9...の散歩道

9.1 九去法

たとえば 28705 が 9 で割れるかどうか知るには、実際に割ってみればよい。もし、商や余りに関心がなく、ただ割れるかどうかだけ知りたいのなら別の方法がある。それには、各桁の数を足してみることだ。$2+8+7+0+5=22$ は 9 で割れないので、もとの 28705 も 9 で割れない。このような調べ方は**九去法**と呼ばれ、古くから知られている。

なぜそうなるかを、たとえば 3 桁の数 $100a+10b+c$ を用いて説明しよう。$100a+10b+c$ は

$$100a+10b+c=(99a+a)+(9b+b)+c=(99a+9b)+(a+b+c)$$

と書き換えることができる。$99a+9b$ が 9 で割り切れることは明らかだから、$100a+10b+c$ が 9 で割り切れるかどうかは、$a+b+c$ 次第ということになる。$a+b+c$ は各桁の和であるから、これが九去法の判断材料になるのだ。これを **Python3** で実現してみよう。

[py script]

```
>>def nimethod(n):
...   s = str(n)
...   while len(s) > 1:
...     sum = 0
...     for c in s:
...       sum += int(c)
...     s = str(sum)
...   if s == '9':
...     print('9 で割れる')
...   else:
...     print('9 で割れない')
...
>>> nimethod(12345)
9 で割れない
>>> nimethod(9999909)
9 で割れる
```

スクリプトを書く際に、各桁の和を求めて、それが 9 で割れるかどうか調べるのでは芸がない。そんなことをするくらいなら、始めから 9 で割ってしまえばよいからだ。そこで、スクリプトは和

に対しても九去法を当てはめ、最後に一桁の数になるまで繰り返すことにした。一桁の数が 9 かどうかを見れば、もとの数が 9 で割れるかどうか分かるからだ。

各桁を取り出す方法はいろいろある。たとえば、次のようなアルゴリズムを核にしてもよいだろう。

[py script]
```
while n:
  sum += n % 10
  n = n // 10
```

これは、まず n を 10 で割った余りを変数 sum に加算する。次に n を 10 で割った整数部分の値に代える。このことを n がなくなる—つまり n = 0 になる—まで続ければよい。n = 0 は同時に while ブロックを抜ける合図でもある。

しかし、ここではちょっと変わったことをしている。最初に、9 で割れるかどうか調べる数値を文字列に変換しているのだ。なぜなら、文字列をスライシングして、各桁を抜き出したいからである。

九去法を最後の一桁まで続けるので、s が 1 より大きい間は while 構文が行われなくてはいけない。len は文字列の長さを返す関数だ。

for c in s が特徴的だろう。これは、c の値に s の各々の値を当てはめ、s のすべてにおいて繰り返す処理である。たとえば、s = 'ab12' とすると、c は a、b、1、2 となって、for ブロックの処理を繰り返すのである。

いま c には、数値の各桁が入るので、それらを加算すればよいのだが、c は文字列なので足す前に整数値に変換している。その和 sum を求めたら、その桁数が 1 より大きいときは処理を繰り返すので、while 文でチェックを受けることになる。しかし sum は数値だから、文字列にして c に渡している。最後に、一桁の文字が '9' かどうか調べて、結果を表示してやればよい。

数値と文字列の変換が煩わしいだろうが、結果的に分かりやすい処理をしているはずだ。

さて、九去法は 3 の倍数を見つけるのにも役立つ。9 で割れないときは、3 で割れるかどうか調べてやると親切だ。

[py script]
```
...   if s == '9':
...     print('9 で割れる')
...   elif s == '3':
...     print('3 で割れる')
...   else:
...     print('3 で割れない')
```

`if` 構文に `elif` 文を使うと、細かい分岐処理ができる。`if` の条件に合致しなければ、続く `elif` の条件を試す。ここでは `elif` 文をひとつ追加しただけだが、`elif` 文はいくつ書いてもよい。そして、どの `if` にも合致しなければ、最後の `else` 文へと進むのである。

おまけの話になるが、引数に `9*4556` のような計算式を与えても、スクリプトは正常に動作する。引数を文字列で入力するような関数にすることもできるが、それでは使い勝手が悪い。与えた数が何の倍数か調べるのだから、入力する値は数値であるのが自然なのである[1]。

もっとも今回の散歩道は、散策するほどのことはないものだ。電卓で計算すれば済むことを、わざわざスクリプトを書いて調べてるのだから。まるで、部屋から見える景色を、わざわざ外に出て鑑賞するみたいなものだ。けれども、外に出ることで気づくこともある。空気の寒暖や風の有無、草木のにおいだって感じることができるかもしれない。スクリプトを書くということは、電卓では味わえないものを感じることでもあるのだ。

[1] スクリプトをファイルに保存して引数を取って実行する場合は、引数が文字列で渡されるので `s = str(n)` は必要ない。しかし今度は `9*4556` のような引数はエラーとなる。