2 計算の基本

■ 計算の実行（評価）

計算の実行は，計算式に続いて SHIFT RET を入力することで行われる．式を計算することを「式を評価する」という．

■ シンボル（名前）

Mathematica で使用する関数や変数などの名前は，シンボルと呼ばれる．シンボルは英字で始まり，その後に英字もしくは数字が続くものである．たとえば， abc, D, Ef, gH, ijk4, LM5n6 などはシンボルである．しかし， 7xyz のように，数字で始まるものはシンボルではない（ 7xyz は 7 xyz と解釈される）．
大文字と小文字は区別される．すなわち， abC と aBc とは異なるシンボルである．

■ 組み込みのシンボルとユーザー定義のシンボル

■ 組み込みのシンボル

Mathematica の組み込みのシンボルは Sin とか Integrate とかのように，必ず大文字で始まる．また，FactorInteger とか MapAt とかのように，複数の単語からなるときには，それぞれの語の先頭を大文字にしたものを組み合わせてある．

Unixなどのシステムとは違って， Mathematica の組み込みのシンボルには，略語がほとんどない．略語が使用されているのは，その略語が通常よく使用されていて，その略語を使わないと，かえって分かりにくくなるような場合である．

Mathematica の組み込みのシンボルで略語であるものの例

Abs	数の絶対値 (absolute value) を与える関数
C	微分方程式の解についてくる定数 (constant)D 微分 (derivation) をする
Det	行列の行列式 (determinant) を求める関数
GCD	最大公約数 (greatest common divisor) を求める関数
Sqrt	平方根 (square root) を求める関数

■ ユ-ザ-定義のシンボル

ユーザーが独自のシンボルを用いるときには，組み込みのシンボルと区別するために，小文字で始めることが推奨されている．初めて Mathematica を使うときにありがちなトラブルは，A,B,C,...という変数名を使用したときに起こる．試しにA,B,Cに1,2,3の値を割り当て（代入）してみよう．

```
In[1]:= A = 1

Out[1]= 1

In[2]:= B = 2

Out[2]= 2

In[3]:= C = 3

        Set::wrsym : シンボルCはProtectedです. 詳細

Out[3]= 3
```

Cへの割り当てを行おうとしたときに，Symbol C is Protected.という警告メッセージが発せられる．先ほどの表にあるように，Cは組み込みのシンボルとして定義されていて，それは書き換えられないように保護(protect)されている．よって，割り当ては拒否される．このようなことを避けるためにも，ユーザーは大文字で始まるシンボルを定義するのは止めるべきである．

シンボルには，後から読んだときに，あるいは他の人が読んだときに意味が分かりやすいような名前をつけるべきである．また，複数の単語を組み合わせるときには， 単語の先頭を大文字にすると読みやすくなる．たとえばnextPrimeという関数名を付ければ，nextPrime[7]は7の次の素数11を返すだろうということの予測がつく．

■ 括弧

丸括弧()，波括弧{}，角括弧[]， 2 重の角括弧[[]]，丸括弧とアスタリスク *の組み合わせ(**)には，それぞれの使用目的があり，それぞれ異なる意味をもつ． Mathematica を使用するとき最初にとまどうところは，この使用目的が数式においての通常の使用目的と若干異なる点である．

解説に用いる変数をクリアしておこう．これは，変数に値が代入されていた場合，計算結果がここで表示したものと異なる可能性があるからだ．

```
In[4]:= Clear[a, b, c, d, e, f]
```

■ 丸括弧()

丸括弧"(","("は式の部分をまとめる（グルーピングする）ために使用する．

```
In[5]:= a + b / c + d

Out[5]= a + b/c + d

In[6]:= (a + b) / (c + d)

Out[6]= (a + b)/(c + d)
```

丸括弧は，過剰に使用しても害はない．(a+b+c)+(x+y+z) などのように，適当な部分を丸括弧でくくって，プログラムを後から（もしくは他人が）見るときに分かりやすくするために，使用することもできる．

■ 波括弧{}

波括弧"{","}"はリストを表すために使用される.

```
In[7]:= {a, b, c}

Out[7]= {a, b, c}
```

リストは多重にネストすることもできる.

```
In[8]:= {{a, b}, {c, d, {e, f}, g}}

Out[8]= {{a, b}, {c, d, {e, f}, g}}
```

■ 角括弧[ ]

角括弧は，関数の引き数を指定するために使用する． Mathematica を初めて使う場合，関数につける括弧を()としてしまいがちであるが，[]であることに注意しよう.

```
In[9]:= Sqrt[9]

Out[9]= 3
```

Sqrtは平方根を求める関数である．この [] を誤って () にすると，

```
In[10]:= Sqrt (9)

Out[10]= 9 Sqrt
```

となってしまう．これは9×Sqrtである.

関数には，複数個の引き数をとるものもある．この場合には，角括弧の中に続けて入れる.

```
In[11]:= Power[a, b]

Out[11]= a^b
```

また，引き数をまったく必要としない関数もある．この場合には，角括弧の中に何も入れないが，しかし，角括弧を省略してはならない.

```
In[12]:= Random[]

Out[12]= 0.604586
```

Random[]は0から1の間の乱数を返す.

■ ２重角括弧[[]]

２重の角括弧 "[[", "]]" は，式の中の要素を番号（インデックス）で指定するために使用する．f[a1, a2, ...] の右に [[n]] をつけると an が取り出せる.

```
In[13]:= f[a, b, c, d][[3]]

Out[13]= c
```

これはリストの要素を取り出すことによく使われる.

```
In[14]:= {a, b, c, d}[[3]]

Out[14]= c
```

多重リストの場合，「 3 番目の要素の中の 2 番目の要素」という指定がしたいときがある．このようなときは[[3,2]]とすればよい.

```
In[15]:= {{a, b}, c, {d, e, f}, g}[[3, 2]]

Out[15]= e
```

■ コメント(**)

"(*"と"*)"とでくくられた部分は，計算評価の対象とはみなされずコメントとして扱われる.

```
In[16]:= 1 (* one *) + (* plus *) 2 (* two *)

Out[16]= 3
```

■ 数式の入力

■ 四則演算

加法，減法，除法は +,-, /である．乗法は * でも空白でもよく，a bは a*b と同じである．乗法，除法は加法，減法よりも結合順位が高いので a+b c は a+(b*c) と解釈される．加法，減法あるいは乗法，除法などの結合順位の同じものが続く場合には，左から順に計算される．たとえばa+b-c+d は((a+b)-c)+d と， また a b/c d は ((a*b)/c)*d と解釈される.

```
In[17]:= a b / c d

Out[17]= a b d
         -----
           c
```

■ ベキ乗

ベキ乗 a^b は a^b と書く． ^は乗法，除法よりも結合順位が高い．たとえばa b^c dはa(b^c)dと解釈される．また，^が続いている場合に，は加減乗除法とは違って，右から順に計算される．たとえばa^b^cはa^(b^c)と解釈される.

```
In[18]:= a b ^ c d

Out[18]= a b^c d

In[19]:= a^b^c

Out[19]= a^(b^c)
```

■ 階乗

整数nの階乗はn!とかく．!はベキ乗よりも結合順位が高い．たとえばa^b!はa^(b!)と解釈される．

```
In[20]:= a^b!
```

$$a^{b!}$$

■ 等式，不等式

等号は = を空白を挟まずに二つ続けて == と書く．これは次に述べる割り当ての記号 = と区別するためである．不等号は <，>であるが，等号を含む不等号は <=，>= と，必ず = を右側に書く．等号および不等号は，連ねて書いてもよい．また，等しくないという関係は != とかく．

```
In[21]:= 2 + 3 == 5 <= 7 != 3
```

```
Out[21]= True
```

<= や != などの記号は，入力したとたんに，数式に通常用いられる次のような記号に置き換わる．

```
In[22]:= 2 + 3 == 5 ≤ 7 ≠ 3
```

```
Out[22]= True
```

```
In[23]:=
```

■ 割り当て（代入）

割り当てが行われていない変数は，評価されるとそのシンボル名をそのまま返す．

```
In[24]:= x
```

```
Out[24]= x
```

変数への値の割り当て（代入）は =を用いる．

```
In[25]:= x = 3
```

```
Out[25]= 3
```

この割り当てを行った後は，xが評価される度に x が 3に置き換えられる．

```
In[26]:= x + y
```

$$3 + y$$

この割り当てを行った後は，xが評価される度に x が 3に置き換えられる．

```
In[27]:= x = 4
```

```
Out[27]= 4
```

```
In[28]:= x + y

Out[28]= 4 + y
```

割り当てを解除するにはClear[x]もしくはx=.（等号とピリオド）を実行する．

```
In[29]:= Clear[x]
```

割り当ては解除されたので，xの値はxのままである．

```
In[30]:= x + y

Out[30]= x + y
```

■ 計算履歴

入力および出力（計算結果）は，すべて保持されている．それはIn[...]およびOut[...] または %により参照できる．

解説に用いる変数をクリアしてから始めよう．

```
In[31]:= Clear[a, b, c]
```

一つ計算をしてみる．

```
In[32]:= Expand[(a + b)^2]

Out[32]= a^2 + 2 a b + b^2
```

この，直前の計算結果を参照したいときは，% を用いる．

```
In[33]:= % / (a - b)

Out[33]= (a^2 + 2 a b + b^2)/(a - b)
```

もう一度，初めの計算結果 $a^2 + 2\,a\,b + b^2$ を参照したいとする．それは，一つ前ではなくて，二つ前の計算結果となってしまった．二つ前の計算結果を参照するには，%% を用いる．

```
In[34]:= %% / (a - 2 b)

Out[34]= (a^2 + 2 a b + b^2)/(a - 2 b)
```

同様に，% の数だけ前方の計算結果を参照できる．

```
In[35]:= %%% / (a - 3 b)

Out[35]= (a^2 + 2 a b + b^2)/(a - 3 b)
```

入力式および出力式には In[..]:= および Out[..]= というタグがついている．これを用いて番号を直接指定することにより参照することもできる．

```
In[36]:= Out[32]

Out[36]= a^2 + 2 a b + b^2
```

計算履歴を利用するのは便利ではあるが，カーソルを動かして，あちこちの入力式を再計算させた場合には，直前の結果がどれであったのか不明瞭になるし，保存したファイルを開いた後，先頭から再計算を始めた場合，In,Outについている番号は，以前のものと異なる可能性があるので，同一の計算が再現できない可能性がある．

■ 頭部と要素

始めに，解説に使う変数への割り当てをクリアしておこう．

```
In[37]:= Clear[a, b, c, d, x, y]
```

Mathematica では，すべてのものはf[a,b,...]という形をしている．fをf[a,b,...]のヘッド（頭部）と呼ぶ．a,b,...はf[a,b,...]の要素と呼ばれる．a+b+c dのような式も，実際はこのような形をしており，完全形式と呼ばれる．それを見るには，FullFormという関数を使う．また，式の頭部はHeadを用いると得られる．

```
In[38]:= FullForm[a + b + c d]

Out[38]//FullForm=
        Plus[a, b, Times[c, d]]

In[39]:= Head[a + b + c d]

Out[39]= Plus
```

要素には左から順に1,2,3,...と，インデックス（番号）が付けられている．なお，ヘッドには便宜上0番というインデックスが付けられている．式の右に[[n]]を付けることにより，n番目の要素を取り出すことができる．

```
In[40]:= f[a, b, c, d][[3]]

Out[40]= c

In[41]:= f[a, b, c, d][[0]]

Out[41]= f
```

a + b x + c x^2 の完全形式を表示しc x^2 が3番目の要素であることを確認して，それを取り出してみる．

```
In[42]:= FullForm[a + b x + c x^2]

Out[42]//FullForm=
        Plus[a, Times[b, x], Times[c, Power[x, 2]]]

In[43]:= (a + b x + c x^2)[[3]]

Out[43]= c x^2
```

また，式の完全形式を分かりやすく表示するには，TreeFormを用いればよい．

```
In[44]:= TreeForm[a + b x + c x^2]

Out[44]//TreeForm=
        Plus[a, |          , |                    ]
                Times[b, x]   Times[c, |          ]
                                      Power[x, 2]
```

■ ルール（変換規則）とその適用

Mathematica において，他の計算機言語にはない有用なものに，ルール（変換規則）というものがある．変換規則は規則の適用とセットにして使われる．

ルールとはx→3のようなものであり，これは「xを3に変換する」という変換規則を表している．ルールは，x^2+5/.x→3のように，ルールが適用される式とルールとの間に，「適用する」ということを意味する記号 /.を挟むことにより，実際に適用される．

expr/.lhs→rhs はexpr中のlhsをすべてrhsに置き換えた式を返す．

```
In[45]:= x^2 + 5 /. x -> 3

Out[45]= 14

In[46]:= x^2 + Sin[x] /. x -> a + b

Out[46]= (a + b)^2 + Sin[a + b]

In[47]:= a + b + c /. a + b -> Sin[x]

Out[47]= c + Sin[x]
```

expr/.{rule1,rule2,...}はexprに複数個のルールを適用した式を返す．

```
In[48]:= a + b + c /. {b -> x, c -> y z}

Out[48]= a + x + y z
```

割り当て（代入）と変換規則の適用との違いは，割り当ては解除しない限り無期限に有効であるのに対して，変換規則の適用はその場限りのものであるということである．次のように x->3 という変換規則を適用しても， x の値そのものが 3 になるわけではなく，以後の計算には全く影響しない．

```
In[49]:= x + y /. x -> 3

Out[49]= 3 + y

In[50]:= x + y

Out[50]= x + y
```

■ 演習問題

[2-1] a-b*c の完全形式を FullForm を使って求めよ．

[2-2] 等しくないという関係は != で表される. a != b の完全形式は何か.

[2-3] Times[Plus[a,Times[b,Power[c,2]]],Plus[1,d]] を通常の数式表現に直せ.

[2-4] Log[a + b^3 Cos[3 + 2 t^3] + Sin[1 + t]] の完全形式は何か. また, この式中から 3 + 2 t^3 の部分を [[ ]] を使って取り出せ.

■ この章で出てきた関数

Clear[expr]	expr に割り当てられた値や定義を除去する
In[n]	n 番目の入力式
Out[n]	n 番目の出力値
FullForm[expr]	expr の完全形式