## リストによる成績処理

本章の冒頭では、**List 7-1**（p.162）を例にして、単一の変数の寄せ集めで学生の点数を表すプログラムの限界や、リストの必要性などを学習しました。

リストを使って書きかえたプログラムが、**List 7-7** です。なお、人数を５人固定ではなく、キーボードから読み込むように拡張しています。

**List 7-7** chap07/list0707.py

```
# 点数を読み込んで合計点・平均点を表示（その１）

print('合計点と平均点を求めます。')
number = int(input('学生の人数：'))

tensu = [None] * number                                    ←1

for i in range(number):                                    ←2
    tensu[i] = int(input('{}番の点数：'.format(i + 1)))

total = 0                                                  ←3
for i in range(number):
    total += tensu[i]

print('合計は{}点です。'.format(total))
print('平均は{}点です。'.format(total / number))
```

**実行例**

```
合計点と平均点を求めます。
学生の人数：5↵
1番の点数：32↵
2番の点数：68↵
3番の点数：72↵
4番の点数：54↵
5番の点数：92↵
合計は318点です。
平均は63.6点です。
```

まず最初に、学生の人数を変数 *number* に読み込みます。

1では、要素数が *number* で、全要素が None のリストを生成します（p.167）。

2では、カウンタ用変数 *i* の値を 0 から *number* - 1 までインクリメントしながら、*tensu*[*i*] に点数を読み込みます。

▶ 入力を促す際は、たとえば *i* が 0 のときに「1 番の点数：」と表示しますので、*i* ではなく *i* + 1 の値を出力します。

3では、合計を求めます。まず *total* を 0 にします。その後、カウンタ用変数 *i* の値を 0 から *number* - 1 までインクリメントしながら、*tensu*[*i*] の値を *total* に加えます。

**人数が可変となり、各学生の点数をインデックスでアクセスできるようになったため、柔軟性が大幅にアップしました。しかもプログラムが短くなっています。**

＊

点数を読み込む2の for 文を enumerate 関数を使って書きかえたのが、**List 7-8** です。このプログラムを実行すると、書きかえた2ではなくて、3の実行時にエラーが発生します。

**List 7-8** chap07/list0708.py

```
# 点数を読み込んで合計点・平均点を表示（その２：エラー）

for i, point in enumerate(tensu):
    point = int(input('{}番の点数：'.format(i + 1)))
```

エラーが発生するのは、*i* と *point* に取り出される (*i*, *point*) が、リストから取り出されて新しく作られたペアだからです。*point* は *tensu*[0] や *tensu*[1] のコピーですから、そこに値を書き込んでも、オリジナルの *tensu*[0] や *tensu*[1] は、None のままです。

そのため、続く3では、*tensu*[0] から取り出された None を整数 *total* に加えようとします。これが、エラーとなる原因です。

▶ この部分がよく理解できなければ、次章の p.210 の学習が終了してから、戻ってくるようにします。

*

一方、合計を求める3では、リストの要素の値を読み取るだけで、書き込みません（左辺には置きません）。そのため、この for 文は、enumerate 関数を使って実現できます。

**List 7-9** に示すのが、そのプログラムです。



**List 7-9** chap07/list0709.py

```
# 点数を読み込んで合計点・平均点を表示（その３：enumerateで走査して合計を求める）

total = 0
for i, point in enumerate(tensu):
    total += point                          # iの値は不要
```

ただし、インデックスの値がなくても、全要素の合計は求められますので、enumerate 関数は使わずに、単なる in を使うべきです。

そのように書きかえたプログラムが **List 7-10** です。

**List 7-10** chap07/list0710.py

```
# 点数を読み込んで合計点・平均点を表示（その４：inで走査して合計を求める）

total = 0
for point in tensu:
    total += point
```

これで、コードがすっきりしました。しかし、まだ改良の余地があります。組込み関数である sum 関数にリスト（やタプルなど）を与えると、その全要素の値の合計が求められます。

そのことを利用すると、プログラムは、**List 7-11** となります。

**List 7-11** chap07/list0711.py

```
# 点数を読み込んで合計点・平均点を表示（その５：sum関数を使って合計を求める）

total = sum(tensu)
```

▶ （リストを含めた、各種の）イテラブルオブジェクトを受け取った sum 関数は、その総和を求めます。**List 4-3**（p.92）では、1 から *n* までの和を while 文で求める方法を学習しましたが、以下の関数呼び出し式で求められます。

```
sum(range(1, n + 1))            # 1からnまでの和を求める
```