Python で楽しむ初等整数論

Hayao Suzuki

PyCon mini Hiroshima 2019

October 12, 2019

Contents

自己紹介

お前誰よ

名前	Hayao Suzuki（鈴木　駿）
Twitter	@CardinalXaro
ブログ	https://xaro.hatenablog.jp/
専門	数学 (組合せ論・グラフ理論)
学位	修士（工学)、電気通信大学
仕事	株式会社アイリッジで Python プログラマをしている

自己紹介

技術書の査読

- 『Effective Python』（オライリージャパン）など
- https://xaro.hatenablog.jp/ に一覧あります。

いろんな発表

- 「SymPy による数式処理」（PyCon JP 2018）など
- https://xaro.hatenablog.jp/ に一覧あります。

初等整数論とは

整数論とは

数の性質に関する数学の一分野

初等整数論とは

代数的な手法や解析的な手法を使わない数論

初等だから簡単ですね？ よかった！

初等とは予備知識がいらないという意味であり決して簡単ではありません。

今日の発表

なぜ Python を使うのか

- 比較的素直にプログラミングできる
- 標準ライブラリも外部ライブラリも豊富にある
- Guido van Rossum 氏はアムステルダム大学で数学と計算機科学を専攻していた

今日の発表

Python で楽しむ初等整数論

- 定理の結果を Python でプログラミングしてみよう
- 定理の証明から Python でプログラミングしてみよう

でも... 難しいんでしょう？

割り算ができれば大丈夫！（個人差があります）

定理の結果を Python でプログラミングしてみよう

ピタゴラスの定理

定理 (Pythagoras)

直角三角形の斜辺の長さを c とし、それ以外の辺の長さを a, b とする。そのとき、

$$a^2 + b^2 = c^2$$

が成り立つ。

ピタゴラス数

定義（ピタゴラス数）

$a^2 + b^2 = c^2$ が成り立つ自然数の組 (a, b, c) をピタゴラス数と呼ぶ。

ピタゴラス数を計算しよう：根性編

根性で計算だ！

```
from itertools import product

for a, b, c in product(range(1, 20), repeat=3):
    if a ** 2 + b ** 2 == c ** 2:
        print(a, b, c)
```

ピタゴラス数を計算しよう：根性編

根性で計算したぞ！

3 4 5
4 3 5
5 12 13
6 8 10
8 6 10
8 15 17
9 12 15
12 5 13
12 9 15
15 8 17

既約ピタゴラス数

命題（ピタゴラス数からピタゴラス数を作る）

自然数の組 (a, b, c) がピタゴラス数ならば、任意の自然数 n に対して (na, nb, nc) もまたピタゴラス数である。

証明.

$$(na)^2 + (nb)^2 = n^2(a^2 + b^2) = (nc)^2.$$

□

定義（既約ピタゴラス数）

互いに素であるピタゴラス数を既約ピタゴラス数と呼ぶ。

既約ピタゴラス数を計算しよう

ちょっと工夫した

```
from itertools import combinations
from math import gcd

for a, b, c in combinations(range(1, 50), 3):
    if a ** 2 + b ** 2 == c ** 2 and gcd(a, b) == 1:
        print(a, b, c)
```

ちょっと工夫した

- a, b, c はそれぞれ違う数である
- a, b は互いに素である

既約ピタゴラス数を計算しよう

既約ピタゴラス数の計算

3 4 5
5 12 13
7 24 25
8 15 17
9 40 41
12 35 37
20 21 29

既約ピタゴラス数をもっと上手く計算しよう

補題

既約ピタゴラス数 (a, b, c) の a, b の一方は偶数でもう一方は奇数である。また、c は常に奇数である。

証明.

自然数の組 (a, b, c) を既約ピタゴラス数とする。
まず、a, b が共に偶数ならば、ピタゴラス数の定義から c も偶数となり、(a, b, c) は共通因数 2 を持つことになり仮定に反する。次に、a, b が共に奇数ならば、ピタゴラス数の定義から c は偶数となる。このとき、$a = 2x + 1, b = 2y + 1, c = 2z$ とすると

$$a^2 + b^2 = c^2 \Rightarrow 2(x^2 + x + y^2 + y) + 1 = 2z^2$$

となり矛盾。
よって、a, b の内、一方は偶数でもう一方は奇数である。また、ピタゴラス数の定義から c は奇数となる。 □

既約ピタゴラス数をもっと上手く計算しよう

補題

既約ピタゴラス数 (a, b, c) において、a を奇数、b を偶数とする。このとき、$(c-b)(c+b)$ は互いに素である。

証明.

自然数の組 (a, b, c) を既約ピタゴラス数とし、a を奇数、b を偶数とする。仮定から、$a^2 = c^2 - b^2 = (c-b)(c+b)$ とできる。
$c-b, c+b$ の共通因数を d とすると、d は
$(c+b)+(c-b) = 2c, (c+b)-(c-b) = 2b$ も割り切る。仮定から b, c は既約なので d は 1 または 2 である。しかし、
$(c-b)(c+b) = a^2$ であり、a は奇数なので d は 1 に他ならない。

□

既約ピタゴラス数をもっと上手く計算しよう

補題

既約ピタゴラス数 (a, b, c) において、a を奇数、b を偶数とする。このとき、$c - b, c + b$ はそれぞれ平方数となる。

証明.

自然数の組 (a, b, c) を既約ピタゴラス数とし、a を奇数、b を偶数とする。仮定から、$a^2 = c^2 - b^2 = (c - b)(c + b)$ とできる。$c - b, c + b$ は互いに素であり、かつ $(c - b)(c + b) = a^2$ なので、$(c - b)(c + b)$ は平方数となる。よって、$c - b, c + b$ がそれぞれ平方数となる。 □

既約ピタゴラス数をもっと上手く計算しよう

命題

互いに素な奇数 $s, t (s > t)$ 対して、
$a = st, b = \dfrac{s^2 - t^2}{2}, c = \dfrac{s^2 + t^2}{2}$ は既約ピタゴラス数となる。

証明.

自然数の組 (a, b, c) を既約ピタゴラス数とし、a を奇数、b を偶数とする。仮定から、$a^2 = c^2 - b^2 = (c - b)(c + b)$ とできる。$c - b, c + b$ はそれぞれ平方数なので、$c + b = s^2, c - b = t^2$ とできる。よって、

$$c = \frac{s^2 + t^2}{2}, b = \frac{s^2 - t^2}{2}$$

となる。また、

$$a = \sqrt{(c - b)(c + b)} = st$$

となる。 □

既約ピタゴラス数をもっと上手く計算しよう

上手く工夫した

```
from itertools import combinations
from math import gcd

for t, s in filter(
    lambda x: gcd(*x) == 1,
    combinations(range(1, 10, 2), r=2),
):
    print(
        s * t,
        int((s ** 2 - t ** 2) / 2),
        int((s ** 2 + t ** 2) / 2),
    )
```

既約ピタゴラス数をもっと上手く計算しよう

既約ピタゴラス数をもっと上手く計算した

3 4 5
5 12 13
7 24 25
9 40 41
15 8 17
21 20 29
35 12 37
45 28 53
63 16 65

Python で楽しむ初等整数論

定理の結果を Python でプログラミングしてみよう

- 結果をそのまま実装して具体例を観察する
- 意外な法則が見つかる（かも）

定理の証明から Python でプログラミングしてみよう

素数は無限個ある

定理 (Euclid)

素数は無限に存在する。

素数は無限個存在する。

証明.

素数は有限個あると仮定し、それを $p_1, p_2, \ldots, p_n$ とする。また、$P = \prod_{i=1}^{n} p_i + 1$ とする。このとき、P は素数が合成数のいずれかである。P が素数の場合、定義から P は $p_1, p_2, \ldots, p_n$ のいずれよりも大きい素数である。P が合成数である場合、ある素数 q で割り切れるが、P の定義から P は $p_1, p_2, \ldots, p_n$ で割り切れるので $P + 1$ を割り切ることができない。よって q は $p_1, p_2, \ldots, p_n$ とは異なる素数である。よって、P が素数、合成数のいずれであっても $p_1, p_2, \ldots, p_n$ とは異なる新たなる素数が得られる。以上より、素数は無限に存在する。 □

証明からアルゴリズムを作る

証明のポイント

- 素数のリスト $\{p_i\}$ から $P = \prod_{i=1}^{n} p_i + 1$ を計算する
- P が素数ならば、素数リストに新たに加える
- P が合成数ならば素因数分解して素数を手に入れる（既存の素数リストには存在しない）

証明からアルゴリズムを作る

素数から素数をつくる

```
from sympy import prod
from sympy.ntheory import factorint, isprime

primes = [2]

for _ in range(10):
    P = prod(primes) + 1
    if isprime(P):
        primes.append(P)
    else:
        new_prime = min(set(factorint(P, multiple=True)))
        primes.append(new_prime)
print(primes)
```

証明からアルゴリズムを作る

素数から素数をつくる

[2, 3, 7, 43, 13, 53, 5, 6221671, 38709183810571, 139, 2801]

「素数は無限個存在する」を乗り越える

定理

$p \equiv 3 \pmod 4$ である素数 p は無限に存在する。

$p \equiv 3 \pmod 4$ である素数 p は無限に存在する

証明.

$p \equiv 3 \pmod 4$ である素数は有限個あると仮定し、それを $p_1, p_2, \ldots, p_n$ とする。また、$P = 4 \prod_{i=1}^{n} p_i + 3$ とする。P が素数の場合、定義から P は $p_1, p_2, \ldots, p_n$ のいずれよりも大きい $P \equiv 3 \pmod 4$ なる素数である。

P が合成数である場合、$P = \prod_{i=1}^{m} q_i$ と素因数分解できるが、q_i の少なくとも 1 つは $q \equiv 3 \pmod 4$ となる。もし、すべての q_i が $p \equiv 1 \pmod 4$ ならば、$(4n+1)(4m+1) = 4(nm+nn+m)+1$ より $P \equiv 3 \pmod 4$ であるが、P の定義から明らかに $P \equiv 3 \pmod 4$ であるので矛盾する。

以上より、$p \equiv 3 \pmod 4$ なる素数は無限に存在する。 □

証明からアルゴリズムを作る

$p \equiv 3 \pmod 4$ な素数から $p \equiv 3 \pmod 4$ な素数をつくる

```
primes = [7]

for _ in range(5):
    P = 4 * prod(primes) + 3
    if isprime(P):
        primes.append(P)
    else:
        new_prime = min(
            p
            for p in factorint(P, multiple=True)
            if p % 4 == 3
        )
        primes.append(new_prime)
print(primes)
```

証明からアルゴリズムを作る

$p \equiv 3 \pmod 4$ な素数から $p \equiv 3 \pmod 4$ な素数をつくる

[7, 31, 67, 19, 179, 197788559]

Dirichlet の算術級数定理

定理 (Dirichlet)

a, b を互いに素な数とする。そのとき、$p \equiv b \pmod{a}$ である素数 p は無限に存在する。

証明.

複素函数論を勉強しよう！ □

Python で楽しむ初等整数論

定理の証明から Python でプログラミングしてみよう

- 構成的な証明はプログラミングできる（こともある）
- プログラミングできれば証明を理解できる（かもしれない）

ピタゴラス数（復習）

定義（ピタゴラス数）

$a^2 + b^2 = c^2$ が成り立つ自然数の組 (a, b, c) をピタゴラス数と呼ぶ。

定義（既約ピタゴラス数）

互いに素であるピタゴラス数を既約ピタゴラス数と呼ぶ。

既約ピタゴラス数の斜辺の性質

既約ピタゴラス数の斜辺を 4 で割ると...?

```
from itertools import combinations
from math import gcd

for t, s in filter(
    lambda x: gcd(*x) == 1,
    combinations(range(1, 10, 2), r=2),
):
    c = int((s ** 2 + t ** 2) / 2)
    print(f"{c} を 4 で割った余りは{c % 4}")
```

既約ピタゴラス数の斜辺の性質

既約ピタゴラス数の斜辺を 4 で割ると...?

5 を 4 で割った余りは 1
13 を 4 で割った余りは 1
25 を 4 で割った余りは 1
41 を 4 で割った余りは 1
17 を 4 で割った余りは 1
29 を 4 で割った余りは 1
37 を 4 で割った余りは 1
53 を 4 で割った余りは 1
65 を 4 で割った余りは 1

2 つの平方数の和で表せる素数

定理 (Fermat の 2 平方和定理)

奇素数 p が 2 つの平方数の和で表せることの必要十分条件は $p \equiv 1 \pmod 4$ である。

系 (Fermat の 2 平方和定理の系)

自然数 c が既約ピタゴラス数 (a, b, c) となる必要十分条件は c が $p \equiv 1 \pmod 4$ な素数の積になることである。

まとめ

定理の結果を Python でプログラミングしてみよう

- 結果をそのまま実装して具体例を観察する
- 意外な法則が見つかる（かも）

定理の証明から Python でプログラミングしてみよう

- 構成的な証明はプログラミングできる（こともある）
- プログラミングできれば証明を理解できる（かもしれない）