論理演算子

整数値を読み込んで、ゼロなのか／1桁の値なのか／それ以上の桁数の値なのかを判定して表示するプログラムを作りましょう。それが **List 3-11** に示すプログラムです。

List 3-11 chap03/list0311.py

```
# 整数値の桁数（ゼロ／1桁／2桁以上）を判定

n = int(input('整数値：'))

if n == 0:                              # ゼロ
    print('その値はゼロです。')
elif n >= -9 and n <= 9:                # 1桁
    print('その値は1桁です。')
else:                                   # 2桁以上
    print('その値は2桁以上です。')
```

実行例

```
① 整数値：0↵
その値はゼロです。
② 整数値：5↵
その値は1桁です。
③ 整数値：-25↵
その値は2桁以上です。
```

論理積演算子 and

読み込んだ値が1桁かどうかの判定を行うのが、網かけ部です。

and 演算子を使った式“x and y”は、日本語の《xかつy》に相当します（**Fig.3-4** a）。そのため、nが-9以上でかつ9以下のときに、『その値は1桁です。』と表示されます。

▶ 網かけ部の判定が正しく行えるのは、and演算子の優先度が、値比較演算子>=と<=よりも低いからです。優先度を含めた全演算子の一覧は、**Table 3-5**（p.76）に示しています。

なお、nが0のときは、『その値はゼロです。』の表示後にif文が終了するため、『その値は1桁です。』と表示されるのは、nが-9, -8, …, -2, -1, 1, 2, …, 8, 9のいずれかのときです。

論理和演算子 or

今度は、読み込んだ値が2桁以上かどうかを判定して表示するプログラムを作ります。それが、**List 3-12** のプログラムです。

List 3-12 chap03/list0312.py

```
# 読み込んだ整数値が2桁以上かどうかを判定（その1）

n = int(input('整数値：'))

if n <= -10 or n >= 10:                 # 2桁以上
    print('その値は2桁以上です。')
else:                                   # 2桁未満
    print('その値は2桁未満です。')
```

実行例

```
① 整数値：-15↵
その値は2桁以上です。
② 整数値：5↵
その値は2桁未満です。
```

or 演算子を使った式“x or y”は、日本語の《xまたはy》に相当します（図b）。ただし、“いずれか一方のみ”ではなく、“**いずれか一方でも**”というニュアンスです。

そのため、変数nの値が-10以下または10以上の値であるときにのみ、『その値は2桁以上です。』と表示されます。

論理否定演算子 not

このプログラムの判定を反転する（裏返す）ことを考えましょう。論理値の反転（図c）を行う **not 演算子**使って書きかえたのが、**List 3-13** のプログラムです。

List 3-13 chap03/list0313.py

```
# 読み込んだ整数値が２桁以上かどうかを判定（その２）

n = int(input('整数値：'))

if not (n <= -10 or n >= 10):          # ２桁未満
    print('その値は２桁未満です。')
else:                                  # ２桁以上
    print('その値は２桁以上です。')
```

List 3-12 と順序が逆

実行例

1 整数値：-15↵
その値は２桁以上です。

2 整数値：5↵
その値は２桁未満です。

本プログラムの網かけ部は、前のプログラムの網かけ部の否定であり、変数 n の値が -10 以下または 10 以上の値でないときに真と評価され『その値は2桁未満です。』と表示されます。

ここで学習した、三つの演算子の総称は、**論理演算子**（*bool operator*）です。

実は、**Fig.3-4** は一般的な論理積、論理和、論理否定の演算結果です。極めて紛らわしいことに、Python の **and** 演算子と **or** 演算子は、この表とは異なる働きをします。

要注意：Python の and 演算子と or 演算子の働きとはまったく異なる

a 論理積（AND）

x	y	x AND y
真	真	真
真	偽	偽
偽	真	偽
偽	偽	偽

両方とも真であれば真

b 論理和（OR）

x	y	x OR y
真	真	真
真	偽	真
偽	真	真
偽	偽	偽

一方でも真であれば真

c 論理否定（NOT）

x	NOT x
真	偽
偽	真

偽であれば真

Fig.3-4 一般的な論理積と論理和と論理否定の真理値表

Table 3-2 に示すのが、Python の論理演算子の概要です。**True** や **False** を生成するのは、**not** 演算子だけです。論理演算子を正確に理解していきましょう。

Table 3-2 論理演算子

x and y	x を評価して偽であれば、その値を生成。そうでなければ y を評価して、その値を生成。
x or y	x を評価して真であれば、その値を生成。そうでなければ y を評価して、その値を生成。
not x	x が真であれば False を、そうでなければ True を生成。

▶ 優先度は、高いほうから順に **not** 演算子、**and** 演算子、**or** 演算子です。