7... の夢見鳥

7.1 不思議な循環小数

$\frac{1}{7} = 0.142857\ldots$ は不思議な性質を持っている。正確には、循環節 142857 の性質が面白いのだ。それは

$$\begin{aligned} 142857 \times 2 &= 285714 \\ 142857 \times 3 &= 428571 \\ 142857 \times 4 &= 571428 \\ 142857 \times 5 &= 714285 \\ 142857 \times 6 &= 857142 \end{aligned}$$

となるからだ。なんと、循環節の数字が巡回しているではないか。

こうなってくると、他にも不思議な循環節を持つ分数を調べたくなるだろう。以前、循環節を調べたことを覚えているかい？　宙に舞い始めたばかりの頃は、$\frac{1}{113}$ がどの程度の循環節を持つか調べるのに苦労したね。余りをザーッと表示させて、最初の余りと同じものがあるかを調べたんだ。これは手間のかかる作業だし、第一、商がどうなっているのか分からずじまいだった。当時は、循環節を表示する力量が不足していたので、仕方なく余りを表示することで目的を達していたのだ。ところが、いまや力量はかなり上昇している。循環小数の循環節を見せてくれるスクリプトを書いてみよう。

対象とする分数は $\frac{1}{n}$ の形で十分だが、n の値によっては困ることがある。循環しないで割り切れてしまう場合があるからだ。割り切れてしまう分数は、必ず $\frac{1}{2^r 5^s}$ の形をしている。したがって、n には 2 と 5 が含まれていないことが条件だ。ただ、この条件の分数だけを対象とすると、$\frac{1}{6}$ のような循環小数も除外されてしまう。しかし $\frac{1}{6}$ は、$\frac{1}{3}$ の循環小数を $\frac{1}{2}$ で分割していると見れば、本質は $\frac{1}{3}$ と同じだ。よって 2 や 5 が一つでも含まれている分数は、対象から外してかまわないだろう。そのほうがスクリプトも簡単にできる。そこで『

```
\begin{luacode*}
function cyc(n)
    r = 10 % n  c = 1     -- /*
    while r ~= 1 do
        r = (r * 10) % n
        c = c + 1
    end                   -- */
    return c
end

function cydec(n)
    --（cyc(n) の /*〜*/ まで挿入、↓ cyc(n) を c に変更。）
    if (n % 2 ~= 0) and (n % 5 ~= 0) then
        r = 1
        for i = 1, cyc(n) do
            tex.print((r * 10) // n .. ' ')
            r = (r * 10) % n
        end
        tex.print('\\dots')
    end
end
\end{luacode*}
```

』として、『\directlua{cydec(7)}』と書けば『1 4 2 8 5 7 ...』が出力される。... 以下が循環するという意味である。もし、小数を表示していることを明確にしたければ、最初に tex.print('0.' .. ' ') を書いておけばよいだろう。

以前のスクリプトでは、$\frac{1}{n}$ の余りを $(n-1)$ 回表示させた。これだと $\frac{1}{9}$ のような分数では、最初の余りから 8 回目の余りまで 1 が表示されてしまう。今回は商の表示を目的としているので、$\frac{1}{9}$ は何も『1 1 1 1 1 1 1 1 ...』でなく『1 ...』となれば十分だ。それに、余分な循環節を表示するようでは、$\frac{1}{113}$ が本当に 112 の循環節を持つ小数かどうかは、綿密に調べなくてはならず効率が悪い。

そのために、循環節がいくつか調べる関数 cyc(n) が必要になったのだ。先に、そこから説明しよう。

cyc 関数は $\frac{1}{n}$ の分母を変数 n で受け取る。このとき最低でも 1 回の割り算が実行されるはずだから、それを r = 10 % n でしている。r は余りが代入される変数だが、この時点では分子の 1 を分母の n で割るわけなので、0 を下ろす意味で 10 % n となっているのだ。また、変数 c は循環の回数を記憶するために必要で、ここで 1 回目の割り算が行われたので 1 で初期化してある。

余り r が 1 になれば一回りしたことになるので、while 構文は r が 1 にならない間繰り返され

る。その際の処理は、以前のスクリプトで使った手法と同じである。違いは循環の回数をカウントしていることだ。`r = 1` となったら、`return` 文で循環の回数である値が返される。

ところで変数 `r` は、`cyc` 関数にも `cydec` 関数にも使われているけど大丈夫？　大丈夫である。それは、`cyc` 関数と `cydec` 関数は連動していないからだ。Lua はグローバル変数を使うので、二つの関数で使われている `r` は同じものである。しかしこの場合は、`cyc` 関数は単に別の場所に書いただけである。

コメントに示したように、`cyc` 関数の一部を `cydec` 関数に移動し、`cydec` 関数一つで実行することを考えてみよう。プログラムを追ってみれば分かるように、前半で使われる `r` と後半に使われる `r` には無関係の値が代入されるのだ。そのため、同じ変数名を使っても問題はない。心配なら異なる文字を使えばよいが、文字の種類には限りがある。大事なのは、処理の流れがきちんと把握できているかどうかだ。

次に、`cydec` 関数を見よう。冒頭に述べたように、分母に 2 または 5 を**因数**に持つ数は除外したい。そこで、`if` 文によって入力を制限している。ここで使われている記号 `and` は以前も使った論理演算子である。ここでは、`n` が 2 で割れない、かつ、5 でも割れないときに限り処理をすることになる。したがって、分母に 2 や 5 を因数に持つ数を入力したら何も表示されないことになる。

さあ、問題ない `n` の値を手に入れたら割り算の開始だ。始めの `r = 1` は分子の 1 だが、計算が進むたびに分子の値が変わるので、変数 `r` を用いているのだ。

商の表示と、次回使う余りの計算も以前のスクリプトと同じだ。最後に `...` を付けて体裁を整えたら完了だ。これで安心して循環節が分かるようになった。遠慮せずに $\frac{1}{n}$ を計算させよう。