1.2 フィボナッチ数列の 2 項間の比

フィボナッチ数列には面白い性質がある。そのひとつは、隣り合う 2 項の比を調べることで見えてくる。1, 1 から始まるフィボナッチ数列を 2 項ずつペアにし、

$$\frac{1}{1},\ \frac{2}{1},\ \frac{3}{2},\ \frac{5}{3},\ \frac{8}{5},\ \frac{13}{8},\ \frac{21}{13},\ \frac{34}{21},\ \cdots$$

のように分数を作る。この程度は暗算や電卓で計算できるだろう。

もちろん **Python3** でも調べることができる。

[py script]

```
>>> a = 1
>>> b = 1
>>> for i in range(1, 20):
...   print((a + b) / b)
...   a, b = b, a + b
...
2.0
1.5
1.6666666666666667

（途中省略）

1.6180339631667064
1.6180339985218033
```

スクリプトは基本的に前に見たものの 4 行目を変えただけである。新しいフィボナッチ数を現在の `b` で割れば、2 項の比になる。スクリプトを実行すると、フィボナッチ数列の隣り合う 2 項の比だけが表示されることになる。`print` 文に “`, end=’ ’`” を含めなかったのは、改行を入れて表示するほうが値を比較しやすいと思ったからだ。蛇足ながら、version 2.x を使っているなら `a`、`b` への代入は `1.` でなくてはならない。ここに小数点がないと、version 2.x は割り算の結果を整数値にしてしまうからだ。で、そのせいで計算結果は `1` が並ぶことになる。まさか、そんなことになってないよね。

フィボナッチ数列というのは、何も 1, 1 から始まると決められているわけではない。たとえば 10, 3 から始めてもよい。すると

$$10,\ 3,\ 13,\ 16,\ 29,\ 45,\ 74,\ 119,\ \ldots$$

のようになって、1, 1 から始まる数列とは違うものになる。しかし、2 項の比についてはそうでもない。

[py script]
```
>>> a = 10
>>> b = 3
>>> for i in range(1, 20):
...   print((a + b) / b)
...   a, b = b, a + b
...
```

実際、a、b の値を変えて実行してみれば、似たような値に収斂（しゅうれん）していく様子が分かるだろう。

この散歩はファイルに保存しないことを前提にしているのだが、さすがに a、b の値だけ変更した振る舞いを調べるのに、再び全部を入力するのはつらいと思われる。コードに変更を加えないけれど、与える数値を変えてシミュレーションしたいときには困るのも事実だ。せめて、そのような場合に役立つコードぐらいは教えておこう。

[py script]
```
>>> while 1:
...   a = float(input('a = '))
...   b = float(input('b = '))
...   for i in range(1, 20):
...     print((a + b) / b)
...     a, b = b, a + b
...
a = 1
b = 1
```

a、b の値を始めに決めず、その都度入力要求させるには input() を使うとよい[1]。ただ、このまま使うと入力を促すプロンプトが何も表示されないので、input('a = ') と書いてプロンプトを表示させている。input() に float をかぶせているのは、input() で入力したものは文字列扱いになるためである。そこで、文字を実数に変換するために float が必要になった。だから、“a = ” のプロンプトに対して 1 と入力しても、**Python3** は実数計算とみなしてくれる。ちなみに、整数値に変換する場合は、int を使うとよい。

これ以外は前のスクリプトと同じなのだが、“a = float(...” から入力を始めて enter キーを押すと、即座に “a = ” のプロンプトが表示されてしまい、input() を使った意味がない。そこで、全体を while 1:でブロック化してある。そのために、それぞれの文がさらなる字下げを余儀なくされた。while 1:というのは、繰り返しを永久に続けるときに使う常套句のようなものだ。したがって、このスクリプトは永久に “a = ” と “b = ” のプロンプトで入力要求を促すことになる。

[1] version 2.x の場合は、input() ではなく raw_input() を用いる。

もちろん ctrl-c で終了できる。

ブロックというのは **Python3** にとって、大変重要な概念である。今後、ブロックが重要な意味を持つスクリプトに出会うと思うので、いまのうちに述べておこう。このスクリプトには、ふたつのブロックが存在する。`while` 構文と `for` 構文のブロックである。ブロックというのは、次にそれと同じ字下げ位置にスクリプトが書かれる直前までのかたまりをいう（正確には少々違う部分があるが、それはあとで登場する）。この場合、`while` ブロックはスクリプトの最後までで、`for` ブロックは続く 2 行までだ。つまり、`while` ブロックによって `a` と `b` の入力と `for` 構文が永久に繰り返されることになる。ただし、その間に `for` ブロックの処理が始まると、`i` が `1` から `19` になるまで所定の処理が続くことになる。慣れるまで大変だろうが、ここは散歩の肝である。早めに身につけてもらいたい。

さて、いろいろな入力を試してみれば、いつでも $1.6\cdots$ 程度の値に収まるように思える。1 にまつわる地を巡っているので、目にする数値は当然 1 から始まる。この数値が何かは、もうしばらく散策を続けたときにはっきりするだろう。