

# Contents

<b>1</b>	<b>Functions</b>	<b>2</b>
1.1	prime – 素数判定, 素数生成	2
1.1.1	trialDivision – 試し割り算	2
1.1.2	spsp – 強擬素数テスト	2
1.1.3	smallSpsp – 小さい数に対する強擬素数テスト	2
1.1.4	millers – Miller の素数判定	3
1.1.5	millersRabin – Miller-Rabin の素数判定	3
1.1.6	lpsp – Lucas テスト	3
1.1.7	fpsp – Frobenius テスト	3
1.1.8	aprs – Jacobi 和テスト	3
1.1.9	primeq – 自動的な素数判定	4
1.1.10	prime – $n$ 番目の素数	4
1.1.11	nextPrime – 次の素数を生成	4
1.1.12	randPrime – ランダムに素数を生成	4
1.1.13	generator – 素数生成	4
1.1.14	generator_eratosthenes – Eratosthenes の篩を使っている 素数生成	5
1.1.15	primorial – 素数の積	5
1.1.16	properDivisors – 真の約数	5
1.1.17	primitive_root – 平方根	5
1.1.18	Lucas_chain – Lucas 数列	5

# Chapter 1

## Functions

### 1.1 prime – 素数判定, 素数生成

#### 1.1.1 trialDivision – 試し割り算

`trialDivision(n: integer, bound: integer/float=0) → True/False`

奇数に対する試し割り算.

bound は素数の探索範囲. もし bound が与えられ,  $n$  の平方根よりも小さいという条件のもと 1 を返せば, それは bound 以下に素因数がないことを意味する.

#### 1.1.2 spsp – 強擬素数テスト

`spsp(n: integer, base: integer, s: integer=None, t: integer=None) → True/False`

base を基にした強擬素数テスト.

$s$  と  $t$  は  $n - 1 = 2^s t$  かつ  $t$  は奇数, となるような数.

#### 1.1.3 smallSpsp – 小さい数に対する強擬素数テスト

`smallSpsp(n: integer) → True/False`

$10^{12}$  より小さい整数  $n$  に対する強擬素数テスト.

4 回の強擬素数テストによって  $10^{12}$  より小さい整数が素数かどうか決定するには十分なものである.

#### 1.1.4 miller – Miller の素数判定

`miller(n: integer) → True/False`

Miller の素数判定.

このテストは GRH のもと有効です.config を見てください.

#### 1.1.5 millerRabin – Miller-Rabin の素数判定

`millerRabin(n: integer, times: integer=20) → True/False`

Miller の素数判定.

**miller** との違いは, Miller-Rabin メソッドは早いですが確率的なアルゴリズムであり, 一方で, **miller** は GRH のもと決定性アルゴリズムとなる.

times (初期設定は 20) は繰り返しの数です. エラーの確率は多くても  $4^{-\text{times}}$  です.

#### 1.1.6 lsp – Lucas テスト

`lsp(n: integer, a: integer, b: integer) → True/False`

Lucas 擬素数テスト.

もし  $n$  がパラメータ  $a$  と  $b$  の Lucas 擬素数なら True を返す, すなわち,  $x^2 - ax + b$  についての.

#### 1.1.7 fsp – Frobenius テスト

`fsp(n: integer, a: integer, b: integer) → True/False`

Frobenius 擬素数テスト.

もし  $n$  がパラメータ  $a$  と  $b$  の Frobenius 擬素数なら True を返す, すなわち,  $x^2 - ax + b$  についての.

#### 1.1.8 apr – Jacobi 和テスト

`apr(n: integer) → True/False`

APR (Adleman-Pomerance-Rumery) 素数判定または Jacobi 和テスト.

$n$  は 32 より小さい素因数がないと仮定する. また  $n$  がいくつかの底に対する `spsp` (強擬素数テスト) を通過したと仮定する.

#### 1.1.9 `primeq` – 自動的な素数判定

`primeq(n: integer) → True/False`

素数判定に対する便利な関数.

$n$  のサイズに依存して `trialDivision`, `smallSpsp` または `apr` を使う.

#### 1.1.10 `prime` – $n$ 番目の素数

`prime(n: integer) → integer`

$n$  番目の素数を返す.

#### 1.1.11 `nextPrime` – 次の素数を生成

`nextPrime(n: integer) → integer`

与えられた整数  $n$  より大きい数の中で, 最も小さい素数を返す.

#### 1.1.12 `randPrime` – ランダムに素数を生成

`randPrime(n: integer) → integer`

10 進  $n$  桁の素数をランダムに返す.

#### 1.1.13 `generator` – 素数生成

`generator((None)) → generator`

2 から  $\infty$  までの素数を生成する (ジェネレータとして).

#### 1.1.14 generator\_eratosthenes – Eratosthenes の篩を使って いる素数生成

`generator_eratosthenes(n: integer) → generator`

Eratosthenes の篩を使って  $n$  までの素数を順に生成する.

#### 1.1.15 primonial – 素数の積

`primonial(p: integer) → integer`

以下の積を返す

$$\prod_{q \in \mathbb{P}_{\leq p}} q = 2 \cdot 3 \cdot 5 \cdots p.$$

#### 1.1.16 properDivisors – 真の約数

`properDivisors(n: integer) → list`

$n$  の真の約数を返す ( $1$  と  $n$  を除いた  $n$  の全ての約数).

小さな素数の積に対してのみ役に立つ. より一般的な場合には `proper_divisors` を使用.

出力は全ての真の約数のリスト.

#### 1.1.17 primitive\_root – 平方根

`primitive_root(p: integer) → integer`

$p$  の平方根を返す.

$p$  は奇素数でなければならない.

#### 1.1.18 Lucas\_chain – Lucas 数列

`Lucas_chain(n: integer, f: function, g: function, x_0: integer, x_1: integer) → (integer, integer)`

以下のように定義される数列  $\{x_i\}$  に対する値  $(x_n, x_{n+1})$  を返す:

$$\begin{aligned}x_{2i} &= f(x_i) \\ x_{2i+1} &= g(x_i, x_{i+1}),\end{aligned}$$

初項は `x_0, x_1`.

`f` は 1 変数の整数関数. `g` は 2 変数の整数関数.

## Examples

```
>>> prime.primeq(131)
True
>>> prime.primeq(133)
False
>>> g = prime.generator()
>>> g.next()
2
>>> g.next()
3
>>> prime.prime(10)
29
>>> prime.nextPrime(100)
101
>>> prime.primitive_root(23)
5
```

# Bibliography