2...の散歩道

2.1 関数

数学に**関数**は付きものである。関数は特別難しいものではない。記述の仕方に慣れさえすれば、関数はむしろ便利で使いやすいのである。

関数 $f(x) = x^2$ があるとする。これは、f という名の関数に (何がし)^2 の機能を持たせている。そのため関数 f に 5 を与えると、25 という値が返ってくるのだ。また関数 $g(x) = 2^x$ は、g という名の関数に $2^{(\text{何がし})}$ の機能を持たせている。よって関数 g に 10 を与えると、1024 という値が返ってくるのだ。

ここでちょっと変な感じを持った人がいるだろうか？　私が関数 f, g と言ったことに対してだ。だが、これでよい。$f(x)$ と書くのは、関数 f が変数 x を使うことを明らかにするためなのだ。もし皆が皆、変数には x しか使わないと取り決めていたら、$f = x^2$, $g = 2^x$ で十分である。しかし、変数には x 以外の文字を使うことはよくあるし、変数が 2 個以上ある関数だってある。さらには文字定数を利用する場合があることを思えば、関数名に続けて ( ) を付け加えることが不可欠なのである。

関数の例をあとふたつ提示しよう。

最初は $f(x,\ y) = x^2 - 3y$ でどうだろう。関数 f は変数に $x,\ y$ を使う。その結果返ってくるのが $x^2 - 3y$ で計算された値だ。だから $f(5,\ 10)$ と書けば、$x = 5$, $y = 10$ を代入して計算した値 -5 が返ってくることになる。一般に数学で使う関数は、実数値を受け取って実数値を返すので、$f(0.8,\ 0.1)$ でも計算可能で 0.34 が返ってくる。

ふたつ目の関数は $g(x) = \lfloor x \rfloor$ でどうだろう。見慣れない記号 $\lfloor\ \rfloor$ はフロア記号と呼ばれる。フロア記号を簡単に説明すると、与えられた値を超えない整数のうち、最大のものを返す機能を持っている。むむ、難しい表現だ。具体的には $g(3.14)$ と書けば 3 が返され、$g(-3.14)$ と書けば -4 が返されるのだ。いわゆる “切り捨て” である。フロア（floor）の名称から想像できるように、これは建物の “床” を意味する。3.14 階という表現はおかしいが、3 階の部屋の中空に 3.14 階があると思えば、その床は 3 階だから $\lfloor 3.14 \rfloor = 3$ とみなすのである。

フロアと対（つい）になる関数はシーリング（ceiling）と呼ばれ、$\lceil x \rceil$ で表される。$\lceil x \rceil$ は、x を下回らない最小の整数である。具体的には $\lceil 3.14 \rceil = 4$、$\lceil -3.14 \rceil = -3$ などである。**Python3** で利用するときは、`import math` をした上で `math.floor(3.14)` や `math.ceil(-3.14)` と記述する。

いまふたつの関数を例に出してみたが、値を求めるためにする記述は、いずれも $f(5,\ 10)$ や $g(3.14)$ のように簡便だ。にもかかわらず -5 や 3 の値が返ってくるのは、裏方で f や g がせっせと $5^2 - 3 \times 10$ や $\lfloor 3.14 \rfloor$ の計算をしているからである。また、$f(3,\ 8)$ のように別の値を与えれば、それに応じた値が返ってくる。これは、関数がする仕事がきちんと決められているからである。

何のことはない。関数とは、与えられた値を別の値に加工する手続きなのだ。

この道では**指数関数**を話題に取り上げる。それも、もっとも基本的な $f(x) = 2^x$ にしておこう。しかしその前に、**Python3** における関数の仕組みを知るために、x の値を入力すると x^2 の値を表示するスクリプトを書いておこう。

[py script]

```
>>> def sqf(x):
...   return x**2
...
>>> sqf(5)
25
```

Python3 の関数は `def` によって定義する。`def sqf(x):` と `return x**2` が **Python3** による、関数 $f(x) = x^2$ の表現である。もう少し細かく分析すると、$f(x)$ に当たるのが `sqf(x)` で、x^2 に当たるのが `x**2` である。x の平方（square）を求める関数（function）の意味で、関数名を `sqf` としただけだから、`sqf` の文字に特別な意味があるわけではない。また、関数もブロック単位で処理されるので、“`:`” をつけ忘れてはいけない。

数学では $f(x) = x^2$ の x に値が代入されたとき、右辺の x^2 により実質的な計算が行われ、答が返ってくる。スクリプトも同じだ。関数 `sqf(x)` に具体的な値が代入されると、`x**2` により実質的な計算が行われる。黙っていても値を返してくれないので、`return` 文を用いて値を返さなくてはならない。

関数の使い方は $f(x) = x^2$ の場合と同じだ。`x` に適当な数値を与えればよい。関数に与える `x` は実数値も受け付けてくれる。

[py script]

```
>>> sqf(1.23)
1.5129
```

バージョンの違いで結果が `1.5128999999999999` のようになるかもしれない。**Python3** は内部

では2進数で計算を行っていて、それを素直に返せば10進数表示で誤差がでるのは道理だが、そんなときは`print sqf(1.23)`とすれば`1.5129`と表示される。さもなくば、スクリプトの`return x**2`を`print(x**2)`に変えてもかまわない。10進数で有限の小数であっても2進数では無限小数になることはザラにある。コンピュータの特性には気をつけなければいけない。

これまで、数値を変えるだけで何度も同じ計算をしたいときは、`while 1:`構文と`input`関数の組み合わせでしのいできた。でも、関数を一度定義してしまえば、関数に与える数値を変えるだけで、好きなだけ試すことができることに気づいただろう。そう、関数は再利用可能な部品のようなものである。もっとも、散歩中はファイルとして保存しないので、**Python3**を終了させてしまえば関数も消えてなくなる運命だけれど、少なくとも`while 1:`構文と`input`関数の組み合わせよりは使い勝手が良くなっただろう。