9章 繰り返し処理のwhile文を理解しよう

9章 繰り返し処理のwhile文を理解しよう

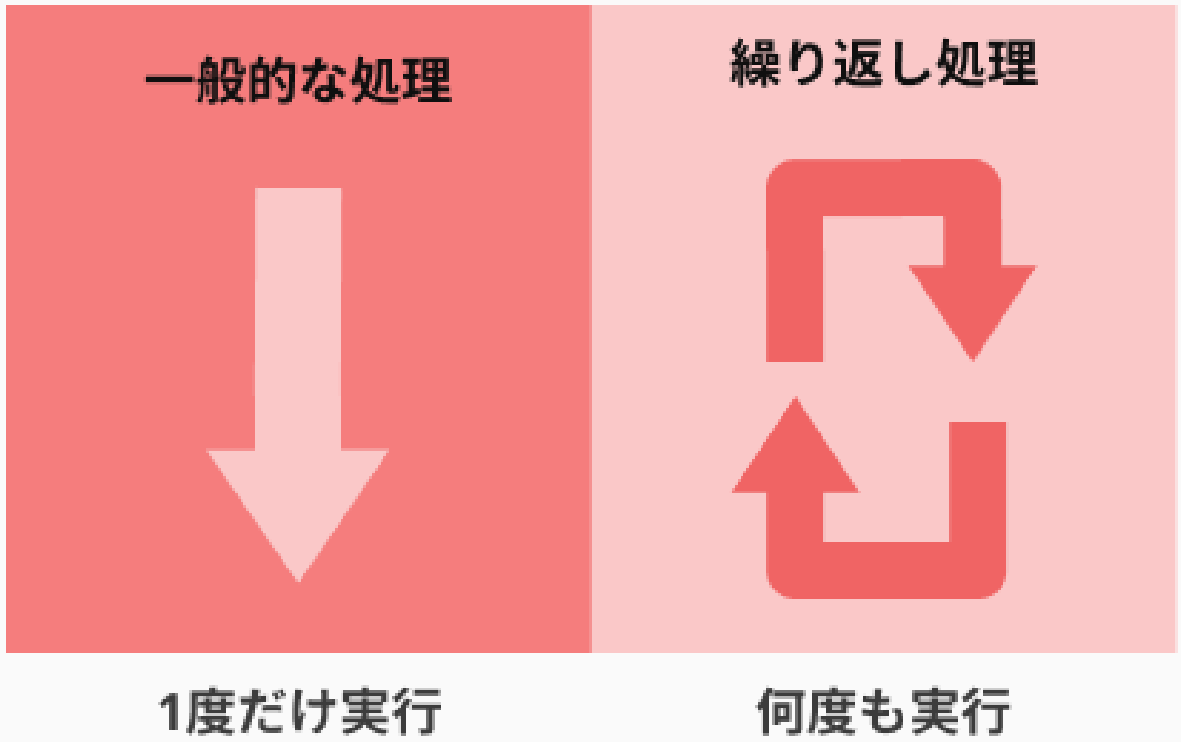
while文の書き方を知り、実際にコードを書いてみます

本章の目標

- while文の書き方を知り、実際にコードを書いてみること
- 無限ループの危険性を理解すること

繰り

繰り返し処理とは



9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	1〜10までの数字を表示
while文	繰り返す回数があらかじめわからない場合	6の目が出るまで繰り返す

本章では while文 について学びます

9章 while文とは

「条件を満たしている間」同じ処理を繰り返し行える構文です

```
while 条件式:
    条件を満たしている間、繰り返す処理
```

- if文やfor文と同じように 条件式 を使う
- 条件式がTrueの間、処理を繰り返す

9章 比較演算子の復習

条件式に使う比較演算子を確認しましょう

比較演算子	処理の内容
`==`	2つの値が等しい場合はTrueを返す
`!=`	2つの値が等しくない場合はTrueを返す
`>` `>=`	左辺が右辺より大きい / 以上の場合はTrue
`<` `<=`	左辺が右辺より小さい / 以下の場合はTrue

9章 while文の使用例

変数numの値が0以外である間、処理を繰り返す

```
import random

# 変数numに0〜4までのランダムな整数を代入
num = random.randint(0, 4)

# 変数numの値が0以外である間、繰り返す
while num != 0:
    num = random.randint(0, 4)
    print(num)
```

9章 while文の使用例

条件式の解説

```
while num != 0:
```

- `num != 0` が条件式
- 変数numの値が0以外のときにTrueを返す
- Trueの間、ブロック内の処理が繰り返される

Visual Studio Codeで実行してみましょう

```
import random

num = random.randint(0, 4)
print(f"最初の値は{num}です。")

while num != 0:
    num = random.randint(0, 4)
    print(f"現在の値は{num}です。")
```

9章 while文を書いてみよう

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

# 変数numに0~4までのランダムな整数を代入する
num = random.randint(0, 4)

# 変数numの最初の値を出力する(確認用)
print(f"最初の値は{num}です。")

# 変数numの値が0以外である間、変数numの値を出力し続ける
while num != 0:
  # 変数numに0~4までのランダムな整数を代入する
  num = random.randint(0, 4)

  # 次の条件式で比較される、変数numの現在の値を出力する
  print(f"現在の値は{num}です。")
```

```
最初の値は2です。
現在の値は1です。
現在の値は2です。
現在の値は1です。
現在の値は3です。
現在の値は4です。
現在の値は1です。
現在の値は4です。
現在の値は0です。
```

while文で注意すべき最も重要なポイントです

```
num = 5

# 条件式が常にTrueを返すので、無限ループになる
while num == 5:
    print(num)
```

- 変数numの値は常に5なので、条件式は常にTrue
- 処理が永遠に終わらない 無限ループ が発生

9章 無限ループを防ぐには

「条件式がFalseを返す可能性はあるか」を必ず確認

無限ループになる

```
num = 5
while num == 5:
    print(num)
```

numが変化しない

正常に終了する

```
num = 5
while num == 5:
    num = random.randint(0, 10)
    print(num)
```

numが変化する可能性あり

9章 while文でのbreak文

for文と同様にbreak文が使えます

```
import random

num = random.randint(0, 4)
i = 1

while num != 0:
    num = random.randint(0, 4)

    if i == 5:
        print("5回目なので強制終了します。")
        break

    print(f"現在の値は{num}です。")
    i = i + 1
```

9章 while文でのbreak文

```
    if i == 5:
        print("5回目なので繰り返し処理を強制終了します。")
        break

    # 次の条件式で比較される、変数numの現在の値を出力する
    print(f"現在の値は{num}です。")

    # カウンタ変数のの値を1増やす
    i = i + 1
```

```
最初の値は4です。
現在の値は2です。
現在の値は3です。
現在の値は2です。
現在の値は3です。
5回目なので繰り返し処理を強制終了します。
```

9章 while文でのbreak文

```
    # カウンタ変数iの値が5であれば、break文で繰り返し処理を矯正終了する
    if i == 5:
        print("5回目なので繰り返し処理を強制終了します。")
        break

    # 次の条件式で比較される、変数numの現在の値を出力する
    print(f"現在の値は{num}です。")

    # カウンタ変数のの値を1増やす
    i = i + 1
```

```
最初の値は1です。
現在の値は2です。
現在の値は0です。
```

9章 while文でのcontinue文

for文と同様にcontinue文も使えます

```
import random

sum = 0

while sum < 20:
    num = random.randint(1, 10)
    print(f"{num}が出ました。")

    if num % 2 == 0:
        print("偶数なので加算しません。")
        continue

    sum = sum + num
    print(f"現在の合計は{sum}です。")
```

9章 while文でのcontinue文

```
    print("偶数なので加算しません。")
    continue

# 変数sumに変数numの値を加算する
sum = sum + num
print(f"現在の合計は{sum}です。")
```

```
9が出ました。
現在の合計は9です。
2が出ました。
偶数なので加算しません。
7が出ました。
現在の合計は16です。
7が出ました。
現在の合計は23です。
```

9章 まとめ

本章では以下の内容を学習しました

while文

- 「条件を満たしている間」同じ処理を繰り返す
- 繰り返す回数があらかじめわからない場合に使う

無限ループ

- 条件式が常にTrueだと処理が永遠に終わらない
- 「条件式がFalseを返す可能性はあるか」を必ず確認

9章 まとめ（続き）

本章では以下の内容を学習しました

break文とcontinue文

- while文でもfor文と同様に使える
- break文: ループを強制終了
- continue文: 今回のループをスキップして次へ

while文は条件に基づく繰り返しに便利ですが、無限ループに注意しましょう