

```
print("1")
print("2")
print("3")
print("4")
print("5")
# ... 続く
```

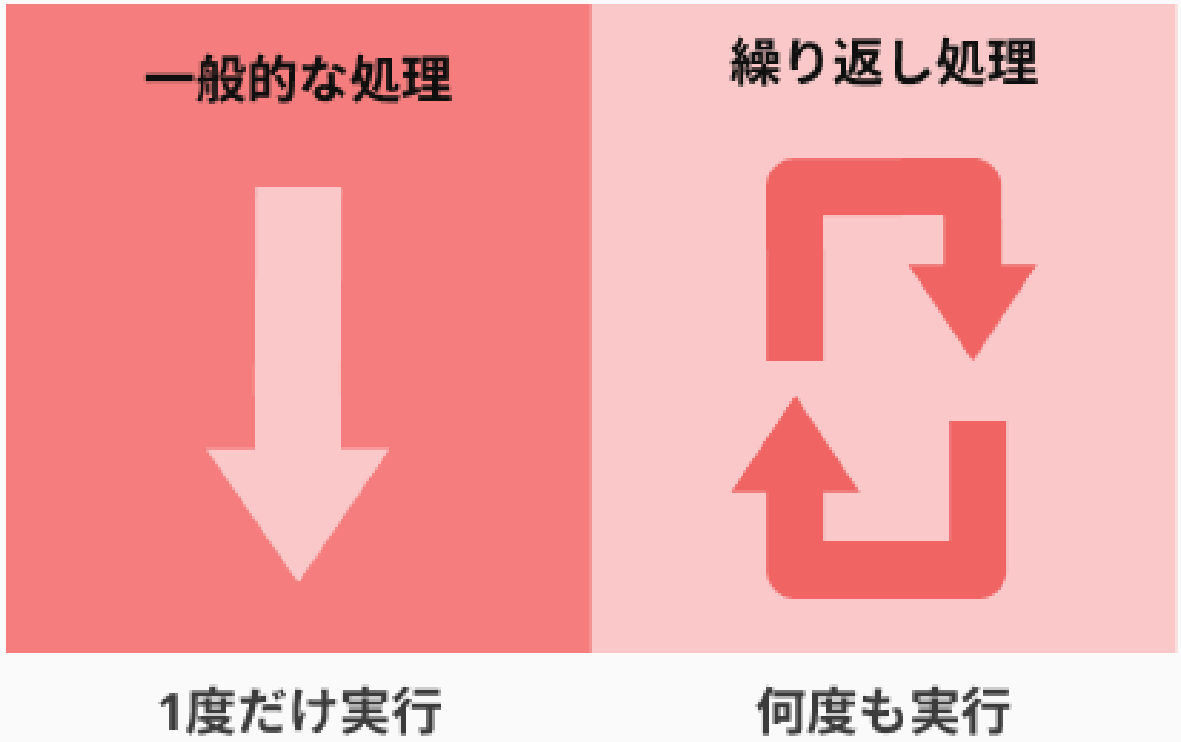
同じようなコードを何度も書くのは手間がかかり、コードが無駄に長くなる

コードをスッキリさせられます

```
for i in range(1, 11):
    print(i)
```

- たった2行で1〜10までの整数を出力できる
- コードが短く、読みやすくなる

繰り返し処理とは



8章 Pythonの繰り返し処理

代表的な繰り返し処理はfor文とwhile文の2つです

構文	使うケース
for文	繰り返す回数があらかじめわかっている場合
while文	繰り返す回数があらかじめわからない場合

本章ではまず for文 について学びます

それぞれの具体例

for文の例

- 1〜10までの数字を順番に表示する
- 配列の中の都道府県名を順番に取り出す

while文の例

- サイコロで6の目が出るまで繰り返す
- 条件を満たすまで処理を続ける

8章 for文の書き方

for文は「決まった回数」同じ処理を繰り返し行いたいときに使います

```
for ループ変数 in 反復可能オブジェクト:
    ブロック（繰り返し行いたい処理）
```

- 反復可能オブジェクトの後には `:（コロン）` を記載
- 繰り返し行われる処理はインデントを下げる

8章 反復可能オブジェクトとは

複数の情報を持ち、繰り返し処理ができるオブジェクトのことです

- 配列（リスト、タプル、セット）
- 辞書（ディクショナリ）
- 文字列
- range型

これらから情報を1つずつ取り出して処理を繰り返す

8章 for文の処理の流れ

反復可能オブジェクトから情報を取り出し、ループ変数に格納して処理します

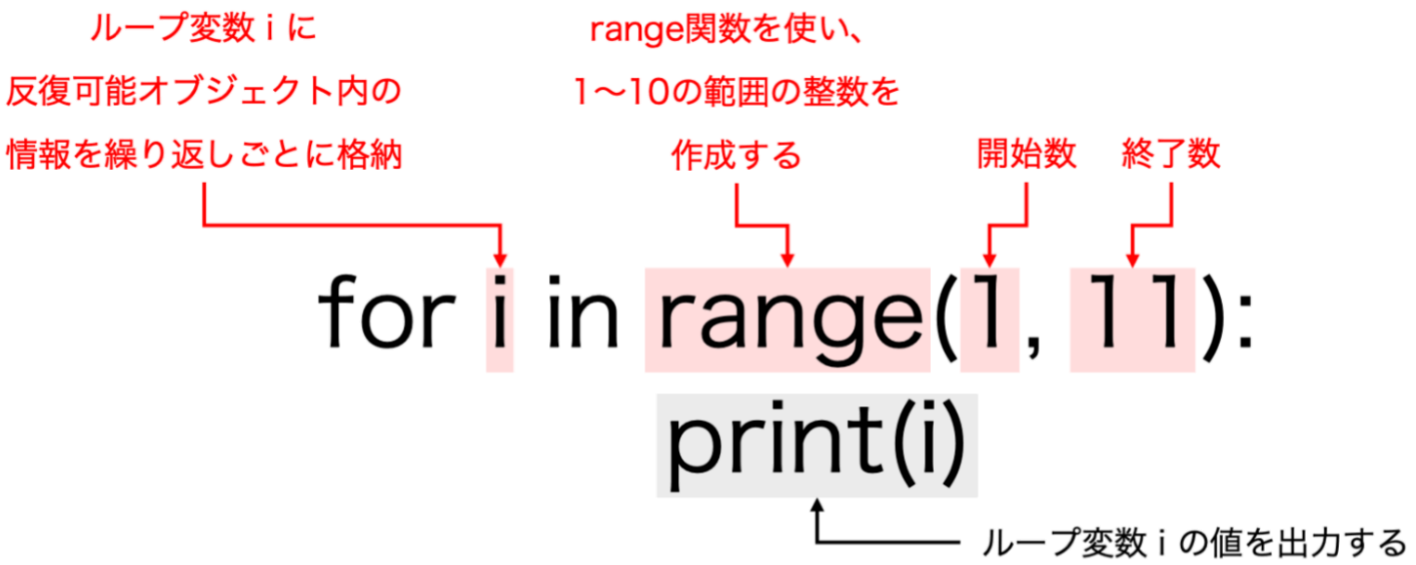
1. 反復可能オブジェクトから情報を取り出す
2. 取り出した情報をループ変数に格納
3. ブロック内で処理する
4. 1〜3を反復可能オブジェクトの中身の分だけ繰り返す

1〜10までの整数を順番に出力するコード

```
for i in range(1, 11):
    print(i)
```

- `i`：ループ変数
- `range(1, 11)`：1から10までを含むrange型を生成
- `print(i)`：ループ変数の値を出力

8章 for文の使用例



range関数で1～10を含む反復可能オブジェクトを生成。
繰り返しごとにループ変数に値が格納され、出力される。
反復可能オブジェクトの中身が全て処理されると繰り返し処理が終了。

8章 range関数とは

開始数から終了数までの連続した数値を要素として持つrange型を生成します

```
range(終了数)
range(開始数, 終了数)
range(開始数, 終了数, ステップ)
```

- 開始数を指定しない場合は 0 から開始
- 終了数は要素に含まれない 点に注意
- ステップは間隔（省略すると1ずつ増える）

8章 range関数の例

様々な書き方とその結果

```
range(3)
# 要素：0, 1, 2

range(6, 9)
# 要素：6, 7, 8

range(5, 11, 2)
# 要素：5, 7, 9（2つおきに増加）
```

8章 for文を書いてみよう

Visual Studio Codeで実行してみましょう

```
for i in range(1, 11):
    print(i)
```

8章 for文を書いてみよう

実行結果

0秒

```
for i in range(1, 11):
  print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

8章 break文とは

繰り返し処理の途中で強制終了し、ループから抜け出す命令です

```
for i in range(1, 11):
    print(i)

    # ループ変数の値が5であれば、繰り返し処理を強制終了
    if i == 5:
        break
```

- 一般的に条件分岐の if文 と組み合わせて使う

20が出たら繰り返し処理を終了するプログラム

```
import random

for i in range(1, 11):
    num = random.randint(1, 20)
    print(f"{i}回目の結果は{num}です。")

    if num == 20:
        print("20が出たので繰り返し処理を強制終了します。")
        break
```

8章 break文を使ってみよう

```python
#ランダムな整数を利用するため、記述が必要です。
import random

for i in range(1, 11):
  #変数numに1~20までのランダムな整数を代入する
  num = random.randint(1,20)

  print(f"{i}回目の結果は{num}です。")

  #変数numの値が20であれば、break文で繰り返し処理を強制終了する
  if num == 20:
    print("20が出たので繰り返し処理を強制終了します。")
    break
```

```
1回目の結果は10です。
2回目の結果は18です。
3回目の結果は15です。
4回目の結果は11です。
5回目の結果は2です。
6回目の結果は20です。
20が出たので繰り返し処理を強制終了します。
```

8章 break文を使ってみよう

```python
#ランダムな整数を利用するため、記述が必要です。
import random

for i in range(1, 11):
  #変数numに1~20までのランダムな整数を代入する
  num = random.randint(1,20)

  print(f"{i}回目の結果は{num}です。")

  #変数numの値が20であれば、break文で繰り返し処理を強制終了する
  if num == 20:
    print("20が出たので繰り返し処理を強制終了します。")
    break
```

```
1回目の結果は13です。
2回目の結果は7です。
3回目の結果は16です。
4回目の結果は3です。
5回目の結果は1です。
6回目の結果は11です。
7回目の結果は7です。
8回目の結果は16です。
9回目の結果は3です。
10回目の結果は1です。
```

8章 continue文とは

繰り返し処理の途中で中断し、次のループに進む命令です

break文	continue文
ループを完全に抜け出す	今回のループをスキップして次へ進む

8章 continue文の使用例

1〜10のうち偶数のみを出力するコード

```
for i in range(1, 11):
    # 奇数（2で割った余りが1）であれば次のループへ
    if i % 2 == 1:
        continue

    print(i)
```

奇数のときはprint(i)が実行されずにスキップされる

8章 continue文の使用例

```
for i in range(1, 11):
 # ループ変数iの値が奇数(2で割った余りが1)であれば、値を出力せずにcontinue文で次のループに進む
 if i % 2 == 1:
   continue

 print(i)
```

```
2
4
6
8
10
```

8章 break文とcontinue文の違い

処理の流れを比較しましょう

文	動作	使用例
break	ループを完全に終了	条件を満たしたら処理を終える
continue	今回のループをスキップ	特定の条件のときだけ処理をスキップ

8章 まとめ

本章では以下の内容を学習しました

繰り返し処理とは

- 「決まった回数」または「条件を満たしている間」同じ処理を繰り返すこと
- Pythonではfor文とwhile文がある

for文

- 繰り返す回数があらかじめわかっている場合に使う
- range関数で連続した数値を生成できる

8章 まとめ（続き）

本章では以下の内容を学習しました

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

どちらも条件分岐のif文と組み合わせて使うことが多い