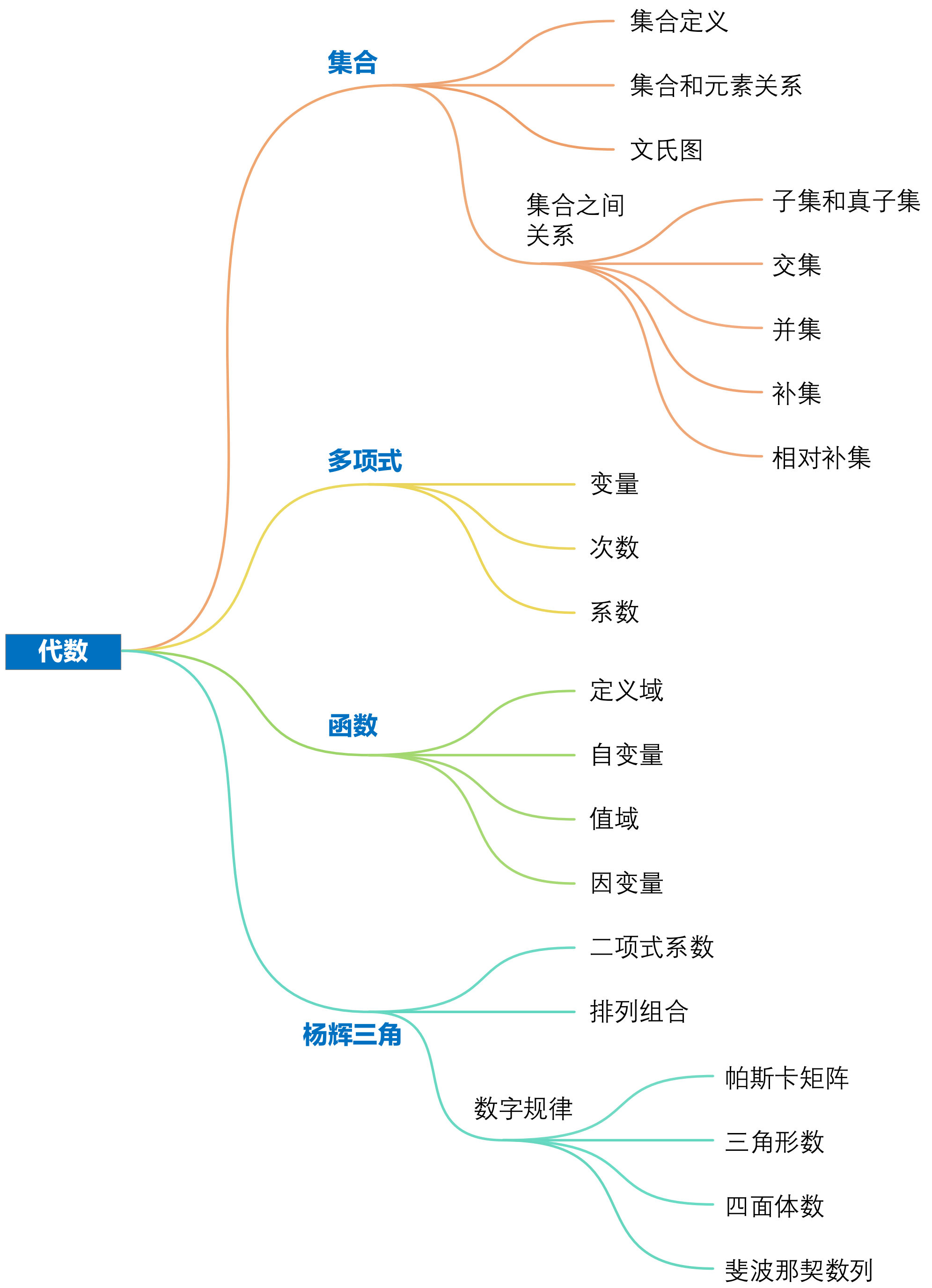
4 Algebra 代数

代数不过是公式化的几何

> 代数不过是公式化的几何；几何不过是图形化的代数。
>
> ***Algebra is but written geometry and geometry is but figured algebra.***
>
> —— 索菲·热尔曼 (Sophie Germain) | 法国女性数学家 | 1776 ~ 1831

- `difference()` 计算集合的相对补集
- `interaction()` 计算集合的交集
- `numpy.roots()` 多项式求根
- `set()` 构造集合
- `subs()` 符号代数式中替换
- `sympy.abc` 引入符号变量
- `sympy.collect()` 合并同类项
- `sympy.cos()` 符号运算中余弦
- `sympy.expand()` 展开代数式
- `sympy.factor()` 对代数式进行因式分解
- `sympy.simplify()` 简化代数式
- `sympy.sin()` 符号运算中正弦
- `sympy.solvers.solve()` 符号方程求根
- `sympy.symbols()` 定义符号变量
- `sympy.utilities.lambdify.lambdify()` 将符号代数式转化为函数
- `union()` 计算集合的并集


代数
集合
集合定义
集合和元素关系
文氏图
集合之间关系
子集和真子集
交集
并集
补集
相对补集
多项式
变量
次数
系数
函数
定义域
自变量
值域
因变量
杨辉三角
二项式系数
排列组合
数字规律
帕斯卡矩阵
三角形数
四面体数
斐波那契数列

4.1 代数的前世今生：薪火相传

思想的传播像火种的接续传递——首先是星星之火，然后是闪烁的炬火，最后是燎原烈焰，排山倒海、势不可挡。

亚历山大图书馆是古希腊最重要的图书馆，也是一个学术和文化中心；亚历山大图书馆，并不在希腊本图，而是在埃及境内。

公元前 47 年，亚历山大图书馆失火，大部分馆藏经典被焚毁。公元 529 年，柏拉图学园和其他所有雅典学校都被迫关闭。

可以想象，柏拉图学园断壁残垣，杂草丛生，物是人非；巢倾卵破，数学家、哲学家们鸟兽散去，远走他乡，衣食无着，晚景凄凉。

古希腊学术圣火如风中之烛，渐渐燃灭，欧洲则一步步陷入漫漫暗夜。

庆幸的是，西方不亮东方亮；希腊典籍被翻译成阿拉伯语，人类思想的火种在另外一个避风港湾得以保全——巴格达“**智慧宫** (House of Wisdom)”。

在智慧宫，东西方科技知识交融发展。在九世纪至十三世纪，智慧宫可以说是整个世界举足轻重的教育学术机构。

花拉子密 (Muhammad ibn Musa al-Khwarizmi) 是一位波斯数学家、智慧宫的代表性学者。花拉子密在约 820 年，创作完成《代数学》 (*Al-Jabr*)，代数学自此成为一门独立学科；他第一次系统性求解一次方程及一元二次方程，因而被称为“代数之父”。英文中的代数 algebra 一词源自 Al-Jabr。值得一提的是，“算法”的英文 algorithm 一词来自于花拉子密 (al-Khwarizmi) 的名字。

图 1. 花拉子密《代数学》封面

印度十进制数字系统就是在阿拉伯进一步发展，并引入欧洲；因此，十进制数字也被称作阿拉伯数字。

好景不长，1258 年蒙古帝国军队的铁蹄大张挞伐，洗劫巴格达，焚毁智慧宫；据说，智慧宫珍贵藏书被丢弃在底格里斯河，河水被染黑长达六个月之久。

代码及 PDF 文件下载：https://github.com/Visualize-ML
本书配套微课视频均发布在 B 站——生姜 DrGinger：https://space.bilibili.com/513194466
欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

相比玉楼金殿、奇珍异宝，记录人类智慧的捆捆羊皮卷、叠叠莎草纸可能显得一分不值；这盏风中摇曳的烛火在两河流域被生生掐灭。

值得宽慰的是，十一世纪开始，十字远征军一次次远征，从阿拉伯人手中取回科学的火种，欧洲也渐渐从几百年的暗夜中苏醒。

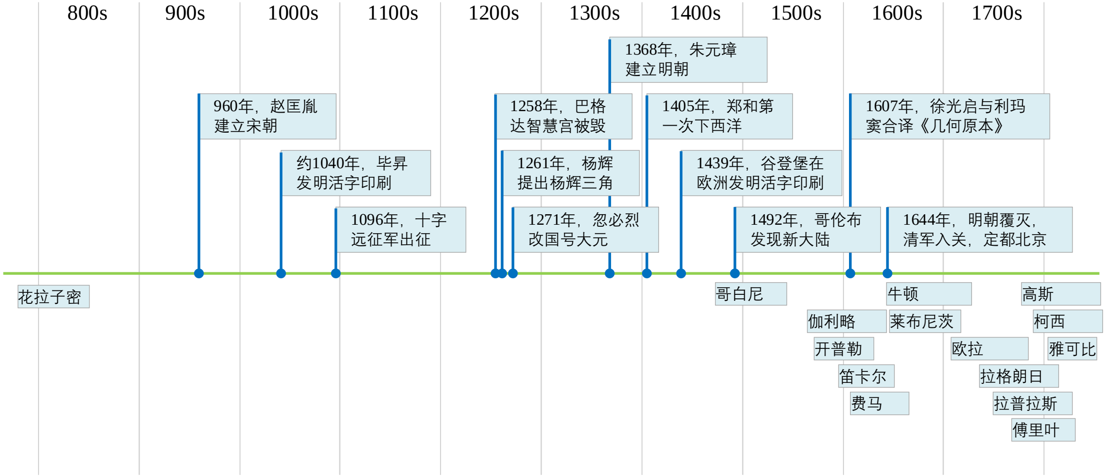


图 2. 西方数学复兴时间轴

十字远征军带回来不仅仅是古希腊的典籍，还有古印度的数学、古代中国的技术发明。这些科学知识在欧洲传播、发展，最终燃成人类思想的烈焰。

这片思想的火海中绽放出众多绚丽的火焰——伽利略、开普勒、笛卡尔、费马、帕斯卡、牛顿、莱布尼兹、伯努利、欧拉、拉格朗日、拉普拉斯、傅里叶、高斯、柯西 ...。本系列丛书会提到他们的名字，以及他们给后来者留下的宝贵知识火种。

4.2 集合：确定的一堆东西

本节回顾集合这个概念。相信本书读者对**集合** (set) 的概念已经不陌生，我们在本书第一章介绍过复数集、实数集、有理数集等等，它们都是集合。

集合是由一个或多个确定的**元素** (member 或 element) 所构成的整体。集合可以分为：**有限元素集合** (finite set)、**无限元素集合** (infinite set) 和**空集** (empty set 或 null set)。

集合与元素

集合与元素的关系有两种：**属于** (belong to) $\in$，**不属于** (not belong to) $\notin$。

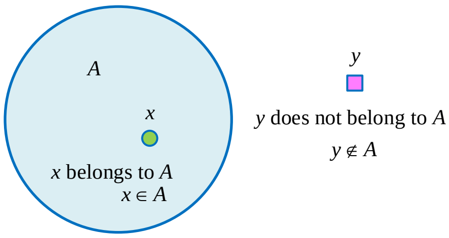


图 3. 属于和不属于

表 1. 集合与元素关系的中英文表达

英文表达	中文表达
x belongs to capital A.	x 属于 A。
x is a member/element of the set capital A.	x 是集合 A 元素。
x is/lies in the set capital A.	x 在集合 A 之内。
The set capital A includes x.	集合 A 包含 x。
y does not belong to the set capital A.	y 不属于集合 A。
y is not a member of the set capital A.	y 不是集合 A 元素。

集合与集合

如果集合 A 中的每一个元素也都是集合 B 中的元素，那么 A 是 B 的**子集** (subset)， 记作 $A \subseteq B$； 而 B 是 A 的**母集**，也称**超集** (superset)。如果同时满足 $A \subseteq B$ 和 $A \neq B$， 则称 A 是 B 的**真子集** (A is a proper subset of B)， 记作 $A \subset B$ 。

给定 A 和 B 两个集合，由所有属于 A 且属于 B 的元素所组成的集合，叫做 A 与 B 的**交集** (intersection)，记作 $A \cap B$ 。把 A 和 B 所有的元素合并组成的集合，叫做 A 和 B 的**并集** (union)，记作 $A \cup B$ 。

补集 (complement) 一般指**绝对补集** (absolute complement)；设 Ω 是一个集合，A 是 Ω 的一个子集，由 Ω 中所有不属于 A 的元素组成的集合，叫做子集 A 在 Ω 中的绝对补集。

A 中 B 的**相对补集** (relative complement)，是所有属于 A 但不属于 B 的元素组成的集合，记做 $A \backslash B$ 或 $A - B$ (set difference of A and B)，也可以读作“B 在 A 中的相对补集 (the relative complement of B with respect to set A)”。

表 2. 集合与集合关系英文表达

数学表达	英文表达
$A \subset B$	The set capital A is a subset of the set capital B. The set capital B is a superset of the set capital A. The set capital A is contained in the set capital B.
$A \subseteq B$	The set capital A is a subset of or equal to the set capital B.
$A \supset B$	The set capital A contains the set capital B.
$A \supseteq B$	The set capital A contains or is equal to the set capital B.
$A \cap B$	The intersection of the set capital A and the set capital B. A intersection B The intersection of A and B
$A \cup B$	The union of the set capital A and the set capital B. Capital A union capital B.

	The union of A and B.
$\overline{A}$	The complement of the set capital A.
$A-B$	The relative complement of the set capital B in the set capital A. The relative complement of set capital B with respect to set capital A.
$A\cap(B\cup C)$	The intersection of capital A and the set capital B union capital C.
$\overline{(A\cup B)}$	The complement of the set capital A union capital B.
$\overline{A}\cap\overline{B}$	The intersection of the complement of capital A and the complement of capital B.

集合之间的关系也可以用**文氏图** (Venn diagram) 来表达。图 4 给出的是两个集合常见关系文氏图。

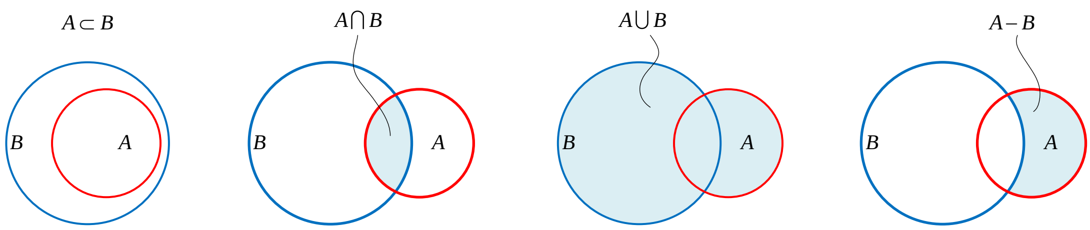


图 4. 两个集合关系文氏图

掷骰子

举个例子，如图 5 所示，掷一枚色子，点数结果构成的集合 Ω 为。

$$\Omega=\{1,2,3,4,5,6\} \tag{1}$$

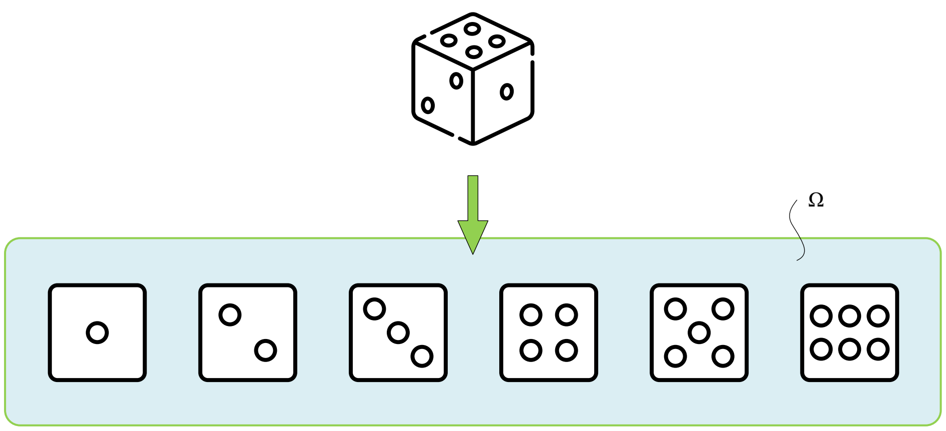


图 5. 投一枚色子点数结果

定义集合 A 为色子点数为奇数。

$$A=\{1,3,5\} \tag{2}$$

集合 B 为色子点数为偶数。

$$B = \{2,4,6\} \tag{3}$$

集合 C 为色子点数小于 4。

$$C = \{1,2,3\} \tag{4}$$

图 6 所示为 A、B、C 三个集合关系。

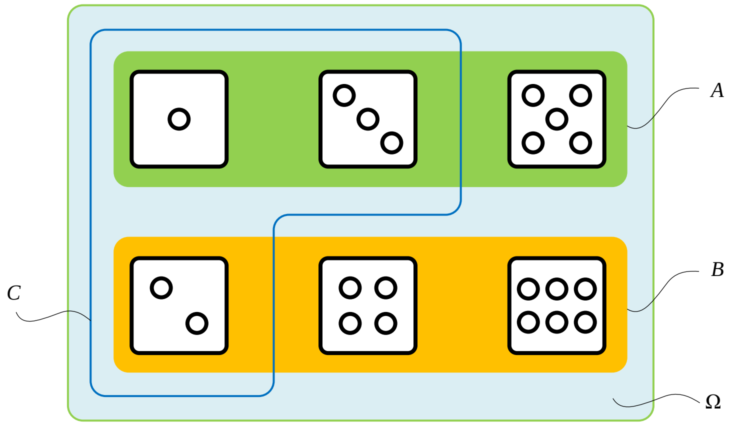


图 6. 色子点数 A、B、C 三个集合关系

显然，A 和 B 的交集为空，即。

$$A \cap B = \varnothing \tag{5}$$

A 和 B 的并集为全集 Ω，即。

$$A \cup B = \Omega = \{1,2,3,4,5,6\} \tag{6}$$

也就是说，A 在 Ω 中的绝对补集为 B。

A 和 C 的交集有两个元素。

$$A \cap C = \{1,3\} \tag{7}$$

A 和 C 的并集有四个元素，即。

$$A \cup C = \{1,2,3,5\} \tag{8}$$

A 中 C 的相对补集为。

$$A - C = \{5\} \tag{9}$$

C 中 A 的相对补集为。

$$C - A = \{2\} \tag{10}$$

下面用 Python 来完成上述计算。

```
# Bk_Ch4_01

# set A: odd
A = set([1,3,5])

# set B: even
B = set([2,4,6])

# set C: less than 4
C = set([1,2,3])

# A union B
A_union_B = A.union(B) #A|B, A or B

# A intersects (meets) B
A_meet_B = A.intersection(B) #A&B, A and B

# A minus B
A_minus_B = A.difference(B) # A - B, set difference

# A union C
A_union_C = A.union(C) #A|C, A or C

# A intersects (meets) C
A_meet_C = A.intersection(C) #A&C, A and C

# A minus C
A_minus_C = A.difference(C) # A - C, set difference

# C minus A
C_minus_A = C.difference(A) # C - A, set difference
```

4.3 从代数式到函数

算数 (arithmetic) 基于**已知量** (known values)；而**代数** (algebra) 基于**未知量** (unknown values)，也称作**变量** (variables)。当然，代数中既有数字也有字母。

现代人一般用 a、b、c 等代表常数，用 x、y、z 等代表未知量；这种记法正是约 400 年前笛卡尔提出的。

引入未知量这种数学工具，有助于将数学问题抽象化、一般化。也就是说，2 + 1、6 + 12、100 + 150 等，都可以抽象地写成 $x + y$。

如图 7 所示，这五个圆形虽然大小明显不同；但是，引入半径 r 这个变量，这些圆形的周长都可以写成 $2\pi r$，面积可以写成 πr^2。将不同的 r 值代入代数式，便可以求得对应圆的周长和面积。

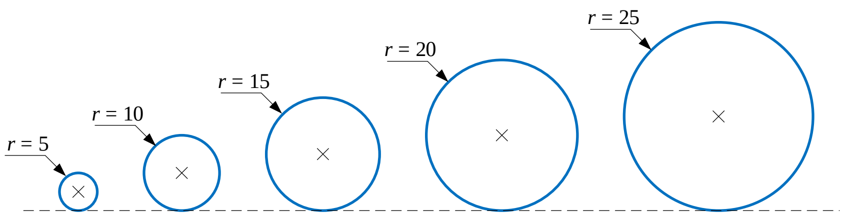


图 7. 不同半径的圆形

多项式

本书最常见的代数式是**多项式** (polynomial)，形如。

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \tag{11}$$

其中，x 为**变量** (variable)，n 为**多项式次数** (degree of a polynomial)，a_0、a_1 … a_n 为**系数** (coefficient)。

系数之所以会使用**下角标** (subscript)，是因为字母不够用；类似地，变量多时，x、y、z 肯定不够用，本系列丛书会用变量加下角标序号来表达变量，比如 x_1、x_2、x_3、x_i、x_j 等等。

多项式是由一个个单项式加减构成。由数和字母的积组成的代数式叫做**单项式** (monomial)，比如 $a_n x^n$；单独的一个数或一个字母也叫做单项式，比如 a_0 和 5。

一个单项式中，所有变量指数之和，叫做这个单项式的次数；比如，$3x^5$ 的次数是 5，$2xy$ 的次数 2 (= 1 + 1)。

只有一个变量的多项式被称作**一元多项式** (univariate polynomial)。变量多于一个的多项式统称**多元多项式** (multivariate polynomials)；其中，有两个变量的多项式常被称作**二元多项式** (bivariate polynomial)。

最高项次数较小的多项式都有特殊的名字，比如：**常数式** (constant equation)、**一次式** (linear equation)、**二次式** (quadratic equation)、**三次式** (cubic equation)、**四次式** (quartic equation) 和**五次式** (quintic equation) 等等。常见多项式及名称总结于表 3 中。

表 3. 常见多项式举例

次数	英文名称	例子
1	linear	$ax+b,\quad a \neq 0$
2	quadratic	$ax^2+bx+c,\quad a \neq 0$
3	cubic	$ax^3+bx^2+cx+d,\quad a \neq 0$
4	quartic	$ax^4+bx^3+cx^2+dx+e,\quad a \neq 0$

5	quintic	$ax^5+bx^4+cx^3+dx^2+ex+f, \quad a \neq 0$

给定如下变量为 x 的三次式。

$$x^3+2x^2-x-2 \tag{12}$$

令 $x = 1$，代入 (12)，得到如下结果。

$$1^3+2\times1^2-1-2=0 \tag{13}$$

编写代码用 SymPy 中函数来完成上述计算。其中 sympy.abc 引入符号变量 x 和 y。SymPy 是重要的符号运算库，本书将会用到其中的代数式定义、求根、求极限、求导、求积分、级数展开等功能。

```
# Bk_Ch4_02_A
from sympy.abc import x,y
expr = x**3 + 2*x**2 - x - 2
print(expr.subs(x,1))
```

同样，可以利用.subs() 将 (12) 中的 x 替换成其他符号变量、甚至代数表达式，比如 $x = \cos(y)$。

$$\left(\cos(y)\right)^3+2\left(\cos(y)\right)^2-\cos(y)-2 \tag{14}$$

接着上述代码，以下代码完成变量替换，将 x 替换为 cos(y)。

```
# Bk_Ch4_02_B
from sympy import cos
expr_cos_y = expr_x.subs(x,cos(y))
print(expr_cos_y)
```

也可以使用 sympy.symbols() 定义更复杂的符号变量，比如下例。

```
# Bk_Ch4_02_C
from sympy import symbols
x,y = symbols('x,y')
```

```
expr_1 = x + y
print(expr_1)

x1,x2 = symbols('x1,x2')
expr_2 = x1 + x2
print(expr_2)
```

再介绍几个 SymPy 中处理代数式的函数。sympy.simplify() 函数可以简化代数式；sympy.expand() 可以用来展开代数式；sympy.factor() 函数则可以对代数式进行因式分解；sympy.collect() 函数用来合并同类项。请大家自行学习如下代码。

```
# Bk_Ch4_03

from sympy import *
x,y,z = symbols('x y z')

# simplify mathematical expressions
expr_1 = sin(x)**2 - cos(x)**2
print(simplify(expr_1))

# expand polynomial expressions
expr_2 = (x + 1)**3
print(expand(expr_2))

# take a polynomial and factors it into irreducible factors
expr_3 = x**3 + 2*x**2 - x - 2
print(factor(expr_3))

# collect common powers of a term in an expression
expr_collected = collect(expr_3 - x**2 - 2*x, x)
print(expr_collected)
```

表 4 总结常用代数式的英文表达。

表 4. 常用代数英文表达

数学表达	英文表达
$x-y$	*x* minus *y*
$-x-y-z$	minus *x* minus *y* minus *z*
$x-(y-z)$	*x* minus the quantity *y* minus *z*. *x* minus open parenthesis *y* minus *z* close parenthesis.
$x-(y+z)-t$	*x* minus the quantity *y* plus *z* end of quantity minus *t*. *x* minus open parenthesis minus *y* plus *z* close parenthesis minus *t*.
$x(x-y+z)$	*x* times the quantity *x* minus *y* plus *z*. *x* times open parenthesis *x* minus *y* plus *z* close parenthesis.
$(x+y)^2$	*x* plus *y* all squared
x^3	*x* to the third *x* to the third power *x* raised to the power of three *x* cubed
$(x+y)(z+t)$	The sum *x* plus *y* times the sum *z* plus *t*. The product of the sum *x* plus *y* and the sum *z* plus *t*. Open parenthesis *x* plus *y* close parenthesis times open parenthesis *z* plus *t* close parenthesis.
$\left(\frac{x}{y}\right)^2$	*x* over *y* all squared

$\frac{x+z}{y}$	The quantity of x plus z divided by y.
$x+\frac{z}{y}$	x plus the fraction z over y.
$t\left(z+\frac{x}{y}\right)$	t times the sum z plus the fraction x over y.

函数

函数 (function) 一词由莱布尼兹 (Gottfried Wilhelm Leibniz) 引入；而 $f(x)$ 这个函数记号由**欧拉** (Leonhard Paul Euler) 发明。欧拉还引入三角函数现代符号，以 e 表记自然对数的底，用希腊字母 Σ 表记累加和以 i 表示虚数单位。

中文“函数”则是由清朝数学家李善兰 (1810 ~ 1882) 翻译。代数、系数、指数、多项式等数学名词中文翻译也是出自李善兰之手。

给定一个集合 X，对 X 中元素 x 施加映射法则 f，记作函数 $f(x)$，得到的结果 $y = f(x)$ 属于集合 Y。集合 X 称作**定义域** (domain)，Y 称为**值域** (codomain)；x 称作**自变量** (an argument of a function 或 an independent variable)，y 称作**因变量** (dependent variable)。

大家应该已经发现，函数 $f: X \to Y$ 有三个关键要素：定义域 X、值域 Y 和函数映射规则 f。

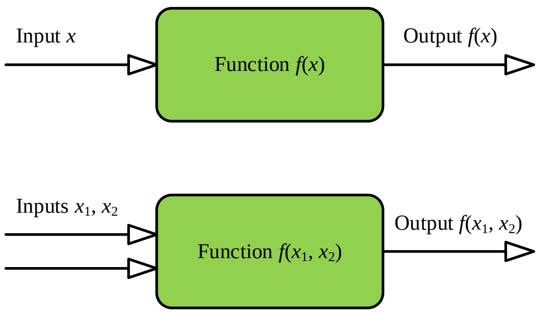


图 8. 一元函数、二元函数的映射

函数的自变量为两个或两个以上时，叫做**多元函数** (multivariate function)。本丛书一般会使用 x 加下角标序号来表达多个自变量，比如 $f(x_1, x_2, \ldots, x_D)$ 函数有 D 个自变量。

为了方便将不同的 x 值代入 (12)，我们可以定义一个函数 $f(x)$。

$$f(x) = x^3 + 2x^2 - x - 2 \tag{15}$$

以下代码将代数式转化为函数，并给 x 赋值得到函数值。

```
# Bk_Ch4_04

from sympy.abc import x, y, z
```

```
expr = x**3 + 2*x**2 - x - 2

from sympy.utilities.lambdify import lambdify
f_x = lambdify(x, expr)

print(f_x(1))
```

表 5 给出有关函数常用英文表达。

表 5. 常用函数英文表达

数学表达	英文表达
$f(x)$	*f x* *f* of *x* the function *f* of *x*
$f(g(x))$	*f* composed with *g* of *x* *f* of *g* of *x*
$f\circ g(x)$	*f* composed with *g* of *x* *f* of *g* of *x*
$f(x+a)$	*f* of the quantity *x* plus *a*
$f(x,y)$	*f* of *x*, *y*
$f(x_1,x_2,\dots,x_n)$	*f* of *x* sub one, *x* sub two, dot dot dot, *x* sub *n*
$f(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$	*f* of *x* equals *a* sub *n* times *x* to the *n*, plus *a* sub *n* minus one times *x* to the *n* minus one, plus dot dot dot, plus *a* sub one times *x*, plus *a* sub zero. *f* of *x* equals *a* sub *n* times *x* raised to the power of *n*, plus *a* sub *n* minus one times *x* raised to the power of *n* minus one, plus dot dot dot, plus *a* sub one times *x*, plus *a* sub zero.
$f(x)=3x+5$	*f* of *x* equals three times x plus five.
$f(x)=x^2+2x+1$	*f* of *x* equals x squared plus two times *x* plus one.
$f(x)=x^3-x+1$	*f* of *x* equals *x* cubed minus *x* plus one.

为了进一步探讨函数性质，我们亟需一个重要的数学工具——**坐标系** (coordinate system)；坐标系是下一章要探讨的内容。

4.4 杨辉三角：代数和几何的完美合体

杨辉三角，也称贾宪三角，又称**帕斯卡三角** (Pascal's triangle)，是二项式系数的一种写法。

如下，$(x + 1)^n$ 展开后，按单项 x 的次数从高到低排列，发现单项式系数呈现出特定规律。

$$
\begin{aligned}
(x+1)^0 &= 1 \\
(x+1)^1 &= x+1 \\
(x+1)^2 &= x^2+2x+1 \\
(x+1)^3 &= x^3+3x^2+3x+1 \\
(x+1)^4 &= x^4+4x^3+6x^2+4x+1 \\
(x+1)^5 &= x^5+5x^4+10x^3+10x^2+5x+1 \\
(x+1)^6 &= x^6+6x^5+15x^4+20x^3+15x^2+6x+1 \\
&\cdots
\end{aligned}
\tag{16}
$$

图 9 将 (16) 单项式系数以金字塔的结构展示，请读者注意以下规律。

- 三角形系数呈现对称性，第 k 行有 $k + 1$ 个系数；
- 三角形每一行左右最外侧系数为 1；
- 除最外两侧系数以外，三角形内部任意系数为左上方和右上方两个系数之和；
- 第 k 行系数之和为 2^k。

杨辉三角中，我们将会看到几何、代数、概率等知识的有趣联系。

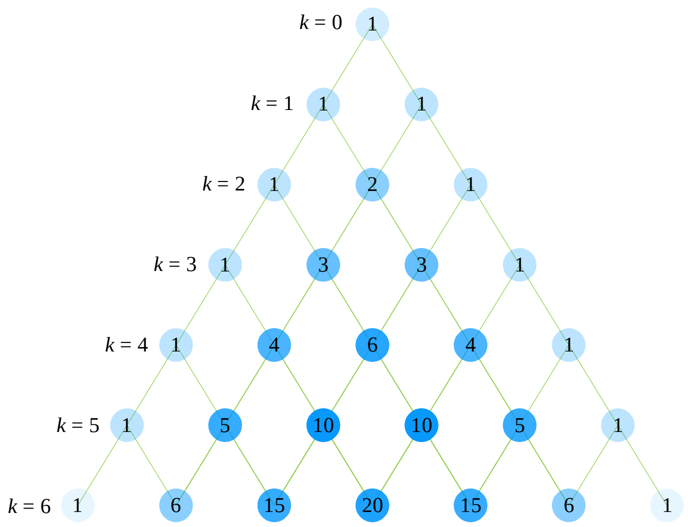


图 9. 杨辉三角

比帕斯卡提前 300 年左右，杨辉在自己书中介绍了这个数字规律。杨辉也不是第一位发现者，他在书中也说的很清楚，这个规律引自贾宪的一部叫做《释锁算术》的数学作品。

按照时间先后顺序，贾宪在 11 世纪北宋时期就发现并推广了这一规律，杨辉只是在 13 世纪南宋时期对其进行了再次的解释，而帕斯卡等到 1655 年在自己的作品中介绍二项式系数规律。

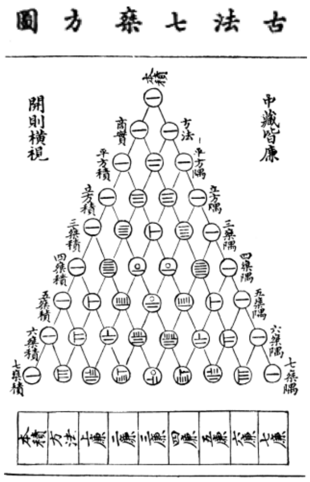


图 10. 元代数学家朱世杰《四元玉鉴》中绘制的杨辉三角

火柴梗图 (stem plot) 方便可视化杨辉三角每行单项式系数的规律。图 11 所示为 $n = 4$、8、12 时，二项式展开单项系数规律。

火柴梗图明显呈现中心对称性；n 为偶数时，对称轴处系数最大；如图 11 所示，n 为奇数时，对称轴附近两个系数为最大值。对称轴左右两侧系数先快速减小，然后再缓慢减小。

随着 n 增大，这一现象更加明显，如图 13 所示。图 13 中，一条优化的曲线呼之欲出，这条曲线就是**高斯分布** (Gaussian distribution) **概率密度函数** (Probability Density Function, PDF)。本书后面会简要介绍这条曲线，本系列丛书《概率统计》一册会从各种角度介绍高斯分布。

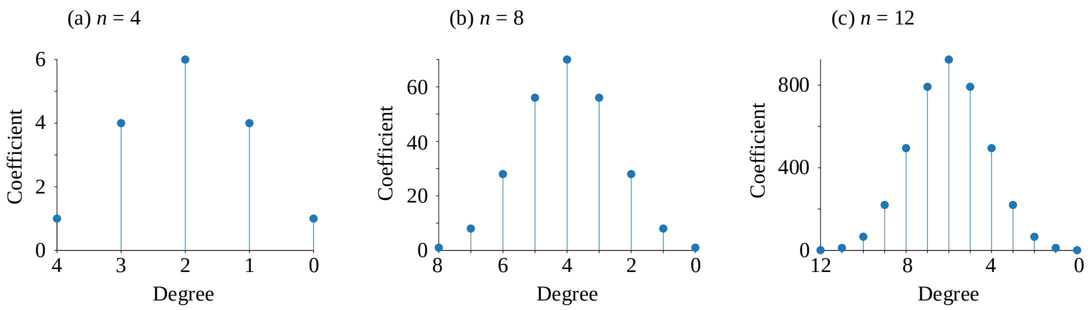


图 11. $n = 4$、8、12 等偶数时，二项式展开单项系数

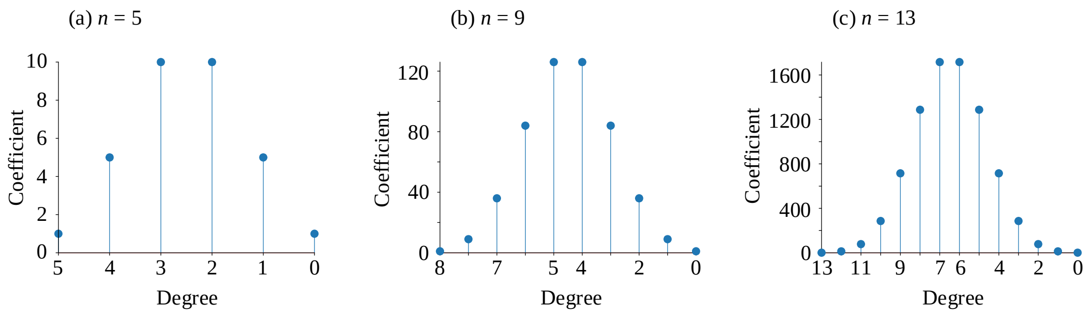


图 12. $n = 5$、9、13 等奇数时，二项式展开单项系数

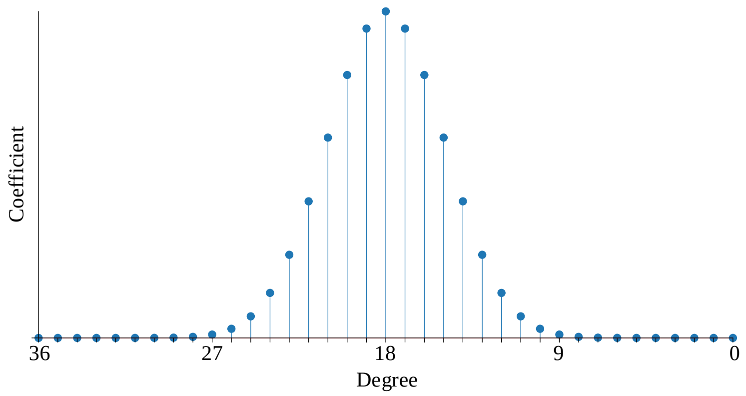


图 13. n = 36 时，二项式展开单项系数

以下代码可以用来产生图 11 和图 13 两图。

```
# Bk_Ch4_05

from sympy.abc import x, a
from sympy import Poly
import matplotlib.pyplot as plt
import numpy as np

for n in [4, 8, 12, 5, 9, 13, 36]:

    expr = (x + 1)**n

    expr_expand = expr.expand()
    expr_expand = Poly(expr_expand)

    poly_coeffs = expr_expand.coeffs()

    print(poly_coeffs)

    degrees = np.linspace(n,0,n + 1)

    fig, ax = plt.subplots()

    plt.stem(degrees, poly_coeffs)
    plt.show()
    plt.xlim(0,n)
    plt.xticks(np.arange(0,n+1))
    y_max = max(poly_coeffs)
    y_max = float(y_max)
    plt.ylim(0,y_max)

    ax.spines['right'].set_visible(False)
    ax.spines['top'].set_visible(False)
    ax.invert_xaxis()
    plt.xlabel('Degree')
    plt.ylabel('Coefficient')
```

4.5 排列组合让系数更具意义

组合数

从 n 个不同元素中，取 m $(m \le n)$ 个元素构成一组，被称作 n 个不同元素中取出 m 个元素的一个**组合** (combination)；注意，组内的元素排序并不重要。

n 个不同元素中取出 m 个元素的所有组合个数叫做组合数，常记做 C_n^m 。

$$C_n^m = C(n,m) = \frac{n!}{(n-m)!m!} \tag{17}$$

其中，! 运算符就是本书前文介绍的**阶乘** (factorial)。

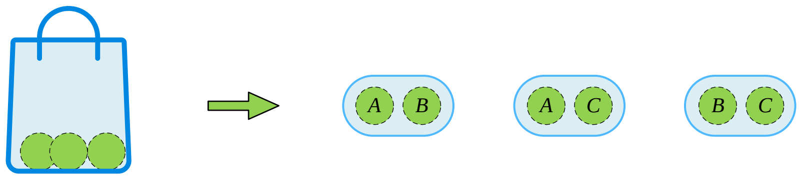


图 14. A、B、C 三个元素无放回抽取两个，结果有三个组合

举个例子，如图 14 所示，从 A、B、C 三个元素无放回抽取两个，只要元素相同，不管次序是否相同都算作相同结果。结果有三个组合 AB、AC、BC，对应的组合数为。

$$C_3^2 = \frac{3!}{(3-2)!2!} = \frac{6}{2} = 3 \tag{18}$$

以下代码完成上述无放回抽取组合实验。

```
# Bk_Ch4_06

import itertools

letters = "ABC"

# find all combinations containing 2 letters

cmb = itertools.combinations(letters, 2)

for val in cmb:
    print(val)
```

结果如下。

```
('A', 'B')
```

```
('A', 'C')
('B', 'C')
```

组合数表达杨辉三角

用组合数可以将 $(x+y)^n$ 展开写成。

$$(x+y)^n = C_n^0 x^n y^0 + C_n^1 x^{n-1} y^1 + C_n^2 x^{n-2} y^2 + \cdots + C_n^{n-2} x^2 y^{n-2} + C_n^{n-1} x^1 y^{n-1} + C_n^0 x^0 y^n \tag{19}$$

因此，杨辉三角还可以如图 15 这样写。

组合数方便解释 (19) 中各项系数。每一项 x 和 y 的次数之和为 n，如果某一单项 y 的次数为 k，x 的次数为 $n-k$，这一项为 $C_n^k x^{n-k} y^k$；该项系数为 C_n^k，相当于在 n 个 x 或 y 连乘中，选取 k 个为 y。

将 $x = y = 1$ 代入 (19)，可以发现组合数的一个重要规律。

$$2^n = C_n^0 + C_n^1 + C_n^2 + \cdots + C_n^{n-2} + C_n^{n-1} + C_n^0 \tag{20}$$

观察图 15 的对称性，容易发现另外一个组合数规律。

$$C_n^k = C_n^{n-k} \tag{21}$$

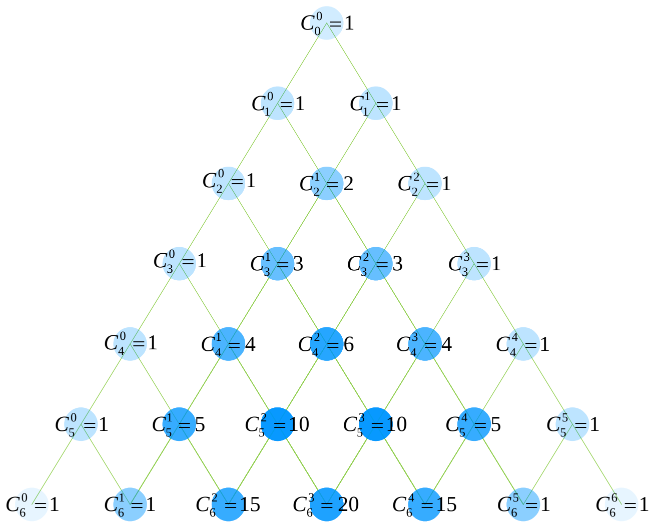


图 15. 用组合数来写杨辉三角

排列数

从 n 个不同元素中，先后取 m ($m \le n$) 个元素排成一列，叫做从 n 个元素中取出 m 个元素的一个**排列** (permutation)。排列中，元素的排序很重要。

n 个不同元素中取出 m 个元素的所有排列的个数叫做排列数，常记做 P_n^m。

$$P_n^m = P(n,m) = \frac{n!}{(n-m)!} \tag{22}$$

同样，如图 16 所示，从 A、B、C 三个元素无放回先后抽取两个，结果有 6 个排列 AB、BA、AC、CA、BC、CB，即。

$$P_3^2 = \frac{3!}{(3-2)!} = 6 \tag{23}$$

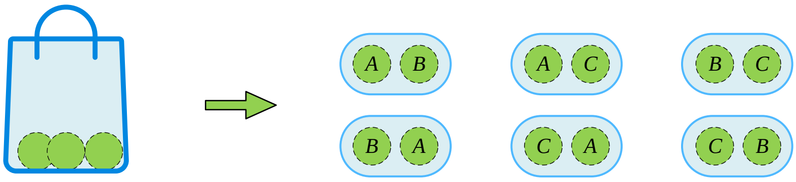


图 16. A、B、C 三个元素无放回抽取两个，结果有 6 个排列

以下代码完成上述无放回排列实验。

```
# Bk_Ch4_07

import itertools

letters = "ABC"

# Arranging 2 elements out of 3

per = itertools.permutations(letters, 2)

for val in per:
    print(val)
```

结果为。

```
('A', 'B')
('A', 'C')
('B', 'A')
('B', 'C')
('C', 'A')
('C', 'B')
```

组合数和排列数

比较 (17) 和 (22)，发现排列和组合的关系为。

$$P_n^m = C_n^m \cdot m! \tag{24}$$

可以这样解释上式，先从 n 个元素取出 m 进行组合，组合数为 C_n^m；然后，把 m 个元素全部进行排列 (也叫全排列)，排列数为 $m!$。这样，C_n^m 和 $m!$乘积便是 n 个元素取出 m 的排列数。

$m = n$ 时，所有的排列情况叫全排列。

$$P_n^n = P(n,n) = n! \tag{25}$$

从 A、B、C 三个元素全排列的结果为 ABC、ACB、BAC、BCA、CAB、CBA。以下代码完成上述计算并打印全排列结果。

```python
# Bk_Ch4_08

import itertools

letters = "ABC"

# Arranging all 3 elements

per = itertools.permutations(letters)

for val in per:
    print(val)
```

结果为。

```
('A', 'B', 'C')
('A', 'C', 'B')
('B', 'A', 'C')
('B', 'C', 'A')
('C', 'A', 'B')
('C', 'B', 'A')
```

4.6 杨辉三角隐藏的数字规律

杨辉三角中隐藏着大量有趣的数字规律。

帕斯卡矩阵

将杨辉三角数字左对齐，可以得到如下矩阵；这个矩阵常被称作**帕斯卡矩阵** (Pascal matrix)。

$$\begin{bmatrix} 1 & & & & & & \\ 1 & 1 & & & & & \\ 1 & 2 & 1 & & & & \\ 1 & 3 & 3 & 1 & & & \\ 1 & 4 & 6 & 4 & 1 & & \\ 1 & 5 & 10 & 10 & 5 & 1 & \\ 1 & 6 & 15 & 20 & 15 & 6 & 1 \end{bmatrix} \tag{26}$$

三角形数

上式矩阵的第一列均为 1，第二列为自然数，第三列为**三角形数** (triangular number)。如图 17 所示，如果一定数量圆形紧密排列，可以形成一个等边三角形，这个数量就叫做三角形数。

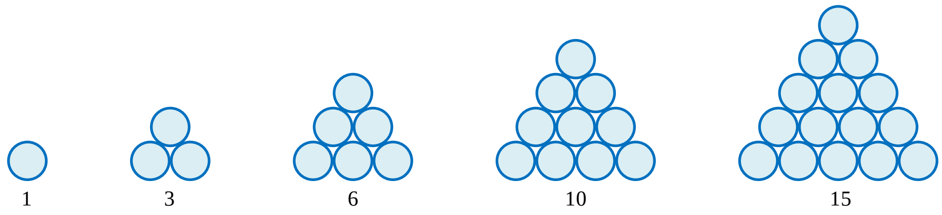


图 17. 三角形数

四面体数

(26) 第四列是叫做**四面体数** (tetrahedral number 或 triangular pyramidal number)；顾名思义，四面体数就是圆球紧密堆成四面体对应的数字。三角数从 1 累加便可以得到四面体数；也就是把图 17 看成是圆球，将它们一层层摞起来，便得到正四面体。

斐波那契数列

按照图 18 浅黄色线条方向，将杨辉三角每一排数字相加，可以得到如下数字序列。

$$1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55 \tag{27}$$

这便是**斐波那契数列** (Fibonacci sequence)。

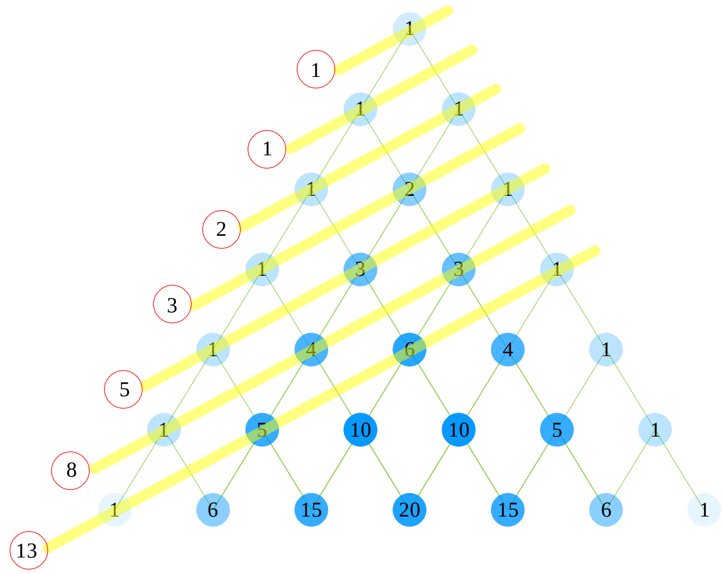


图 18. 杨辉三角和斐波那契数列关系

4.7 方程：求解鸡兔同笼问题

方程 (equation) 就是含有未知量的等式，比如 $x + 5 = 8$。使得等式成立的未知量的值叫做方程的**根** (root) 或**解** (solution)。

一元一次方程 (linear equation in one variable) 可以写成。

$$ax + b = c \tag{28}$$

其中，x 为未知变量，a、b、c 为实数，且 $a \neq 0$。

二元一次方程 (linear equation in two variables)，可以写成。

$$ax + by = c \tag{29}$$

其中，x 和 y 为未知变量，a、b、c 为实数，$a \neq 0$ 且 $b \neq 0$。

方程组 (system of equations) 是指两个或两个以上的方程，一般也会对应两个或两个以上未知量。

鸡兔同笼问题就可以写成二元一次方程组。

鸡兔同笼问题原文是"今有雉兔同笼，上有三十五头，下有九十四足，问雉兔各几何？"

用现代汉语来说就是：现在笼子里有鸡 (雉读作 zhì) 和兔子在一起。从上面数一共有三十五个头，从下面数一共有九十四只脚，问一共有多少只鸡、多少只兔子？

用 x 代表鸡，y 代表兔；有 35 个头对应如下方程式。

$$x + y = 35 \tag{30}$$

有 94 只脚对应如下方程式。

$$2x + 4y = 94 \tag{31}$$

联立两个等式得到方程组。

$$\begin{cases} x + y = 35 \\ 2x + 4y = 94 \end{cases} \tag{32}$$

很容易求得。

$$\begin{cases} x = 23 \\ y = 12 \end{cases} \tag{33}$$

也就是，笼子里有 23 只鸡，12 只兔。

本书会在在一次函数、线性代数这两个话题中继续有关鸡兔同笼故事。

一元二次方程 (quadratic equation in one variable) 可以写成。

$$ax^2 + bx + c = 0 \tag{34}$$

其中，a、b、c 都是实数，且 $a \neq 0$。

(34) 的求根公式可以写成。

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{35}$$

(34) **判别式** (discriminant) 是。

$$\Delta = b^2 - 4ac \tag{36}$$

$\Delta > 0$，一元二次方程有两个实数根；$\Delta = 0$，一元二次方程有两个相同实数根；$\Delta < 0$，一元二次方程有两个不同的复数根，不存在实数根。

`sympy.solvers.solve()` 函数可以用来求根。采用 `numpy.roots()` 也可以计算多项式方程的根。给定如下多项式等式。

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0 \tag{37}$$

单项式次数从高到低各项系数作为输入，用 `numpy.roots`($[a_0, a_1, \ldots, a_{n-1}, a_n]$) 函数来求根。

举个例子，如下三次多项式。

$$f(x) = -x^3 + 0 \cdot x^2 + x + 0 = 0 \tag{38}$$

以下代码可以求解三个根。

```python
# Bk_Ch4_09

# use sympy to solve
from sympy.solvers import solve
from sympy import Symbol
x = Symbol('x')
roots = solve(-x**3 + x, x)

# use numpy to solve
import numpy as np

coeff = [-1, 0, 1, 0]
roots_V2 = np.roots(coeff)
```

表 6 所示为方程相关的英文表达。

表 6. 方程相关英文表达

$x(y+1)=5$	*x* times the quantity of *y* plus one equals five.
$(x+a)(x+b)=0$	The quantity of *x* plus a times the quantity of *x* plus b equals zero.
$2x+y=5$	Two *x* plus *y* equals five.
$2x^2+3x+4=0$	Two *x* squared plus three *x* plus four equals zero.
$2x^3+3x^2+4=0$	Two *x* cubed plus three *x* squared plus four equals zero.
$(x+y)/2x=0$	The quantity of *x* plus *y* over two *x* equals zero.
$(x+y)^n=1$	The quantity *x* plus *y* to the nth power equals one.
$x^n+x^{n-1}=5$	*x* to the *n*, plus *x* to the *n* minus one equals five.

有时候，知识的传播好似随风潜入夜；更多时候，是水火不容的碰撞和残酷血腥的争夺。然而，这场抢占知识高地的竞争从未偃旗息鼓，大有愈演愈烈之势。

拿破仑曾感叹“数学的发展与国运息息相关”。在图 2 中西方代数复兴的时间轴上，一方面我们看到数学无国界，她在世界各地辗转腾挪、断续发展；另一方面，这个历史的脉络也让人们看到人才培养、聚集、转移，伴随着军事和政治影响力的转移，随之而来的是生产力和社会财富的转移。

阿基米德的血肉之躯不能挡住罗马士兵的刀刃；但是，他的精巧发明曾一度让强敌闻之色变。掌握利用知识，和汗牛充栋的藏书，是天壤之差、云泥之别。让数学思想之火熊熊燃烧的是一代代栋梁之才，和席卷八荒、保护矩火的强大力量；而两者互为给养、风雨同舟、荣辱与共。