

## もっと使ってみよう

### 前回の復習

- 評価するときは `(Shift)+(Enter)` とする .
- 組み込み関数は全て大文字で始まる .
- 微分は `D`, 積分は `Integrate`. 次の命令で,  $\sqrt{x^2+1}$  を積分したのち微分する . 一見元に戻ってないようだが, 式を簡単にせよと命令すると, 元の関数が現れる .

```
Integrate[Sqrt[x^2 + 1], x]
```

```
D[%, x]
```

```
Simplify[%]
```

(1)

### さらなる組み込み関数

- Mathematica は多項式を扱うことができる . 展開 (`Expand`) や因数分解 (`Factor`) もできる .

```
x + 2x
```

```
3 x
```

```
Expand[(x + 2y)^5]
```

```
x^5 + 10 x^4 y + 40 x^3 y^2 + 80 x^2 y^3 + 80 x y^4 + 32 y^5
```

```
Factor[x^100 - 1]
```

```
(-1 + x) (1 + x) (1 + x^2) (1 - x + x^2 - x^3 + x^4) (1 + x + x^2 + x^3 + x^4) (1 - x^2 + x^4 - x^6 + x^8)
(1 - x^5 + x^10 - x^15 + x^20) (1 + x^5 + x^10 + x^15 + x^20) (1 - x^10 + x^20 - x^30 + x^40)
```

(2)

- 次の計算を試してみよ .

```
Factor[x^2 + 2xy + y^2]
```

(3)

この式では,  $x$  と  $y$  の間に空白を入れなければ因数分解してくれない . なぜなら, 空白を入れなければ,  $xy$  は一つの変数と認識されてしまうからである . 少し長いプログラムを書くと,  $a$  から  $z$  の 26 個の変数では足りなくなってしまうことがよくあるから, 2 文字以上で名前が付けられた変数も許されているのである . 通常の数学では  $xy$  と書けば  $x \times y$  のことだが, プログラム言語においてはそうではない . ちなみに,  $2x$  は間に空白を入れなくても  $2 \times x$  と認識してくれる .

- 分数式も扱える . 関数 `Together` は通分を行う .

```
Together[1/(x + 1) + 1/(x - 1) + 1/(x^2 - 1)]
```

```
1 + 2 x
-----
(-1 + x) (1 + x)
```

(4)

## 練習問題 1

部分分数分解する関数をヘルプで調べ、次の式を部分分数分解せよ。

$$\frac{3x}{x^3 - 1}.$$

■ 方程式、連立方程式を解くこともできる。式の中で、 $=$  (イコール) が 2 つ重なっていることに注意すること。

```
Solve[x2 + x - 1 == 0, x]
```

```
{ {x -> 1/2 (-1 - Sqrt[5])}, {x -> 1/2 (-1 + Sqrt[5])} }
```

```
Solve[{x2 + y - 1 == 0, y2 + x - 1 == 0}, {x, y}]
```

```
{ {x -> 0, y -> 1}, {x -> 1, y -> 0}, {x -> -1/2 - Sqrt[5]/2, y -> 1/2 (-1 - Sqrt[5])}, {x -> -1/2 + Sqrt[5]/2, y -> 1/2 (-1 + Sqrt[5])} }
```

(5)

■ Solve は代数的に解けないものは単なるシンボルを返してくる。NSolve で近似解を求めることならできる。なお、4 次以下の方程式は解の公式があるので、必ず Solve で解ける。

```
Solve[x5 + 2x + 1 == 0, x]
```

```
{ {x -> Root[1 + 2 #1 + #15&, 1]}, {x -> Root[1 + 2 #1 + #15&, 2]}, {x -> Root[1 + 2 #1 + #15&, 3]},  
  {x -> Root[1 + 2 #1 + #15&, 4]}, {x -> Root[1 + 2 #1 + #15&, 5]} }
```

```
NSolve[x5 + 2x + 1 == 0, x]
```

```
{ {x -> -0.701874 - 0.879697 i}, {x -> -0.701874 + 0.879697 i}, {x -> -0.486389},  
  {x -> 0.945068 - 0.854518 i}, {x -> 0.945068 + 0.854518 i} }
```

(6)

## 練習問題 2

$\cos x = x$  の近似解を求めるにはどうしたら良いだろうか。また、 $\cos x = x^2$  の近似解も求めてみよ。

## 計算を止める方法

非常に時間のかかる命令を Mathematica に与えると、画面上ではフリーズしたようになってしまう。ここでは、その状態から脱する方法を示す。

■ 次は、123456789 の素因数分解が  $3^2 \cdot 3607 \cdot 3803$  であることを表す。

```
FactorInteger[123456789]
```

```
{ {3, 2}, {3607, 1}, {3803, 1} }
```

(7)

さて、かの Fermat は  $2^{2^n} + 1$  は全て素数であると予想したが、Euler が  $2^{2^5} + 1$  の自明でない素因子を見付けた。

```
2^(2^5) + 1
4294967297

FactorInteger[%]
{{641, 1}, {6700417, 1}}
```

(8)

$2^{2^n} + 1$  ( $n = 0, 1, \dots, 10$ ) を素因数分解してみよう。Do[exp,{i,min, max}] は exp を i=min から i=max の範囲で繰り返し評価する。

```
Do[Print[n, " ", FactorInteger[2^(2^n) + 1], {n, 0, 10}]
```

(9)

この命令を実行すると、ノートブックのタイトルに「実行中」と記されたまま、時間だけが過ぎていくであろう。この状況から脱するには、メニューバーの「カーネル」から「評価を放棄」をクリックすればよい。

## 関数の定義と変数の扱い

### 簡単な関数の定義

■ 次のようにして関数を自分で定義することができる。\_と: を忘れないように。組み込み関数は大文字で始まるので、区別のために、自己定義関数は小文字で始めることを勧める。

```
f[x_] := x + 2

f[1]
```

(10)

■ 次の 2 つの数列は、どちらも自然対数の底  $e$  に収束する。

$$\left(1 + \frac{1}{n}\right)^n, \quad \sum_{i=0}^n \frac{1}{i!} \quad \left(= 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!}\right)$$

しかし、その収束の速さはまるで違う。次でそのことを体感せよ。

```
N[E]

a[n_] :=  $\left(1 + \frac{1}{n}\right)^n$ 

Do[Print[n, " ", N[a[n]]], {n, 1, 100}]

b[n_] := Sum[ $\frac{1}{i!}$ , {i, 0, n}]

Do[Print[n, " ", N[b[n]]], {n, 1, 10}]
```

(11)

## 変数

■ Mathematica は、高級な電卓としての機能のみならず、プログラム言語としての機能も有する。Mathematica に限らず、プログラムを組むには、変数の扱いに慣れることが必須である。

■  $x=1$  と入力すれば  $x$  に 1 を代入する。Clear[x] (または  $x=.$ ) は  $x$  を単なる変数に戻す。

```
x = 1
1

Expand[(x + y)^5]
1 + 5 y + 10 y^2 + 10 y^3 + 5 y^4 + y^5

Clear[x]

Expand[(x + y)^5]
x^5 + 5 x^4 y + 10 x^3 y^2 + 10 x^2 y^3 + 5 x y^4 + y^5
```

(12)

■  $x=1$  が  $x$  に 1 を代入する一方、 $x==1$  は  $x$  と 1 が等しい、という命題を表す。

```
x = 1
1

x == 1
True
```

(13)

■ (重要) 次が意味するところをよく理解してほしい。ここがプログラムを理解する上で一つの関門である。

```
x = x + 1
2
```

(14)

数学において、 $x = x + 1$  という式は無意味である。恒等式ではないし、方程式として見ても解が存在しない。しかし、そのような先入観は一切捨てて頂きたい。プログラム言語において、 $x=x+1$  とは、(新しい)  $x$  に (古い)  $x+1$  を代入しなさい、という命令なのである。今までは  $x$  は 1 であったから、新しい  $x$  は 2 となる。繰り返すが、イコールひとつは「左辺に右辺を代入せよ」という意味である。それ以外の意味ではイコールを重ねるのが原則である。意味が理解できたなら、もう一度  $x=x+1$  とやってみよ。どうなるか想像が付くであろう。

■ 次を実行し、違いを観察せよ (2 つ以上の命令を 1 行に書くには、間を ; (セミコロン) で区切る)。

```
x = .; y = x; x = 2; Print[y]

?y

x = 1; y = x; x = 2; Print[y]

?y
```

(15)

一つ目の例では,  $x$  が単なる変数のときに  $y=x$  としたので,  $x$  に値が代入される毎に  $y$  にもその値が代入される. 二つ目の例では,  $x$  が 1 のときに  $y=x$  としたので, これは  $y=1$  と解釈される. よって,  $x$  に新しい値が代入されても, その値は  $y$  には代入されない. これらの事情は  $?y$  とすれば確かめられるであろう.  $x$  に何か値が代入されているときに,  $y$  を常に  $x$  と等しい値にしたいならば,  $y:=x$  とすればよい. 次を試してみよ.

```
x = 1; y := x; x = 2; Print[y]
```

(16)

$y=x$  のような命令を即時評価,  $y:=x$  のような命令を遅延評価という. 最後にクリアしておこう.

```
Clear[x, y]
```

(17)

### (参考) 関数定義における即時評価と遅延評価

■ 次のようにすると, Expand による展開は即時に行われ, その計算結果を  $f[x]$  と定義する.

```
f[x_] = Expand[(1 + x)^2]
1 + 2 x + x^2

f[1 + y]
```

(18)

一方, 次のようにすると, 関数  $f$  が呼び出された時点で Expand の命令が発動するので, 結果が異なる. 通常, 遅延評価の意味で関数を定義することが多いので, 常に: (コロン) を付けておくことを勧める.

```
f[x_] := Expand[(1 + x)^2]

f[1 + y]
```

(19)

### (参考) パッケージ

Mathematica には, 最初から使える組み込み関数の他に, より専門的な組み込み関数がパッケージの中に収められている. パッケージをロードすることにより, そのパッケージの中の組み込み関数を使うことができるようになる. 例えば前節で見たように, 整数の素因数分解には時間がかかるので, より効率的な素因数分解の方法を開発することが, 現在でも研究の対象であり, その一つである Elliptic Curve Method は Mathematica のパッケージに収められている. ちなみに, 素因数分解に時間がかかることは, 現在よく使われている「暗号」の安全性の根拠になっている.

■ まず, 次の命令でパッケージ FactorIntegerECM をロードせよ.

```
<< NumberTheory`FactorIntegerECM`
```

(20)

上の命令はやや長いので, 次のようにしてもよい. ヘルプを呼び出し「アドオンとリンク」をクリックする(アド

オンはパッケージとほぼ同義語) . “標準パッケージ” → “Number Theory” → “FactorIntegerECM” と順に選び , ヘルプブラウザ上に上の命令があるので , そこをクリックして `[Shift]+[Enter]` を押す . これで関数 `FactorIntegerECM` が使用可能になる . この関数は , 入力された整数の素因子を一つだけ出力する .

ロードできたら , 続けて次を評価せよ . ただし , 一つ目の計算には少し時間がかかる . 最後の `PrimeQ` は素数かどうかを判定する . この計算結果により ,  $2^{2^8} + 1$  は 2 つの素数の積であることが分かる .

```
FactorIntegerECM[2^(2^8) + 1]
```

```
(2^(2^8) + 1) / %
```

```
PrimeQ[%]
```

(21)

$2^{2^9} + 1$  の素因数分解は , Mathematica にさせることは難しいが , 次のようになることが知られている .

$$2^{2^9+1} = 2424833 \times 7455602825647884208337395736200454918783366342657 \times 741640062627530801524787141901937474059940781097519023905821316144415759504705008092818711693940737$$

## 本日の課題

■ 次の数列を表す関数 `c[n]` を定義せよ .

$$\sum_{i=1}^n \frac{1}{i} - \log n.$$

この数列はコーシー列であり , その収束値をオイラー定数という . オイラー定数は , Mathematica では `EulerGamma` で表す . (11) を参考に , この数列がオイラー定数に収束する様子を観察せよ (収束はかなり遅いので ,  $n$  を大きく取る必要があるだろう) .

■ 上の計算が , `math2.nb` の最後に位置するようにしてセーブせよ . 次にメーラーを起動し , 次の 2 点を記入せよ .

- `FindRoot` も `NSolve` と同じく近似解を計算する関数であるが , それぞれの長所と短所を箇条書きにしてまとめよ .
- 今日の内容について何か感想を述べよ (一言でもよい) .

■ メールの Subject は「学籍番号+math2」とせよ . 例えば , 学籍番号が 6101999 ならば , 6101999+math2 となる . 署名を付け , `math2.nb` を添付し , 今日中に `j-goto` に送ること .