第1讲 符号计算系统简介

1-1 走进 Mathematica

Mathematica是一款科学计算软件，一个集成化的数学软件系统

- 数值和符号计算引擎
- 图形系统
- 编程语言
- 文本编辑系统
- 与其他应用程序的高级连接

软件的应用

- 数学公式的推导工具

符号计算系统的数学对象几乎涉及所有的数学基础学科
初等数学，微积分，线性代数，微分方程，数值计算等

- 理论研究的实验工具

对于数学思想进行验证，节省科研人员大量繁重的计算工作；
通过对精确图形，实验结果的仔细观察，对理论知识的研究带来启发和灵感

- 数学教学的辅助工具

Mathematica软件可以绘制各种类型的图形，培养学生的几何直观
软件的界面可以提供包含互动功能的演示

1-2符号计算系统的组成

- 数值计算、符号计算、图形演示和程序设计语言

Axiom 免费开源软件，http : // axiom - developer.org
Maxima 免费开源软件，http : // maxima.sourceforge.net
Maple 商业软件，http : // www.maplesoft.com
Reduce 免费开源软件，http : // reduce - algebra.sourceforge.net

- Mathematica是世界上通用符号计算系统中最强大的系统
- 自从1988发布以来，它已经对如何在科技和其它领域运用计算机产生了深刻的影响
- 基于 30 多年来的持续开发，Mathematica 在所有技术计算领域表现卓著
- 涵盖所有技术计算领域的将近 5,000 个内置函数

1-3初识 Mathematica

- Mathematica 分为两部分：内核（kernel）和用户操作界面
- 用户可以创建并且编辑 "笔记本文档", 该笔记本文档可以包含程序代码和其它格式化的文本 （比如公式、图像、表格、声音等）
- 文档可以划分章节，也可以表示为幻灯片形式，便于进行演讲。

第1讲 绪论

1－2 常用操作

1. 运行命令 和cell（单元）的标记

计算(V) → 计算单元(E)　shift + Enter

```
111 * 111
```

```
Plot[Sin[x], {x, -3, 3}]  (* Sin → Cos *)
```

问题：shift + Enter 与 Enter的区别？

2. 插入，删除，收起，打开 cell

中间插入cell，　插入 → 相同样式的单元(Y)

3. 数学文字输入

（1）二维输入方式

常用方式：面板(P) → 其它

其它的子菜单包括：代数操作 ，基本排版 ， 数学输入

希腊字母等 ：　插入(I) → 特殊字符

数学格式：面板(P) → 数学助手 $\sqrt[3]{21}$ ， $\int_{\square}^{\square} \square \, d\square$ ， $\sum_{\square=\square}^{\square} \square$

上标和下标：插入(I) → 排版(T)

表格和矩阵：插入(I) → 表格／矩阵(M)

（2）一维输入方式

单元(Y) → 转换成 → TraditionalForm

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

4. 获取帮助

(1) 知道问题所属范围：帮助 (H) → Wolfram 参考资料 (D)

(2) 知道具体函数或指令：在帮助文本框中输入 Integrate

(3) 网站

* Wolfram 在线文档库 (http : // library.wolfram.com /)

收集了历年来 Wolfram 技术大会的讲稿，还有用户分享的课件、数据包、技术笔记等，

还有相关书籍噢！很多都是可以免费下载使用的！

* Wolfram 在线视频库 （ http : // www.wolfram.com / broadcast / ）

*《Wolfram 语言快速编程入门》

http : //
www.wolfram.com / language / fast - introduction - for - programmers / zh /

*《Mathematica 和 Wolfram 语言面向数学学习的快速入门指南》

http : //
www.wolfram.com / language / fast - introduction - for - math - students / zh /

5. 关于版本

向上兼容

6. 两个参考文件

关于教材.pdf 关于版本.pdf

```
11 111 * 11 111 (* 像用一个计算器    *)
```

- 怎样应用Mathematica软件
  - Mathematica的内置函数名多采用自然输入法，函数名就是英文单词，首字母要大写
  - 函数的变量或运算的内容用 [ ]
  - 列表，变量的范围用 { }
  - 运行函数用 Shift + Enter

因式分解

```
Factor[x^3 + 36 x^2 + 431 x + 1716]
```

解线性方程组

```
Solve[{3 x - 2 y == 5, x + y == 5}, {x, y}]
```

计算导数

```
D[x^2 Sin[x], x]
```

计算不定积分

```
Integrate[x^3 Sin[x], x]
```

计算矩阵特征值

```
A = {{1., 3, 5}, {3, 2, 7}, {1, 2, -1}};
Eigenvalues[A]
```

绘制函数图像

```
Plot[Sin[x], {x, -3.5, 3.5}]
```

```
Manipulate[ Plot[ a Sin[b x + c], {x, -6, 6}, PlotRange → {-5, 5}],
 {a, 1, 5, 0.2}, {b, 1, 5, 0.2}, {c, 0, 2 Pi, Pi / 100}]
```

绘制单叶双曲面

$$\frac{x^2}{4}+\frac{y^2}{3}-\frac{z^2}{2}=1$$

```
ContourPlot3D[x^2 / 4 + y^2 / 3 - z^2 / 2 == 1, {x, -4, 4}, {y, -3, 3},
 {z, -2, 2}, ColorFunction → Function[{x, y, z}, Hue[z / 4]]]
```

I-4课程介绍

- 一方面按照数学的进程，从初等数学到高等数学，介绍如何调用Mathematica的函数做初等数学、微积分、线性代数和微分方程中的计算题，验证数学公式的推导，演示函数图形
- 另一方面按照计算机语言的结构，从简单的命令行输入到构建复杂的程序包，学习过程编程和函数编程
- 希望大家通过这门课程的学习能够□□□会用□□□和□□□用好□□□高等数学，给以后的专业课学习及科学研究带来帮助。
- 1994年中国科学技术大学是国内最早为本科生开设符号计算语言mathematica课程的几个高校之一，现作为数学学院和全校本科生公共选修课，已有近30届的学生选修了本课程。部分学生已将

Mathematica作为工具用在高等数学、数学实验和专业课程学习、数学模型竞赛、大学生研究计划和毕业论文中。
2009年□□□符号计算系统软件□□□评为安徽省精品课程。
2010年□□□符号计算研究和实践□□□获安徽省教学成果一等奖。

- 主要参考教材：张韵华，王新茂，Mathematica 7 实用教程，中国科学技术大学出版社

第2讲 Mathematica 的基本量

2－1 数的表示及其函数

1．简单数值类型

Mathematica中的简单数值类型有整数、分数（有理数）、实数和复数 四种。

整数： Integer， 没有误差，任意长度的准确数.

```
{2^5, 2^10}
```

```
2^2016
```

```
IntegerDigits[%]    (* IntegerDigits[2^2016] *)
```

```
Length[%]
```

例：计算 48、105、120 的最大公约数和最小公倍数.

```
{GCD[48, 105, 120], LCM[48, 105, 120]}
```

例：判断11117和13117是否为素数.

```
{PrimeQ[11117], PrimeQ[13117]}
```

例：第1个和第101个素数是多少？

```
{Prime[1], Prime[101]}
```

有理数：Rational，既约分数

```
12345678987654321/234321
```

实数：Real，有限精度的浮点数。实数的输出也既可以是小数形式，
又可以是指数形式。

```
a=-.618
```

```
b=98765432123456789.98765432123456789
```

```
{IntegerPart[1.5], FractionalPart[1.5]}
```

复数：Complex，x + y I，虚数单位 I，实部x和虚部y都可以
是整数、有理数或实数.

```
z = (2 + 5 I)^2
```

例：计算复数的幅角和模。

```
{Arg[z], Abs[z]}
```

2．数学常数

Degree	角度，45 Degree 表示 45^0	E	自然对数的底数，e
GoldenRatio	黄金分割数 1.618	I	虚数单位 $\sqrt{-1}$
Infinity	无穷大 ∞	Pi	圆周率 π

3. 数的转换

. 转为整数

```
{Round[1.5],Floor[1.5], Ceiling[1.5] }      (*向下取整，向上取整 *)
```

```
{Round[-1.5],Floor[-1.5], Ceiling[-1.5] }
```

. 转为实数

```
{N[Pi],N[Pi,30]}
```

. 实数转为分数

```
{Rationalize[3.14], Rationalize[3.1415926]}
```

```
Rationalize[3.1415926, 0.000001]
```

4. 常用初等函数 -- 面对所有数值类型

`Abs[x]`	实数的绝对值或复数的模
`Re[z]、Im[z]、Arg[z]、Conjugate[z]`	复数的实部、虚部、幅角、共轭
`Power[x, y]、Sqrt[x]`	幂函数、平方根
`Exp[x]、Log[x]、Log[b, x]`	指数函数、自然对数函数、对数函数
`Max[x1, x2, ...]、Min[x1, x2, ...]`	最大值、最小值
`Sign[x]`	符号函数
`Sin[x]、Cos[x]、Tan[x]、Csc[x]、Sec[x]、Cot[x]`	三角函数
`ArcSin[x]、ArcCos[x]、ArcTan[x]、ArcCsc[x]、 ArcSec[x]、ArcCot[x]`	反三角函数
`Sinh[x]、Cosh[x]、Tanh[x]、Csch[x]、Sech[x]、Coth[x]`	双曲函数
`ArcSinh[x]、ArcCosh[x]、ArcTanh[x]、ArcCsch[x]、 ArcSech[x]、ArcCoth[x]`	反双曲函数
`Binomial[m, n]、Multinomial[n1, n2, ...]`	二项式组合系数 $\binom{n}{m}$
`Factorial[n]、Factorial2[n]`	阶乘!、双阶乘!!
`FactorInteger[n]`	整数分解
`GCD[n1, n2, ...]、LCM[n1, n2, ...]`	最大公约数、最小公倍数
`Mod[m, n]、Mod[m, n, d]`	余数
`Prime[n]、PrimeQ[n]、PrimePi[n]`	素数生成、素数检验、素数计数

例题：

```
{Sqrt[4. + 9 I], Cos[2.1 + 3 I]}
```

5. 字符串

用双引号 " " 括起的字符，字符串中可以包含任意编码的字符，如希腊字母、中文字符等还可以包含一些特殊字符，如换行符 "\n "、制表符 "\t".

```
{ToLowerCase["ABCD"], ToUpperCase["abcd"]}
```

```
StringReverse["ABCD"]
```

```
a = "中国科学技术大学现代教育技术中心";
Print["制作单位:", a]
```

有关字符串的编辑，对字符串插入、删除、查找、替换等，在此不再展开内容。
有兴趣的读者请参看
中国科学技术大学出版社， **Mathematica 7** 实用教程（第二版）， **24** 页 **- 29** 页。

本课内容

1. 简单数值类型： 整数、有理数、实数和复数

2. 数学常数

3. 数的转换

4. 常用初等函数

5. 字符串

第2讲 Mathematica的基本量

2－2 列表生成

列表表示对象：数组（向量）、矩阵、集合、数据库中的记录、
数据结构中的树和图等。

列表形式：用花括号围起来的有限个元素，元素之间用逗号分割。
一个列表可以包含任意多个元素，列表中的元素可以是不同类型的
任何Mathematica对象。

如果一个列表的某个元素是列表，我们称之为嵌套列表（例In[2]）。

本讲侧重一维数组，在第5讲线性代数中侧重二维数组。

1．枚举元素

方式简单明了，列表中元素较少时使用.

```
a = {学号, 姓名, 身高, 体重}; (*类似于数据库中的记录*)
b = {1, 2, 3}; d = {2, 3, 4}; A = {b, d}
```

2．Range

Range[n]	生成列表{1,2,...,,n}, m为1可省略
Range[m,n]	生成{m,m+1,...,n}
Range[m,n,d]	以m为首项, 以d为增量，{m,m+d,..., }

例题：

```
Range[5]
s^Range[5]
Range[12.5,2,-2.5]
Clear[a,b];Range[a,b,(b-a)/4]
Simplify[%]
Range[{3, 4}]    (* Range[3,4]?*)
Range[Range[5]] (* Range[{1,2,3,4,5}]  *)
```

3. Table

```
Table[expr, {k, m, n, d}]    循环变量 k，首项 m，增量为 d，终止值是 n .
Table[expr, {k, m, n}]       增量为1
Table[expr, {k, n}]          首项和增量都为1
Table[expr, n]
```

例题：

```
Table[i^2, {i, 3}]
X = Table[x[k], {k, 3}]  (*注意x的大写和小写*)
A = Table[Sin[i], {i, 3}]
a = {1, 2, 3}; A = Sin[a]  (*初等函数自动作用到列表的每一个元素*)
a^3
B = Table[10 i + j, {i, 3}, {j, 4}]
MatrixForm[B]
```

例：列表的元素是函数图像，Table中的函数名为循环变量，取值枚举列表。

```
Table[Plot[f[x], {x, -Pi, Pi}], {f, {Sin, Cos, Tan, Cot}}]
```

4. 生成随机列表

RandomInteger 和 RandomReal 分别生成随机整数和随机实数.

例题：

```
RandomReal[]          (* 0~1之间的随机实数 *)
RandomInteger[10]     (* 0~10之间的随机整数 *)
RandomReal[{1, 2}, 3]     (* 生成3个1~2之间的随机实数 *)
RandomInteger[1, {2, 3}]    (* 生成2行3列的随机0-1矩阵 *)
```

5* 递归表 RecurrenceTable

```
RecurrenceTable[{f[n]==f[n-1]+f[n-2],f[1]==1,f[2]==1},f,
  {n,8}]
Table[Fibonacci[n], {n, 8}]
```

6. 列表元素表示

```
Clear[A, a, B, b]; A = Table[a[k], {k, 3}]
{a[1] = 111, A[[2]] = 222, A[[-1]] = 333}; A
B = Table[b[i, j], {i, 3}, {j, 3}]
{B[[1]], B[[1, 1]], b[1, 1]}
{B[[2, 1]], b[2, 2], b[2, 3]} = {21, 22, 23}; B[[2]]
B
```

TableForm[B]

列表生成及其分量表示

枚举元素

Range

Table

RandomInteger 或 **RandomReal**

RecurrenceTable

第2讲 Mathematica的基本量

2－3 列表相关函数

1．列表元素编辑

例： Drop 去掉倒数3个元素

```
t = {a, b, c, d, e, f}; Drop[t, -3]
```

例： Delete 删除倒数第3个元素

```
s = {2, 3, 4, 5, 6}; Delete[s, -3]
```

```
{t, s}
```

```
t = Drop[t, -3]; s = Delete[s, -3]
```

```
Delete[s, {{-1}, {-2}, {-3}}]
```

```
B = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
```

例： 在 B[[3]] 之前插入

```
Insert[B, x, 3]
```

例： 在B[[3]] 之后插入

```
Insert[B, x, -1]
```

```
u = v = {1, 2, 3}; Append[u, x]
```

```
AppendTo[v, x]; {u, v}
```

2．列表元素排序

例：请观察排序和排序选项.

```
Sort[{9, 3, 1, 2, 4, 6, 3, 4, 7}]
```

```
Reverse[%]
```

```
Sort[{9, 3, 1, 2, 4, 6, 3, 4, 7}, Greater]
```

```
Sort[{9, 3, 1, 2, 4, 6, 3, 4, 7}, #1 > #2 &]
```

```
Sort[{a + b, a + c, a + b + c, a - b}]
```

3．列表元素求和

Apply 常常出现在列表运算中，Apply 的用法简单灵活，应用范围广.

Apply[f, expr]	函数或算子f作用于表达式expr
Apply[Plus, list]	把list 中的所有元素加在一起
Apply[Times, list]	把list 中的所有元素乘在一起

例：将表中所有的元素相加

```
b = {5, 8, 5, 7, 2, 6}; Apply[Plus, b]
```

例：将表中所有的元素相乘

```
Apply[Times, b]
```

```
Apply[Sin, b]
```

```
Map[Sin, b]
```

4. 列表合并与拆分

`Flatten[a]`	把嵌套列表压平为1维列表
`Partition[a, n]`	把列表拆分成若干长度为 n 的子列表

例：将表a 的元素分别拆分成每行2个和3个元素的矩阵.

```
a = {{1, 2, 3, 4}, {4, 5, 6, 7}}; b = Flatten[a]
```

```
{Partition[b, 2], Partition[b, 3]}
```

5. 列表的集合运算

`Union[a1, a2, ...]`	多个集合的和（并）集，删除重复元素并排序
`Intersection[a1, a2, ...]`	多个集合的交集，删除重复元素并排序
`Complement[a, b1, b2, ...]`	删除a中的b1, b2, ... 元素和重复元素并排序

```
a = {2, 3, 5, 1, 5, 1, 0, 2, 4}; b = {3, 1, 0, 2, 0, 3, 0, 3, 2};
c = {2, 3, 4, 3, 2, 1, 2, 4, 1};
```

例：计算集合的a, b, c 的和集，交集，a 的补集。

```
Union[a, b, c]
```

```
Intersection[a, b, c]
```

```
Complement[a, b, c]
```

列表运算

- 列表元素插入和删除
- 列表元素排序
- 列表元素求和 **Apply**
- 列表合并与拆分
- 列表的集合运算

第2讲 Mathematica的基本量

2－4 表达式

几乎所有的Mathematica对象都可以被认为是表达式。

数字、字符串、符号（数学常数、变量名或函数名）是最基本的

表达式单元。多个表达式通过函数或运算符连接成为一个复合表达式。

Mathematica的运算过程就是表达式求值的过程。

1．变量和变量替换

变量名

在Mathematica中，变量名通常以英文字母开头，后跟字母或数字，变量名的字符长度不限。

希腊字母和中文字符也可以用在变量名中。

Mathematica区分英文字母的大小写，因此A与a表示两个不同的变量。

建议变量名以小写字母开头。如果用大写字母表示变量，请避免

使用 C、D、E、I 等系统已用的字符.

例：变量赋值。

```
a = 22; b = 77; {a, b} = {b, a}
{a, b} = {22, 77}; c = 1122; {a, b, c} = {b, c, a}
```

变量替换

表达式 /. 规则 或 ReplaceAll[表达式, 规则]

其中替换规则是一个或一组形如lhs → rhs的表达式（详情请看第7讲）.

例：计算三角形的面积有以下Heron公式。

$$\left(s = \frac{1}{2}(a+b+c);\ A = \sqrt{s(s-a)(s-b)(s-c)}\right)$$

```
Clear[a, b, c]; s = (a + b + c) / 2; A = Sqrt[s (s - a) (s - b) (s - c)];
A /. {a → 1, b → 1, c → 1}      (*把a,b,c都换成1*)
A /. {a → 3, b → 4, c → 5}
```

```
2 + 3 x + x^2 /. {2 → 3, 3 → 4}
```

```
ReplaceAll[2 + 3 x + x^2, {2 → 3, 3 → 4}]
```

2. 算术表达式

一个算术表达式通常是由数或变量经算术运算符或函数连接而成，其中变量的类型可以是数值、符号、列表等，数可以是系统内嵌函数或者是用户自定义函数.

运算优先级	运算符	说 明
1	[ ]、{ }、( )	函数、列表、分隔符
2	!、!!	阶乘、双阶乘
3	++、--	变量自加1、自减1
4	+=、 -= 、 *= 、 /=	运算后赋值给左边变量
5	^	方幂
6	.	矩阵乘积或向量内积
7	*、/	乘、除
8	+、-	加、减

在不引起误解的情况下，算术表达式中的乘号可以忽略不写。

例如：2 a、2 a、2 * a 的意义是相同的，

(a - b) (c + d) 与 (a - b) * (c + d) 的意义也是相同的，

但是 a2 与 a 2 的意义却是不同的。

算术运算的优先级遵从数学习惯，同级运算符按照从左到右的顺序，赋值则按照从右到左的顺序。

3. 逻辑表达式

逻辑表达式通常是由逻辑运算符或关系运算符连接而成。

逻辑表达式的值：True、False 、无法确定。

例：

```
Clear[a, b]; a < b
```

```
Infinity == 2 Infinity
```

```
x^2 + 2 x * y + y^2 == (x + y) ^2
```

```
Simplify[x^2 + 2 x * y + y^2 == (x + y) ^2]
```

关系运算符

< 、 ≤ 、 == 、 ≥ 、 > 、 ≠

逻辑运算符

And 、 &&	逻辑与
Or 、 ||	逻辑或
Not 、 !	逻辑非
Equivalent	逻辑等价

例：写出与下列数学条件等价的Mathematica逻辑表达式。

$m \in (s,t)$, $x \notin (-10,10)$

第2讲 Mathematica的基本量

2－1 数的表示及其函数

2－2 列表生成

2－3 列表相关函数

2－4 表达式

第 3 讲 初等函数运算

3 - 1 多项式运算

3.1.1多项式的基本运算

多项式的基本代数运算有加法、减法、乘法、除法、模运算

```
t1 = x^2 - 2 x - 3; t2 = x - 3;
```

```
t1 + t2
```

```
t1 - t2
```

```
t1 * t2
```

```
t1 / t2
```

给定两个单变量多项式a(x) 和b(x)，存在唯一的多项式q(x) 和r(x) 使得a(x)=b(x)q(x)+r(x)

- PolynomialQuotient　两个多项式相除的商式
- PolynomialRemainder　两个多项式相除的余式
- PolynomialQuotientRemainder
  同时给出两个多项式相除的商式和余式

```
PolynomialQuotient[x^3 + y^3, x - y, x]
```

```
PolynomialRemainder[x^3 + y^3, x - y, x]
```

```
PolynomialQuotientRemainder[x^3 + y^3, x - y, x]
```

3.1.2多项式元素提取

- PolynomialQ[expr,x] 或PolynomialQ[expr,{x,y,...}]
  检验expr是否关于变元x,y,...的多项式

```
PolynomialQ[x^2 - y^2, {x, y}]
```

```
PolynomialQ[x^2 + x + 2 / y, {x, y}]
```

```
PolynomialQ[x^2 + x + 2 / y, x]
```

```
poly1 = (x - 1) ^ (15); poly2 = x^4 + 5 x^3 y^2 - 3 x y + 6 y^4;
```

- Variables[poly] 多项式poly的变元列表

```
Variables[poly2]
```

- Coefficient[poly,x,n]　多项式poly中x^n项的系数，n缺省值为1
- CoefficientList[poly,x] 或CoefficientList[poly,{x,y,...}]
  多项式poly关于变元x,y,...的系数列表

poly1=(x-1)^(15)

poly2=x^4+5 x^3 y^2-3 x y+6 y^4;

```
Coefficient[poly1, x, 5]
```

```
CoefficientList[poly2, x]
```

例题：在展开$(x+y+z)^6$后 ，$x^2\,yz^3$项的系数是？

```
Coefficient[(x + y + z)^6, x^2 * y * z^3]
```

3.1.3多项式展开与合并

- Expand[expr]
  展开表达式expr中的乘积和正整数方幂,
  按照幂次由低至高的顺序，将表达式展开成为单项之和
- ExpandAll[expr] 展开表达式expr中的乘积和整数方幂
- PowerExpand[expr,x] 或PowerExpand[expr,{x,y,...}]
  展开表达式expr中与变元x或{x,y,...}有关的乘积的方幂,例如对数，
  开方表达式中的嵌套幂次.
- ExpandNumerator[expr] 展开表达式expr中的分式的分子
- ExpandDenominator[expr] 展开表达式expr中的分式的分母

例题：展开多项式$(x+2y+1)^2$

```
Expand[(x + 2 y + 1)^2]
```

例题：比较 Expand[ ], ExpandAll[ ], PowerExpand[ ]的区别.

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

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

```
ExpandAll[Sin[(x + y)^-2]]
```

```
PowerExpand[Log[(x * y)^z]]
```

例题：分别展开表达式$t=\frac{(1-x)}{(x+1)^2}+\frac{(x+1)^2}{x(1-x)}$中的分子和分母的多项式.

```
t = (1 + x)^2 / (x (1 - x)) + (1 - x) / (1 + x)^2;
```

```
ExpandDenominator[t]
```

$$\frac{(1+x)^2}{x-x^2}+\frac{1-x}{1+2x+x^2}$$

```
ExpandNumerator[t]
```

$$\frac{1-x}{(1+x)^2}+\frac{1+2x+x^2}{(1-x)\,x}$$

- Collect[expr,x] 或Collect[expr,{x,y,...}]
  合并表达式expr中与变元x或{x,y,...}有关的同类项

- Apart[expr,x]
  把表达式expr写成部分分式之和的形式
- ApartSquareFree[expr]
  把表达式expr写成部分分式之和的形式，
  其中分母是无重根多项式的方幂的形式
- Cancel[expr]
  约分表达式expr中的分式
- Together[expr]
  通分表达式expr中的分式

例题：展开$(x+a+2)^4$, 按x的幂次合并同类项.

```
Collect[(x + a + 2)^4, x]
```

例题：将有理式$t=\frac{x^2}{1-2x^2+x^4}$展开成部分分式的和.

```
t = x^2 / (1 - 2 x^2 + x^4);
```

```
Apart[t]
```

```
ApartSquareFree[t]
```

例题：求$\frac{2x+1}{5x-7}$, $\frac{x-2}{3x+2}$, $\frac{x^2}{x^2+3}$的和，分子分母都是展开形式.

```
Together[(2 x + 1)/(5 x - 7) + (x - 2)/(3 x + 2) + x^2/(x^2 + 3)]
```

例题：约分$\frac{x^2-y^2}{x^3-y^3}$

```
Cancel[(x^2 - y^2) / (x^3 - y^3)]
```

3.1.4多项式因式分解

- Factor[poly] 在整数环上分解多项式
- FactorSquareFree[poly] 提出多重因式
- FactorTerms[poly,x] 或FactorTerms[poly,{x,y,...}]
  分解poly为常数与本原多项式乘积的形式，x或{x,y,...}缺省为所有变元
- FactorList[poly]
  poly的不可约因子及其方幂列表
- FactorSquareFreeList[poly]
  poly的无重根因子及其方幂列表

例题：对多项式$16-32x-8x^2+28x^3-3x^4+9x^5$作因式分解

```
poly = 16 - 32 x - 8 x^2 + 28 x^3 - 3 x^4 + 9 x^5;

Factor[poly]

FactorSquareFree[poly]

Factor[poly, Extension → Sqrt[-1]](* 在复数域分解多项式*)

FactorList[poly]

FactorSquareFreeList[poly]
```

3.1.5多项式组公因式与公倍式

- PolynomialGCD[p1,p2,...]
  多项式组{p1,p2,...}的最大公因式
- PolynomialLCM[p1,p2,...]
  多项式组{p1,p2,...}的最小公倍式
- PolynomialExtendedGCD[f,g,x]
  一元多项式f(x)和g(x)的扩展最大公因式
  - 语句PolynomialExtendedGCD[f,g,x]返回一个形如{h,{a,b}}的列表，
    其中a和b都是关于变元x的有理式，h是f和g的最大公因式，并且满足h=af+bg

```
PolynomialLCM[x^2 + x^2 y - y - 1, x^2 y + x^2 z - y - z](*最小公倍式*)

PolynomialGCD[x^2 + x^2 y - y - 1, x^2 y + x^2 z - y - z](*最大公因式*)

PolynomialExtendedGCD[x^2 - 2 x * y + y^2 + x - y, x^2 - y^2 - x + y, x]
```

第3讲 初等函数运算

3 - 2 三角函数运算

- 3-1节中介绍的几个函数例如：Expand、Cancle、Together 等也适用于三角函数运算，但运算过程中并没有用三角函数公式进行简化， 函数中设置Trig->True 选项，使之可以将三角函数恒等式与这些运算结合起来.

- 例如

```
Cancel[Sin[x] / (1 - Cos[x]^2)]
```

```
Cancel[Sin[x] / (1 - Cos[x]^2), Trig → True]
```

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

```
Together[Sin[x]^2 / (1 - Cos[x]^2) + Cos[x]^2 / (1 - Sin[x]^2), Trig → True]
```

- Trig->True 同样适用于双曲函数.

```
Expand[(Cosh[x]^2 + Sinh[x]^2) (Cosh[x]^2 - Sinh[x]^2)]
```

```
Expand[(Cosh[x]^2 + Sinh[x]^2) (Cosh[x]^2 - Sinh[x]^2), Trig → True]
```

3-2-1适用于三角函数的特殊函数

- TrigExpand[expr]三角函数和差化积
- TrigFactor[expr]三角函数因式分解
- TrigFactorList[expr]三角函数因子列表
- TrigReduce[expr]三角函数积化和差
- TrigToExp[expr]化三角函数为指数函数
- ExpToTrig[expr]化指数函数为三角函数

```
Expand[Sin[x] + Sin[2 x] + Sin[3 x], Trig → True]
```

```
TrigExpand[Sin[x]+Sin[2x]+Sin[3x]]
```

```
TrigFactor[Sin[x]+Sin[2x]+Sin[3x]]
```

```
TrigFactorList[Sin[x]+Sin[2x]+Sin[3x]]
```

```
TrigReduce[Sin[x]Sin[2x]Sin[3x]]
```

```
TrigToExp[Sin[x]+Sin[2x]+Sin[3x]]
```

```
ExpToTrig[(-1)^x]
```

例题：求和并化简 $\frac{\text{Cos}[x]}{1+\text{Sin}[x]}+\text{Tan}[x]$

```
Together[Cos[x]/(1 + Sin[x]) + Tan[x], Trig → True]
```

```
TrigReduce[%]
```

例题：构造 Sin[nx], Cos[nx]的n倍角公式表, n=2,3,4

```
A = Table[{n, TrigExpand[Sin[n x]], TrigExpand[Cos[n x]]}, {n, 2, 4}]
```

```
TableForm[A, TableHeadings → {None, {"n", "sin nx", "cos nx"}}]
```

3-2-2表达式化简

- Simplify[expr] 化简表达式expr
- Simplify[expr,assum] 依据假设assum化简表达式expr
- FullSimplify[expr] 深入化简表达式expr
- FullSimplify[expr,assum] 依据假设assum深入化简表达式expr
- Assuming[assum,expr] 依据假设assum执行表达式expr

```
Simplify[x^2 + 3 (x + 2 / 3)]
```

```
FullSimplify[x^2 + 3 (x + 2 / 3)]
```

```
Simplify[(x + y) / 2 > Sqrt[x * y], x > y > 0]
```

第3讲 初等函数的运算

3-3 解方程运算

3-3-1 方程的表示

- 方程通常表示为: "lhs==rhs"------逻辑表达式.
- 方程组通常表示为 {eqn1,eqn2,...} 或eqn1 && eqn2 && .
- 将方程中的等号"=="换成不等号"<", ">","<=", ">=", "!="就得到不等式.
- 方程和不等式也可以被展开、合并、或化简

```
eqn = (1 - x) (1 + y) == (1 + x) (1 - y); Expand[eqn]
```

```
Collect[eqn, x]
```

- Eliminate[eqns,vars] 将方程组eqns中的部分变元消去

```
p1 = x^2 + y^2 + z^2 - (a^2 + b^2 + c^2); p2 = x + y + z - (a + b + c);
p3 = a^2 + b^2 + c^2 - 2 a + 2 b + 1; p4 = a + b - 2 c;
Eliminate[{p1, p2, p3, p4} == {0, 0, 0, 0}, {a, b, c}]
```

3-3-2 一般方程的求解

- Solve[eqns,vars] 求方程（组）eqns的所有准确解vars
  方程中只有一个未知量时，vars可不写。

```
Solve[a * x^2 + b * x + c == 0, x]
```

例题：求解方程组$\begin{cases} 2x+3y=7 \\ 3x+4y=10 \end{cases}$

```
Solve[2 x + 3 y == 7 && 3 x + 4 y == 10, {x, y}]
```

```
Solve[{2 x + 3 y == 7, 3 x + 4 y == 10}, {x, y}]
```

- 说明：1.如果要利用方程的解代入表达式求值，可直接用取代运算符/.
  2.方程的解以列表形式给出，可以用Part提取列表元素.

例题： 求解方程组$\begin{cases} x^2+y=4 \\ 2x-y=1 \end{cases}$，并计算表达式$\sqrt{x^2+y^2}$在这些解上的值.

```
solution = Solve[{x^2 + y == 4, 2 x - y == 1}, {x, y}]
```

```
√(x^2 + y^2) /. solution
```

```
Simplify[%]
```

例题：计算方程$-735+1337x-778x^2+198x^3-23x^4+x^5=0$所有根的平方和.

```
solution = Solve[-735 + 1337 x - 778 x^2 + 198 x^3 - 23 x^4 + x^5 == 0, x]
```

```
{{x → 1}, {x → 3}, {x → 5}, {x → 7}, {x → 7}}
```

```
solution[[1]]
```

```
solution[[1, 1]]
```

```
solution[[1, 1, 2]]
```

```
Sum[solution[[k, 1, 2]]^2, {k, 1, 5}]
```

- NSolve[eqns, vars] 求方程eqns的所有数值解vars
- NSolve[eqns, vars,n]
  求方程eqns的所有数值解vars, 达到n位精度

```
NSolve[x^5 == x + 1, x]
```

```
NSolve[x^5 == x + 1, x, 20]
```

- Roots[eqn, var]
  求一元多项式方程eqn的所有准确解var
- NRoots[eqn, var]
  求一元多项式方程eqn的所有数值解var
- Root[f,k]
  求一元多项式f的第k个根

```
Roots[x^5 == x + 1.0, x]
```

```
NRoots[x^5 == x + 1.0, x, 20]
```

- FindRoot[f,{x,a}] 以a为初值，求函数f(x)的一个根x
- FindRoot[eqns,{x,a}]以a为初值，求方程eqns的一个解x

例题：求解方程 $\sin x = x^2 - 1$

```
Plot[{Sin[x], x^2 - 1}, {x, -Pi, Pi}]
```

```
FindRoot[Sin[x] == x^2 - 1, {x, -1}]
```

例题：求方程x*sin (x) = 1 在区间[-10，10] 上的解。

```
Plot[x * Sin[x] - 1, {x, -10, 10}]
```

```
a = {-9, -6, -3, -1, 1, 3, 6, 9}
```

```
Table[FindRoot[x * Sin[x] == 1, {x, a[[i]]}], {i, 8}]
```

例题： 求方程组 $\begin{cases} e^x + \ln y = 2 \\ \sin x + \sin y = 1 \end{cases}$ 的近似解.

```
ContourPlot[{Exp[x] + Log[y] == 2, Sin[x] + Sin[y] == 1}, {x, 0, 4}, {y, 0, 4}]
```

```
sol1 = FindRoot[{Exp[x] + Log[y] == 2, Sin[x] + Sin[y] == 1}, {x, 0.4}, {y, 2}]
```

```
sol2 = FindRoot[{Exp[x] + Log[y] == 2, Sin[x] + Sin[y] == 1}, {x, 1.6}, {y, 0.2}]
```

- Reduce[expr, vars, dom]
  化简方程或不等式expr并求所有解vars

- FindInstance[expr, vars, dom, n]
  求方程或不等式expr的n个特解vars

```
Reduce[a * x^2 + b * x + c == 0, x]
```

```
Reduce[3 x - 2 y > a && x + y < b, {x, y}]
```

```
FindInstance[x^2 + y^2 == z^2 && 0 < x < y < z < 20, {x, y, z}, Integers, 3]
```

3-3-3 递归方程的求解

- RecurrenceTable[eqns, expr, nspec]
  由递归关系eqns求表达式expr生成的数列
- RSolve[eqns, a[n], n]
  由递归关系eqns求数列an的通项公式

```
RecurrenceTable[{a[n + 2] == a[n + 1] + a[n], a[1] == a[2] == 1}, a[n], {n, 10, 20}]
```

```
RSolve[b[1 + n] == 1/3 (x * b[n] + y), b[n], n]
```

第 4 讲 微积分

4 - 1 求极限

计算函数极限的形式：Limit [expr, x->x0]

例1 ： 求下列极限

(1) $\lim\limits_{x\to 0}\dfrac{\sin(a\,x)}{x}$　　(2) $\lim\limits_{x\to 1}\left(\dfrac{m}{1-x^m}-\dfrac{n}{1-x^n}\right)$

```
Limit[Sin[a x] / x, x -> 0]
```

```
Limit[m / (1 - x^m) - n / (1 - x^n),
 x → 1]
```

不是所有的函数都有确定的极限

例如 $\lim\limits_{x\to 0}\sin\left(\dfrac{1}{x}\right)$，极限不存在，在0附近，函数在[-1, 1] 之间波动，

Limit运算的结果是一个区间。

```
Limit[Sin[1/x], x → 0]
```

```
(1 + %) ^2 * 5
```

数列的极限，可以用同样的形式进行计算。

例2：计算数列的极限

(1) $\lim\limits_{n\to\infty}\left(\sqrt{n+\sqrt{n}}-\sqrt{n}\right)$

(2) $\lim\limits_{n\to\infty}\dfrac{n!}{n^n}$

```
Limit[Sqrt[n + Sqrt[n]]
  - Sqrt[n], n -> Infinity]
```

```
Limit[n! / n^n, n -> Infinity]
```

- 递归定义的数列的极限

例3： 设 $x_1=\sqrt{2}$ ，$x_n=\sqrt{(2+x_{n-1})}$，求 $\lim\limits_{n\to\infty}x_n$

```
f[1] = N[Sqrt[2], 10];
f[n_] := N[Sqrt[2 + f[n - 1]], 10];
f[10]
```

```
1.999997647
```

```
xn = Table[f[n], {n, 1, 20}]
ListPlot[xn, PlotStyle → {Red, Thick}, Joined -> True]
```

计算函数极限的一般形式是

- Limit[expr, x -> x0]　　计算 x → x0时函数expr 的极限
- Limit[expr, x -> x0, Direction -> 1]　　计算x → x0时函数expr的左极限
- Limit[expr, x -> x0, Direction -> -1]　　计算x → x0时函数expr的右极限

例4：计算下列极限：

$$(1)\ \lim_{x\to 0^+}\frac{\log(x)}{x}\qquad (2)\ \lim_{x\to\infty}\frac{\Gamma\left(x+\frac{1}{2}\right)}{\sqrt{x}\,\Gamma(x)}$$

```
Limit[Log[x] / x, x -> 0, Direction -> -1]
```

```
Limit[Gamma[x + 1 / 2] / (Sqrt[x] * Gamma[x]),
  x -> Infinity]
```

- 从 Mathematica 语言的语法上说，
  Limit 函数自己不带对多个变量取极限的功能即不能计算重极限，可以计算累次极限。

例5： 计算 $\lim_{y\to\infty}\lim_{x\to\infty}\left(\frac{x\,y}{x^2+y^2}\right)^{x^2}$

```
Limit[Limit[((x * y) / (x^2 + y^2)) ^ (x^2),
    x -> Infinity], y -> Infinity]
```

```
Limit[Limit[((x * y) / (x^2 + y^2)) ^ (x^2),
    y -> Infinity], x -> Infinity]
```

- 还可以计算自变量沿某一固定路径趋向于固定点时，表达式的极限

```
Limit[(x^2 + y^2) / (1 - (1 + x^2 + y^2) ^ (1 / 2)) /. {y → a t, x → t}, t → 0]
```

例6：画出函数 $f(x)=\frac{(x-5)^2}{3\,(x+1)}$ 的斜渐近线。

```
(* 函数f(x)的渐近线是指：当x->∞时，f(x)无限接近某一
   直线y=x+b即lim_{x→∞}(f(x)-(a x+b))=0，需要用极限
   的思想确定参数a和b的值。*)
```

```
f[x_] := (x - 5) ^ 2 / (3 * (x + 1));
a = Limit[f[x] / x, x -> Infinity]
```

```
b = Limit[f[x] - a x, x -> Infinity]
```

```
Plot[{f[x], a x + b}, {x, 0, 30}]
```

第4讲 微积分

4 - 2 微商和微分

4-2-1 微商（导数）

- 计算导数的命令是D[f, x] 和D[f, {x, n}]
  分别表示 f'(x) 和$f^{(n)}(x)$。
- 计算导数和偏导数是同一命令。
  如果f是一元函数，*D*[*f*, *x*] 表示 $\frac{df}{dx}$
  如果f是多元函数，*D*[*f*, *x*] 表示 $\frac{\partial f}{\partial x}$
- 微商函数的常用形式如下
  - D[f, x]　　偏导数 $\frac{\partial f}{\partial x}$
  - D[f, x1, x2, ..., xn]　　高阶偏导数 $\frac{\partial^n f}{\partial x1\,\partial x2\,\ldots\,\partial xn}$
  - D[f, {x, n}]　　n阶导数 $\frac{\partial^n f}{\partial x^n}$
  - D[f, x, NonConstants → {y1, y2, ..., ym}]
    复合函数的偏导数 $\frac{\partial f}{\partial x}$，y1, ..., ym是x的函数
  - D[f, {{x, y, z, ...}}]　　偏导向量 $\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}, \ldots\right)$

```
D[x^n, x]

D[x^n, {x, 2}]

D[z (x^2 + y^2), x, y]

D[z (x^2 + y^2), {x, 2}]

D[z (x^2 + y^2), {{x, y}}]

D[z (x^2 + y^2), {{x, y, z}, 2}] (*Hesen 矩阵*)

D[z (x^2 + y^2), {x}, {y},
 NonConstants → {z}]
```

例1：求函数 $f(x) = \sin\left(x^2 + \sqrt{x}\right)$ 在x = 2 的导数

```
D[Sin[x^2 + Sqrt[x]], x] /. → 2

f[x_] := Sin[x^2 + Sqrt[x]];
f'[2]
```

在表达式中也能使用微积分中计算导数的单引号标记。请注意表示求导符号的单引号要求在英文输入状态下

例2：使用数学求导符号

```
Clear[f];
f[x_] := Sin[x^2];
f[x] + f'[x] + f''[x]
```

4-2-2 全微分

- 在Mathematica中，D[f, x] 计算变量为f关于x的偏导数，系统默认f 中的其它变量与 x无关；
- Dt[f] 给出f的全微分形式
- Dt[f, x] 给出f的全导数，系统默认 f 中所有变量是 x 的函数。
- 对于f中不依赖于x的常量，要用选项 Constants –> {常量1, 常量2, ...}作出说明。

```
D[x^2 + y^2, x]
```

```
Dt[x^2 + y^2, x]
Dt[x^2 + y^2, x, Constants → {y}]
```

```
Dt[x^2 + y^2]
```

```
Dt[x^2 + y^2 + z^2, {x, 2}]
```

例3：x, y的函数关系由参数方程 $x = 2 t^2$, $y = Sin(t)$ 确定，求 y 关于 x 的二阶导数。

```
x[t_] := 2 * t^2; y[t_] := Sin[t];
```

```
dx := D[x[t], t]
dy := D[y[t], t]
d1 = dy / dx
```

```
Y[t_] := dy / dx
d2 = D[Y[t], t] / D[x[t], t]
```

第4讲 微积分

4 - 3 不定积分和定积分

4-3-1 不定积分

- 计算不定积分的命令是Integrate[f, x]
  输出结果中省略积分常数。
- 计算二重积分的命令是Integrate[f, x, y]
  积分的顺序是从右自左，先对变量y做积分计算，再对变量x做积分计算
- Integreate主要计算只含有 "简单函数" 的被积函数。"简单函数" 包括有理函数、指数函数、对数函数、
  三角和反三角函数。

例1：计算下列不定积分

(1)$\int 3ax^2 dx$　　(2)$\int \sqrt{x^2-a^2}\,dx$

(3)$\int \sqrt{\cos x}\,dx$　　(4)$\int \frac{1}{1+x^4} dx$

```
Integrate[3 a x^2, x]
```

```
Integrate[Sqrt[x^2 - a^2], x]
```

```
Integrate[Sqrt[Cos[x]], x]
```

```
Integrate[1 / (1 + x^4), x]
```

```
TraditionalForm[%]
```

- Integrate可以计算形式上的积分

例2：$\int e^x (f(x) + f'(x))\,dx$

```
Integrate[E^x * (f[x] +
  Derivative[1][f][x]), x]
```

- Integrate可以计算分段函数积分

例3：$f(x)=\begin{cases} x & x \geqslant 1 \\ 1 & x < 1 \end{cases}$，求 $\int f(x)\,dx$.

```
Integrate[Piecewise[{{1, x < 1}, {x, x ⩾ 1}}], x]
```

- Mathematica可以对向量值函数积分

```
Integrate[{x, E^x, Sin[x]}, x]
```

- Mathematica提供如Bessel函数、Gamma函数 和Beta函数等二三十个数学物理特殊函数可以用来表示积分结果。

例4：$\int e^{-x^2}\,dx$

```
Integrate[E^(-x^2), x]
```

■ Mathematica算不出结果的积分对被积函数做些化简外仍按Integrate形式输出.

```
Integrate[Sin[Sin[x]], x]
```

4-3-2 定积分

■ 计算定积分的函数形式是:
Integrate[f[x], {x, a, b}]
计算 $\int_a^b f(x)dx$的准确解

■ NIntegrate[f[x], {x, a, b}] 计算 $\int_a^b f(x)\,dx$的数值解

```
Integrate[Max[x, x^2 - 2], {x, -2, 3}]
```

```
{Integrate[x / (1 + Cos[x]), {x, 0, π / 2}], NIntegrate[x / (1 + Cos[x]), {x, 0, π / 2}]}
```

```
Integrate[Cos[x]^n, {x, 0, π / 2}, Assumptions → n > 1]
```

```
Integrate[Exp[-x^2], {x, 0, Infinity}]
```

4-3-3 多重积分

■ Integrate[f, {x, a, b}, {y, c, d}]
计算累次积分 $\int_a^b dx \int_c^d f(x, y)\,dy$ 的准确解

■ NIntegrate[f, {x, a, b}, {y, c, d}]
计算定积分 $\int_a^b dx \int_c^d f(x, y)\,dy$ 的数值解

例5：计算区域D上的二重积分， D由 y = x以及x轴， x = 1 围成.

$$\iint_D \text{Sin}[x + 2y]\,dx\,dy$$

```
ParametricPlot[{{x, x}, {1, x}}, {x, 0, 1}]
```

```
Integrate[x^2 + y^2, {x, 0, 1}, {y, 0, x}]
```

```
Integrate[(x^2 + y^2) Boole[y ≤ x], {x, 0, 1}, {y, 0, 1}]
```

例6：计算 $\iiint_V xy^2 z^3\,dx\,dy\,dz$
其中V是由曲面z = xy, z = 0, y = x, x = 1 围成。

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

```
Integrate[x * (y^2) * (z^3), {x, 0, 1}, {y, 0, x}, {z, 0, (x * y)}]
```

例7 计算两个圆柱体$x^2 + y^2 = 1$, $x^2 + z^2 \le 1$ 相交部分的体积

```
ineqa = x^2 + y^2 ≤ 1 && x^2 + z^2 ≤ 1;
Integrate[Boole[ineqa], {x, -Infinity, +Infinity},
 {y, -Infinity, +Infinity}, {z, -Infinity, +Infinity}]
```

例8：计算曲线积分 $\int_L \left(x^2 + x \cos x\right) ds$

```
Integrate[x^2 + x Cos[x], {x, y} ∈ Circle[]]
```

例9：计算单位球面面积

```
Integrate[1, {x, y, z} ∈ Sphere[]]
```

第 4 讲 微积分

4 - 4 级数

4-4-1 幂级数展开

- Series[expr, {x, x0, n}]　　将expr在x = x0点展开到n阶的幂级数
- Series[expr, {x, x0, n}, {y, y0, m}]
  先对y展开到 m阶再对x展开n阶幂级数

```
Series[Sin[x^2], {x, 0, 15}]
```

```
Series[f[x], {x, a, 3}]
```

```
Series[Sin[x] * Cos[y], {x, 0, 5}, {y, 0, 5}]
```

- 展开式中可能将分数指数，对数函数等作为基本元素。

```
Series[Sqrt[x], {x, 0, 10}]
```

```
Series[∛(x^2), {x, 0, 10}]
```

```
Series[Exp[Sqrt[x]], {x, 0, 6}]
```

```
Series[Exp[Sqrt[x]], {x, 1, 6}]
```

```
Series[x^x, {x, 0, 4}]
```

- Series命令可以计算无界函数在瑕点的洛朗展开式,展开式的最高次数可以是负数。

```
Series[1 / (E^x - 1), {x, 0, 10}]
```

```
Series[1 / ((E^x - 1)^10), {x, 0, -2}]
```

- Series命令可以计算函数在无穷远点的洛朗展开式

```
Series[1 / (x - 2), {x, Infinity, 4}]
```

4-4-2 幂级数的运算

- 幂级数求和
  - Sum[f, {i, imin, imax}]
    计算 $\sum_{i=imin}^{imax} f(i)$
  - Sum[f, {i, imin, imax, di}]
    计算 $f(imin) + f(imin + di) + f(imin + 2\,di) + \ldots f\left(imin + \left[\frac{imax}{imin}\right] di\right)$
  - Sum[f, {i, imin, imax}, {j, jmin, jmax}]　　计算 $\sum_{i=imin}^{imax}\left(\sum_{j=jmin}^{jmax} f(i, j)\right)$
- 无穷乘积

■ Product[f, {i, imin, imax}]

计算 $\prod_{i=imin}^{imax} f(i)$

```
Sum[x^i / i, {i, 1, 7}]
Sum[x^i / i, {i, 7}] (*省略求和下界时，默认从1开始  *)
Sum[x^i / i, {i, 1, 7 , 2}]
Sum[x^n / n!, {n, 0, Infinity}]
Sum[1 / i^4, {i, 1, Infinity}]
Sum[1 / (i! + (2 i)!), {i, 1, Infinity}]
N[%]
```

例：求幂级数 $\sum_{n=0}^{\infty} \frac{x^{4n+1}}{4n+1}$ 的收敛域与和函数。

```
a[n_] := 1 / (4 * n + 1);
f[x_, n_] := a[n] * x^(4 n + 1);
r = Limit[Abs[a[n + 1] / a[n]], n → Infinity]
R = 1 / r^(1 / 4)
```

检查端点是否收敛

```
Sum[f[1, n], {n, 1, Infinity}]
Sum[f[-1, n], {n, 1, Infinity}]
```

两个端点都是发散点。收敛域 (-1,1)

```
Sum[f[x, n], {n, 1, Infinity}]
Sum[D[f[x, n], x], {n, 1, Infinity}]
Integrate[%, x]
```

■ 给出幂级数中某一项的系数

SeriesCoefficient[f, n] 级数f中x^n的系数

SeriesCoefficient[f, {x, x0, n}] 函数f在x0的展开式中 (x - x0)^n的系数

```
SeriesCoefficient[Sin[x], {x, 0, 3}]
SeriesCoefficient[Exp[x], {x, 0, n}]
Series[Log[1 + x], {x, 0, 5}]
SeriesCoefficient[%, 2]
```

■ 反函数级数

InverseSeries[s, {x, x0, n}] 给出级数s的反函数的幂级数展开式

```
t = Series[Sin[x], {x, 0, 7}]
InverseSeries[t]
Series[ArcSin[x], {x, 0, 7}]
```

- 幂级数复合

ComposeSeries[s1, s2] 表示用幂级数s2代换幂级数s1中的变量x

```
ComposeSeries[Series[Cos[x], {x, 0, 10}], Series[Sin[x], {x, 0, 10}]]
```

```
Series[Cos[Sin[x]], {x, 0, 10}]
```

```
Series[Cos[x], {x, 0, 10}] /. x → Series[Sin[x], {x, 0, 10}]
```

4-4-3 Fourier级数

- FourierSeries[f[x], x, n]
  计算以2Pi为周期的函数f[x]的n阶傅里叶级数 $\sum_{k=-n}^{n} c_k e^{i k t}$
  其中 $c_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t)\, e^{-i k t}\, dt$.
- FourierSeries[f[x, y, z ..], { x, y, z ..}, {n1, n2, n3 ..}]
  计算多元函数的傅里叶级数

```
a = FourierSeries[x^2, x, 3]
```

```
Plot[a, {x, -3 Pi, 3 Pi}]
```

```
b = FourierSeries[x * y, {x, y}, {2, 2}]
```

```
Plot3D[b, {x, -3 Pi, 3 Pi}, {y, -3 Pi, 3 Pi}]
```

对于以2L为周期的函数，要用FourierParameters->{1,2Pi/L}说明

```
FourierSeries[Abs[x], x, 3, FourierParameters → {1, 2 Pi}]
```

```
Plot[%, {x, -3, 3}]
```

- 正弦级数与余弦级数

FourierSinSeries[f[x], x, n]
f[x]是以2π为周期的函数，在[-π,π]上是奇函数

FourierCosSeries[f[x], x, n]
f[x]是以2π为周期的函数，在[-π,π]上是偶函数

若函数周期不是2π，同样用FourierParameters说明

```
c = FourierSinSeries[x, x, 5]
```

```
Plot[c, {x, -3 π, 3 π}]
```

```
d = FourierCosSeries[x^2, x, 5, FourierParameters → {1, π}]
```

```
Plot[d, {x, -4, 4}]
```

第 4 讲 微积分

4-5 微分方程

4-5-1 求解常微分方程

求解常微分方程和常微分方程组的一般形式：

- DSolve[eqns, y[x], x]
  解y(x)的微分方程或方程组 eqns，x为变量。
- DSolve[eqns, y, x]　　在纯函数的形式下求解，纯函数部分请看第7讲。

```
DSolve[y'[x] == x * y[x], y[x], x]
```

```
DSolve[{y[x] == -z'[x], z[x] == -y'[x]}, {y[x], z[x]}, x]
```

一般一个一阶线性微分方程都可以通过积分运算求解，但是如果方程的个数大于1，或阶数大于2，求解就没有固定的方法，一些简单的二阶线性方程的解被当做特殊函数来表示其它方程的解。

```
DSolve[y''[x] - Cot[x]^2 y[x] == 0, y[x], x]
```

一些特殊类型的二阶线性微分方程可能有形式简单的解

```
DSolve[x^2 y''[x] + y[x] == 0, y[x], x]
```

三阶以上的方程只有极少的方程能够被解出

```
DSolve[y'''[x] == x * Sin[y[x]], y[x], x]
```

例1：求解初值问题 $\left(1+x^2\right) y''+2xy'-6x^2-2=0,\ y(-1)=0,\ y'(-1)=0.$

```
sol =
 DSolve[{(1 + x^2) * y''[x] + 2 x * y'[x] - 6 * x^2 - 2 == 0, y[-1] == 0, y'[-1] == 0}, y[x], x]
```

```
Plot[y[x] /. sol, {x, -1, 4}]
```

- 纯函数形式的解可以代入微分的运算 DSolve[eqns,y,x]

```
sol1 = DSolve[{y'[x] + y[x] == a Sin[x], y[0] == 0}, y, x]
```

```
FullSimplify[y''[x] + y[x]^2 /. sol1]
```

```
sol2 = DSolve[{y'[x] + y[x] == a Sin[x], y[0] == 0}, y[x], x]
```

```
FullSimplify[y''[x] + y[x]^2 /. sol2]
```

4-5-2 求解偏微分方程

求解偏微分方程的命令是
DSolve[eqn, y, {x1, x2, ..}]
DSolve[eqns, {y1,y2,..}, {x1, x2, ..}]

例2：求解偏微分方程 $\frac{\partial y(x1,\ x2)}{\partial x1}+\frac{\partial y(x1,\ x2)}{\partial x2}=\frac{1}{x1\ x2}$

```
DSolve[D[y[x1, x2], {x1}] +
  D[y[x1, x2], {x2}] ==
 1 / (x1 * x2), y[x1, x2],
 {x1, x2}]
```

例3：求解调和方程$u_{xx} + u_{yy} = 0$

```
DSolve[D[u[x, y], {x, 2}] + D[u[x, y], {y, 2}] == 0, u, {x, y}]
```

第5讲　线性代数

5-1 矩阵的定义

5-1-1 矩阵的输入（1）

- 在Mathematica中， 向量与矩阵是用列表表示的，向量是简单的列表，矩阵是列表的列表。
- 向量：
  - Table[expr,{i, a,b,d}] 生成列表{expr(a),expr(a+d),expr(a+2d),...expr(b)}
    a，d的缺省值为1
  - Array[f,n]　生成列表{f[1], f[2],.....f[n]}
    f是一元函数.
- 矩阵：
  - Table[expr, {i,ai,bi,di} , {j,aj,bj,dj}]
    expr为矩阵元素的通项公式，是循环变量i,j的表达式，
    a为循环初值，b为循环终值上界，d为循环步长。
    当a或d缺省时，其值为1.
  - Array[f, {m,n}]
    定义m行n列的矩阵，矩阵元素f[i,j], f是二元函数.
    Array[f, m,n]
    定义m维向量，{f[n],f[n+1],...f[n+m-1]}.

例题：Table, Array函数的用法与区别

```
a[i_, j_] := 1 / (i + j - 1);
```

```
A1 = Table[a, {2}, {2}]
```

```
A2 = Array[a, {2, 2}]
```

- MatrixForm[列表] 将列表用矩阵形式表示.

```
A2 // MatrixForm
```

```
MatrixForm[A2]
```

例题：构造一个由0-9随机数字构成的5阶方阵

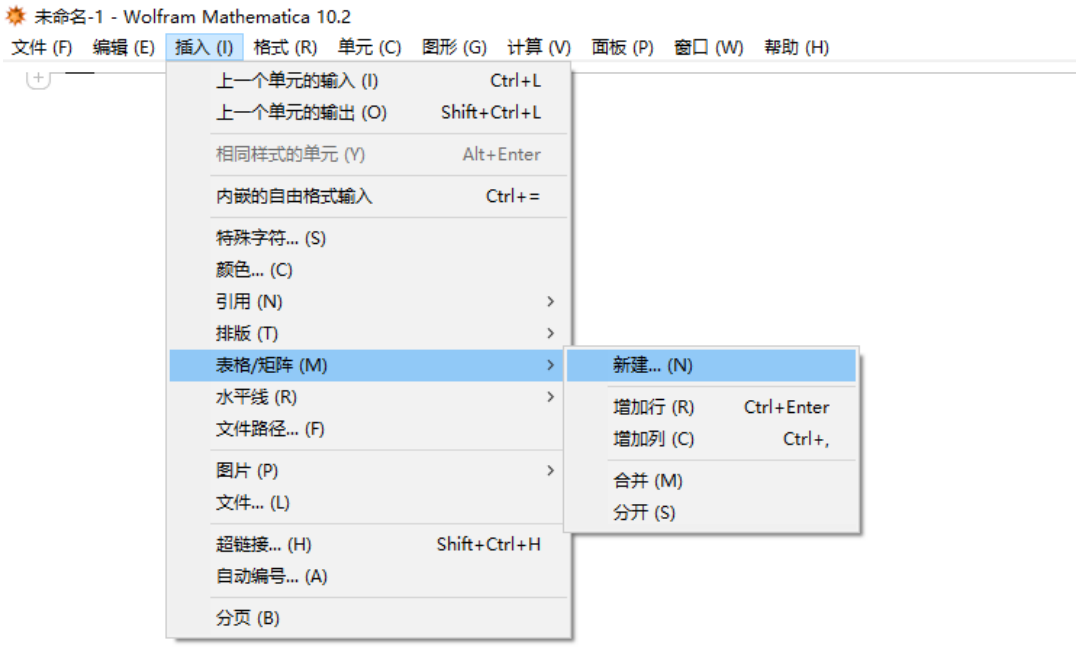
```
Table[RandomInteger[{0, 9}, 5], {i, 1, 5, 1}] // MatrixForm
```

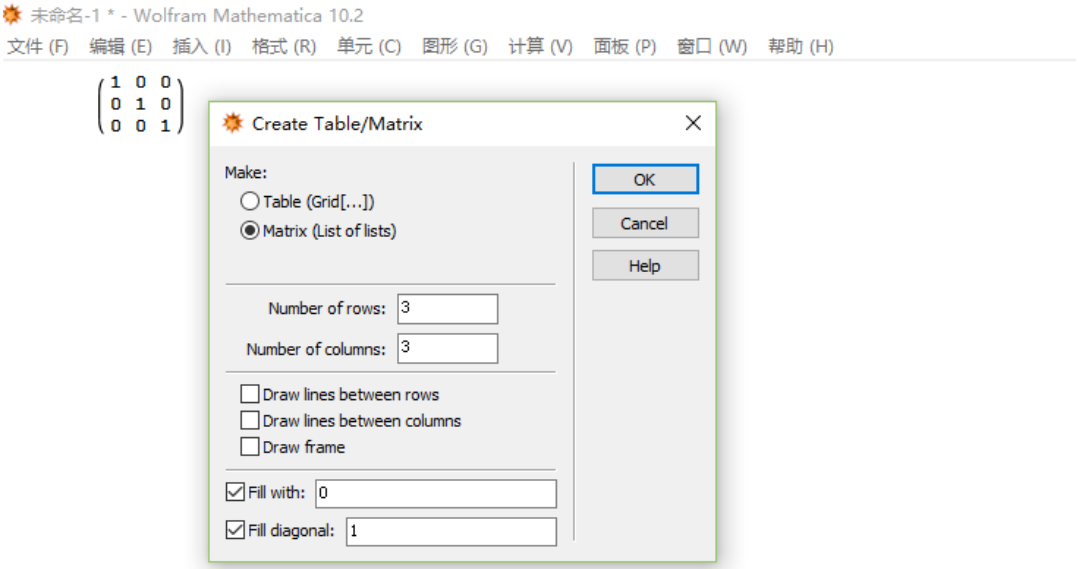
例题：构造一个5阶上三角方阵，上三角元素为行号与列号之和。

```
Table[If[i ≤ j, i + j, 0], {i, 5}, {j, 5}] // MatrixForm
```

5-1-1 矩阵的输入（2）

- 可以通过Mathematica的菜单项□□□Insert->Table/Matrix->New...□□□来插入一个矩阵
  然后直接修改矩阵的元素





- 矩阵元素不多时，可直接列表输入

```
A = {{1, 2, 3}, {2, 3, 4}} // MatrixForm
```

5-1-2 定义特殊矩阵

- 定义对角阵DiagonalMatrix[对角线元素列表]

```
DiagonalMatrix[{a, b, c}] // MatrixForm
```

- 定义n阶单位矩阵 IdentityMatrix[n]

```
IdentityMatrix[5] // MatrixForm
```

- 定义稀疏矩阵
  - SparseArray[rules,dims,val]
    由rules定义的具有维数dims的稀疏矩阵，
    未指明的矩阵元素取值val。
  - SparseArray[{{i1,ji}->v1,{i2,j2}->v2,...},{m,n}]
    按下标位置定义稀疏矩阵元素

例题：定义7阶准对角阵，主对角线元素均为2，与主对角线元素相邻的元素为1，其余元素为0

```
SparseArray[{{i_, i_} → 2, {i_, j_} /; j == i + 1 → 1, {i_, j_} /; j == i - 1 → 1},
  {7, 7}] // MatrixForm
```

例题：定义4×5阶稀疏矩阵M，其中M[2,3]=1，M[4,2]=5.

```
M = SparseArray[{{2, 3} → 1, {4, 2} → 5}, {4, 5}] // MatrixForm
```

```
HilbertMatrix[{3, 4}] // MatrixForm
```

```
HankelMatrix[4] // MatrixForm
```

```
HankelMatrix[{a, b, c, d}] // MatrixForm
```

```
RotationMatrix[θ] // MatrixForm
RotationMatrix[θ, {1, 1, 1}] // MatrixForm
```

5-1-3 矩阵分量的操作

- 由于矩阵是一个特殊的表，对矩阵分量的操作可以通过对表的操作进行,表的操作函数都同样适用于矩阵.
- A[[i,j]], Part[A,i,j]
  A的i行j列元素或子矩阵，i、j为指标或指标集
  - 语句中的i,j可以是一个整数（正数i表示第i个位置，负数-i表示倒数第i个位置），
    也可以是一个指标集{i1,...,ik}或All
- Take[A,i,j] A的子矩阵，i、j代表指标集
  - 语句中的i,j可以是一个整数（正数i表示取前i个位置，负数-i表示取后i个位置），
    或有形式{m,n,s}（表示以步长s取从m到n的位置）

例题：对题中的5阶方阵完成指定的操作。

$$\begin{pmatrix} 11 & 12 & 13 & 14 & 15 \\ 21 & 22 & 23 & 24 & 25 \\ 31 & 32 & 33 & 34 & 35 \\ 41 & 42 & 43 & 44 & 45 \\ 51 & 52 & 53 & 54 & 55 \end{pmatrix}$$

1. 提取第2行第3列元素　　2. 提取第4行全部元素
3. 提取第5列全部元素　　4.提取第2,4行，1，3，5列的子矩阵
5. 将第一行元素乘以-2，加到第三行上.

```
matrix = Table[10 i + j, {i, 1, 5}, {j, 1, 5}]

matrix[[2, 3]]

matrix[[4]]

matrix[[All, 5]]

Part[matrix, {2, 4}, {1, 3, 5}]

Take[matrix, {2, 4, 2}, {1, 5, 2}]

matrix[[3]] -= 2 matrix[[1]];
matrix // MatrixForm
```

第5讲　线性代数

5-2 矩阵的基本运算

5-2-1 矩阵的算术运算

- A+B或Plus[A,B]　　矩阵或向量加法
- A-B或Subtract[A,B]　矩阵或向量减法
- -A或Minus[A]　　负矩阵或负向量
- A*B或Times[A,B]　　对应元素相乘
- A/B或Divide[A,B]　　对应元素相除
- A^n或Power[A,n]　　每个元素方幂

```
A = {{1, 2}, {3, 4}}; MatrixForm[A]
B = {{5, 6}, {7, 8}}; MatrixForm[B]
A + B
A - B
A * B
A / B
A^3
```

```
A = {{1, 2}, {3, 4}} // MatrixForm
 B = {{5, 6}, {7, 8}} // MatrixForm
A + B
```

```
5 + A
```

```
{5, 6} + A // MatrixForm
```

```
{{5}, {6}} + A
```

- A.B或Dot[A,B]矩阵乘法或向量的内积
- Cross[A,B]向量的外积
- MatrixPower[A,n]方阵A的方幂
- MatrixExp[A]方阵A的指数函数e^A
- Transpose[A]矩阵A的转置
- ConjugateTranspose[A]复矩阵A的共轭转置

```
a = {1, 2, 3}; b = {4, 5, 6}; {a.b, Cross[a, b]}
```

```
MatrixPower[A, 2]
```

```
A // MatrixForm
Transpose[A] // MatrixForm
```

5-2-2 行列式和逆矩阵

- Det[A]方阵A的行列式
- Inverse[A]方阵A的逆矩阵
- Minors[A]方阵A的余子式
  - Minors[A]也是一个n阶方阵,它在(i,j)位置上的元素是A关于(n-i+1,n-j+1)位置的余子式
- Minors[A,k]方阵A的所有k阶子矩阵的行列式
  - Minors[A,k]生成一个矩阵，其元素是矩阵所有k阶子方阵的行列式.

例题 计算行列式 $\begin{vmatrix} a & b & c & d \\ a & a+b & a+b+c & a+b+c+d \\ a & 2a+b & 3a+2b+c & 4a+3b+2c+d \\ a & 3a+b & 6a+3b+c & 10a+6b+3c+d \end{vmatrix}$

```
A = {{a, b, c, d}, {a, a + b, a + b + c, a + b + c + d}, {a, 2 a + b,
    3 a + 2 b + c, 4 a + 3 b + 2 c + d}, {a, 3 a + b, 6 a + 3 b + c, 10 a + 6 b + 3 c + d}};
A // MatrixForm
Det[A]
```

例题 计算范德蒙行列式 $\begin{vmatrix} 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 & x_5^2 \\ x_1^3 & x_2^3 & x_3^3 & x_4^3 & x_5^3 \\ x_1^4 & x_2^4 & x_3^4 & x_4^4 & x_5^4 \end{vmatrix}$

```
van = Table[x[i]^j, {j, 0, 4}, {i, 1, 5}]
% // MatrixForm
Det[van]
```

```
Det[van] // Simplify
```

例题： 设A=$\begin{pmatrix} 1 & 2 & 2 \\ 2 & 3 & 3 \\ 3 & 4 & 5 \end{pmatrix}$，求其逆矩阵, 余子式.

```
A = {{1, 2, 2}, {2, 3, 3}, {3, 4, 5}}
```

```
Inverse[A] // MatrixForm
```

```
Minors[A]
```

```
{{-1, -1, 0}, {-2, -1, 2}, {-1, 1, 3}}
```

```
Minors[A, 1]
Minors[A, 2]
Minors[A, 3]
```

5-2-3 方阵的特征值和特征向量

- CharacteristicPolynomial[A,x]　方阵A的特征多项式
- Eigenvalues[A]　计算出方阵A的所有特征值；
- Eigenvectors[A]　计算出方阵A的所有特征向量
- Eigensystem[A]　同时计算出方阵A的所有特征值和特征向量

例题： 计算方阵的特征多项式，特征值，与特征向量，验证矩阵满足特征方程

```
A = {{-2, 4, 1, 0, 2}, {-4, 6, 1, -1, 0},
    {0, 0, 3, 0, 0}, {0, 0, 2, 3, 0}, {0, 0, 1, 0, 3}};
A // MatrixForm
```

$$\begin{pmatrix} -2 & 4 & 1 & 0 & 2 \\ -4 & 6 & 1 & -1 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 1 & 0 & 3 \end{pmatrix}$$

```
CharacteristicPolynomial[A, x]
```

```
108 IdentityMatrix[5] - 216 A + 171 MatrixPower[A, 2] -
  67 MatrixPower[A, 3] + 13 MatrixPower[A, 4] - MatrixPower[A, 5] // MatrixForm
```

```
Eigenvalues[A]
Eigenvectors[A]
```

```
Eigensystem[A]
```

5-2-4 矩阵的秩、迹、范数

- MatrixRank[A]　矩阵A的秩
- Norm[A,p]　矩阵A的范数，p=1,2,Infinity，缺省值2
- Norm[v,p]　向量v的范数，p>=1或p=Infinity，缺省值2
- Total[v]　向量v的元素和v1+...+vn
- Tr[A]矩阵A的迹（对角元素和）a11+...+ann
- Tr[A,f]矩阵A的广义迹f[a11,...,ann]

$$\mathbf{A} = \begin{pmatrix} -2 & 4 & 1 & 0 & 2 \\ -4 & 6 & 1 & -1 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 1 & 0 & 3 \end{pmatrix};$$

```
{MatrixRank[A], Tr[A], Total[A]}
```

```
{5, 13, {-6, 10, 8, 2, 5}}
```

例题：计算矩阵， 向量的范数.

```
matrix = {{a + 1, b + 2, c + 3}, {a + 2, b + 3, c + 4}};
Norm[matrix, 1]
Norm[matrix, Infinity]
```

第5讲　线性代数

5-3 矩阵的高级运算

5.3.1线性方程组的求解(I)

- 求解线性方程组 AX=B
  - 当方阵A的行列式不为零时，线性方程组AX=B有唯一解$X=A^{-1}B$
  - 当A的行列式为零或者A不是方阵的时候，线性方程组可能无解或有无穷多解。
  - 线性方程组有无穷多解的时候，由基础解系向量的线性组合加上一个特解组成线性方程组的全部解

LinearSolve[A, B]	求方程组AX = B的一个特解X
NullSpace[A]	求齐次方程组AX = 0 的一个基础解系

例题1： 求解线性方程组 $\begin{cases} x+3y+2z=9 \\ 2x-y+3z=3 \\ 3x+2y-z=6 \end{cases}$

```
A = {{1, 3, 2}, {2, -1, 3}, {3, 2, -1}};
B = {9, 3, 6};
Det[A]

LinearSolve[A, B]
```

求解线性方程组 $\begin{cases} x+3y+2z=9 \\ 2x-y+3z=3 \\ 3x+2y+5z=6 \end{cases}$

```
u = {{1, 3, 2}, {2, -1, 3}, {3, 2, 5}};

LinearSolve[u, B]
```

例题2： 求解线性方程组 $\begin{cases} x+3y+z=1 \\ 2x-y+3z=4 \\ x-4y+2z=3 \end{cases}$

```
Clear[A, B]; A = {{1, 3, 1}, {2, -1, 3}, {1, -4, 2}};
B = {1, 4, 3};
Det[A]

particular = LinearSolve[A, B]

basis = NullSpace[A]

sol = basis[[1]] * t + particular

Print["线性方程组的通解为   x=", sol[[1]], ",    y=", sol[[2]], ",   z=", sol[[3]]]
```

检验结果是否正确

```
A.sol // Expand
```

5.3.1线性方程组的求解（2）

在线性代数课程中，我们求解线性方程组 AX=B的方法是 Gauss—Jordan方法
将增广矩阵(A|B)化为行阶梯型，从而可以看出方程组的通解。

- RowReduce[A]　将矩阵A化为行最简形

例题3：求解线性方程组 $\begin{cases} x+y+z-u-v=5 \\ 3x-2y+z+2u-v=3 \\ 2x-3y+3u=-2 \end{cases}$

```
Clear[A, sol];
A = {{1, 1, 1, -1, -1, 5}, {3, -2, 1, 2, -1, 3}, {2, -3, 0, 3, 0, -2}};
MatrixForm[A]
r = RowReduce[A] // MatrixForm
```

从系数矩阵中可以看出方程组的解.

5.3.2线性空间相关运算(I)

- 考虑由一组向量生成的线性空间，我们需要了解一下问题：
  - 线性空间的维数
  - 线性空间的一组标准正交基
  - 生成线性空间的向量的极大无关组

这些问题可以用一下命令给出解答

- 维数 = Rank[{α_1, α_2, ... α_n}], n个向量组成的矩阵的秩
- Orthogonalize[{α_1, α_2, ... α_n}] ,对n个行向量作标准正交化
- 从最简行梯形矩阵中可以看出极大无关组

例题4：设A = $\begin{pmatrix} 2 & -1 & -1 & 1 & 2 \\ 1 & 1 & -2 & 1 & 4 \\ 4 & -6 & 2 & -2 & 4 \\ 3 & 6 & -9 & 7 & 9 \\ 2 & -5 & 3 & -3 & 2 \end{pmatrix}$ ，求的列向量组的一个极大无关组.

```
Clear[];
A = {{2, -1, -1, 1, 2}, {1, 1, -2, 1, 4},
   {4, -6, 2, -2, 4}, {3, 6, -9, 7, 9}, {2, -5, 3, -3, 2}};
A // MatrixForm
RowReduce[A] // MatrixForm
```

例题5：将向量 a1=(1,1,0,0),a2=(1,0,1,0), a3=(-1,0,0,1),a4=(1,-1,-1,1)转化为标准正交基。

```
Clear[];
A = {{1, 1, 0, 0}, {1, 0, 1, 0}, {-1, 0, 0, 1}, {1, -1, -1, 1}};
MatrixForm[A]

Orthogonalize[A] // MatrixForm
```

5.3.2线性空间相关运算(2)

- Normalize[v,f]　　将向量单位化
- Projection[u,v,f]　求向量u在v方向上的投影分量

f是线性空间中内积的定义，f 缺省时使用标准复内积和L2范数

```
Normalize[{a, b, c}]
```

```
Projection[{x, y, z}, {a, b, c}, Dot]
```

例题6：在区间[0,1]上定义连续函数的内积

$(f, g) = \int_0^1 x f(x) g(x) dx$，

求多项式空间{a+bx+cx^2 | a, b, c 是任意实数}的一组标准正交基.

```
F[f_, g_] := Integrate[x * f * g, {x, 0, 1}];
Orthogonalize[{1, x, x^2}, F]
```

```
Projection[f[x], g[x], F]
```

5.3.3矩阵的乘积分解

- JordanDecomposition[A]
  对方阵A返回{P,J}使A=PJP-1，J是Jordan标准形。
- LUDecomposition[A]
  对方阵A返回{lu,p,c}使A=P.L.U。
  L是单位下三角，U是上三角，L和U被压缩到一个矩阵中表示，置换方阵P，表示对A进行行位置的变换，c是矩阵A的L^∞条件数。

例题7：设A = $\begin{pmatrix} -2 & 4 & 1 & 0 & 2 \\ -4 & 6 & 1 & -1 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 1 & 0 & 3 \end{pmatrix}$，求A的Jordan标准型和过渡矩阵。

```
Clear[]; A = {{-2, 4, 1, 0, 2}, {-4, 6, 1, -1, 0}, {0, 0, 3, 0, 0}, {0, 0, 2, 3, 0}, {0, 0, 1, 0, 3}};
```

```
MatrixForm /@ JordanDecomposition[A]
```

例题8：设A = $\begin{pmatrix} 2 & 3 & 4 \\ 4 & 11 & 14 \\ 6 & 29 & 43 \end{pmatrix}$，求A的LU分解。

```
Clear[]; A = {{2, 3, 4}, {4, 11, 14}, {6, 29, 43}};
```

```
{lu, p, c} = MatrixForm /@ LUDecomposition[A]
```

其中L=$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{pmatrix}$，U=$\begin{pmatrix} 2 & 3 & 4 \\ 0 & 5 & 6 \\ 0 & 0 & 7 \end{pmatrix}$，p=$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$表示没有进行行交换.

$$\mathbf{l} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{pmatrix}; \quad \mathbf{u} = \begin{pmatrix} 2 & 3 & 4 \\ 0 & 5 & 6 \\ 0 & 0 & 7 \end{pmatrix}; \quad \mathbf{l.u}$$

$$\mathbf{Clear[]; A} = \begin{pmatrix} 3 & 1 & 2 \\ 6 & 12 & 11 \\ 9 & 5 & 7 \end{pmatrix};$$

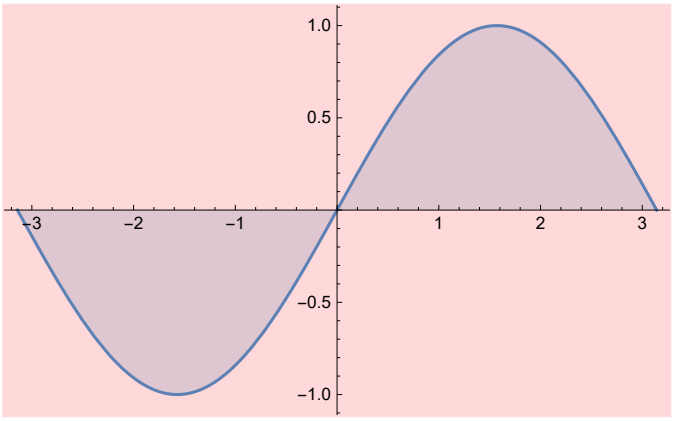
```
{lu, p, c} = MatrixForm /@ LUDecomposition[A]
```

$$\mathrm{L} = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 2 & 5 & 1 \end{pmatrix}, \mathrm{U} = \begin{pmatrix} 3 & 1 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{pmatrix}$$

$$\mathbf{l} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 5 & 1 \\ 3 & 1 & 0 \end{pmatrix}; \quad \mathbf{u} = \begin{pmatrix} 3 & 1 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{pmatrix}; \quad \mathbf{l.u // MatrixForm}$$

第6讲 在Mathematica 中作图

6－1 一元函数作图Plot



Plot应用对象：在直角坐标系中绘制单变量函数f (x) 在指定区间上的图形。

Plot命令形式：

Plot[f, {x, xmin, xmax}, 选项]
在区间 {xmin, xmax} 上按选项定义值绘制单变量函数f (x) 的图形

Plot[{f1, f2, ...}, {x, xmin, xmax}, 选项]
在区间 {xmin, xmax} 上按选项定义同时绘制函数f1 (x) , f2 (x) , ...的图形

1. 关于单变量函数 f(x) ：显式函数

例：画出f (x) = sinxcos2x, 在区间[－2 Pi, 2 Pi] 上的图形。

```
Plot[Sin[x] Cos[2 x], {x, -2 Pi, 2 Pi}]
```

例：在区间[－5, 5] , 在同一坐标系中画出 sinx, cosx, sinxcosx的函数曲线。

```
Plot[{Sin[x], Cos[x], Sin[x] Cos[x]}, {x, -5, 5}]
```

例：画出 $\left(\int \sin(x)\cos(2x)\,dx\right)$, 在区间[－6.3, 6.3] 上的图形.

```
Plot[Integrate[Sin[x] Cos[2 x], x], {x, -6.3, 6.3}]
```

```
Plot[Evaluate[Integrate[Sin[x] Cos[2 x], x]], {x, -6.3, 6.3}]
```

```
g = Integrate[Sin[x] Cos[2 x], x]; Plot[g, {x, -6.3, 6.3}]
```

```
g[x_] = Integrate[Sin[x] Cos[2 x], x]; Plot[g[x], {x, -6.3, 6.3}]
```

2. 关于绘图区间

(1) 连续区间 {变量名, 表达式1, 表达式2}

例：
```
Plot[Sin[x], {x, -1, 1}]
```

```
a = 3; b = 7; Plot[Sin[y], {y, a - b, a + b}]
```

(2) 画图变量在几何区域中取值 Plot[f, x ∈ S]

例：

```
S = ImplicitRegion[x ≤ -2 || x ≥ 3, {x}]
```

```
Plot[Cos[x], {x} ∈ S]
```

3. 关于选项：可有可无，可多可少.

例：给x、y坐标轴分别加标记 "x", "f(x)"，设置背景色为浅红色。

```
Plot[(x^2 - x) Sin[x], {x, 2, 16}, AxesLabel → {"x", "f(x)"}, Background → LightRed]
```

例：给图形加上框线和网格.

```
Plot[x Sin[x], {x, 0, 3}, Frame → True,
 GridLines → Automatic, Background → LightBlue]
```

```
Plot[{Sin[x], Cos[x], Sin[x] Cos[x]}, {x, -5, 5}, PlotStyle → {Thick, Red, Dashed}]
(* ,PlotLegends→"Expressions"    *)
```

例：观察和比较 Floor[x], Round[x], Ceiling[x] 函数。

```
Plot[Floor[x], {x, -3.5, 3.5}, Filling → Axis]
```

```
Plot[Round[x], {x, -3.5, 3.5}, Filling → Axis]
```

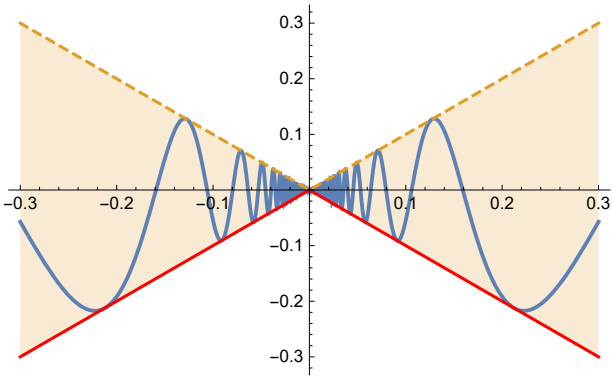
```
Plot[Ceiling[x], {x, -3.5, 3.5}, Filling → Axis]
```

选项名	默认值	说明
AspectRatio	1 / GoldenRatio	高宽比
Axes	True	是否绘制轴
ClippingStyle	None	如何绘制曲线被剪切的区域 »
ColorFunction	Automatic	确定曲线颜色的方法
ColorFunctionScaling	True	是否缩放ColorFunction的变量
EvaluationMonitor	None	在每次函数计算时，需要计算的表达式
Exclusions	Automatic	x 中排除的点
ExclusionsStyle	None	排除点的绘制样式
Filling	None	每条曲线下填充
FillingStyle	Automatic	填充的样式
MaxRecursion	Automatic	递归子划分的最大数量
Mesh	None	在每条曲线上绘制多少个网格点
MeshFunctions	{#1 &}	如何决定网格点的放置位置
MeshShading	None	如何在网格点间绘制阴影区域
MeshStyle	Automatic	网格点的样式
Method	Automatic	修饰曲线的方法
PerformanceGoal	$PerformanceGoal	试图优化哪些方面的性能
PlotLegends	None	曲线的图例
PlotPoints	Automatic	样本点的初始数量
PlotRange	{Full, Automatic}	y 的范围或包含的其它值
PlotRangeClipping	True	是否在曲线范围内剪切
PlotStyle	Automatic	指定每条曲线样式的图形指令
PlotTheme	$PlotTheme	绘图的整体外观主题
RegionFunction	(True &)	如何确定是否包含一个点
TargetUnits	Automatic	在绘图中显示的单位
WorkingPrecision	MachinePrecision	内部计算使用的精度

例：填实第2条曲线到第3条曲线之间的区域。

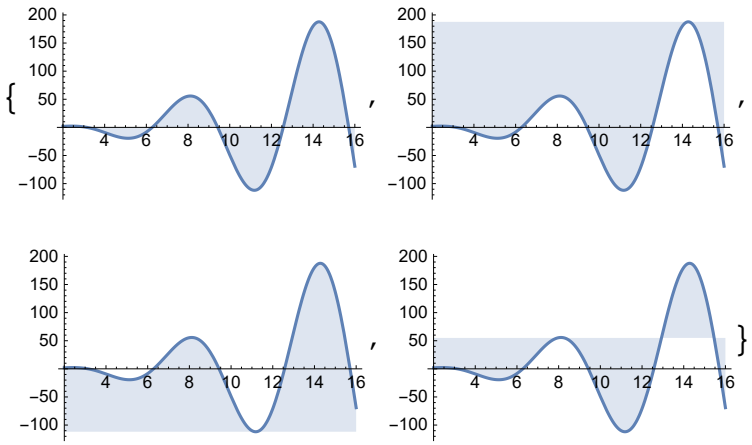
```
Plot[{x Sin[1/x], Abs[x], -Abs[x]}, {x, -0.3, 0.3}, Filling → {2 → {3}}]
```

```
Plot[{x Sin[1/x], Abs[x], -Abs[x]}, {x, -0.3, 0.3},
 Filling → {2 → {3}}, PlotStyle -> {Thick, Dashed, Red}]
```



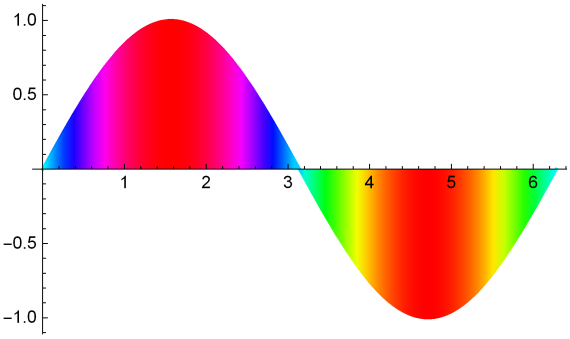
例：观察4种方式填实方式。

```
Table[Plot[(x^2 - x) Sin[x], {x, 2, 16}, Filling → f], {f, {Axis, Top, Bottom, 55}}]
```



例：给曲线增彩加色.

```
Plot[Sin[x], {x, 0, 2 Pi},
 ColorFunction -> Function[{x, y}, Hue[y]], Filling → Axis]
```



第6讲　在 Mathematica 中作图

6－2 二维参数作图 和极坐标作图

1. 二维参数作图 ParametricPlot

```
ParametricPlot[{fx, fy}, {u, umin, umax}, 选项]
```
按照选项值，在[umin, umax] 范围内画一条参数曲线

```
ParametricPlot[{{fx, fy}, {gx, gy}, ...}, {u, umin, umax}, 选项]
```
按照选项值，在[umin, umax] 范围内画一组参数曲线

```
ParametricPlot[{fx, fy}, {u, umin, umax}, {v, vmin, vmax}, 选项]
```
按照选项值，画出参数所示函数的区域

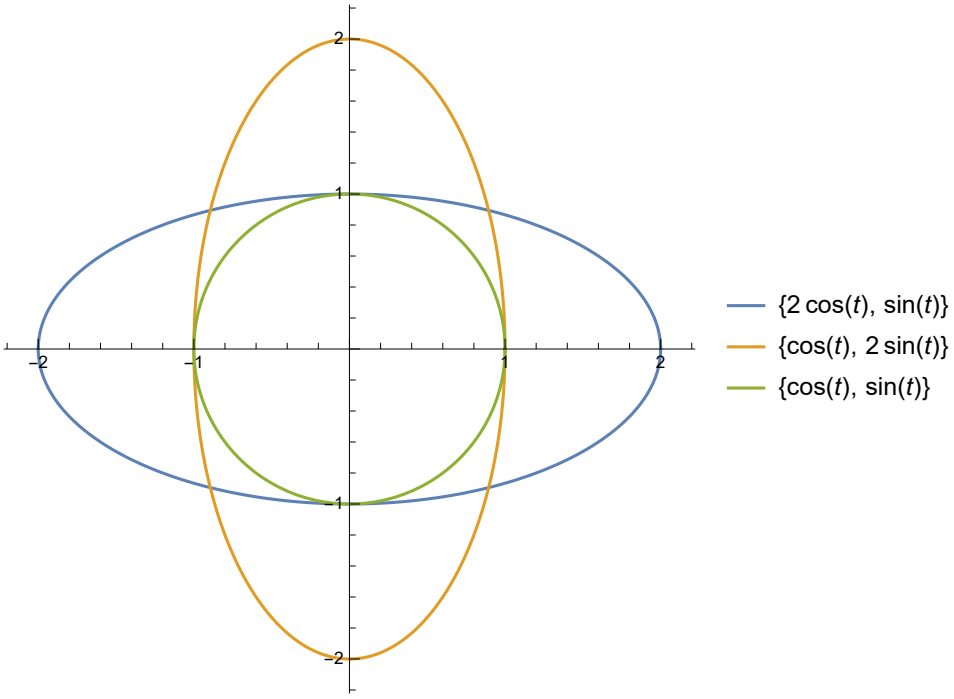
```
ParametricPlot[{{fx, fy}, {gx, gy}, ...}, {u, umin, umax}, {v, vmin, vmax}]
```
绘制一组参数区域

例1：画一条参数曲线.

```
ParametricPlot[{Sin[t], Sin[2 t]}, {t, 0, 2 Pi}]
```

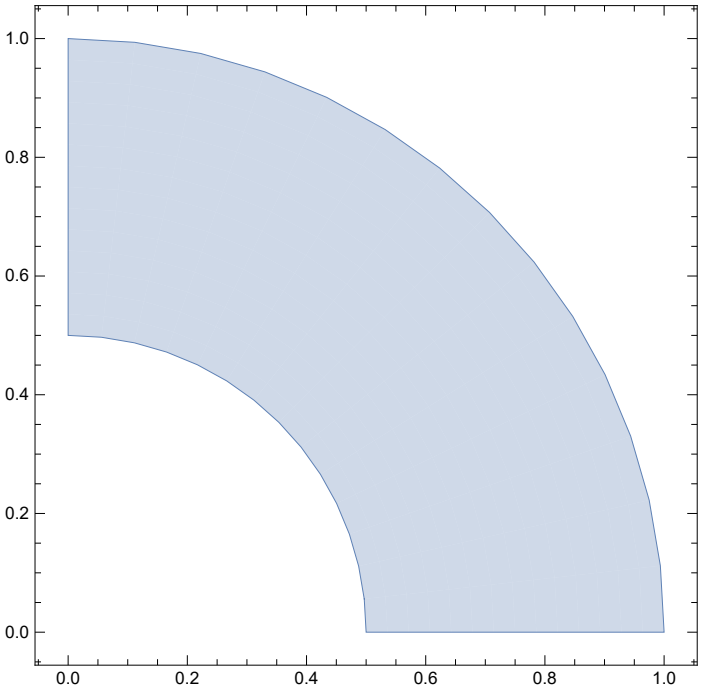
例2：画一组参数曲线.

```
ParametricPlot[{{2 Cos[t], Sin[t]}, {Cos[t], 2 Sin[t]}, {Cos[t], Sin[t]}},
 {t, 0, 2 Pi}, PlotLegends → "Expressions"]
```



例3：画扇形区域。

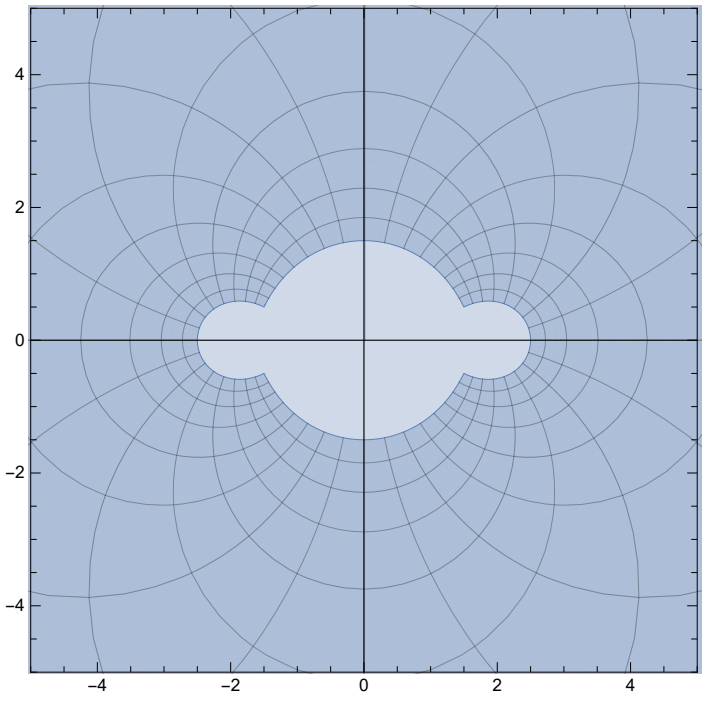
```
ParametricPlot[{v Cos[u], v Sin[u]}, {u, 0, Pi / 2}, {v, 1 / 2, 1}, Axes → False]
```




```
ParametricPlot[{v Cos[u], v Sin[u]}, {u, 0, Pi / 2}, {v, 0, 1}, Axes → False]
```

例4：画复函数图，请观察选项 Mesh → Automatic 的作用.

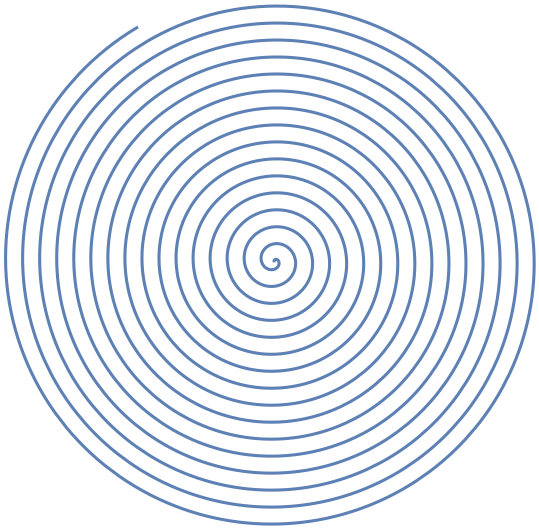
```
ParametricPlot[With[{z = u + I v}, {Re[z + 1 / z], Im[z + 1 / z]}],
 {u, -1 / 2, 1 / 2}, {v, -1 / 2, 1 / 2}, PlotRange -> 5, Mesh → Automatic]
```



```
ParametricPlot[With[{z = u + I v}, {Re[z + 1 / z], Im[z + 1 / z]}],
 {u, -1 / 2, 1 / 2}, {v, -1 / 2, 1 / 2}, PlotRange -> 5]
```

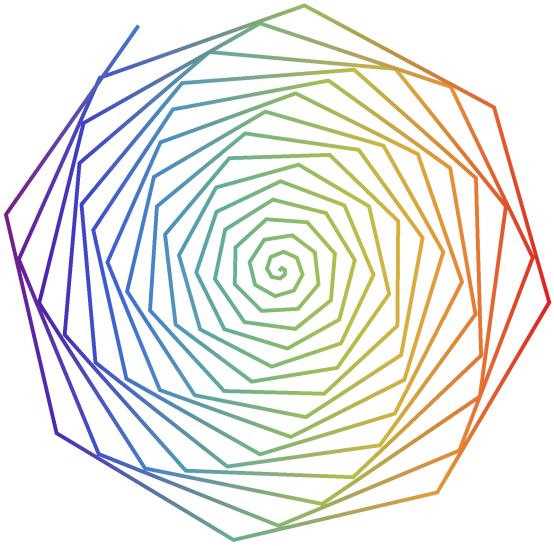
例5：请观察选项 **MaxRecursion → 0** 的作用.

```
ParametricPlot[{u Sin[u], u Cos[u]}, {u, 0, 100}, Axes → False]
```



选项 **MaxRecursion → 0** 设置系统不对曲线做光滑处理.

```
ParametricPlot[{u Sin[u] , u Cos[u]},
 {u, 0, 100}, PlotPoints → 125, Axes → False, MaxRecursion → 0,
 PlotStyle → Thick, ColorFunction → ColorData["Rainbow"]]
```



2. 重现和重组图形

`Show[pic]` 显示图形表达式pic

`Show[pic, 选项名 → 选项值]` 按选项显示图形表达式pic

`Show[pic1, pic2, ..., picn]` 将图pic1，pic2，...，picn 放在一幅图中显示

`Show[GraphicsGrid[{{p11, p12, ...}, {p21, p22, ...}, ...}]]`
按矩阵元素排列形式显示每个图形

Show 的选项主要定义察看图形的方式，例如，定义坐标轴的刻度，
曲线或曲面的显示范围.有些选项则不起作用，例如，设置曲线样式
选项 PlotStyle.

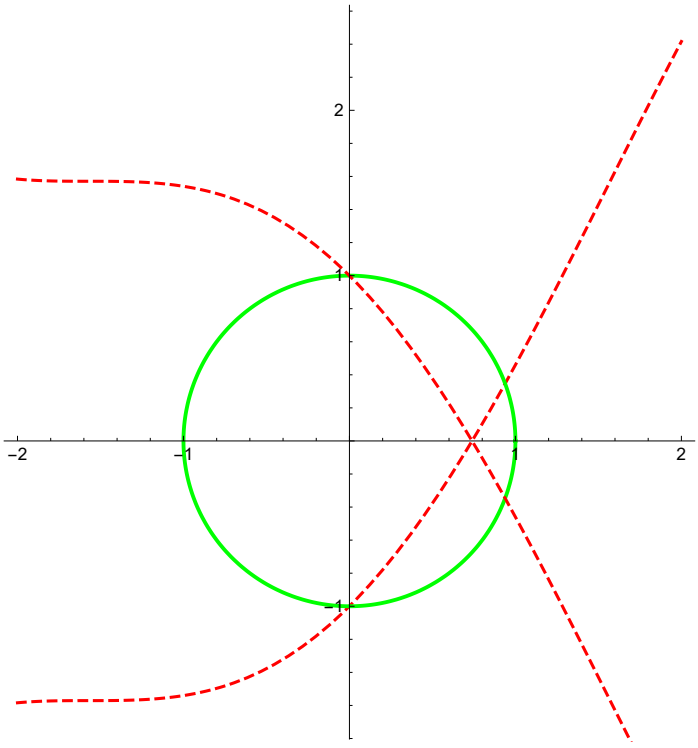
例6：PlotRange 在 Show 中.

In[1]:=
```
Plot[Sin[x^2] / x, {x, 0, 6}]
Show[%, PlotRange → {{3, 5}, {-5, 5.}}]
```

例7：用 Show 组合图形。

In[3]:=
```
pic1 = Plot[x - Cos[x], {x, -2, 2}, PlotStyle → {Red, Dashed}];
pic2 =
  ParametricPlot[{Cos[x], Sin[x]}, {x, 0, 2 Pi}, PlotStyle → {Green, Thick}];
pic3 = Plot[-x + Cos[x], {x, -2, 2}, PlotStyle → {Red, Dashed}];
Show[{pic1, pic2, pic3}, Framed → True,
 Grad → Automatic, AspectRatio → Automatic]
```

Out[6]=



例8：按3个图形一行显示图形数组

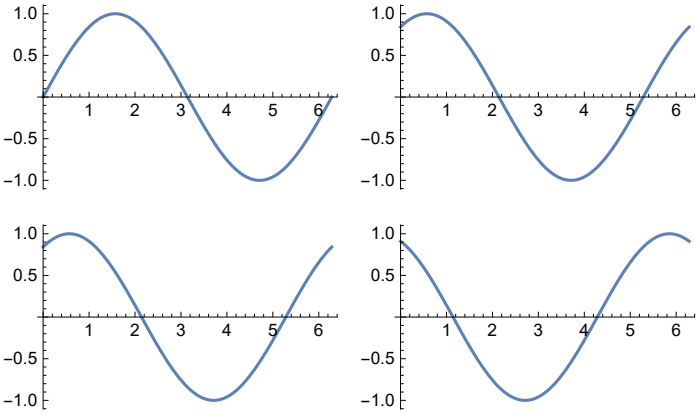
```
tt = Table[Plot[Sin[x + t], {x, 0, 2 Pi}], {t, 0, 8}];
```

```
Show[tt]
```

```
Show[GraphicsGrid[Partition[tt, 3]]]
```

两两比较图形

```
Show[GraphicsGrid[{{tt[[1]], tt[[2]]}, {tt[[2]], tt[[3]]}}]]
```



第6讲　在 Mathematica 中作图

6 - 3　二元函数作图　Plot3D

Plot3D 应用对象：

在直角坐标系中绘制二元函数 f (x, y) 在指定区间上的图形。

Plot3D 命令形式：

Plot3D[f[x, y], {x, x0, x1}, {y, y0, y1}, 选项]

在 x和y的区域上，按选项画出空间曲面实数值表达式f (x, y) 的三维图形

Plot3D[{f[x, y], g[x, y]}, {x, x0, x1}, {y, y0, y1}, 选项]

按选项值在同一坐标系中绘制函数f (x, y) 和g (x, y) 的三维图形

1. 关于绘图区间

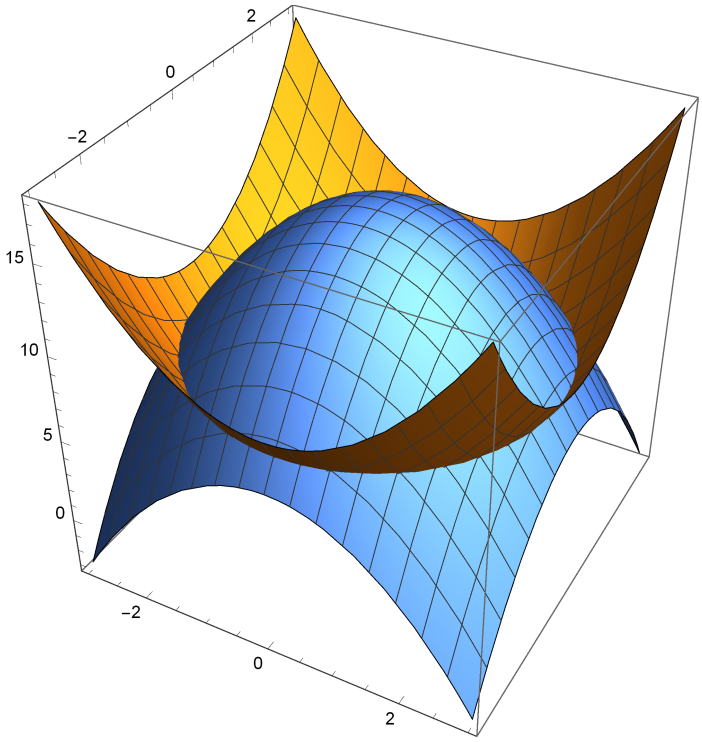
Plot3D[f[x, y], {x, x0, x1}, {y, y0, y1}]	变量 $\{x, y\}$ 在矩形区域中取值
Plot3D[f[x, y], $\{x, y\} \in reg$]	变量 $\{x, y\}$ 在几何区域 reg取值.

例1：绘图区域是矩形区域

```
Plot3D[Sin[x + y^2/3], {x, -Pi, Pi}, {y, -4.5, 4.5}]
```

```
Plot3D[{x^2 + y^2, 15 - x^2 - y^2}, {x, -3, 3}, {y, -3, 3}]
```

```
Plot3D[{x^2 + y^2, 15 - x^2 - y^2}, {x, -3, 3}, {y, -3, 3}, BoxRatios -> {1, 1, 1}]
```

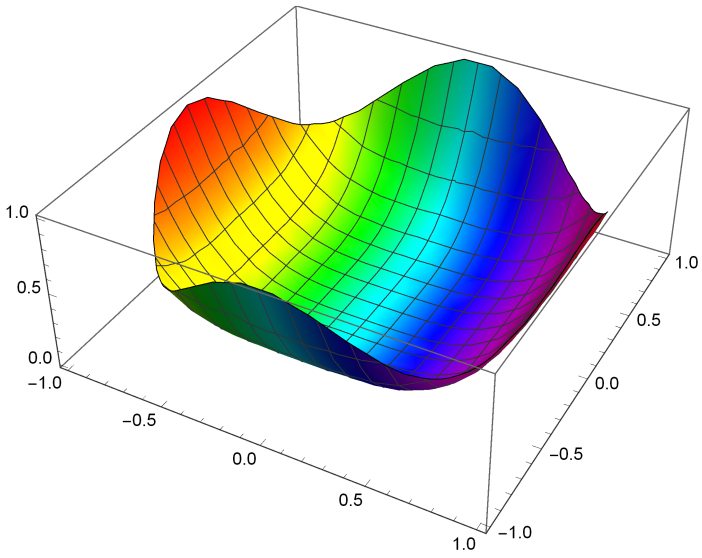


例2：绘图区域是单位圆.

```
Plot3D[2 x^2 + y, {x, y} ∈ Disk[]]
```

```
Plot3D[2 x^2 + y, {x, -3, 3}, {y, -5, 5}]
```

```
Plot3D[x^4 + y^4, {x, y} ∈ Disk[], ColorFunction → Function[{x, y}, Hue[x]]]
```



例3：在环形区域上绘制图形. `RegionFunction → Function[ ...]`

```
Plot3D[{x^2 + y^2, 15 - x^2 - y^2}, {x, -2.2, 2.2}, {y, -2.2, 2.2},
 RegionFunction → Function[{x, y}, 0.2 < x^2 + y^2 < 4.2], BoxRatios -> {1, 1, 1}]
```

```
Plot3D[Sin[x + y^2], {x, -2, 2}, {y, -2, 2},
 RegionFunction → Function[{x, y}, 1 < x^2 + y^2 < 5]]
```

```
Plot3D[Sin[x + y^2], {x, -2, 2}, {y, -2, 2},
 RegionFunction → Function[{x, y}, x^2 + y^2 < 5]]
```

例4：用 `Exclusions` 排除矩形的部分定义域

```
Plot3D[Tan[x y] + 1 / (y^2 - x^3 + 3 x - 3), {x, -2, 2}, {y, -2, 2}]
```

```
Plot3D[Tan[x y] + 1 / (y^2 - x^3 + 3 x - 3), {x, -2, 2},
 {y, -2, 2}, Exclusions → {Cos[x y] == 0, y^2 - x^3 + 3 x - 3 == 0}]
```

```
Plot3D[Tan[x y] + 1 / (y^2 - x^3 + 3 x - 3), {x, -2, 2}, {y, -2, 2},
 BoundaryStyle → Thick, RegionFunction → Function[{x, y, z}, x^2 + y^2 >= 1]]
```

2. 关于选项

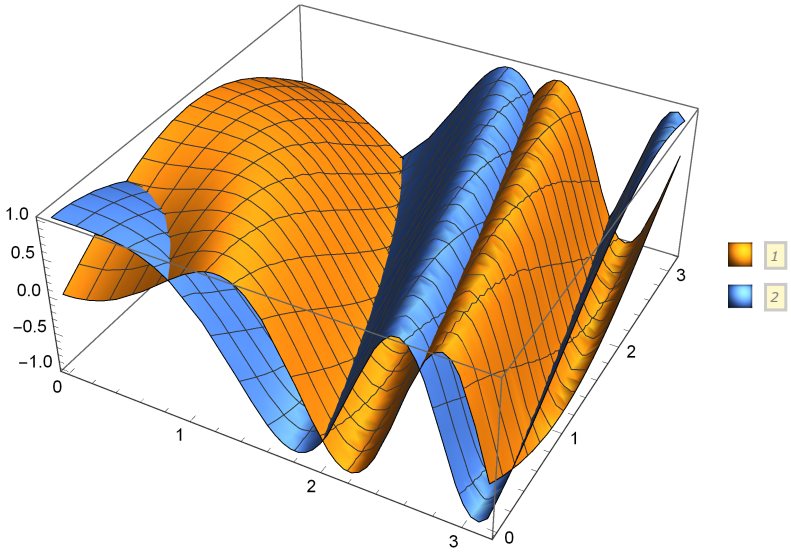
选项名	默认值	说明
Axes	True	是否绘制轴
BoundaryStyle	Automatic	如何绘制曲面的边界线
BoxRatios	{1, 1, 0.4}	有界3D box 比例
ClippingStyle	Automatic	如何绘制曲面的剪切部分
ColorFunction	Automatic	如何决定曲面的颜色
ColorFunctionScaling	True	是否用函数ColorFunction做
EvaluationMonitor	None	在每次函数计算时需要计算的表
Exclusions	Automatic	排除的 x、y 曲线
ExclusionsStyle	None	如何绘制排除曲线
Filling	None	每个曲面下的填充
FillingStyle	Opacity[0.5]	填充使用的样式
MaxRecursion	Automatic	递归子划分的最大数量
Mesh	Automatic	每个方向上绘制网格线的数量
MeshFunctions	{#1 &, #2 &}	如何决定网格线的放置位置
MeshShading	None	如何设置网格线之间的阴影区域
MeshStyle	Automatic	网格线的样式
Method	Automatic	细化曲面的方式
NormalsFunction	Automatic	如何决定有效的法向量
PerformanceGoal	$PerformanceGoal	优化执行的性能
PlotLegends	None	曲面的图例
PlotPoints	Automatic	每个方向上样本点的最初数量
PlotRange	{Full, Full, Automatic}	包括 z 范围或其它值
PlotStyle	Automatic	每个曲面样式的图形指令
PlotTheme	$PlotTheme	overall theme for the p
RegionFunction	(True &)	如何确定是否包含一个点
TextureCoordinateFunction	Automatic	如何确定纹理坐标
TextureCoordinateScaling	True	是否将参数缩放至
WorkingPrecision	MachinePrecision	内部计算使用的精度

例5：关于选项 BoxRatios

```
{Plot3D[x^2 - y^2, {x, -1, 1}, {y, -1, 1}],
 Plot3D[x^2 - y^2, {x, -1, 1}, {y, -1, 1}, BoxRatios -> {1, 1, 1}]}
```

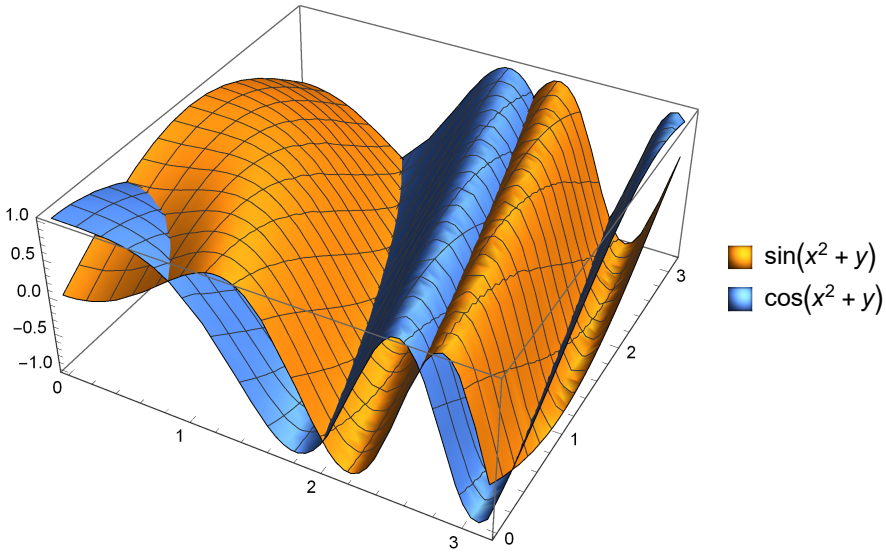
例6：关于图列选项 PlotLegends.

```
Plot3D[{Sin[x^2 + y], Cos[x^2 + y]}, {x, 0, Pi}, {y, 0, Pi}, PlotLegends → Automatic]
```




```
Plot3D[{Sin[x^2 + y], Cos[x^2 + y]},
 {x, 0, Pi}, {y, 0, Pi}, PlotLegends → {"one", "two"}]
```

```
Plot3D[{Sin[x^2 + y], Cos[x^2 + y]},
 {x, 0, Pi}, {y, 0, Pi}, PlotLegends → "Expressions"]
```



例7：加背景色、去网格线、去边界框.

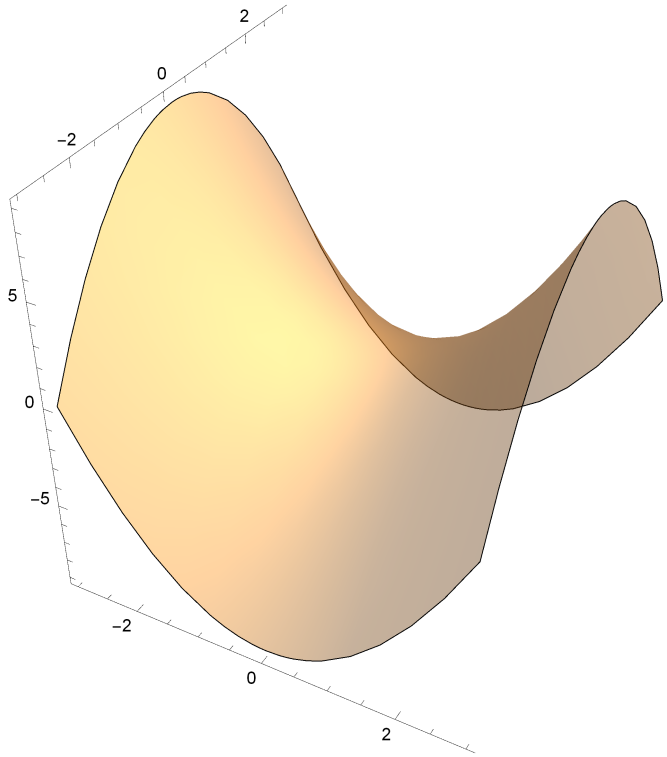
```
Plot3D[Sin[x + y^2], {x, -3, 3}, {y, -2, 2}, Background → Blue,
 Mesh → None, BoxRatios -> {1, 1, 1}, Boxed → False]
```

例8：设置区域边界样式.

```
Plot3D[Sin[x + y^2], {x, -3, 3}, {y, -2, 2},
 RegionFunction → Function[{x, y, z}, x^2 + y^2 ≥ 1],
 BoundaryStyle → Directive[Red, Thick]](*,Boxed→False,Axes→None]*)
```

例9：让曲面透明来查看内部结构.

```
Plot3D[ x^2 - y^2, {x, -3, 3}, {y, -3, 3}, PlotStyle → Opacity[0.4],
 Mesh → None, BoxRatios -> {1, 1, 1}, Boxed → False]
```



第6讲　在 Mathematica 中作图

6－4　三维参数、极坐标、球坐标作图

1．三维参数函数作图

ParametricPlot3D命令形式：

ParametricPlot3D[{**x** (**t**), **y** (**t**), **z** (**t**)}, {**t**, **t0**, **t1**}, 选项]
在三维空间中按选项绘制空间参数曲线 {**x** (**t**), **y** (**t**), **z** (**t**)}

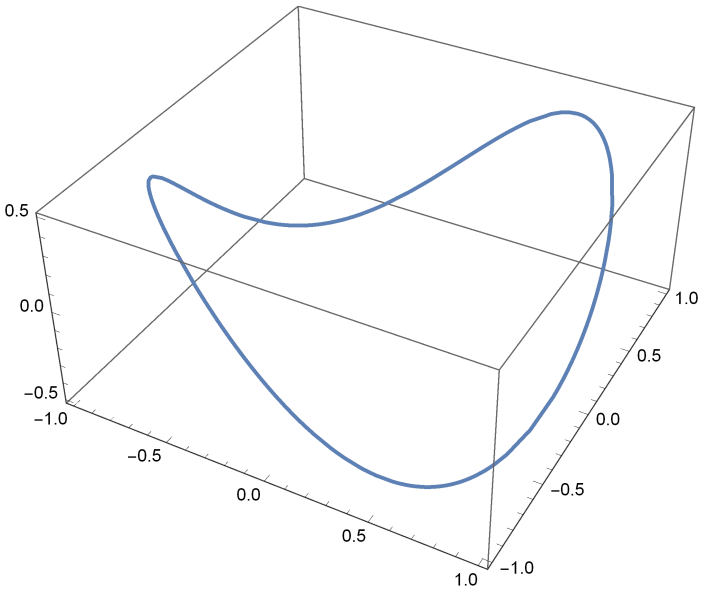
ParametricPlot3D[{**x**, **y**, **z**}, {**u**, **u0**, **u1**}, {**v**, **v0**, **v1**}, 选项]
绘制参数u和v的三维空间曲面**x** = **x** (**u**, **v**), **y** = **y** (**u**, **v**), **z** = **z** (**u**, **v**)

ParametricPlot3D[{{**xa**, **ya**, **za**}, {**xb**, **yb**, **zb**}}, {**u**, **u0**, **u1**}, {**v**, **v0**, **v1**}, 选项]
在同一坐标系中绘制两张曲面

ParametricPlot3D[{**x**, **y**, **z**}, {u, v} ∈ *reg*]　从几何区域**reg**取值画图

例1：画空间曲线．

```
ParametricPlot3D[{Cos[u], Sin[u], Cos[u] Sin[u]}, {u, 0, 2 Pi}]
```



例2：画二次锥面．

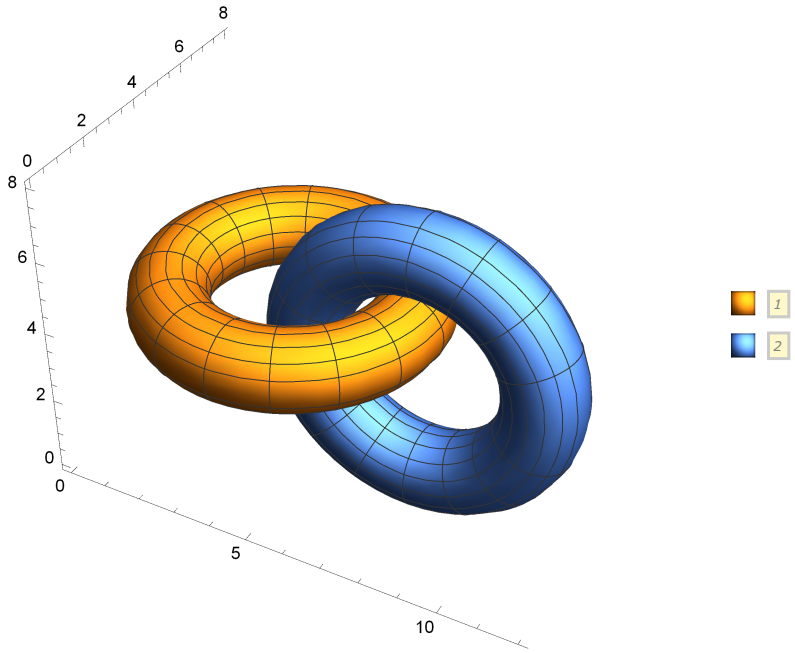
```
ParametricPlot3D[{r * Cos[a], r * Sin[a], r}, {a, 0, 2 Pi}, {r, -4, 4}, Boxed -> False]
```

例3：给三维参数图设置玫瑰色．

```
ParametricPlot3D[
 {2 * (Cos[o + p] + p * Sin[o + p]), 2 * (Sin[o + p] - p * Cos[o + p]), o / (2 * Pi)},
 {o, 0, 4 * Pi}, {p, 0, 4 * Pi}, PlotPoints → 40,
 BoxRatios -> {1, 1, 1.5}, Mesh -> None, ColorFunction → "RoseColors"]
```

例4：两个交叉的圆环.

```
{fx, fy, fz} = {4 + (3 + Cos[v]) Sin[u], 4 + (3 + Cos[v]) Cos[u], 4 + Sin[v]};
{gx, gy, gz} = {8 + (3 + Cos[v]) Cos[u], 3 + Sin[v], 4 + (3 + Cos[v]) Sin[u]};
ParametricPlot3D[{{fx, fy, fz}, {gx, gy, gz}}, {u, 0, 2 Pi}, {v, 0, 2 Pi},
 PlotLegends → Automatic, Boxed -> False]
```



例5：请观察参数曲线定义域的影响力.

```
{ParametricPlot3D[
   {Cos[u] Sin[v], Sin[u] Sin[v], Cos[v]}, {v, 0, Pi}, {u, 0, 2 Pi}],
ParametricPlot3D[{Cos[u] Sin[v], Sin[u] Sin[v], Cos[v]}, {v, 0, Pi}, {u, 0, 16 Pi}]}
```

2. 极坐标作图

点M的极坐标由半径r和幅角θ确定，直角坐标M (x, y) 和极坐标的转换关系：

(x, y) = (r cos θ , r sinθ)。

PolarPlot 形式：

PolarPlot[r, {θ-, θa, θb}]

在幅角θ定义域上绘制r = r (θ) 的曲线

例6：画出阿基米德螺线r = θ

```
{PolarPlot[θ, {θ, 0, 5 Pi}],
 PolarPlot[θ, {θ, -5 Pi, 5 Pi}], PolarPlot[θ, {θ, 0, -5 Pi}]}
```

例7：看看 $\sqrt{u}$ 在极坐标和直角坐标中的表现.

```
{PolarPlot[Sqrt[u], {u, 0, 3 Pi}, PlotStyle → {Orange, Thick}],
 Plot[Sqrt[u], {u, 0, 3 Pi}, PlotStyle → {Green, Thick}]}
```

例8：画一组极坐标曲线.

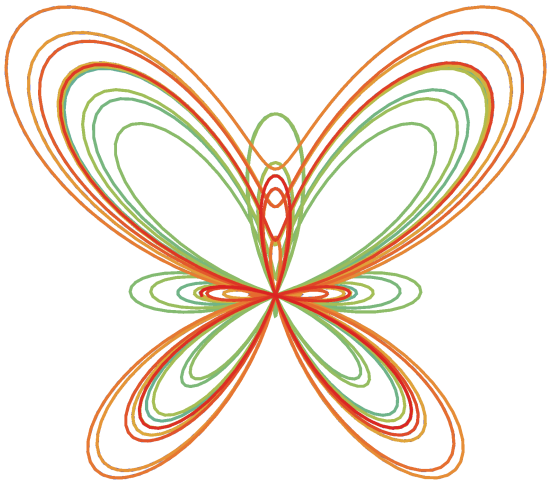
```
r = {1, 1 + 1 / 12 Sin[12 t], 1 / 2, 1 / 2 + 1 / 24 Sin[12 t]};
PolarPlot[r, {t, 0, 2 Pi}, PlotStyle → {Green, Dashed}]

PolarPlot[{Sin[5 t], Sin[4 t]}, {t, 0, 2 Pi}, PlotStyle → {Red, Purple}]
```

例9：看看函数r 在直角坐标和极坐标中的表现？

In[1]:=
```
r = Exp[Cos[t - Pi / 2]] - 2 * Cos[4 * (t - Pi / 2)] + Sin[(t - Pi / 2) / 12]^5;
Plot[r, {t, 0, 36 Pi}]
```

In[3]:=
```
PolarPlot[r, {t, 0, 36 Pi}, ColorFunction → "Rainbow", Axes -> None]
```

Out[3]=



3. 球坐标作图

三维空间点M的球坐标由半径r经度θ纬度φ唯一确定，其中θ 的范围从0到π，φ的范围从0 到2 π。直角坐标M (x, y, z) 和球坐标的转换关系：

(x, y, z) = (r sin θ cosφ, r sinθ sinφ, r cosθ)

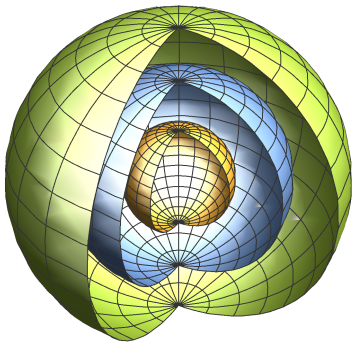
```
SphericalPlot3D[r, {θ, θa, θb}, {φ, φa, φb}]

SphericalPlot3D[{r1, r2, ...}, {θ, θa, θb}, {φ, φa, φb}]
```

例10：画单位球.

```
SphericalPlot3D[1, {θ, 0, Pi}, {φ, 0, 2 Pi}, Boxed -> False]
```

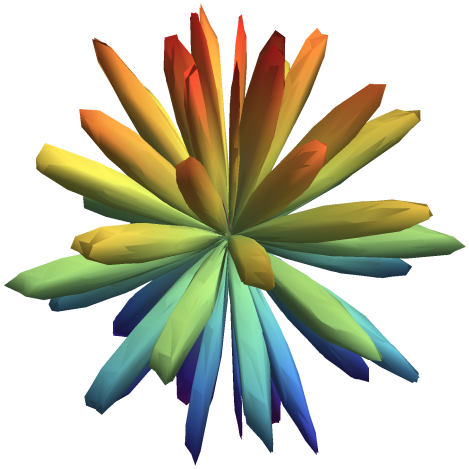
例11：大半球套小半球.

```
SphericalPlot3D[{1, 2, 3}, {θ, 0, Pi}, {φ, 0, 3 Pi / 2},
 Boxed -> False, Axes -> False, ColorFunction → Hue[x], Axes → None]
```



例11：一束花.

```
SphericalPlot3D[Sin[7 u] Cos[7 v], {u, 0.2, Pi}, {v, 0.2, 2 Pi},
 Boxed -> False, Mesh -> None, Axes -> False, ColorFunction → "Rainbow"]
```



第6讲　在 Mathematica 中作图

6－5　数据绘图

6－5－1　二维数据绘图

二维数据data表示：

{{x1, y1}, {x2, y2}, ...}	数据表 {{xi, yi}, i = 1, ..., n}
{y1, y2, ...}	数据表　{{1, y1}, {2, y2}, ...}

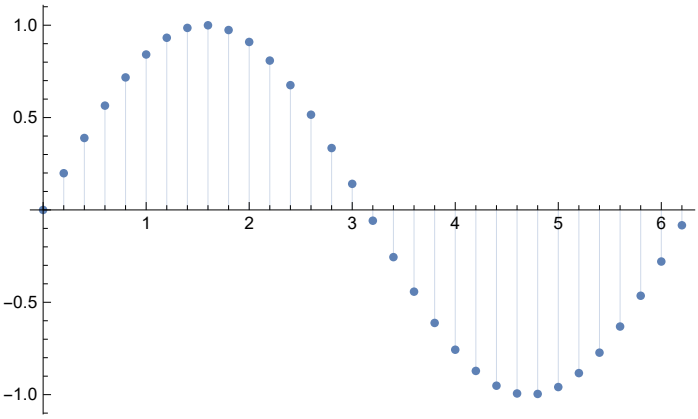
二维数据绘图命令：

ListPlot[data, 选项]	按选项用data绘制数据点集
ListPlot [data, Joined → True]	画一条通过数据点的光滑曲线
ListLinePlot[data, 选项]	按选项用数据data绘制曲线
ListPolarPlot[data, 选项]	在极坐标系下画离散点集data

例1：数据点列表.

In[14]:=
```
data = Table[{x, Sin[x]}, {x, 0, 2 Pi, 0.2}];
ListPlot[data, Filling → Axis]
```

Out[15]=



In[16]:=
```
ListPlot[data, Joined -> True, Filling → Axis]
```

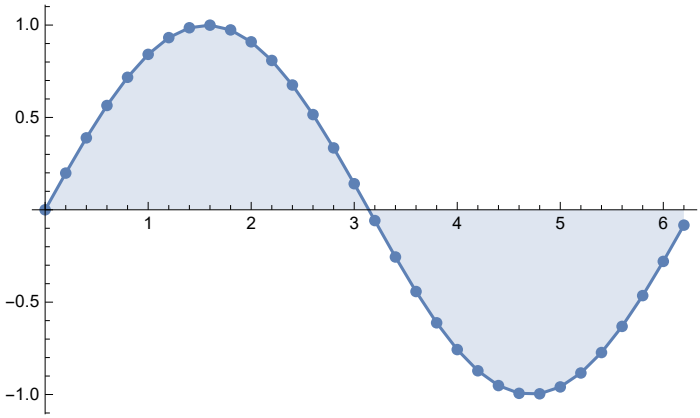
例2：数据曲线图.

```
ListLinePlot[data, Filling → Axis]
```

例3：选项：**Mesh** 曲面的网格线，曲线的取值点.

In[18]:=
```
ListPlot[data, Joined -> True, Filling → Axis, Mesh → Full]
```

Out[18]=



In[19]:=
```
ListLinePlot[data, Filling → Axis, Mesh → Full]
```

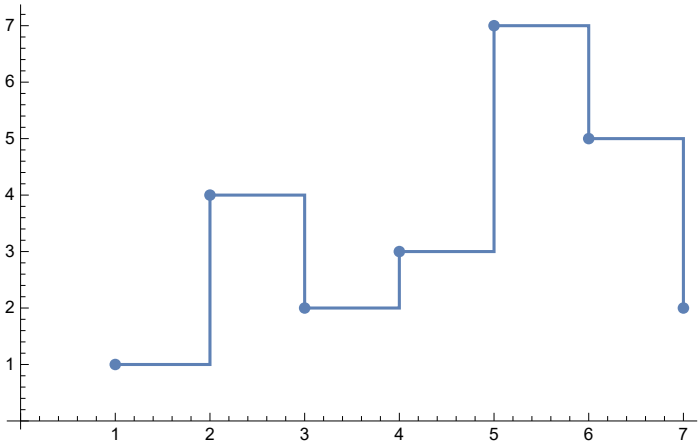
例4：将点连成线并标出点的位置.

In[20]:=
```
ListLinePlot[{{1, 4, 2, 3, 7, 5, 2}}, Mesh -> Full,
 MeshStyle -> Directive[PointSize[Large], Purple]]
```

例5：用分段常量 (零次插值) 连接离散点.

In[24]:=
```
ListLinePlot[{1, 4, 2, 3, 7, 5, 2}, InterpolationOrder → 0, Mesh → Full]
```

Out[24]=



例6：设置选项 **InterpolationOrder → 3**，用三次插值多项式近似离散点序列.

```
ListLinePlot[{1, 4, 2, 3, 7, 5, 2}, InterpolationOrder -> 3,
 Mesh -> Full, MeshStyle -> Directive[PointSize[0.02], Red]]
```

例7：画出函数**f (x) = (x - 1) (x - 1.5) (x - 2.7)** 在 **{0.75, 2.8}** 的极值点.

```
f[x_] = (x - 1) (x - 1.5) (x - 2.7); t = Plot[f[x], {x, 0.75, 2.8}]
```

```
sol = Solve[D[f[x], x] == 0, x]
```

```
u = x /. %
```

```
u = {sol[[1, 1, 2]], sol[[2, 1, 2]]}
```

```
data = Table[{u[[k]], f[u[[k]]]}, {k, 1, Length[u]}]
```

```
Show[t, ListPlot[data, PlotStyle → {Red, PointSize[Large]}]]
```

例8：画极坐标离散点列.

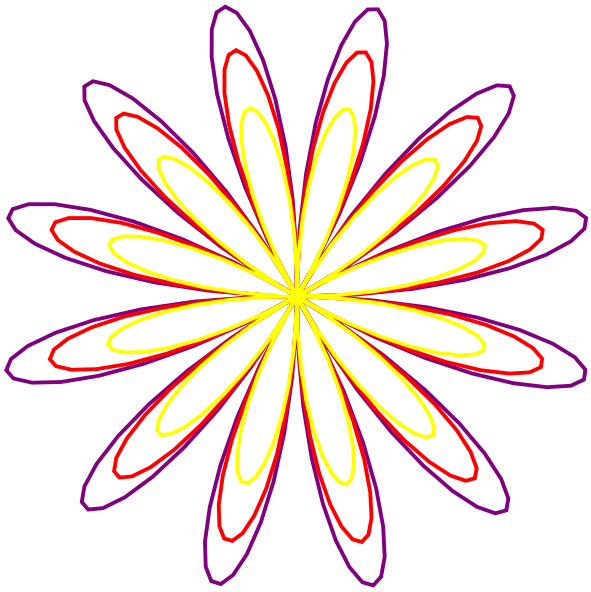
In[21]:=
```
ListPolarPlot[Table[θ, {θ, 0, 2 Pi, 0.1}]]
```

例9：画离散点列的极坐标图。

In[22]:=
```
t = Range[0, 12 Pi, 0.2];
```

In[23]:=
```
ListPolarPlot[{Sin[t], 0.85 Sin[t], 0.65 Sin[t]}, Joined -> True, Axes -> False,
 PlotStyle → {{Purple, Thick}, {Red, Thick}, {Yellow, Thick}}]
```

Out[23]=



6－5－2 三维数据绘图

二维数据data表示：{x, y}

{{x1, y1}, {x2, y2}, ...}　　数据表 {{xi, yi}, i = 1, ..., k}

{y1, y2, ...}　　数据表　{{1, y1}, {2, y2}, ...}

三维空间数据点表示： {x, y, z}

{{x1, y1, z1}, {x2, y2, z2} ...}　　向量点列

{{z11, ..., z1n}, ... {zm1, ..., zmn}}　　矩阵点列

{z11, ..., z1n} 为 {{1, 1, z11}, ..., {1, n, z1n}}

三维数据绘图命令 ：

`ListPointPlot3D[{{x1, y1, z1}, {x2, y2, z2}, ...}]`	点集的三维散点图
`ListPlot3D[data]`	在 $\{x_i, y_i\}$ 处的高度为z_i 的三维曲面图 (点集三维图)
`ListPlot3D[data, Mesh → Full]`	绘制穿过所有每个数据点位置的网格

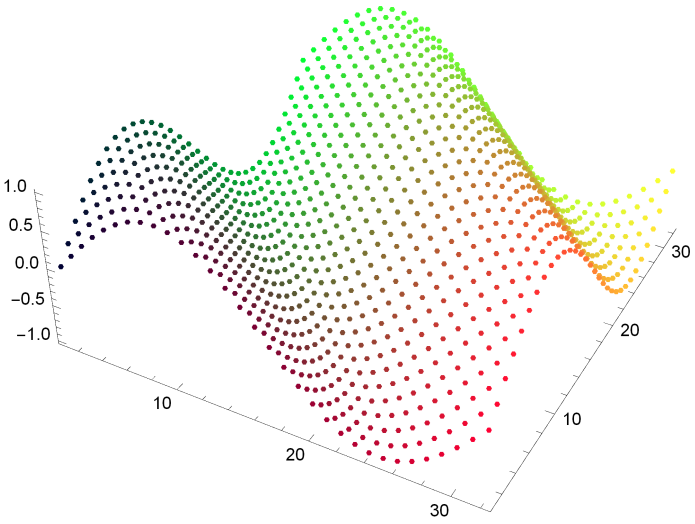
例1： u的点列定义：`Sin[i + j]  {i, 0, 2 Pi, 0.2}, {j, 0, 2 Pi, 0.2}  {i, j, Sin[i + j]}`
绘图时的点列：`{I, J, Sin[i + j]} , {I, 1, m}, {J = 1, m}  m = Length[u]`

In[28]:=
```
u = Table[Sin[i + j], {i, 0, 2 Pi, 0.2}, {j, 0, 2 Pi, 0.2}];
```

In[29]:=
```
ListPointPlot3D[u, Boxed → False]
```

In[30]:=
```
ListPointPlot3D[u,
 ColorFunction → Function[{i, j}, RGBColor[i, j, 0.2]], Boxed → False]
```

Out[30]=



例2：观察和比较点列作图和函数作图.

In[33]:=
```
ListPlot3D[Table[Sin[x + y^2], {x, 0, Pi, 0.1}, {y, 0, Pi, 0.1}],
 ColorFunction → "DarkRainbow"]
```

In[34]:=
```
Plot3D[Sin[x + y^2], {x, 0, Pi}, {y, 0, Pi}, ColorFunction → "DarkRainbow"]
```

In[35]:=
```
ListPlot3D[Table[Sin[i + j^2], {i, 0, Pi, 0.1}, {j, 0, Pi, 0.1}], Filling -> Bottom]
```

例3：观察数据取 225, 625 和 2500 点的图形效果.

In[36]:=
```
fdata[n_] := Table[Sin[0.01 (i + j)] + Cos[0.01 (i * j)], {i, 1, n}, {j, 1, n}];
ListPlot3D[fdata[15], Axes → False, Boxed → False]
```

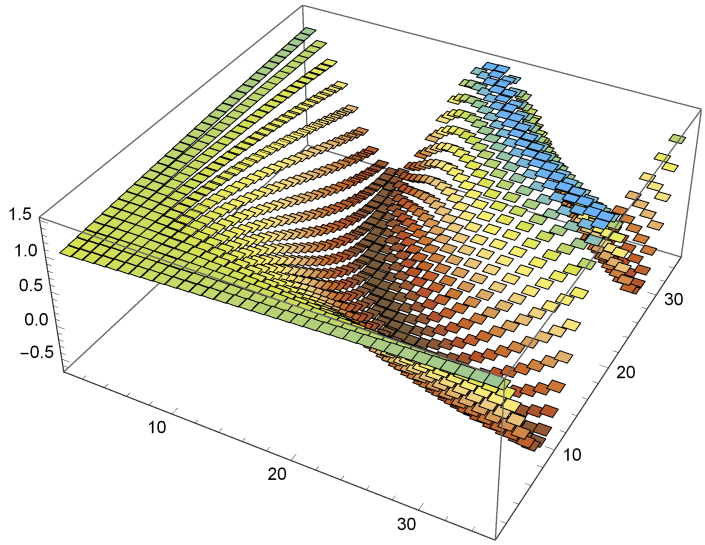
In[38]:=
```
ListPlot3D[fdata[35], Axes → False, Boxed → False]
```

```
ListPlot3D[fdata[50], Axes → False, Boxed → False]
```

例4：观察选项 Mesh → None , Mesh → 8 和 Mesh → All .

In[40]:=
```
ListPlot3D[fdata[35], Mesh → None,
 InterpolationOrder → 0, ColorFunction → "SouthwestColors"]
```

Out[40]=



In[41]:=
```
ListPlot3D[fdata[35], Axes → False, Boxed → False]
```

In[44]:=
```
ListPlot3D[fdata[35], Mesh → All,
 Boxed → False, ColorFunction → "SouthwestColors"]
```

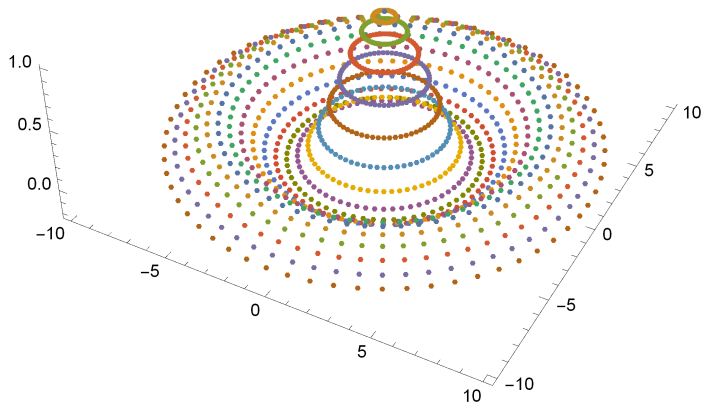
In[43]:=
```
ListPlot3D[fdata[35], Mesh → 8, Boxed → False]
```

例5：画一顶“草帽”.

In[25]:=
```
data = Table[{r Cos[t], r Sin[t], Sinc[r]}, {r, 0, 10, 0.5}, {t, 0, 2 Pi, 0.1}];
```

In[27]:=
```
ListPointPlot3D[data, Boxed → False]
```

Out[27]=



In[26]:=
```
{Plot[Sin[x] / x, {x, -10, 10}], Plot[Sinc[x], {x, -10, 10}]}
```

第6讲　在 Mathematica 中作图

6－6　图元绘图

1．二维 图元绘图

`Graphics[{primitives, 选项}]`　按选项画二维图元素`primitives`

`Graphics[{pri1, 选项1, pri2, 选项2}]`

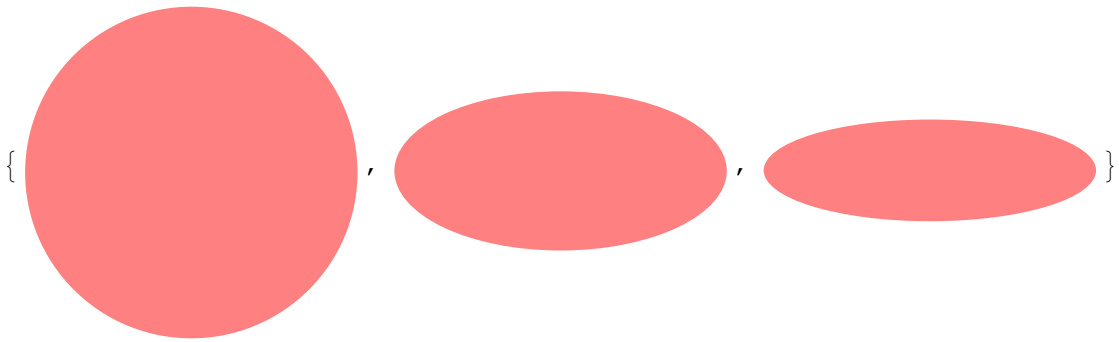
二维图形元素	说 明
`Arrow[{{x1, y1}, ...}]`	箭头
`Circle[{x, y}, {ra, rb}, {t1, t2}]`	圆心在 `{x, y}`，从弧度 `t1`到弧度 `t2`的椭圆弧
`Disk[{x, y}, {ra, rb}, {t1, t2}]`	圆心在 `{x, y}`，从弧度 `t1`到弧度 `t2`的填实椭圆 默认值： `{x, y} = {0, 0}, {t1, t2} = {0, 2 Pi}`
`Line[{{x1, y1}, ...}]`	依次连接相邻两点的线段
`Point[{x, y}]`	点的位置 `{x, y}`
`Polygon[{{x1, y1}, ...}]`	多边形
`Rectangle[{xmin, ymin}, {xmax, ymax}]`	以 `{xmin, ymin}, {xmax, ymax}` 为顶角的矩形
`Inset[obj, ...]`	插入元素
`Text[expr, {x, y}]`	`{x, y}` 坐标处插入表达式`expr`

例1：`Circle[{x, y}, r]` 圆心在 `{x, y}`，半径为`r`的圆弧线

单位圆 `Disk[1]` 或 `Disk[ ]`

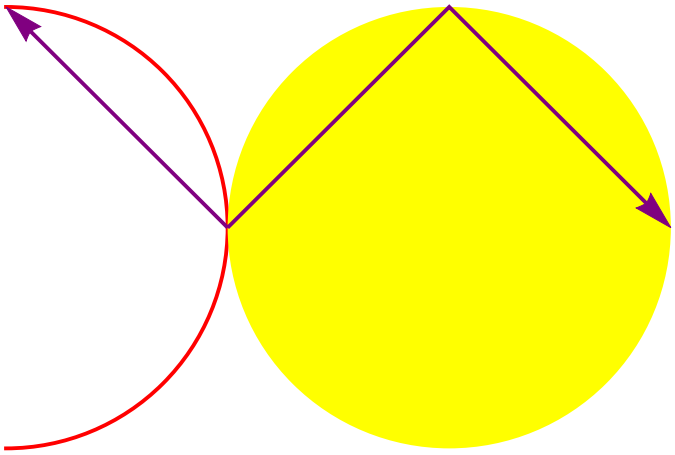
例2：选项 `AspectRatio` 的作用．

```
Table[Graphics[{Pink, Disk[]}, AspectRatio → 1/k], {k, 1, 3}]
```



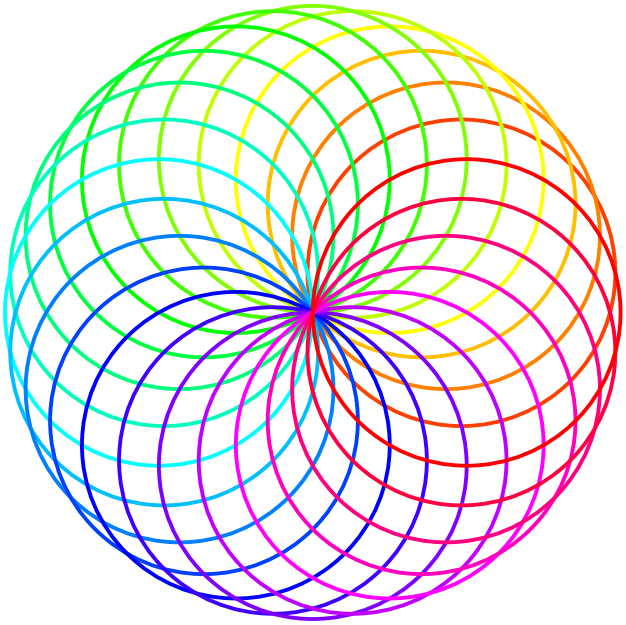
例3：可以转弯的箭头.

```
Graphics[{Thick, Red, Circle[{-1, 0}, 1, {-Pi/2, Pi/2}],
  Yellow, Disk[{1, 0}], Purple, Arrowheads[Large],
  Arrow[{{0, 0}, {-1, 1}}], Arrow[{{0, 0}, {1, 1}, {2, 0}}]}]
```



例4：多环图.

```
Graphics[
 {Thick, Table[{Hue[t/24], Circle[{Cos[2 Pi t/24], Sin[2 Pi t/24]}]}, {t, 24}]}]
```



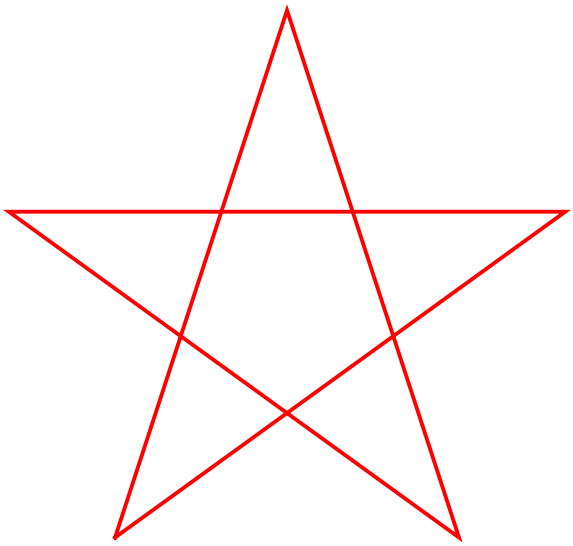
例5：画五边形.

```
data = Table[{Cos[t], Sin[t]}, {t, Pi/2, 2 Pi + Pi/2, 2 Pi / 5}];
Graphics[{Thick, Line[data]}]
```

例6：画五角星.

```
newdata = {data[[3]], data[[1]], data[[4]], data[[2]], data[[5]], data[[3]]};
```

```
tu = Graphics[{Red, Thick, Line[newdata]}]
```



例7：五角星插入到单位圆，正方形中。

```
{Graphics[{LightRed, Disk[{0, 0}, 1.1], Inset[tu]}],
 Graphics[{LightBlue, Rectangle[], Inset[tu]}]}
```

3. 三维 图元绘图

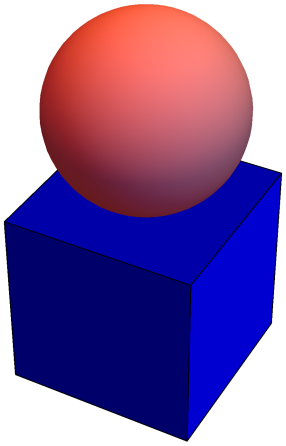
Graphics3D[{选项, 图元素}]

Graphics3D[{选项1, 图元素1, 选项2, 图元素2}]

三维图形元素（primitives）	说 明
Point[{x, y, z}]	点
Line[{{x1, y1, z1}, ...}]	折线段
Arrow[{pt1, pt2, pt3}, ...]	箭头
Polygon[{p_1**, …,** p_n**}]**	顶点为p_i 的填充多边形多边形
Text[expr, {x, y, z}]	在位置 **{x, y, z}** 处插入**expr**
Sphere[p, r]	球心为**p**半径为r的球体
Cuboid[p_{min}**,** p_{max}**]**	左下角为p_{min}，右上角为p_{max} 立方体
Cylinder[{{x_1**,** y_1**,** z_1**}, {**x_2**,** y_2**,** z_2**}}, r]**	表示半径为r围绕从 (x_1, y_1, z_1) 到 (x_2, y_2, z_2) 的线段的圆柱体
Cone[{{x_1**,** y_1**,** z_1**}, {**x_2**,** y_2**,** z_2**}}, r]**	底部半径r，中心为 (x_1, y_1, z_1)，顶点为 (x_2, y_2, z_2) 的圆锥体
Ball[p, r]	表示中心在点p、半径为r的球
Tube[{pt_1**,** pt_2**, …}, r]**	连接一系列点的半径为 r 线状三维管体

例1：画单位球和单位立方体.

```
Graphics3D[
 {Pink, Sphere[{0, 0, 2}], Blue, Cuboid[{-1, -1, -1}, {1, 1, 1}]}, Boxed → False]
```



例2：画多边形和一个点.

```
Graphics3D[{LightRed, Polygon[{{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}],
  Red, PointSize[0.5], Point[{1/2, 1/2, 1/2}]}]
```

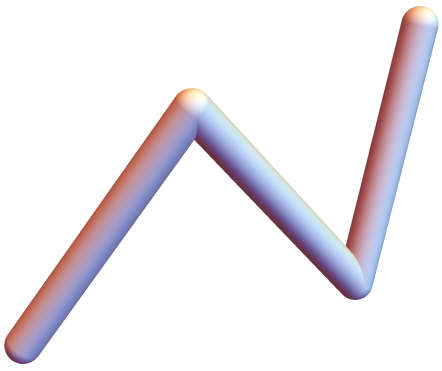
例3：将 $x^2 + y^2 + z^2 \le 1$ 放在单位球中.

```
Graphics3D[{Sphere[], Text[x^2 + y^2 + z^2 ≤ 1, {0, 0, 0}]}, Boxed → False]
```

例4：生成半径为 0.15 的管道

Tube[{pt_1, pt_2, ...}, r]　半径为 r 线段三维管体，线段由点列生成

```
Graphics3D[Tube[{{1, 1, -1}, {2, 2, 1}, {3, 3, -1}, {3, 4, 1}}, 0.15], Boxed → False]
```



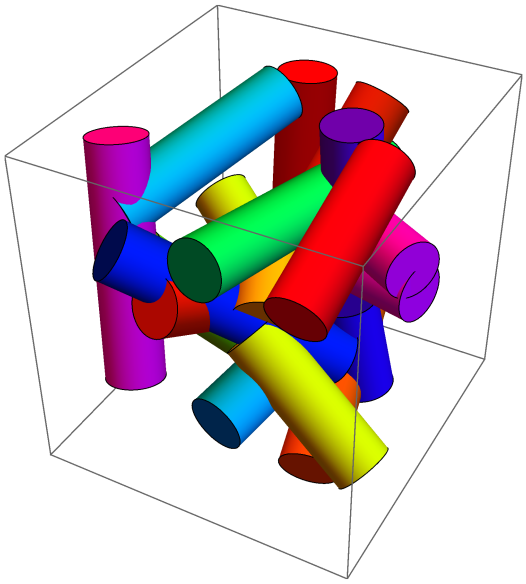
```
Graphics3D[Tube[BSplineCurve[{{1, 1, -1}, {2, 2, 1}, {3, 3, -1}, {3, 4, 1}}, 0.15]],
 Boxed → False]
```

例5：随机生成18个圆柱

圆柱图元 **Cylinder[{{x_1, y_1, z_1}, {x_2, y_2, z_2}}, r]**

生成半径为r围绕从 (x_1, y_1, z_1) 到 (x_2, y_2, z_2) 的线段的圆柱体

```
Graphics3D[Table[{Hue[RandomReal[]], Cylinder[RandomReal[10, {2, 3}]]}, {18}]]
```



第6讲 在 Mathematica 中作图

6－7 图形动画

动画演示提供了交互运行函数和命令的方式，让函数展开、积分运算等数学运算生动起来；

动画演示函数图形简单明了而栩栩如生.

1. Manipulate 交互式演示

Manipulate[expr, {u, ua, ub}] 动画演示表达式**expr**在区间 **{ua, ub}** 的所有值

Manipulate[expr, {u, ua, ub, dstep}] 控制量**u**在区间 **{ua, ub}** 之间以步长**dstep**变化

Manipulate[expr, {u, {u1, u2, ...}}] **u**取离散值**u1, u2, ...**

Manipulate[expr, {u, ...}, {v, ...},] 设置两个或多个控制量

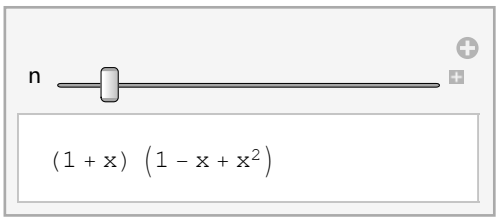
控制量**u**（或称滑杆），**expr**可为函数、图形等多种形式的表达式.

例1：动画演示中的循环变量的步长1常常不能省略.

```
Manipulate[n, {n, 1, 20}]
Manipulate[n, {n, 1, 20, 1}]
```

In[7]:=
```
Manipulate[Factor[1 + x^n], {n, 2, 12, 1}]
```

Out[7]=



例2：选项 **PlotRange** 在动画演示中的作用.

```
Manipulate[Plot[x Sin[a x] / 4, {x, 0, 9}], {a, 0, 2}]
Manipulate[Plot[x Sin[a x] / 4, {x, 0, 9}, PlotRange -> {-3, 3}], {a, 0, 2}]
```

例3：动画演示随初始条件变化的微分方程解函数.

```
Manipulate[
 Plot[Evaluate[y[t] /. First[NDSolve[ {y''[x] == -x y[x], y[0] == a, y'[0] == b},
       y, {x, 0, 4}]]], {t, 0, 4}, PlotRange → 4],
 {{a, 1, TraditionalForm[y[0]]}, -3, 3},
 {{b, 0, TraditionalForm[y'[0]]}, -3, 3}]
```

2. Animate 动画演示

`Animate[expr, {u, umin, umax}]`	在 {umin, umax} 区域内动态演示表达式expr
`Animate[expr, {u, umin, umax, du}]`	按步长du演示表达式expr
`Animate[expr, {u, {u1, u2, ...}}]`	u 取离散值u1, u2
`Animate[expr, {u, ...}, {v, ...}, ...]`	取每一对 {u, v} 的值演示表达式expr

例4：动态演示求导.

```
Animate[D[x^n, x], {n, 2, 10, 1}]
```

```
Animate[D[x^n, x], {n, 2, 10}]
```

例5：双重动画变量.

```
Animate[Plot[Sin[a x] + Sin[b x], {x, 0, 10}, PlotRange → 2], {a, 1, 5}, {b, 1, 5}]
```

例6：演示点绕圆周.

```
Animate[
 Graphics[{Blue, Thick, Circle[], Red, PointSize[0.03], Point[{Cos[t], Sin[t]}]}],
 {t, 0, 2 Pi, Pi/24}]
```

例7：演示一列水波.

```
Animate[Plot3D[Sin[Sqrt[x^2 + y^2] + t * 2 * Pi],
  {x, -8 Pi, 8 Pi}, {y, -8 Pi, 8 Pi}, PlotRange → 10, PlotPoints → 50,
  AspectRatio → 1, Boxed → False], {t, 2, 0}, AnimationRunning → False]
```

例8：直纹面形成单叶双曲面.

```
t = {-1.5, 1.5};
Animate[Graphics3D[Table[Line[{{Cos[t] - Sin[t], Cos[t] + Sin[t], 1},
      {Cos[t] + Sin[t], Sin[t] - Cos[t], -1}}], {t, 0, s, Pi/27}],
  Boxed -> False, PlotRange -> {t, t, t}], {s, 0, 2 Pi, Pi/27}]
```

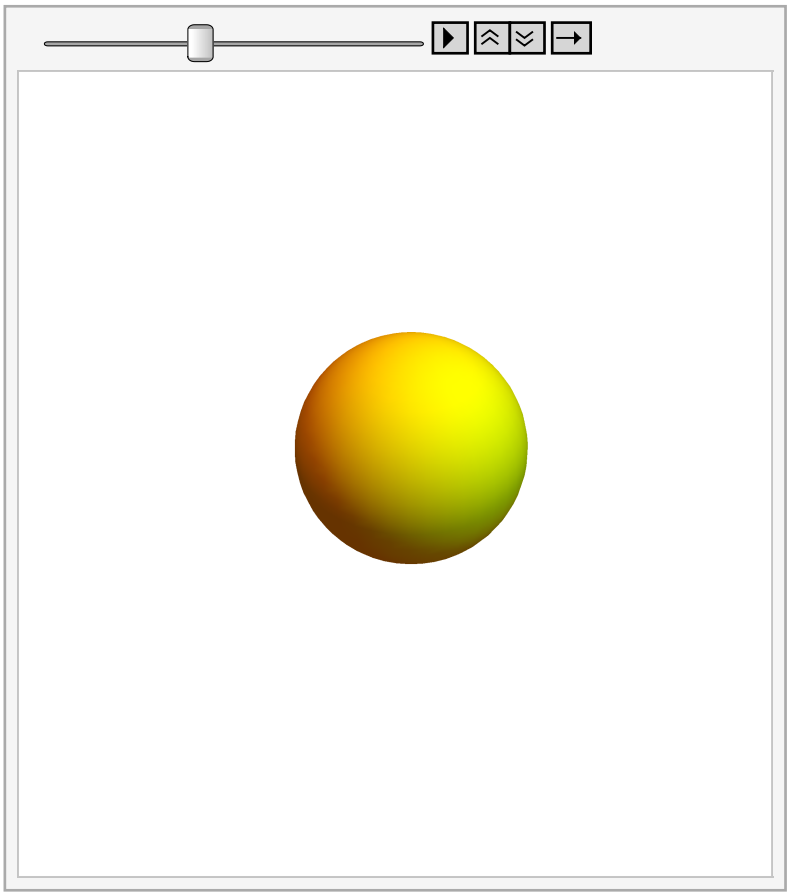
3. ListAnimate

`ListAnimate[{expr1, expr2}]` 依次运行图形表达式序列 `{expr1, expr2}`, 产生动画效果

例9：变换的球.

In[12]:=
```
A = {Red, Blue, Yellow, Green, Orange, Gray};
ListAnimate[Table[Graphics3D[
    {A[[n]], Sphere[{0, 0, 0}, n]}, PlotRange -> 6, Boxed → False], {n, 6}]]
```

Out[13]=



例10：逐点画出函数图

In[8]:=
```
data = Table[{i, Sin[i]}, {i, 0, 2.1 Pi, 0.075 Pi}];
ListAnimate[Table[ListLinePlot[Take[data, i],
    Mesh -> All, PlotRange -> {{0, 6.5}, {-1, 1}}], {i, Length[data]}]]
```

第6讲　在 Mathematica 中作图

6－8　播放声音

（本节内容为选修）

Wolfram 系统处理图形和声音的方法是非常类似的.Play创建一个播放声音的对象，函数值给出作为时间函数的声音的振幅，播放指令将数学函数转化为波形的声音，返回一个Sound 对象。Mathematica可以展示任意的函数及数据的波形分析，以及基于音符的音频合成，和艺术级的声音设置。

```
{Plot[Sin[ t], {t, 0, 2}], Play[Sin[2 Pi 440 t], {t, 0, 1}]}
```

```
Speak[Plot让我们看到数学函数的形体美]
```

```
Speak[Play让我们听到函数美妙的歌声]
```

1. Play 以声音的波形形式播放函数

Play[f, {t, tmin, tmax}]	在[tmin, tmax] 秒之间播放振幅函数f
Play[{f1, f2}, {t, tmin, tmax}]	产生立体声音。首先给出左通道
Play[{f1, f2, ..., ...}]	在多通道上产生声音

例1：自定义音符.(试试m = 256 的效果)

```
m = 512;
freq = {1, 2^(2/12), 2^(4/12), 2^(5/12), 2^(7/12), 2^(9/12), 2^(11/12), 2};

f[m_, t_] :=
  Play[{Sin[512 * 2^(m/12) * 2 Pi * x], Sin[509 * 2^(m/12) * 2 Pi * x]}, {x, 0, t}]
Show[{f[0, 0.618], f[2, 0.618], f[4, 0.618], f[5, 0.618], f[7, 0.618], f[9, 0.618]}]
```

例2：中国科学技术大学校歌“永恒的东风”（片段）

```
Show[{f[7, 0.309], f[7, 0.618], f[9, 0.309], f[7, 0.4635], f[5, 0.1545],
   f[4, 0.309], f[2, 0.309], f[0, 0.927], f[2, 0.309], f[-5, 0.927],
   f[-5, 0.309], f[-3, 0.618], f[-5, 0.618], f[0, 0.4635],
   f[4, 0.1545], f[7, 0.4635], f[4, 0.1545], f[9, 1.854],
   f[7, 0.927], f[9, 0.309], f[7, 0.618], f[4, 0.309], f[0, 0.309],
   f[2, 0.618], f[4, 0.309], f[7, 0.309], f[2, 0.927],
   f[7, 0.309], f[7, 0.618], f[9, 0.309], f[7, 0.4635], f[5, 0.1545], f[4, 0.309],
   f[2, 0.309], f[0, 0.4635], f[0, 0.1545], f[0, 0.309], f[2, 0.309], f[-5, 1.236]}]
```

2. Sound 播放音符

Sound 播放音符的声音基元和指令，它可以组合不同乐器的音符序列。
像Play一样运行指令后显示包含一个表示该声音的图形和一个按钮。点击按钮播放声音。

Sound[primitives]	表示一个声音
Sound[primitives, t]	指定声音持续 t
Sound[primitives, {tmin, tmax}]	从tmin到tmax 播放声音

3. SoundNote 音符表示

给Sound提供播放的声音的基本音符元素

SoundNote[pitch, t]	音符持续的时间长度为 t
SoundNote[pitch, {tmin, tmax}]	音符持续的时间从tmin到tmax
SoundNote[pitch, tspec, "style", *opts*]	指定乐器播放音符

音符规定

"C", "C#", "D" 等	中央C八度音阶的的所有音符
"C*m*", "C#*m*", "D*m*" 等	*m*八度音阶的所有音符 ("C4" 是中央C音)
None	休止 (一次音乐停顿)
" *percussion* "	一次打击

"C+4" 等价于 "C4"；
低音符可以通过 "C-1" 等指定负数用来指定中央 C 音以下的音调.

SoundNote[] 在默认情况下，表示用钢琴风格的中央C音，演奏时间为1秒

例3：产生一个中央 C 音

```
Sound[SoundNote[0]]
```

例4：生成一个一秒的小提琴的中央 G 音：

```
Sound[SoundNote["G", 1, "Violin"]]
```

例5：演奏 135 $\dot{1}$

```
Sound[{SoundNote["C"], SoundNote["E"], SoundNote["G"], SoundNote["C5"]}]
```


```
Sound[{SoundNote[], SoundNote[4], SoundNote[7], SoundNote[12]}]
```

例6：演奏 《来生缘》 (片段)

```
Sound[{SoundNote[-3, 0.2], SoundNote[4, 0.2], SoundNote[2, 0.2],
  SoundNote[4, 0.2], SoundNote[0, 0.2], SoundNote[-1, 0.2],
```

```
SoundNote[-1, 0.2], SoundNote[-3, 0.2], SoundNote[-3, 0.2],
SoundNote[4, 0.2], SoundNote[2, 0.2], SoundNote[4, 0.2], SoundNote[0, 0.2],
SoundNote[-1, 0.2], SoundNote[-1, 0.2], SoundNote[-3, 0.2],

SoundNote[-3, 0.2], SoundNote[4, 0.2], SoundNote[2, 0.2], SoundNote[4, 0.2],
SoundNote[0, 0.2], SoundNote[-1, 0.2], SoundNote[-1, 0.2], SoundNote[-3, 0.2],
SoundNote[9, 0.2], SoundNote[4, 0.2], SoundNote[2, 0.2], SoundNote[4, 0.2],
SoundNote[0, 0.2], SoundNote[-1, 0.2], SoundNote[4, 0.2], SoundNote[2, 0.2],

SoundNote[4, 2.4], SoundNote[2, 0.2],
SoundNote[0, 0.2], SoundNote[-3, 0.2], SoundNote[-5, 0.2],

SoundNote[-3, 2.75],

SoundNote[-3, 0.8], SoundNote[-3, 0.4],
SoundNote[-1, 0.2], SoundNote[0, 1], (*寻寻觅觅*)

SoundNote[None, 0.25], SoundNote[-3, 0.4], SoundNote[4, 0.2],
SoundNote[2, 0.2], SoundNote[2, 0.2], SoundNote[0, 0.2], SoundNote[2, 0.4],
SoundNote[-3, 0.2], SoundNote[2, 1], (*在无声无息中消失*)

SoundNote[-1, 0.2], SoundNote[-1, 0.2], SoundNote[-1, 0.2],
SoundNote[-1, 0.2], SoundNote[-1, 0.4], SoundNote[-3, 0.4],
SoundNote[-5, 0.0], SoundNote[-5, 0.6], (*总是找不到回忆*)

SoundNote[-5, 0.2], SoundNote[-5, 0.4], SoundNote[-3, 0], SoundNote[-1, 0.4],
SoundNote[0, 0.2], SoundNote[-3, 0.2], SoundNote[-3, 0.2],
SoundNote[-3, 0.2], SoundNote[-3, 0.4], SoundNote[-5, 0.2],
SoundNote[-3, 1], SoundNote[None, 0.3], (*找不到曾被遗忘的真实*)

SoundNote[-3, 0.2], SoundNote[-3, 0.2], SoundNote[-3, 0.2],
SoundNote[-3, 0.2], SoundNote[-3, 0.4], SoundNote[-1, 0.2],
SoundNote[0, 1], SoundNote[None, 0.25], (*一段一段的回忆*)

SoundNote[-3, 0.4], SoundNote[4, 0.2], SoundNote[2, 0.2],
SoundNote[2, 0.2], SoundNote[0, 0.2], SoundNote[2, 0.4],
SoundNote[-3, 0.2], SoundNote[2, 1], (*回忆已经没有意义*)

SoundNote[7, 0.2], SoundNote[7, 0.2], SoundNote[7, 0.2],
SoundNote[7, 0.2], SoundNote[7, 0.4], SoundNote[2, 0.4],
SoundNote[7, 0.2], SoundNote[9, 0.4], SoundNote[7, 0.2], SoundNote[9, 0.4],
SoundNote[7, 0.4], SoundNote[4, 1], SoundNote[None, 0.4],

SoundNote[9, 0.2], SoundNote[9, 0.2], SoundNote[9, 0.2], SoundNote[9, 0.2],
SoundNote[9, 0.4], SoundNote[4, 0.2], SoundNote[9, 1], (*也许分开不容易*)
```

```
  SoundNote[None, 0.2], SoundNote[7, 0.4], SoundNote[4, 0.2],
  SoundNote[2, 0.2], SoundNote[2, 0.2], SoundNote[2, 0.2],
  SoundNote[2, 0.2], SoundNote[2, 0.4], SoundNote[-3, 0.2],
  SoundNote[2, 1], SoundNote[None, 0.6], (*也许相亲相爱不可以*)

  SoundNote[7, 0.2], SoundNote[7, 0.2], SoundNote[7, 0.2],
  SoundNote[7, 0.2], SoundNote[7, 0.4], SoundNote[2, 0.4],
  SoundNote[7, 0.2], SoundNote[9, 0.4], SoundNote[7, 0.2], SoundNote[9, 0.4],
  SoundNote[7, 0.4], SoundNote[4, 1], SoundNote[None, 0.4],

  SoundNote[4, 0.2], SoundNote[4, 0.2], SoundNote[4, 0.2],
  SoundNote[4, 0.2], SoundNote[4, 0.4], SoundNote[2, 0.4], SoundNote[4, 0.2],
  SoundNote[2, 0.2], SoundNote[0, 0.8], (*情深缘浅不得已*)

  SoundNote[None, 0.2], SoundNote[-3, 0.4], SoundNote[4, 0.2],
  SoundNote[2, 0.2], SoundNote[2, 0.2], SoundNote[2, 0.2], SoundNote[2, 0.4],
  SoundNote[-3, 0.2], SoundNote[2, 1](*你我也知道去珍惜*)}]
```

4 . Speak 语音

Speak["string"] 播放 "string" 中的文本的语音表示

Button["按钮名", **Speak["string"]**] 设置播放按钮，单击按钮播放string

SpokenString[expr] 给出表达式expr 语音表示的文本字符串

例7：朗读中英文.

```
Speak[InterpolatingPolynomial]
```

Speak[SpokenString给出表达式语音表示的文本字符串]

例8：朗读数学表达式.

```
Speak[1 + x^3 + Cos[x]]
```

```
SpokenString[1 + x^3 + Cos[x]]
```

```
1 plus x cubed plus cosine of x
```

例9：制作语音按钮.

```
Button["press", Speak["Good morning"]]
```

```
BarChart3D[{1, Button[2, Speak[2]], 3}]
```

第7讲 自定义函数和模式替换

7－1 自定义函数

所有的输入的实体都是符号表达式；

所有的操作都是调用变换规则（Transformation rules）对表达式求值；

表达式求值过程就是将表达式从一种表示形式变换为另一种表示形式的过程。

在符号计算环境下并不强调函数与命令的区别。

例：求和函数Sum和绘图Plot等都可看成一个变换规则。

```
xx = 123
```

对表达式xx定义了一条变换规则

1．模式 x_ _blank

本核心语言的一个独特的优势，其强大的、简洁明了的、 高可读性的符号语言

例1： 定义函数 f (x) = x－1．

```
f[x_] := x - 1   (*或 f[x_]=x -1*)

{f[10], f[a]}

s = {{1, 2}, {3, 4}}; f[s]
```

例2：变量是矩阵

```
g[x] = 1 + 2 x

g[10] + g[x]   (* 系统只认识g[x]，g[10]无定义 *)
```

例3：同名函数不同模式

```
f[x_, y_] := x + y
f[x_, x_] := Sin[x] + Cos[x]
{f[11], f[3, 3], f[3, 5]}
```

例4：看看f的所有的函数定义

```
? f
```

模式定 义	说 明
f[x_]	定义时命名为 x的任意表达式，
f[x_, y_]	命名为 x, y的任意表达式
f[x_, x_]	命名为 两个相同的任意表达式
x^n_	x 的任意幂次，幂次为n
x_^n_	任意表达式的任意幂次
{a_, b_}	含有两个表达式的表

2． := 与 = 延时赋给与立即赋给

lhs := *rhs*　　*rhs*保留未计算的形式

例5：观察延时赋给与立即赋给的区别

```
x = 1; fa[x_] := 2 + x

ga[x_] = 2 + x
```

```
{fa[5], ga[5]}
```

例6：计算数据表list的算术平均值.

```
mean[list_] := Apply[Plus, list] / Length[list]
```

```
mean[{1, 3, 5, 7, 9}]
```

例7：计算方阵的算术平均值

```
aver[m_] := (n = Length[m]; Sum[m[[i, j]], {i, 1, n}, {j, 1, n}] / n^2)
aver[{{1, 2}, {3, 10}}]
```

3. x_, x__, x___

x__ 一个或多个变量， x___ 零个、一个或多个变量

例8：观察定义和调用时变量的数量

```
h[x__] := Plus[x]
```

```
h[2, 3, 4]
```

```
h[{2, 3, 4}]
```

4. 模式类型说明

模式	说 明	模式	说 明
x_	任何表达式，	x_Integer	任何整数
x_Real	任何近似实数，	x_Complex	任何复数
x_List	任何表	x _Matrix	变量是矩阵
x _h	任何头为h的表达式		

例9：定义变量为整型

```
ua[x_Integer] := (x - 1) (x + 1)
{ua[a + b], ua[3], ua[1.1]}
```

对于更复杂的模式，例如，模式是y的多项式，元素为数值的矩阵等限定条件。则用

```
pattern?test
```

问号表示对模式测试是否满足test的逻辑条件。仅当测试函数的值为真时才调用函数。

例10：定义变量为 数值类型

```
uv[x_?NumericQ] := x + 2
{uv[1.2], uv[3], uv[4 / 5], uv[2 + 3 I], uv[xyz]}
```

测试函数	是否为
NumberQ[expr]	数
NumericQ[expr]	数值类型
IntegerQ[expr]	整数
OddQ[expr] / EvenQ[expr]	奇数 / 偶数
Positive[x] / Negative[x]	正数 / 负数
NonPositive[x] / NonPositive[x]	非正数 / 非负数
MatrixQ[expr]	矩阵
MemberQ[list, form]	list中是否有form的元素
TrueQ[expr]	逻辑值为真
PolynomialQ[expr, {x1, x2, ..}]	expr是否为x1, x2, ... 的多项式
FreeQ[expr, form]	expr 中是否没有与form匹配的部分
MatchQ[expr, form]	模式 form 是否与 expr 匹配

例11：模式匹配吗？

```
{MatchQ[a^b, _ + _], MatchQ[a^b, _^_], MatchQ[a^b, _], MatchQ[a^b, __]}
```

第7讲 自定义函数和模式替换

7－2 模式替换

1．自定义模式替换

模式替换能建立变换规则；扩大了变换规则的抽象功能，
适合建立数学上各类等价的变换公式和系统控制函数.

例1：变量替换给表达式或变量赋值.

```
f = x^2 + 2 x + 1;  f /. x → 2
```

```
fun[1 - x] + fun[x] /. fun → try
```

例2：请观察替换步骤

```
{a, b, c} /. a → b /. b → d
```

```
{a, b, c} /. {a → b, b → d}
```

例3：模式替换能用自定义函数取代？

```
1 + f[x] + f[y] + f[z - 1] /. f[t_] → t^2
```

```
1 + x^(1/2) + x^2 /. x^n_ → g[m]
```

例4：模式替换化简三角函数

```
t = Sin[2 x] + Sin[2 y z] + Sin[z];
t /. Sin[2 x_] → 2 Sin[x] Cos[x]
```

为模式替换取名并保存到变量中，调用更方便.

```
s = Sin[x_] → 2 Sin[x/2] Cos[x/2]
```

```
%% /. s
```

2．→ 和 :> 立即变换和延迟变换

→ 和 :> 的区别正象定义函数时 = 与 := 一样，分别表示立即变换和延迟变换。
lhs → rhs 在定义规则时 rhs被求值；常用于替换一个具有确定值的表达式；
lhs :> rhs 在定义规则时 rhs不求值，调用规则时才求值。
常用于设定一个数学的关系式或给出执行命令时才计算表达式时

例5：观察随机数在立即变换和延迟变换中的区别

```
{x, x, x} /. x -> RandomReal[]
```

```
{x, x, x} /. x :> RandomReal[]
```

例6：Expand 在立即变换和延迟变换中

```
fa[x_] → Expand[(1 + x)^2]
```

```
s = ga[x_] :> Expand[(1 + x)^2]
```

```
ga[y + 2] /. s
```

```
ga[z + 2] /. ga[x_] :> Expand[(1 + x)^2]
```

3. 非自动调用规则

Mathematica 系统的规则可分成两类：自动调用的规则和非自动调用的规则.
自动调用的规则包括系统的内部制定的规则和用户使用 = 或 := 定义的规则，
非自动使用的规则由 → 或 :> 定义，要用 /. 、 //. 或 Replace 才能调用规则.

(1) /. 与 //.

表达式 /. 规则　　**ReplaceAll** [expr, rules]
对表达式只调用一次，

表达式 //. 规则　　ReplaceRepeated[expr, rules]
对表达式反复调用规则，直到表达式调用不再变化为止.

例7：调用一次和反复调用

```
Clear[x, f]; f[3] /. f[x_] → x f[x - 1]
```

```
f[3] //. {f[1] → 1, f[x_] → x f[x - 1]}
```

例8：对数运算

```
log[a b c] /. log[x_ y_] → log[x] + log[y]
```

```
log[a b c] //. log[x_ y_] → log[x] + log[y]
```

(2) Replace 与 ReplaceAll (/.)

Replace[expr, rules]	应用规则表 rules 来替换整个表达式expr;
Replace[expr, rules, n]	应用规则表来替换表达式expr 的第n层
ReplaceAll[expr, rules]	应用规则表 rules 替换表达式 expr 的每个部分,
expr /. rules	
ReplaceList[expr, lhs → rhs]	以所有可能的方式应用规则表替换整个表达式 expr，并列出所有匹配的形式

例9：Replace，ReplaceAll 和 ReplaceList

```
{Replace[x^2, x^2 → Sin[x]], Replace[1 + x^2, x^2 → Sin[x]]}
```

```
Replace[1 + x^2, x^2 → Sin[x], {1}]
```

```
{ReplaceAll[x^2, x^2 → Sin[x]], ReplaceAll[1 + x^2, x^2 → Sin[x]]}
```

```
ReplaceList[a + b + c, x_ + y_ → g[x, y]]
```

```
ReplaceList[a + b + c, x_ + y_ → x y]
```

4. 给模式替换附加条件 /;

`pattern /; condition`	满足条件condition时模式匹配
`lhs -> rhs /; condition`	模式满足条件 condition 时调用规则
`lhs := rhs /; condition`	模式满足条件 condition 时调用函数
	注： condition 逻辑表达式

例10：模式匹配中附加条件

```
{6, -7, 3.5, -1, -2} /. x_Real → 11 111
```

```
{6, -7, t, 3.5, -I} /. x_?NumericQ → "**"
```

```
{6, -7, 3.5, -1, -2} /. x_ /; x < 0 → H
```

例11：定义函数中附加条件

```
ff[x_] := Sin[x] /; x <= 2
```

```
ff[x_] := Cos[x] /; x > 2
```

```
{ff[-1], ff[3]}
```

```
If[x <= 2, Sin[x], Cos[x]]
```

第7讲 自定义函数和模式替换

7 - 3 重复模式、纯函数

复习：

__	BlankSequence	一个或多个表达式
___	BlankNullSequence	零个或多个表达式

```
f[x_Symbol, k__Integer] := Apply[Plus, x^{k}]
f[t, 3, 5, 7]
```

```
g[x___Integer] := p[x, Plus[x]]; v = 2;
{g[], g[1], g[1, 2, 3], f[u, v, 7]}
```

```
h[x_, y_: 1, z_: 2] := x + y^2 + Cos[z]
h[a, b]   (* 系统默认第三个变量的值为2 *)
h[c]
```

1. 重复模式 (Repeated Patterns)

expr ..	重复一次或多次的模式或表达式	.. Repeated
expr ...	重复零次或多次的模式或表达式	... RepeatedNull

例1：观察 .. 与 ... 的区别.

```
{{}, {a, a}, {a, b}, {a, a, a}, {a}} /. {a ..} -> x
```

```
{{}, {a, a}, {a, b}, {a, a, a}, {a}} /. {a ...} -> x
```

Count[list, form]	给出list中与模式form匹配的元素数目
Count[list, form, n]	给出与模式form匹配的到第n层元素数目

例2：选项 n 层.

```
Count[{{a, a, b}, b, {a, {a, b}, b}}, b]
```

```
Count[{{a, a, b}, b, {a, {a, b}, b}}, b, 2]
```

例3：不同的模式匹配表示.

```
Count[a + Sin[a] / (a + b), _Symbol, 1]
```

```
Count[a + Sin[a] / (a + b), _Symbol, Infinity]
```

```
Count[{1, "f", u, "h", "7", Sin[a]}, _?StringQ]
```

```
{Count[RandomReal[1, {100}], u_ /; u > 0.5],
 Count[RandomReal[1, {1000}], u_ /; u > 0.5]}
```

Cases[list, form]	给出list中与模式form匹配的所有元素列表
Cases[expr, lhs → rhs]	对expr中与模式lhs匹配的元素调用规则
Cases[expr, lhs → rhs, k]	同上，直到第k层元素

例4：给出满足的模式匹配和删除模式匹配元素列表.

```
Cases[{-1, 1, x, x^2, x^4}, x^_]
```

```
DeleteCases[{-1, 1, x, x^2, x^4}, x^_]
```

例5：给出是整数和非整数的列表.

```
Cases[{1, 1.5, f[a], 2, 3, y, f[8], 9, f[10]}, _Integer]
```

```
Cases[{1, 1.5, f[a], 2, 3, y, f[8], 9, f[10]}, Except[_Integer]]
```

例6：重复模式例题.

```
Cases[{f[a], f[a, b, a], f[a, c, a]}, f[(a | b) ..]]
```

```
Cases[{f[a], f[a, b, a], f[a, a, a]}, f[a ..]]
```

```
Cases[{f[a], f[b], f[a, a, b], f[a, b, a], f[a, b, b]}, f[a .., b ..]]
```

```
Cases[{f[a], f[b], f[a, a, b], f[a, b, a], f[a, b, b]}, f[a ..., b ...]]
```

2. 纯函数

Mathematica中提供了一种不取函数名的定义函数方式，称为纯粹函数。
对于多次调用的函数常定义函数以简化计算，定义函数时要给函数和形式变量取个名字.
纯函数的优势：不用给要定义的函数取名，甚至函数的形式变量也无需取名，
用标记 # 表示变量。使用纯粹函数，可使定义函数和调用函数一次完成。

定义形式

Function[变量，表达式]	定义一个变量的纯函数
Function[{变量表}，表达式]	定义多个变量的纯函数

纯函数的简略形式

表达式 &	表达式中的变量为 #, #1, #2, ## 等

例7：定义函数 f (u, v) = u^2 + v^3, 计算f (2, 3).

```
f[u_, v_] := u^2 + v^3; f[2, 3]
```

纯函数定义和调用一步完成

```
Function[{u, v}, u^2 + v^3][2, 3]
```

```
(#1^2 + #2^3) &[2, 3]
```

例8：按照 x 和 y 坐标着色.

```
Plot3D[Sin[x y], {x, 0, 3}, {y, 0, 3},
 ColorFunction → Function[{x, y, z}, RGBColor[x, y, 0.3]]]
```

例9：用纯函数定义矩阵.

```
Array[(#1 10 + #2) &, {3, 4}]
```

例10：按向量第2个元素由大到小排列.

```
data = {{7, 3, 1}, {4, 5, 6}, {2, 4, 3}};
Sort[data, #1[[2]] > #2[[2]] &]
```

例11：表达式的每个元素平方.

```
Map[Function[x, x^2], a + b + c]
```

例12：## 表示所有变量.

```
(Max[##] + Min[##]) &[2, 3, 4, 7, 99]
```

第7讲 自定义函数和模式替换

7 - 4 函数的属性与属性定义

例1：查看函数 Sqrt 属性.

```
Sqrt[{4, 9, 16}]
Attributes[Sqrt]
```

例2：查看算符 Plus 和算符 Times 属性.

```
Attributes[Plus]
Attributes[Times]
```

1. 属性名称

Orderless	交换律，即f[a, b] 与f[b, a] 等价
Flat	结合律，如 f[f[a], b] 等价于f[a, b]
OneInditity	恒等，f[f[a]] 等价与a
Listable	自动分配到表中
Constant	导数为零的常数
Protected	函数定义受到保护
Locked	属性值不能改变
ReadProtected	函数的定义不可读
HoldFirst	函数的第一个参量不被求值
HoldRest	除第一个参量外函数的所有参量不被求值
HoldAll	函数的所有参量不被求值

2. 属性运算函数

`Attributes[f]`	显示函数 f的属性表
`Attributes[f] = {属性1, 属性2}`	设置 f的属性
`SetAttributes[f, 属性a]`	把属性a加到f的属性表中
`ClearAttributes[f, 属性a]`	清除f中的属性a
`Attributes[f] = {}`	设置 f的属性表为空，即 f无任何属性

```
{f[f[a, b], c], f[{u, v, w}]}
```

例3：给 f 设置结合律属性

```
SetAttributes[f, Flat]
```

```
{f[f[a, b], c], f[{u, v, w}]}
```

例4：给 f 设置自动分配到列表中属性

```
SetAttributes[f, Listable]
```

```
f[{u, v, w}]
```

用户一般不要修改Mathematica内部函数的属性。如果你要增加某函数的性能，先去掉保护属性。作出变换规则或定义后，再恢复内部函数的保护属性。

例5：给 Log 增加运算规则

```
Log[a b c]
```

```
Unprotect[Log]
```

```
Log[x_ y_] := Log[x] + Log[y]
```

```
Protect[Log]        (*恢复Log的保护*)
```

```
Log[a b c]
```

3. Mathematica的计算表达式的步骤

* 计算表达式的头部
* 依次计算表达式的每个元素
* 运算Flat、Orderless和Listable等属性有关的法则
* 调用用户定义的规则
* 调用系统定义的规则
* 计算结果

例6：计算矩阵的2范数. $||A||_2 = \max\left\{\sqrt{\lambda_k}\right\}$, λ_k 是 AA^T 的特征值.

```
js2n[A_] := Max[Sqrt[Eigenvalues[Transpose[A].A]]]
```

```
m = {{2, 6}, {1, 3}};
{js2n[m], Norm[m, 2]}
```

第8讲 程序设计

语言概述

Wolfram 语言是一种高度发展的基于知识的语言，支持多种编程模式，
是一个可扩展的系统，可以方便的创建高效、模块化的程序包、
它的符号程序结构和界面体系结构，提供一个良好的软件开发环境.

过程式编程

Wolfram 语言从传统的计算机语言中脱颖而出.Wolfram 语言支持所有标准的过程式编程结构，通过集成把它们扩展到更为普遍的符号编程环境中。

例：条件 If，循环 While.

函数式编程

函数式编程是 Wolfram 语言的核心功能，它为符号功能提供了高效、简洁的特色.

例： Apply，Array，Nest，Select.

程序设计：输入输出 条件结构 循环结构 模块结构

8－1 条件结构

过程化条件：If，Which，Switch

1．If 语句

If[cond，expr1，expr2，expr3]
如果逻辑表达式 cond = True，返回expr1的值；如果cond = False，返回expr2的值；
否则返回expr3的值.expr2和expr3可缺省，所有表达式都是复合表达式.

If[cond，expr1]　如果cond = True，返回expr1的值.
If[cond，expr1，expr2]
　　如果cond = True，返回expr1的值;
　　如果cond = False，返回expr2的值.

例1： expr2 的默认值为Null，expr3的默认值为If表达式本身。

```
fa[x_] := If[x > 0, Green]
{fa[2], fa[-2], fa[t]}
```

例2：定义函数

$$f(x,y)=\begin{cases}x+y, x\cdot y\geq 0\\ x/y, x\cdot y<0\end{cases}$$

```
f[x_, y_] := If[x > 0 && y >= 0, x + y, x / y]
{f[12, 3], f[24, -12], f[2, u]}
```

2 Which 语句

Which[cond1, val1, cond2, val2, ..., condn, valn]
返回第一个满足condi = True的vali的值，
若每个condi都是False，返回Null，若某个condi既非True又非False，
返回表达式 Which[condi, vali, ..., condn, valn].

例3：用 If 嵌套和 Which 语句定义函数

$$g(x)=\begin{cases}\sin x, & x<-1\\ |x|, & -1\le x\le 1\\ \cos x, & x>1\end{cases}$$

```
ga[x_] := Which[x < -1, Sin[x], x ≥ -1 && x ≤ 1, Abs[x], x > 1, Cos[x]]
{ga[-2], ga[-0.5], ga[2]}
```

```
gb[x_] := If[x < -1, Sin[x], If[x <= 1, Abs[x], Cos[x]]]
{gb[-2], gb[-0.5], gb[2], gb[t]}
```

例4：用 Which 语句定义分段函数

$$h(x)=\begin{cases}-x, & x<0\\ \sin(x), & 0\le x<6\\ x/2, & 16\le x<20\\ 0, & \text{其它}\end{cases}$$

```
h[x_] := Which[x < 0, -x, x ≥ 0 && x < 6, Sin[x], x ≥ 16 && x < 20, x / 2, True, 0]
{h[-12], h[5], h[16.2], h[33]}
```

h[t]？课后练习

3. Switch 语句

Switch[expr, patt1, val1, ..., pattn, valn]
计算expr的值，与模式form1，form2...，依次做比较，
找出第一个与expr 匹配的模式formi计算并返回表达式valuei的值。
若无匹配，返回Switch表达式本身。

例5：Mod[x, 3]，余数为0，1，2 对应 sin x，cosx，在 {1，x} 作 Log 图.

```
g[x_] := Switch[Mod[x, 3], 0, Sin[x], 1, Cos[x], 2, Plot[Log[t], {t, 1, x}]]
{g[9], g[16], g[8]}
```

4. 函数式条件

(1) 数学函数条件

```
Piecewise[{{val1, cond1}, ..., {valn, condn}}, val]
```

(2) 条件表达式：ConditionalExpression

```
Simplify[Sqrt[x^2] + x^2, Assumptions → x < 0]
```

例6：定义分段线性函数并绘图.

$$f(x)=\begin{cases} x+4, & x<1 \\ 6-x, & -1\le x<2 \\ 2x, & 2\le x<3 \\ 9-x, & x\ge 3 \end{cases}$$

```
Plot[Which[x < 1, x + 4, x < 2, 6 - x, x < 3, 2 x, x ≥ 3, 9 - x], {x, -1, 4}]
```

```
f[x_] := Piecewise[{{x + 4, x < 1}, {6 - x, 1 ≤ x < 2}, {2 x, 2 ≤ x < 3}, {9 - x, x ≥ 3}}]
Plot[f[x], {x, -1, 4}]
```

(3) 基于列表条件

`Cases[list, patt]` 给出list中所有满足patt的元素

`Select[list, cond]` 选取list中所有满足cond的元素

例7：查找数列中的素数，Cases 和 Select 具有条件和循环的双重功能。

```
data = Range[20]; Cases[data, _?PrimeQ]
```

```
Select[data, PrimeQ]
```

```
Cases[{{a, b}, {1, 2, 3}, {{d, 6}, {d, 10}}}, {_, _}?VectorQ]
```

(4) 模式约束

`_h` 指定头部为h的任意表达式;

`/; pattern /; condition` 当条件满足时，模式才匹配;

`lhs:>rhs/;condition` 当条件满足时,才使用规则;

`p? test` 测试p是否为 True .

```
f[x_Integer] := x + 1
```

```
f[x_?NumericQ] := NIntegrate[Sin[t^3], {t, 0, x}]
```

```
g[x_] := Sin[x] /; x > 0
```

第8讲 程序设计

8－2 循环结构

过程式循环 Do, While, For

1. Do 语句

Do[循环体的表达式, 循环范围]

一重循环

Do[expr, {i, m, n, d}]

在循环范围内按步长d对expr求值。

i 循环变量 , m 初始值, n中止值, d 步长. m, m + d, m + 2 d, ...

循环变量可为整数/小数分数复数枚举. expr 复合表达式.

Do[expr, {i, m, n}] 步长 为1

Do[expr, {i, n}] 初值和步长都为1

Do[expr, {n}] 计算n 次表达式expr

Do[expr, {i, list}] 按循环列表list的元素, 对expr求值

例1：输出：n 从E 到10 按步长π递增.

```
Do[Print[n], {n, E, 10, Pi}]
```

例2：循环变量 n 是个隐含的局部变量, 它的变化不会影响语句外部的变量n的值.

```
n = 100; Do[Print[n], {n, E, 10, Pi}]; n
```

例3：循环变量是枚举类型.

```
Do[Print[t], {t, {Red, Green, Blue, Yellow}}]
```

例4：留意变量是字符的循环.

```
t = "t"; Do[t = 1 / (3 k + t); Print[t], {k, 3}]
```

$$\cfrac{1}{3+t}$$

$$\cfrac{1}{6+\cfrac{1}{3+t}}$$

$$\cfrac{1}{9+\cfrac{1}{6+\cfrac{1}{3+t}}}$$

例5： 循环范围 {k, 1, 3} 与 {k, {1, 3}} 有区别 ?

```
t = "t"; Do[t = 1 / (3 k + t); Print[t], {k, {1, 3}}]
```

$$\cfrac{1}{3+t}$$

$$\cfrac{1}{9+\cfrac{1}{3+t}}$$

二重循环

```
Do[expr, {i, i0, i1, is}, {j, j0, j1, js}]
```

i 从i0 到i1 按步长is递增；对每个i，
j 从j0 到j1 按步长js递增，计算表达式expr，
在前的是外循环，在后的是内循环.

例6 ：输出：i 从1 到3 ，j从1到3的 {i, j}.

```
Do[Print[{i, j}], {i, 3}, {j, 3}]
```

```
Do[ Do[Print[{i, j}], {j, 3}], {i, 3}]
```

例7 ：输出：i 从1 到3 ，对每个i，j从1到i的 {i, j}.

```
Do[Print[{i, j}], {i, 3}, {j, i}]
```

2. While 语句

```
While[cond, expr]
```

若逻辑条件 cond = True，则对循环体表达式求值，
重复对条件判断和对循环体求值过程直到cond ≠ True时退出循环.

例8：计算 1 + 2 + 3 + 4 + 5

```
s = 0; k = 1; While[k ≤ 5, s = s + k; Print[s]; k++]
```

```
s = 0; Do[s = s + k; Print[s], {k, 1, 5}]
```

例9：用辗转相除法计算两个数的最大公约数.

```
{a, b} = {49, 21 }
While[b ≠ 0, {a, b} = {b, Mod[a, b]}; Print[{a, b}]]; a
```

例10：用 Newton 迭代法计算3的平方根.

$$x_{k+1} = x_k - \frac{x_k^2 - 3}{2x_k}$$

```
xb = 1.5; xa = xb + 1;
While[Abs[xb - xa] > 0.00001,
    xa = xb;
    xb = xa - (xa^2 - 3) / 2 / xa;
    Print[xb]]
```

3 For 循环

For[初始值，条件，修正循环变量，循环体]

For[init, cond, incr, expr]

init 最先求值，接下来按照cond、expr、incr的顺序依次对表达式求值，直到cond ≠ True.

init, cond, incr, expr均为复合表达式，逗号是4个部分的分隔符.

例11 ：计算 1 + 2 + 3 + 4 + 5

For[init, cond, incr, expr]　For[init, cond, incr]

```
For[s = 0; n = 1, n ≤ 5, n++, s += n; Print[s]] (*正确的程序*)
```

```
For[s = 0; n = 1, n ≤ 5, ++n; s += n; Print[s]](*错误的程序*)
```

例12 ：与 Do 循环 不同，在 While、For 循环 中没有隐含的局部变量.

```
n = 100; For[n = E, n ≤ 10, n = n + Pi, Print[n]]; Print["n=", n]  (* n+=Pi *)
```

```
e
```

```
e + π
```

```
e + 2 π
```

```
n=e + 3 π
```

例 13：生成多项式序列 f (n) = $1 + \left(1 + \left(1 + x^2\right)^2 + \ldots\right)^2$

```
For[i = 1; f = x, i < 4, i++, f = 1 + f^2; Print["f(", i, ")=", f]]
```

```
f(1)=1 + x^2
```

```
f(2)=1 + (1 + x^2)^2
```

```
f(3)=1 + (1 + (1 + x^2)^2)^2
```

4. ++ 和 --

函数	说明
x++、x--	在使用x后将x增(减)1
++x、--x	在使用x前将x增(减)1
x += y、x -= y	x = x + y、x = x - y
x *= y、x /= y	x = x * y、x = x / y

5. 迭代函数（函数式循环）

Nest[f, x, n] 迭代n次，f (f (f (... f (x) ...)))

NestList[f, x, n] 迭代n次并给出每步迭代值

NestWhile[*f*, *expr*, *test*] 从*expr*开始，重复应用*f*直到*test*不再是True为止.

NestWhileList[f, expr, test] 同上，列出迭代列表

FixedPoint[*f*, *expr*] 从*expr*开始，重复应用*f*直到结果不再改变

FixedPointList[f, x0] 同上，列出迭代列表

TakeWhile[{a1, ..., an}, f] 列出 f (a1) =. . . = f (ak) = True 的ai

LengthWhile[{a1, ..., an}, f] 给出满足 (a1) =. . . = f (ak) = True的元素个数

例14：以1.0 为初值，用Newton迭代法计算3的平方根，做5次迭代并输出计算结果.

$$x_{k+1} = x_k - \frac{x_k^2 - 3}{2x_k}$$

```
f[x_] := (x + 3 / x) / 2; NestList[f, 1.0, 5]
FixedPointList[f, 1., 5]
```

例15：取出列表中的偶数

```
TakeWhile[{2, 4, 6, 1, 2, 3}, EvenQ]
```

例16：判断列表中偶数的个数

```
LengthWhile[{2, 4, 6, 1, 2, 3}, EvenQ]
```

第8讲 程序设计

8－3 程序模块

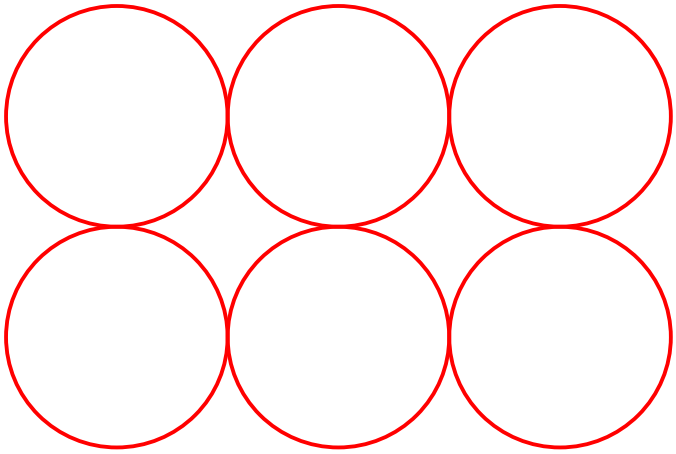
1. Block 块

```
Block[{x, y, ...}, expr]
Block[{x = x0, y = y0, ...}, expr]
```

对expr中的 {x, y, ...} 使用其局部值.Block语句的作用是保护模块外部的同名变量 {x, y, ...} 使得它们的值不受模块内部语句的影响。Block语句的效果相当于先备份 {x, y, ...} 的值，接着赋以新值 {x0, y0, ...}，再执行程序体expr，最后还原 {x, y, ...} 的备份值.

例1：用 Block 定义函数输出图形

```
fa[m_, n_] := Block[{s = 2 m - 1, t = 2 n - 1},
  Graphics[{Red, Thick, Table[Circle[{i, j}], {i, 1, s, 2}, {j, 1, t, 2}]}]]

fa[3, 2]
```



2. Module 模块

```
Module[{x, y, ...}, expr]
Module[{x = x0, y = y0, ...}, expr]
```

对 expr中的 {x, y, ...} 创建局部变量.

Module语句的作用则是保护模块内部的局部变量 {x, y, ...}，使得它们的值不受模块外部语句的影响。每次执行Module语句之前，Mathematica都会自动创建新的变量来代替 {x, y, ...}.

例2：用 Module 定义函数计算最大公约数

```
gcd[m0_, n0_] :=
 Module[{m = m0, n = n0},
  While[n ≠ 0, {m, n} = {n, Mod[m, n]}];
  m]

gcd[105, 126]]
```

例3：观察 Block 和 Module 的区别

```
f[x_] := Block[{t}, Integrate[Exp[x * t], {t, 0, 1}]];
g[x_] := Module[{t}, Integrate[Exp[x * t], {t, 0, 1}]];
{f[x], g[x], f[t], g[t]}
```

3. With　With 的速度比 Module 快

```
With[{x = x0, y = y0, ...}, expr]
```

将expr中的 {x, y, ...} 替换成 {x0, y0, ...}

例4：用 With 定义函数输出图形

```
fb[m_, n_] := With[{s = 2 m - 1, t = 2 n - 1},
  Graphics[{Purple, Thick, Table[Circle[{i, j}], {i, 1, s, 2}, {j, 1, t, 2}]}]]

fb[3, 2]
```

例5：Block 仅局部化值；它并不替代值.Module 创建新符号：

```
{Block[{x = 5}, Hold[x]], With[{x = 5}, Hold[x]], Module[{x = 5}, Hold[x]]}
```

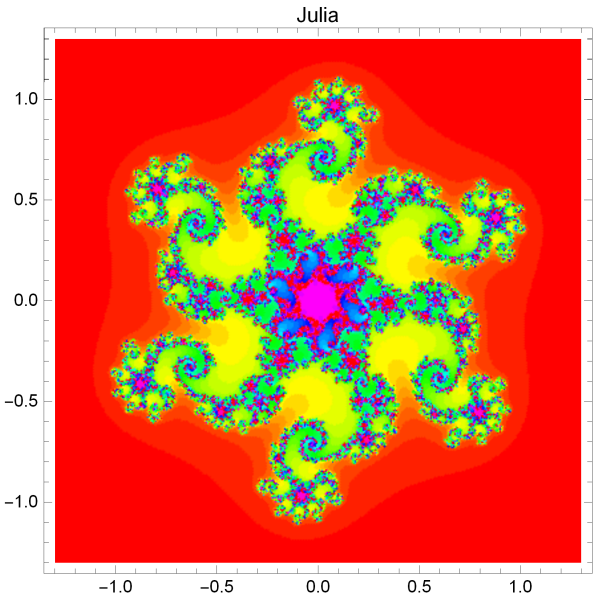
```
{Hold[x], Hold[5], Hold[x$2311]}
```

例6：绘制复映射 $f(z) = z^2 + c$ 的Julia集和Mandelbrot (曼德尔布罗特) 集的图像，

序列 $z_n = z_{n-1}^2 + c$，从 $z_0 = 0$ 开始迭代，其中Julia集是使迭代 不收敛 $z = f(z)$ 的初值 的集合，Mandelbrot集是使迭代 不收敛的所有复数 c 的集合.

Julia： 朱莉娅　　Mandelbrot：曼德尔布罗特

```
F2[x_, y_, Cx_, Cy_, n_] := Block[{z, i = 0}, z = x + y * I;
  While[(Abs[z] < 2.0) && (i < 50), ++i; z = z^n + (Cx + Cy * I)]; Return[i]]
Julia[Cx_, Cy_, n_, xm_List, ym_List] :=
 DensityPlot[F2[x, y, Cx, Cy, n], {x, xm[[2]], xm[[3]]}, {y, ym[[2]], ym[[3]]},
  PlotPoints → 100, PlotLabel → "Julia", Mesh → False, ColorFunction → Hue]

J1 = Julia[0.27334, 0.00742, 2, {x, -1, 1}, {y, -1.2, 1.2}]

J2 = Julia[0.73251, -0.414193, 6, {x, -1.3, 1.3}, {y, -1.3, 1.3}];
Labeled[J2, Style["n=6, C=0.73251-0.414193*I", 18, Bold, Blue]]
```

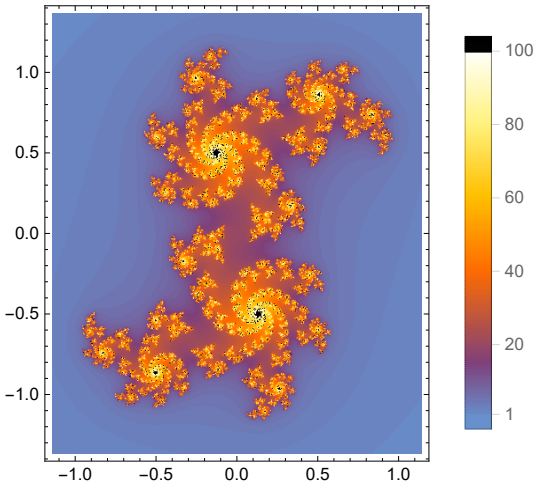


"n=6, C=0.73251-0.414193*I"

4. JuliaSetPlot 绘制朱莉娅集合，选项与Graphics相同
MandelbrotSetPlot 绘制曼德尔布罗特集，选项与Graphics相同

例7：用系统函数画分形图

```
JuliaSetPlot[0.365 - 0.37 i, PlotLegends → Automatic]
```




```
MandelbrotSetPlot[{-0.65 + 0.47 I, -0.4 + 0.72 I}, PlotLegends → Automatic]
```

```
MandelbrotSetPlot[ColorFunction → Hue]
```

5. 中止程序运行

单击 计算 (V) → 放弃计算 (A) 或 Alt + .

系统会退出全部表达式运算，返回 $Aborted

```
x = 1; While[x > 0, x]
```

第8讲 程序设计

*8－4 程序包

1．给函数附加定义信息

例1：显示系统的函数定义说明

```
Sqrt[a, b]
```

例2：给自函数定义加说明

```
fun::sm = "矩阵不可逆";
fun[x_Matrix] := If[Det[x] ≠ 0, MatrixForm[Inverse[x]], Message[fun::sm]]
fun[{{1, 2}, {2, 4}}]
fun[{{1, 2}, {2, 3}}]
fun[{{1, 2, 3}, {2, 4}}]
```

有关信息操作

`s::tag = String`	定义一条信息
`Message[s::tag]`	显示一条信息
`On[s::tag]`	打开s的tag的信息
`Off[s::tag]`	关闭s的tag的信息

2．构建程序包

程序包为纯文本格式，通常后缀名“*.m”，具有下面的一般形式。

```
BeginPackage["package`"]
    f::usage = "text" ...
      Begin["context`"]
            f[args] = values
            ...
      End[]
EndPackage[]
```

`Begin["context`"]` `End[]` 定义了一个上下文（Context）

```
$Context
```

例3：生成一个软件包，用来计算一组数据的算术平均、几何平均、中位数、方差和标准差.

```
BeginPackage["stat`"];
 mean::usage = "计算算术平均";
 geomean::usage = "计算几何平均";
 median::usage = "计算中位数";
 var::usage = "计算方差";
 stdev::usage = "计算标准差";
  Begin[my]
      mean[x_] := Total[x] / Length[x];
      geomean[x_] := Apply[Times, x] ^ (1 / Length[x]);
      median[x_] := Module[{n = Length[x], s = Sort[x]},
                  If[OddQ[n], s[[(n + 1) / 2]], (s[[n / 2]] + s[[n / 2 + 1]]) / 2]];
      var[x_] := Total[(xmean[x]) ^2] / (Length[x] - 1);
     stdev[x_] := Sqrt[var[x]];
  End[]
EndPackage[];
```

点击菜单项 文件 -> 另存为
在“保存类型”中选择 纯文本 或者使用任意文字处理软件, 录入以下内容，
保存为“我的文档”目录下的文件后缀为.m的文件，例如“MyPackage.m”

```
<< MyPackage`
```

或 `Needs["MyPackage`"]` 加载已生成的软件包.

```
data = RandomReal[{0, 1}, 10]
```

```
{mean[data], geomean[x], var[x], stdev[x]}
```

3. Wolfram 系统标准程序包

统计学程序包中包含

Analysis of Variance »

Hierarchical Clustering »

Hypothesis Testing »

Multivariate Statistics »

Statistical Plots »

离散数学程序包

Combinatorica

Computational Geometry »

Graph Utilities »

微积分学程序包中包含

Equation Trekker »

Fourier Series »

Function Approximations »

Numerical Calculus »

Numerical Differential Equation Analysis »

Variational Methods »

还有代数学程序包，多面体程序包，辅助数学程序包，

绘图和日期程序包 ，声音程序包 ，物理量和属性程序包，实用程序包等

例4：调入多面体程序包，演示12面体.（多面体可以通过标准名称来指定）

```
Needs["PolyhedronOperations`"]
{PolyhedronData["Dodecahedron"], Truncate[PolyhedronData["Dodecahedron"]]}
Show[Stellate[PolyhedronData["Dodecahedron"]], Boxed -> False]
$Packages
$Path
```