CHAPTER5　「ブローの判定」の補足

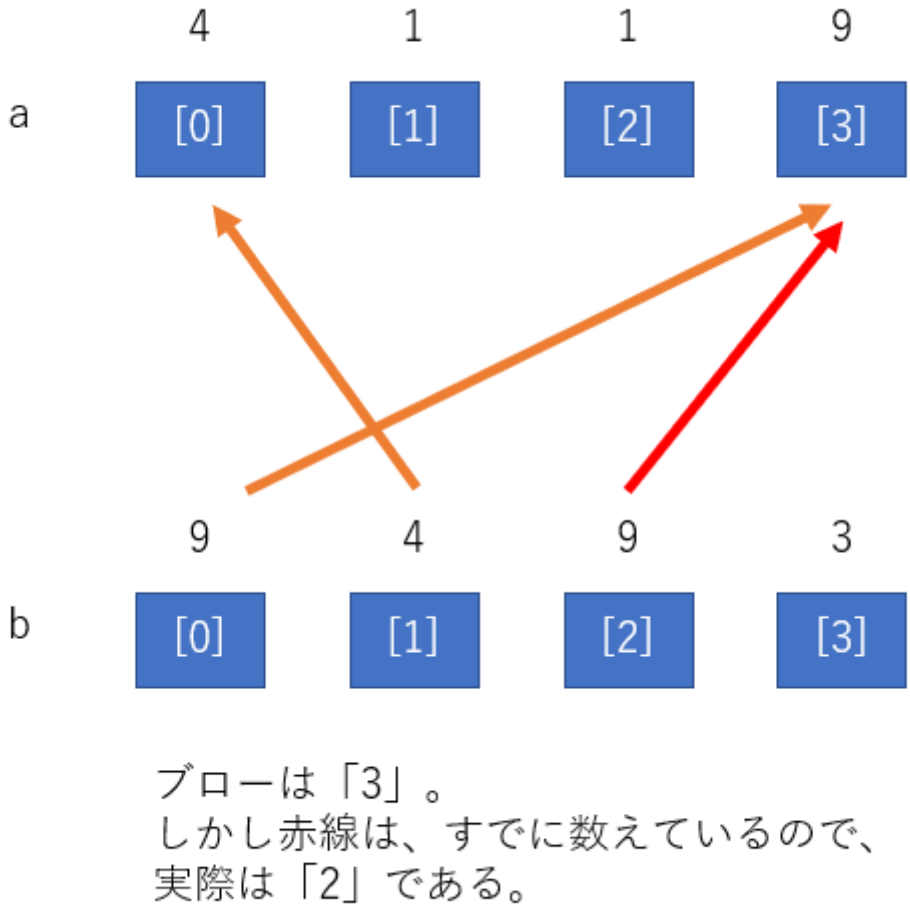
第5章では、「ヒット&ブロー」のゲームを作っています。この例では、次の判定としています。

- ヒットは「位置も数も同じ」
- ブローは「位置は正しくないが、その数字が含まれている」

ヒットについては異論がないかと思いますが、ブローについては、「重複して数えるかどうか」という点が議論の対象となります。

本文中では、話を簡単にするため、ブローについては、「ヒットしたものは除外するけれども、ブローの重複は排除しない」としています（p.149の図5-5-3）。

しかしこれは、正確な判定ではありません。たとえば、aが「4119」、bが「9493」のケースを考えます。この場合、ブローは「9」と「4」なので「2」とカウントすべきですが、本文中のプログラムでは、「9」「4」「9」のように、「9」の重複を無視して「3」とカウントしています。



これを解決する方法は、いくつかありますが、比較の際、「bを比較する際、すでに比較しているならスキップする」のが、簡便な方法です。

example05-05-01.py（p.151）では、次のようにして、ブローを判定しています（37～43行目）。

```
# ブローを判定
blow = 0
for j in range(4):
  for i in range(4):
    if (int(b[j]) == a[i]) and (a[i] != int(b[i])) and (a[j] != int(b[j])):
      blow = blow + 1
      break
```

Pythonでは、「in」という演算子を使うと、リストのなかに値が含まれているかどうかを確認できます。

```
値 in リスト
```

またリストに対して、

```
リスト[開始:終了]
```

という表記を使うと、「開始」から「終了」まで（開始は含まれるが終了は含まれない）の部分的なリストを取得できます。たとえば、bが、

```
b = [9, 4, 9, 3]
```

である場合、

```
b[0:2]
```

と記述すると、

```
[9, 4]
```

が得られます。このようにリストから部分的なリストを取り出す操作を「スライス」と言います。

そしてループ中では「continue」という文を記述することで、以降の処理をスキップして、次のループに進むことができます。

こうした処理を使って、先ほどのブローの判定を次のように変更すると、図の赤線の部分（すでに検索した部分）をスキップして数えられます。

```
# ブローを判定
blow = 0
for j in range(4):
  # すでに同じ数を確認していたら処理をスキップ
  if b[j] in b[0:j]:
    continue
  for i in range(4):
    if (int(b[j]) == a[i]) and (a[i] != int(b[i])) and (a[j] != int(b[j])):
      blow = blow + 1
      break
```

「if b[j] in b[0:j]」で比較することで、「それよりも前に、同じ数を含んでいたとき」、つまり、「その数をすでに比較していたとき」は、continueでスキップすることで、重複して数えなくなります。