5.3 究極の連分数

黄金比が

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

の解であることは述べたね。しかし、違った考えで x の解を求めることもできる。

ややこしい話だけれど、$x = 1 + \dfrac{1}{x}$ の $1 + \dfrac{1}{x}$ を右辺の x に代入してみることだ。つまり

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

ということだ。しかし、右辺にはまだ x が存在するので、再び $x = 1 + \dfrac{1}{x}$ を右辺にの x に代入すると

$$x = 1 + \frac{1}{1 + \dfrac{1}{1 + \frac{1}{x}}}$$

となるだろう。さらに、右辺にはまだ x が存在するので...。きりがないね。だが、これを無限に続ければ

$$x = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cdots}}}$$

となることが理解できるだろうか。

このように、分数の中に分数が幾重にもなる分数を**連分数**と呼んでいる。連分数は一般に

$$a = q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cfrac{1}{q_4 + \cdots}}}$$

で表すが、これでは紙面を使い過ぎるので

$$a = [q_1,\ q_2,\ q_3,\ q_4,\ \ldots]$$

といった表し方をすることが多い。連分数の各分子が常に 1 だから可能な記述なのだ。

ところで、連分数で表された x は $x = 1 + \dfrac{1}{x}$ の解だから $x = \dfrac{1 + \sqrt{5}}{2}$ だ。つまり

$$\frac{1 + \sqrt{5}}{2} = [1,\ 1,\ 1,\ 1,\ \ldots]$$

なんだね。右辺の連分数には 1 しか使われていないので、連分数の中でも特別な連分数だ。

そもそも連分数が特別なんだから、黄金比はまさに究極の連分数と言えるだろう。でも、ちょっと待って。連分数って本当に特別な分数なの？　次の例を見てほしい。

$$\frac{355}{113} = 3 + \cfrac{1}{7 + \cfrac{1}{16}}$$

有限個の分数で終わっているけれど、これも正真正銘の連分数だ。例が正しいことをぜひ確認してほしい。

実はどんな分数も連分数で表せる。いや、無理数だって連分数になるのだ。今回の散歩では深入りはしないが、有理数は有限連分数、無理数は無限連分数ということが知られている。ちょっと前に歩いた場所で、割り切れない分数は循環小数—つまりは規則的な繰り返しをする数—であることを確認したはずだ。それに対して無理数は、不規則な小数だっただろう。でも驚いてはいけない。不規則なのは見せ掛けなのだ。

計算は少々面倒だが、[1, 2, 2, 2, 2, 2, 2, 2, ...] や [1, 1, 2, 1, 2, 1, 2, 1, 2, ...] といった、規則的な連分数がどんな小数になるか計算してみるとよい。もちろん "..." の部分すべては計算できないので、"..." の直前で終わっている連分数として計算するわけだけど。

おそらくどこかで目にした近似値になったのではないだろうか。もし、"..." をすべて計算できれば、正にその数になる。何とも不思議なものだが、詳しくは別の機会に回そう。ここでは、有理数を有限連分数に直すスクリプトに取り組むことにする。

[py script]

```
>>> def contfrac(a, b):
...   print(a // b, end=' ')
...   while a % b > 0:
...     a, b = b, a % b
...     print(a // b, end=' ')
...
>>> contfrac(355, 113)
3 7 16
```

スクリプトを説明する前に、$\frac{355}{113}$ を連分数にする方法を知る必要がある。コンピュータは自ら連分数を作ってくれないのだ。あくまでも人がやる作業と同じことを、短時間でやってのけるだけなのだから。

それは次のようにして求められた。

$$\begin{aligned} \frac{355}{113} &= 3 + \frac{16}{113} \\ &= 3 + \frac{1}{\frac{113}{16}} \end{aligned}$$

$$= \quad 3 + \frac{1}{7 + \frac{1}{16}}$$

いちいち説明の必要はないだろう。ここでは最後に $\frac{1}{16}$ が現れて終了となったが、仮に $\frac{3}{16}$ にでもなろうものなら、さらに $\frac{3}{16} = \frac{1}{\frac{16}{3}}$ としていけばよい。最終的に分子が 1 になったところで終了だ。スクリプトは分子が 1 であることの確認を、分数の逆数が割り切れるかどうかでしているけれど、意味するところは変わらない。

このスクリプトは分子・分母の値を必要とする。ここでは分子を a、分母を b とし、それぞれを引数として関数に与えている。

はじめに `print(a // b)` としているが、これは分子が分母より大きいの場合の処置である。ここでは確実に整数値がほしいので "//" で割っている。`while` 構文は余りが 0 にならない限り、分子と分母を交換しながら処理を続けるようになっている。`a, b = b, a % b` がタプル代入であることは理解したかな？　ところで、この処理は少しややこしいので以下は注意深く読んでほしい。

代入式を見ると、a を b で割った余りが b に代入されている。え？余りは分子である a に代入するんじゃないの？　そう、たしかにそうだが、さらに次の連分数展開ではこの逆数が使われるので、余りは分母にされてしまうはずだ。だから分母である b に代入したのだ。

したがって、いままで分母に使われていた b が新しい分子 a になるから、タプル代入ひとつで分子・分母の入れ換えが完了する。スクリプトはわずか 1 行で連分数展開をこなしているように見える。しかし、人が行う作業が凝集されていることを見逃してはいけない。

このとき連分数展開に現れる次の数は $\frac{a}{b}$ の整数部分である。それを `print(a // b, end=' ')` によってひとつずつ表示することになる。

いろいろな分数の連分数表記を調べると意外な発見があるだろう。このスクリプトは無理数を連分数にできないが、たとえば黄金比 $\frac{1+\sqrt{5}}{2}$ なら近似分数として $\frac{16180339}{10000000}$ を使ってもよい。試しに、この近似分数を連分数にすると次のようになる。

[py script]

```
>>> contfrac(16180339, 10000000)
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 10 15 1 1 1 2
```

さすがに後ろのほうになると誤差が出るので、ずっーと 1 が表示されるわけではない。それでも、だいたいの雰囲気はつかめるはずだ。$\frac{14142135}{10000000}$ や $\frac{17320508}{10000000}$ なども試してみるとよいだろう。