# 条件分岐の応用

Python プログラミング応用

if-elif-else / 三項演算子 / ネスト / 条件の整理

- if-elif-else
- 三項演算子
- ネスト
- 条件の整理

# if-elif-else は「複数の選択肢」から 1つ選ぶ

if-else は2択。if-elif-else なら3つ以上の選択肢に対応できる

80点以上？ → 優

60点以上？ → 良

40点以上？ → 可

それ以外 → 不可

例: 成績評価

CODE

```
score = 75

if score >= 80:
    grade = "優"
elif score >= 60:
    grade = "良"
elif score >= 40:
    grade = "可"
else:
    grade = "不可"

→ 評価: 良
```

上から順にチェック → 最初にTrueになった処理だけ実行

# ⇄ elif と if を間違えると結果が変わる

elif は1つ選んで終了 / if は全部チェック

## elif を使う

```
score = 85

if score >= 80:
    print("優秀です")
elif score >= 60:
    print("合格です")

→ 優秀です
```

1つだけ表示される（期待通り）

## if を2つ使う

```
score = 85

if score >= 80:
    print("優秀です")
if score >= 60:   # elif ではない！
    print("合格です")

→ 優秀です
→ 合格です   ← 意図しない！
```

2つとも表示されてしまう

# 実用例: 年齢別の料金計算

elif で年齢区分を順番に判定する

| | | |
|---|---|---|
| **未就学児** | 6歳未満 | **0円** |
| **小学生** | 6〜11歳 | **500円** |
| **中高生** | 12〜17歳 | **800円** |
| **一般** | 18〜64歳 | **1,200円** |
| **シニア** | 65歳以上 | **800円** |

CODE

```
age = int(input())

if age < 6:
    price = 0       # 未就学児
elif age < 12:
    price = 500     # 小学生
elif age < 18:
    price = 800     # 中高生
elif age < 65:
    price = 1200    # 一般
else:
    price = 800     # シニア
```

# 三項演算子：シンプルな if-else を1行で

変数に値を代入するだけの場面で使える

構文：　変数 = True時の値 if 条件式 else False時の値

通常の if-else（4行）

```
temperature = 28

if temperature >= 25:
    message = "暑いです"
else:
    message = "涼しいです"
```

三項演算子（1行）

```
temperature = 28

message = "暑いです"
  if temperature >= 25
  else "涼しいです"
```

# 三項演算子はシンプルな場面だけ

便利だが、複雑にすると読みにくくなる

## 三項演算子が向いている

```
# 偶数か奇数か
result = "偶数" if n % 2 == 0 else "奇数"

# 合否判定
result = "合格" if score >= 60 else "不合格"

# 在庫チェック
status = "在庫あり" if stock > 0 else "在庫切れ"
```

- 条件が1つだけ
- 処理が代入1つだけ
- 内容がシンプル

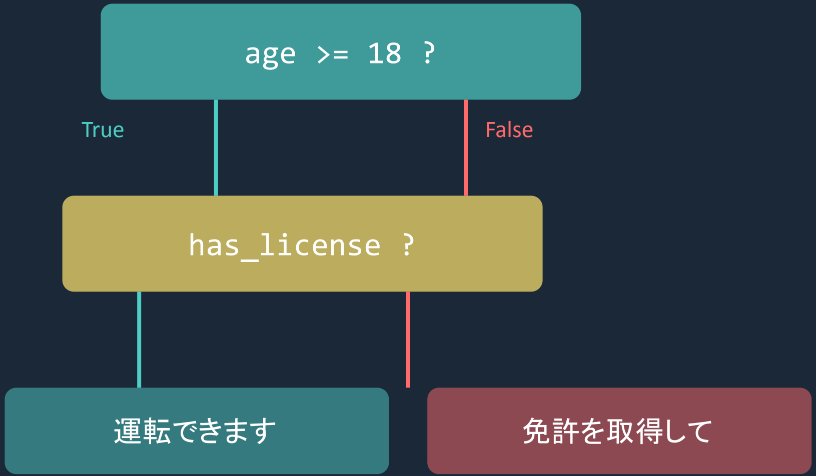
## if-elseを使うべき

```
# 複雑すぎて読みにくい！
price = 1000
  if (age >= 18
      and has_ticket)
     or is_vip
  else 500
    if age >= 12
  else 0
```

- 条件が複雑（and/or複数）
- 各分岐で複数の処理
- elifが必要な場合

# ネスト: 条件の中にさらに条件を書く

大きな分類 → 細かい分類と段階的に判定する




CODE

```
age = 25
has_license = True

if age >= 18:
    # さらに免許の有無をチェック
    if has_license:
        print("運転できます")
    else:
        print("免許を取得して")
else:
    print("18歳未満は不可")
```

インデントの深さに注意: 外側のif = 4スペース、内側のif = 8スペース

# 深いネストは読みにくい → 整理しよう

if-elifで書き直すとフラットで読みやすくなる

**ネストが深い**

```
if age >= 18:
    if has_ticket:
        if has_id:
            print("入場OK")
        else:
            print("ID必要")
    else:
        print("チケット必要")
else:
    print("18歳未満は不可")
```

**if-elifで整理**

```
if age < 18:
    print("18歳未満は不可")
elif not has_ticket:
    print("チケット必要")
elif not has_id:
    print("ID必要")
else:
    print("入場OK")
```

ネスト3段以上 → if-elifで書き直せないか検討しよう

# 複雑な条件は変数に分けて名前をつける

条件の意味がわかりやすくなり、デバッグも簡単になる

条件をそのまま書いた場合

```
if (age >= 20 and age <= 65) and income >= 3000000 and credit_score >= 700:
```

条件を変数に分けた場合

```
# 各条件に名前をつける
is_valid_age = (age >= 20 and age <= 65)
has_sufficient_income = (income >= 3000000)
has_good_credit = (credit_score >= 700)

# 読みやすい！
if is_valid_age and has_sufficient_income and has_good_credit:
    print("審査OK")
```

変数名が条件の説明になり、どの条件を満たさないか個別にチェックもできる

# まとめ：条件分岐（応用編）

## if-elif-else

- 3つ以上の選択肢から1つを選ぶ
- 上から順番にチェック
- 最初にTrueの処理だけ実行

## 三項演算子

- シンプルなif-elseを1行で
- 値 if 条件 else 値 の形式
- 複雑な条件には使わない

## ネスト

- 条件の中にさらに条件
- インデントの深さに注意
- 深すぎたら整理する

## 条件の整理

- 条件を変数に分けて命名
- コードが読みやすくなる
- エラー箇所も特定しやすい

次は「繰り返し処理（ループ）」を学びます