関数名	振る舞い	入力型	出力型
nextprime	次の素数を求める	int	int
lcm	最小公倍数	int, int	int
gcd	最大公約数	int, int	int
rand	乱数生成	int	int
isprime	素数判定	int	boolean
ifactor	素因数分解	int	string
mod	剰余	int, int	int

関数名	振る舞い	入力型	出力型
importmatrix	textファイルから 行列を読み込む	string, stiring	int
matrix	行列生成	int, int, int	array
matrixinverse	逆行列	array	string
determinant	行列式の解	array	float
trancepose	転置行列	array	string
eigenvectors	固有値, 固有ベクトル	array	string

Maple

tmp.mw

```
interface(quiet=true);
writeto("./result.txt");
nextprime(3);
writeto(terminal);
interface(quiet=false);
```

result.txt

```
5
```

test.rb

```
require './mapleruby'
a = 3
RMaple.new.nextprime(a)
```

①

mapleruby.rb

```
class RMaple
  def nextprime(a)
    a = a.to_i
    p Mapleruby.new("nextprime(#{a})").exec_i
  end
end
class Mapleruby

DATA_FILE=File.join(ENV['HOME'],'.mapleruby_rc')
  def initialize(maple_code)
    @maple_code = maple_code
    @src = get_env
    @maple_path=@src[:MAPLE_PATH]
  end
  def exec_i
    result = exec
    return result.to_i
  end
end
def exec
    code0=<<EOS
interface(quiet=true);
writeto("./result.txt");
#{@maple_code};
writeto(terminal);
interface(quiet=false);
EOS
    File.write('tmp.mw',code0)
    command="#{@maple_path} tmp.mw"
    status,stdout,stderr=systemu command
    status,stdout,stderr=systemu 'cat result.txt'
    return stdout
  end
  def get_env
    begin
      file = File.open(DATA_FILE,'r')
    rescue => evar
      p evar
      print "no resource file for mapleruby.\n"
    end
    @src = YAML.load(file.read)
    file.close
    p @src
  end
end
```

②

③

④

⑤

⑥

Ruby

```ruby
require "mapleruby/version"
require 'systemu'
require 'yaml'

class RMaple
  def matrix(a,b,c)
    p a.to_i
    p b.to_i
    p c
    puts text = "with(LinearAlgebra): matrix(#{a}, #{b}, #{c})"
    p x = Mapleruby.new(text).exec_m(b)
  end
end

class Mapleruby
  DATA_FILE=File.join(ENV['HOME'],'.mapleruby_rc')
 def initialize(maple_code)
    @maple_code = maple_code
    @src = get_env
    @maple_path=@src[:MAPLE_PATH]
  end
  def exec_m(b)
    x = exec.split("").map(&:to_i)
    x1 = x.delete_if{|i| i==0}
    result = x1.each_slice(b).to_a
    return result
  end
   def exec
    (省略)
  end
  def get_env
    (省略)
  end
end
```

出力

```
with(LinearAlgebra): matrix(3, 3, [[1, 4, 7], [2, 11, 8], [3, 6, 0]])
{:MAPLE_PATH=>"/Library/Frameworks/Maple.framework/Versions/Current/bin/maple"}
[[1, 4, 7], [2, 1, 1], [8, 3, 6]]
```

```ruby
require "mapleruby/version"
require 'systemu'
require 'yaml'

class RMaple
 def matrix(a,b,c)
    p a.to_i
    p b.to_i
    p c
    puts text = "with(LinearAlgebra): matrix(#{a}, #{b}, #{c})"
    p x = Mapleruby.new(text).exec_m(b)
  end
end

class Mapleruby
  DATA_FILE=File.join(ENV['HOME'],'.mapleruby_rc')
 def initialize(maple_code)
    @maple_code = maple_code
    @src = get_env
    @maple_path=@src[:MAPLE_PATH]
  end
  def exec_m(b)
    x = exec.split("")
    x1 = x.delete_if{|i| i == " "}
    x2 = x1.delete_if{|j| j == "\n"}
    x3 = x2.delete_if{|k| k == "]"}
    x4 = x3.delete_if{|l| l ==“["}
    x5 = x4.map(&:to_i)
    result = x5.each_slice(b).to_a
    return result
  end
  def exec
    (省略)
  end
  def get_env
    (省略)
  end
end
```

出力

```
with(LinearAlgebra): matrix(3, 3, [[1, 4, 7], [2, 11, 8], [3, 6, 0]])
{:MAPLE_PATH=>"/Library/Frameworks/Maple.framework/Versions/Current/bin/maple"}
[[1, 4, 7], [2, 1, 1], [8, 3, 6], [0]]
```

```ruby
require "mapleruby/version"
require 'systemu'
require 'yaml'

class RMaple
  def matrix(a,b,c)
    p a.to_i
    p b.to_i
    p c
    puts text = "with(LinearAlgebra): matrix(#{a}, #{b}, #{c})"
    p x = Mapleruby.new(text).exec_m(b)
  end
end

class Mapleruby
  DATA_FILE=File.join(ENV['HOME'],'.mapleruby_rc')
  def initialize(maple_code)
    @maple_code = maple_code
    @src = get_env
    @maple_path=@src[:MAPLE_PATH]
  end
  def exec_m(b)
    x = exec.gsub(/[^\d]/, " ")
    x1 = x.split(" ").map(&:to_i)
    result = x1.each_slice(b).to_a
    return result
  end
   def exec
    (省略)
  end

  def get_env
    (省略)
  end
end
```

出力結果

```
with(LinearAlgebra): matrix(3, 3, [[1, 4, 7], [2, 11, 8], [3, 6, 0]])
{:MAPLE_PATH=>"/Library/Frameworks/Maple.framework/Versions/Current/bin/maple"}
[[1, 4, 7], [2, 11, 8], [3, 6, 0]]
```

```ruby
require "mapleruby/version"
require 'systemu'
require 'yaml'

class RMaple
  def lcm(a,b)
    a = a.to_i
    b = b.to_i
    p Mapleruby.new("lcm(#{a},#{b})").exec_i
  end
  def mod(a,b)
    a = a.to_i
    b = b.to_i
    p Mapleruby.new("modp(#{a},#{b})").exec_i
  end
end

class Mapleruby
  (省略)
end
```

```ruby
require "mapleruby/version"
require 'systemu'
require 'yaml'

class RMaple
  def lcm(a,b)
    main_i :lcm, a, b
  end
  def mod(a,b)
    main_i :modp, a, b
  end
  def main_i(name,*list_a)
    p name
    p list_a
    p Mapleruby.new("#{name}",list_a).exec_i
  end
end

class Mapleruby
  (省略)
end
```

整数論

main_i
出力がint型
- nextprime
- lcm
- gcd
- rand
- mod

main_b
出力がboolean型
- isprime

main_s
出力がstring型
- ifactor

行列

main_m
出力がstring型
- matrixinverse
- trancepose
- eigenvectors

main_mf
出力がfloat型
- determinant

例外
- matrix
- importmatrix

```
/Users/eri/mapleruby/lib% cat result.txt
                                [1    2    1]
                                [           ]
                                [4    5    6]
                                [           ]
                                [7    8    9]
```

matrix関数で生成された行列①

```
/Users/eri/mapleruby/lib% cat result.txt
                              [-1/2    -5/3    7/6 ]
                              [                    ]
                              [ 1       1/3    -1/3]
                              [                    ]
                              [-1/2      1     -1/2]
```

①の行列を
matrixinverse関数を用いた結果

```
/Users/eri/mapleruby/lib% cat result.txt
[          15.28732829           ]
[                                ]
[-0.143664146 + 0.6097889260 I],
[                                ]
[-0.143664146 - 0.6097889260 I]

    [0.1603653687 , -0.9286675356 + 1.185380029 I ,

    -0.9286675356 - 1.185380029 I]

    [0.6455963335 , -0.3303739256 - 0.9609839090 I ,

    -0.3303739256 + 0.9609839090 I]

    [1. , 1. , 1.]
```

①の行列を
eigenvectors関数を用いた結果

	初期バージョン	バージョン2
利点	Mapleのプログラムに慣れた人には, 一目でMapleに送る式が分かるため 新しい関数を実装するのが容易.	一度重複分をまとめた関数を プログラムしてしまえば, その後新たに関数を実装したとしても プログラム量が少なくて済む. 将来的に見てプログラムが短くなる.
欠点	コピーアンドペーストしたような 同じ内容の関数が多くできてしまう. 結果としてプログラムが長くなる.	例外にあたる関数や, 今までに実装されてない出力を持つ 関数の場合，実装時に結局初期バージョンと同様のプログラムしなければならない.