5...の散歩道

5.1　黄金比

さあ、これから5...の散歩道が始まる。と言っても5から始まるのではなく、散歩中に5と出会うのが、いままでとは違うところだが。

まず、フィボナッチ数列の2項の比

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

に再び登場してもらおう。

以前出会ったことなので分かっているだろうが、比の値は$1.618033\cdots$となる。ところでフィボナッチ数列の2項の比は、$\frac{(\text{後の項})}{(\text{前の項})}$で求めたものだ。しかし、比なんだから別に$\frac{(\text{前の項})}{(\text{後の項})}$で求めたっていいだろうに。そう思ったら、フィボナッチ数列の比を$\frac{(\text{前の項})}{(\text{後の項})}$で計算してほしい。電卓でも計算できるが、だいぶ以前に書いたスクリプトを変更すればすぐだ。

さあ、比の値はどうなった？　ちょっと面白い結果になったのではないかい？　比の値は$0.618033\cdots$である。きっちり1だけ違っているだろう。誤差はないのかって？　そう、まったく誤差はないのだ。この面白い性質を持つ比の値は**黄金比**と呼ばれている。一般には、$1.618033\cdots$の方を黄金比と呼ぶことが多い。

黄金比とは一体どういう数なのだろうか。計算は簡単にできるので確かめてみよう。

まず、$1.618033\cdots$に収束する真の値をxとしよう。あとから求めた比$0.618033\cdots$は、xの逆比$\frac{1}{x}$のことである。それがxより1小さい値に等しいのだから

$$\frac{1}{x} = x - 1$$

が成り立つ。等式は簡単な2次方程式となる。両辺にxを掛けて整理すると$x^2 - x - 1 = 0$だから、解の公式を使えば一発で解ける。フィボナッチ数の比は正の値なので、$x > 0$の解を求めると

$$x = \frac{1 + \sqrt{5}}{2}$$

であることがわかる。黄金比は**無理数** $\sqrt{5}$ を含む数なのだ。ここに 5 が登場してきた。解は正の数だけに限っているものの、方程式からひとつの解しか得られないということは、この性質を満たす数は黄金比だけということである。

Python3 でシミュレーションしてみよう。ここでは $\frac{1}{x} = x - 1$ を移項して

$$x = \frac{1}{x} + 1 \tag{5.1}$$

で x を探ることにする。そのほうが x の値を直接計算できるからだ。ところで、探るという言葉が気になるね。なぜそのような言葉遣いをしたかはスクリプトを見てもらいたい。

[py script]

```
>>> def gratio(x):
...   for i in range(1, 20):
...     x = 1 / x  + 1
...     print(x)
...
>>> gratio(3)
1.3333333333333333
1.75
1.5714285714285714

（途中省略）

1.6180339217722395
1.6180340143330838
1.6180339789779863
```

スクリプトは実に簡単だ。初期値を変えながらいろいろ試してほしい。初期値が何であれ、黄金比に収束する様子が分かると思う。

ここの `for` 文の `i` は、久々に `for` 文のブロックで使われない、独立した単なるカウンタである。その場合は、`for i in range(20)` と書いてもよく、むしろそのほうがよいかもしれない。`range(n)` は `range(0, n)` を意味する。つまり、`0` から `n-1` までの n 回の繰り返しである。そう、n 回の繰り返しであることが一目瞭然になるのだ。

さて、スクリプトを見て、あれ？コンピュータが 2 次方程式を解くんじゃないんだ、と感じただろう。そう、コンピュータは計算をする機械であって、問題を解く機械ではない。問題を解く手順は人間が与えるのだ。では、この手順は何をしているのだろう。どう見ても $\frac{1}{x} = x - 1$ と同等な—つまり両辺に x を掛けた—方程式

$$1 = x^2 - x$$

を解いているのではないね。でも、解である黄金比が求められている。

それなら $1 = x^2 - x$ を移項した式、$x = x^2 - 1$ を使っても同じことだろう。スクリプトの計算式を `x = x * x - 1` に変えてみる。さあ、準備は整った。関数を走らせよう。今度は、与えるどんな値に対しても黄金比に収束する、とは言えないはずだ。ときに、巨大な数が表示されてあわてるかもしれない（そういうときは落ち着いて ctrl-c だよ）。

どうして？　同じ関係の方程式を使ったはずなのに。その秘密はグラフを描けば見えてくる。はじめのスクリプトに使ったのは $x = \dfrac{1}{x} + 1$ で、次のスクリプトに使ったのは $x = x^2 - 1$ だから、それぞれ

$$y = x \quad と \quad y = \frac{1}{x} + 1$$
$$y = x \quad と \quad y = x^2 - 1$$

のグラフの交点を求めていることになる。ただし、スクリプトは一気に交点を求めているのではない。適当な x から始めて、いわばふたつのグラフを渡り歩くような処理がスクリプトのしていることだ。でも、これ以上詳しいことは言わないでおこう。実際にグラフを描いて、x 軸上の点から始めて、ふたつのグラフを縦横縦横...交互に進んでみよう。$y = x$ と $y = \dfrac{1}{x} + 1$ のグラフでは、必ず交点にたどり着くだろう。しかし、$y = x$ と $y = x^2 - 1$ のグラフでは、始めの x 次第で交点にたどり着いたりあさっての方向へ跳んでしまったりするはずだ。こうなったときが、収束しない場合なのである。