

# 附录：Mathematica 简明教程



第1章 Mathematica概述

第2章 Mathematica的基本量

第3章 Mathematica的基本运算

第4章 函数作图

第5章 微积分的基本操作

第6章 微分方程的求解

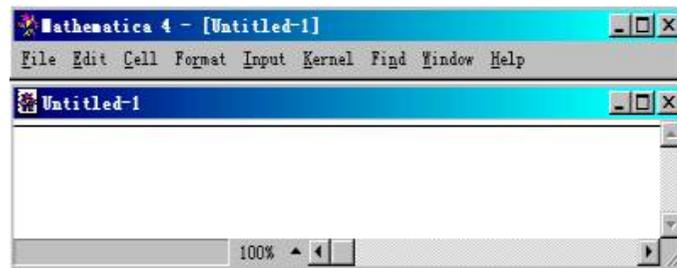
第7章 Mathematica程序设计

第8章 Mathematica中的常用函数

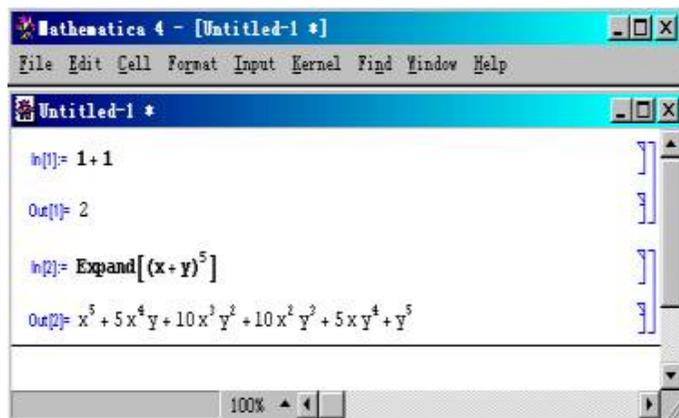
# 第一章：Mathematica概述

## 1.1.1 Mathematica的启动和运行

在Windows环境下已安装好Mathematica4.0，启动Windows后，在“开始”菜单的“程序”中单击  Mathematica 4，就启动了Mathematica4.0，在屏幕上显示如图的Notebook窗口，系统暂时取名Untitled-1，直到用户保存时重新命名为止：



输入1+1，然后按下Shift + Enter键，这时系统开始计算并输出计算结果，并给输入和输出附上次序标识In[1]和Out[1]；再输入第二个表达式，要求系统将一个二项式展开，按Shift+Enter输出计算结果后，系统分别将其标识为In[2]和Out[2]。



在Mathematica的Notebook界面下，可以用这种交互方式完成各种运算，如函数作图，求极限、解方程等，也可以用它编写像C那样的结构化程序。

在Mathematica系统中定义了许多功能强大的函数，可以直接调用这些函数。这些函数分为两类，一类是常用数学函数；第二类是功能函数，如作函数图形的函数

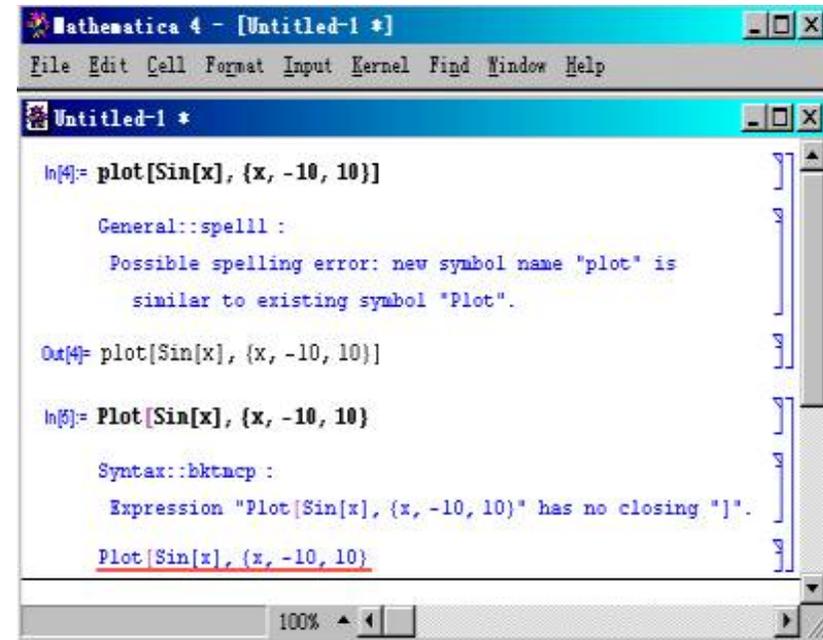
Plot[f[x],{x,xmin,xmax}]，解方程函数Solve[eqn,x]等。 2

# 第一章：Mathematica概述

**错误信息：**如果输入了不合语法规则的表达式，系统会显示出错信息，并且不给出计算结果。

- (1) Mathematica 区分字母的大小写；
- (2) 所有指令的首字母大写；
- (3) 括号的匹配。

例如：要画正弦函数在区间[-10, 10]上的图形，输入`plot[Sin[x],{x,-10,10}]`，则系统提示“可能有拼写错误，新符号‘plot’很像已经存在的符号‘Plot’”，实际上，系统作图命令“Plot”第一个字母必须大写，一般地，系统内建函数首写字母都要大写。再输入`Plot[Sin[x],{x,-10,10}`，系统又提示缺少右方括号，并且将不配对的括号用蓝色显示，如右图



```
Mathematica 4 - [Untitled-1 *]
File Edit Cell Format Input Kernel Find Window Help

Untitled-1 *
In[4]:= plot[Sin[x], {x, -10, 10}]

General::spell1 :
Possible spelling error: new symbol name "plot" is
similar to existing symbol "Plot".

Out[4]= plot[Sin[x], {x, -10, 10}]

In[5]:= Plot[Sin[x], {x, -10, 10}

Syntax::bktncp :
Expression "Plot[Sin[x], {x, -10, 10}" has no closing "]".

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

**运行结束：**

完成各种计算后，点击**File->Exit**退出，如果文件未存盘，系统提示用户存盘，文件名以“.nb”作为后缀，称为Notebook文件。以后想使用本次保存的结果时可以通过**File->Open**菜单读入，也可以直接双击它，系统自动调用Mathematica将它打开。

# 第一章：Mathematica概述

## 1.1.2 表达式的输入

Mathematica 提供了多种输入数学表达式的方法。除了用键盘输入外，还可以使用工具样或者快捷方式键入运算符、矩阵或数学表达式。

### 1. 数学表达式二维格式的输入

Mathematica提供了两种格式的数学表达式。形如 $x/(2+3x)+y*(x-w)$ 的称为一维格式，

形如  $\frac{x}{2+3x} + \frac{y}{x-w}$  的称为二维格式。 可以使用快捷方式输入二维格式，也可用基本

输入工具栏 输入二维格式。下面列出了用快捷方式输入二维格式的方法

数学运算

数学表达式

键盘输入

分式

$$\frac{x}{2}$$

x Ctrl+/ 2

n次方

$$x^n$$

x Ctrl+^ n

开n次方

$$\sqrt[n]{x}$$

Ctrl+2 x Ctrl+5 n

下标

$$x_2$$

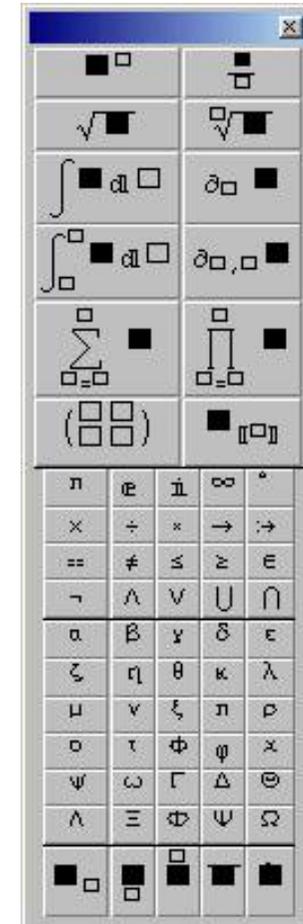
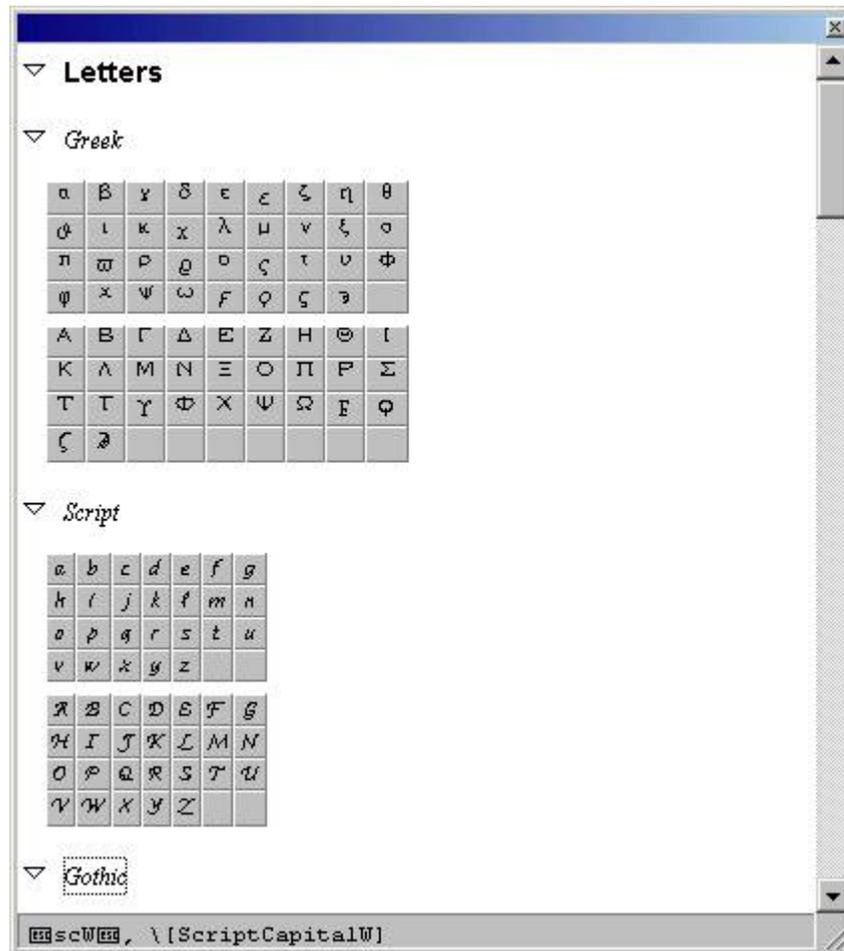
x Ctrl+\_2

# 第一章：Mathematica概述

从FILE菜单中激活Palettes->Basic Input 工具栏，使用工具栏可输入更复杂的数学表达式。

## 2.特殊字符的输入

单击后输入。



# 第一章：Mathematica概述

## 1.2.Mathematica的联机帮助系统

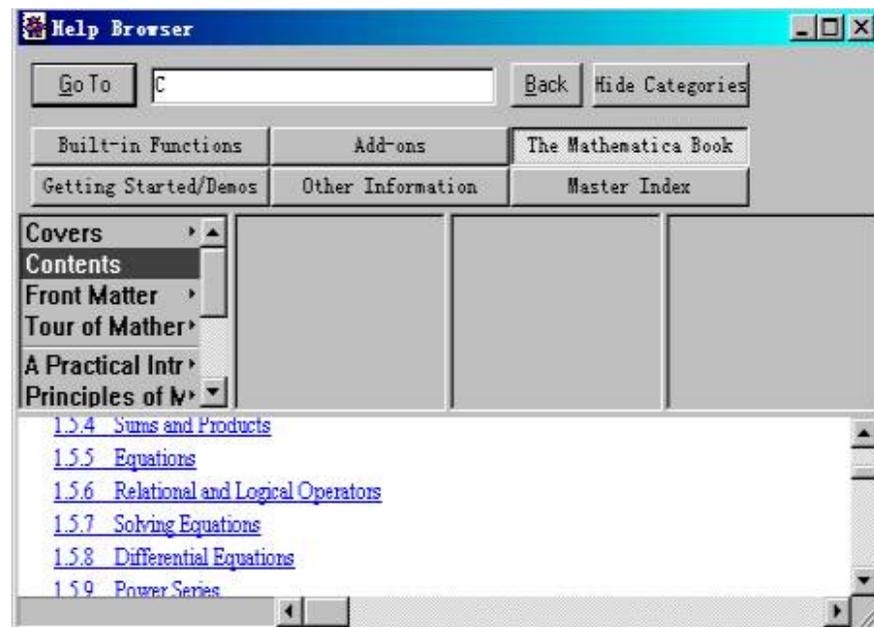
用Mathematica的过程中，常常需要了解一个命令的详细用法，或者想知系统中是否有完成某一计算的命令，联机帮助系统永远是最详细、最方便的资料库。

### 1.获取函数和命令的帮助

在Notebook界面下，用 `?` 或 `??` 可向系统查询运算符、函数和命令的定义和用法，获取简单而直接的帮助信息。例如，向系统查询作图函数Plot命令的用法？`Plot` 系统将给出调用Plot的格式以及Plot命令的功能（如果用两个问号“??”，则信息会更详细一些）。`? Plot*` 给出所有以Plot这四个字母开头的命令。

## 2.Help菜单

可以通过按F1键或点击帮助菜单项Help Browser，调出帮助菜单



# 第一章：Mathematica概述



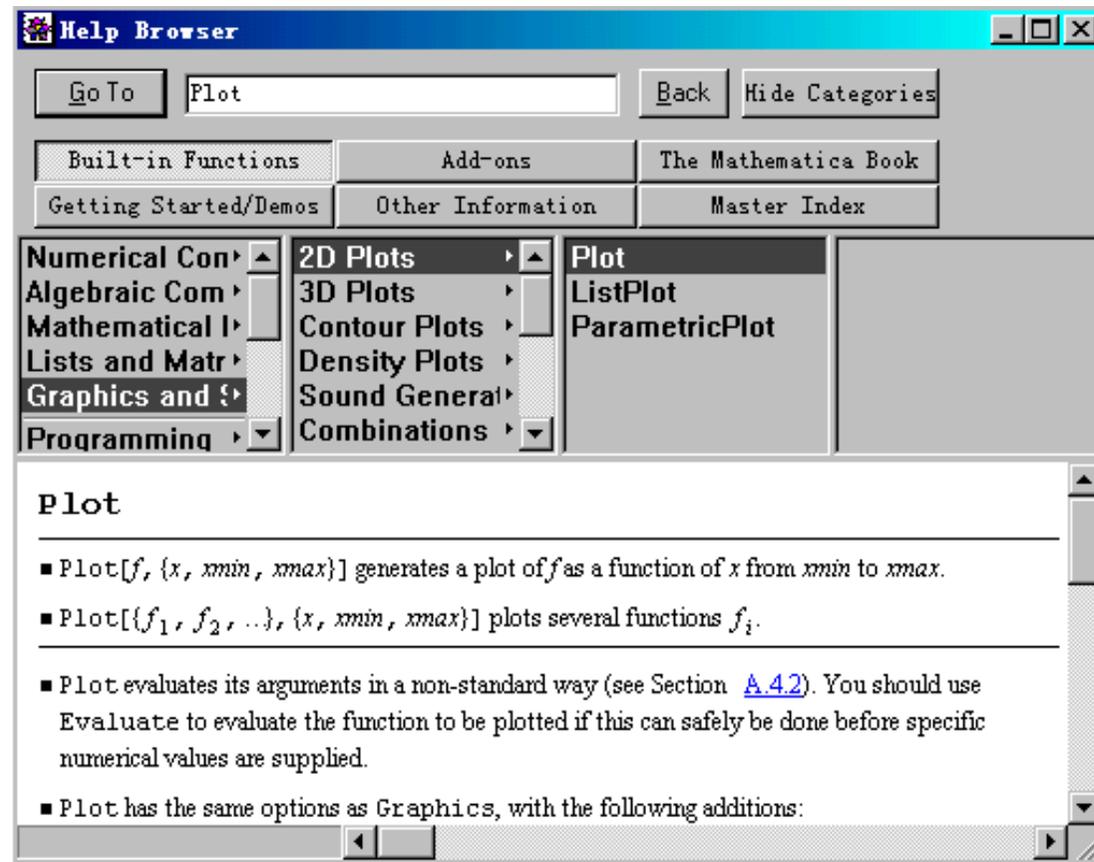
其中的各按钮用途如表所示

<b>Built-in Function</b>	内建函数，按数值计算、代数计算、图形和编程分类存放
<b>Add-ons</b>	程序包（ <b>Standard Packages</b> ） <b>MathLink Library</b> 等内容
<b>The Mathematica Book</b>	一本完整的 <b>Mathematica</b> 使用手册
<b>Getting Started/Demos</b>	初学者入门指南和多种演示
<b>Other Information</b>	菜单命令的快捷键，二维输入格式等
<b>Master Index</b>	按字母命令给出命令、函数和选项的索引表

查找Mathematica中具有某个功能的函数，可通过帮助菜单中的Mathematica使用手册，通过其目录索引可以快速定位到自己要找的帮助信息。

# 第一章：Mathematica概述

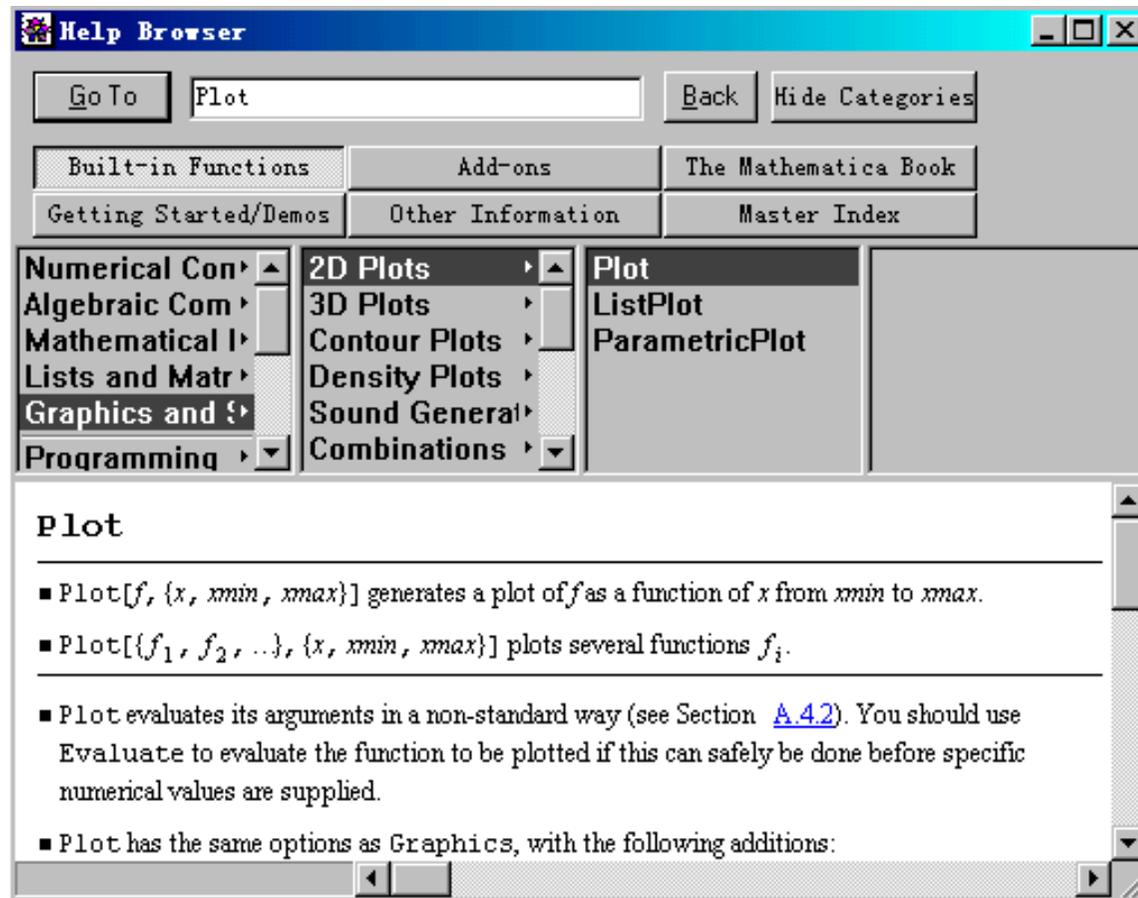
例如：需要查找Mathematica中有关解方程的命令，单击“**The Mathematica Book**”按钮，再单击“**Contents**”，在目录中找到有关解方程的节次，点击相应的超链接，有关内容的详细说明就马上调出来了。如果知道具体的函数名，但不知其详细使用说明，可以在命令按钮 **Goto** 右边的文本框中键入函数名，按回车键后就显示有关函数的定义、例题和相关联的章节。例如，要查找函数**Plot**的用法，只要在文本框中键入**Plot**，按回车键后显示如图的窗口，再按回车键，则显示**Plot**函数的详细用法和例题。



# 第一章：Mathematica概述

如果已经确知Mathematica 中有具有某个功能的函数，但不知具体函数名，可以点击 **Built-in Functions**按钮，再按功能分类从粗到细一步一步找到具体的函数，例如，要画一元函数图形的函数，点击

Built-in Functions ->Graphics and Sound->2D Plots->Plot，找到Plot的帮助信息。



## 第2章： Mathematica的基本量

### 2.1 数据类型和常数

**1.数值类型：**基本的数值类型有四种：整数，有理数、实数和复数

如果计算机的内存足够大,Mathemateica可以表示任意长度的精确实数，而不受所用的计算机字长的影响。整数与整数的计算结果仍是精确的整数或是有理数。

例如：2的100次方是一个31位的整数：

```
In[1]:=2^100
```

```
Out[1]=1267650600228228229401496703205376
```

允许使用分数，也就是用有理数表示化简过的分数。当两个整数相除而又不能整除时，系统就用有理数来表示，即有理数是由两个整数的比来组成如：

```
In[2]:=12345/5555
```

```
Out[2]=2469/1111
```

## 第2章： Mathematica的基本量

实数是用浮点数表示的，Mathematica实数的有效位可取任意位数，是一种具有任意精确度的近似实数，当然在计算的时候也可以控制实数的精度。实数有两种表示方法：一种是小数点另外一种是用指数方法表示的。如：

```
In[3]:=0.239998
```

```
Out[3]=0.23998
```

```
In[4]:=0.12*10^11
```

```
Out[4]=0.12*10^11
```

实数也可以与整数，有理数进行混合运算，结果还是一个实数。

```
In[5]:=2+1/4+0.5
```

```
Out[5]=2.75
```

复数是由实部和虚部组成。实部和虚部可以用整数，实数，有理数表示。在Mathematica中，用i表示虚数单位如：

```
In[6]:=3+0.7i
```

```
Out[6]:=3+0.7i
```

## 第2章： Mathematica的基本量

### 2.不同类型数的转换

在Mathematica的不同应用中，通常对数字的类型要求是不同的。例如在公式推导中的数字常用整数或有理数表示，而在数值计算中的数字常用实数表示。在一般情况下在输出行Out[n]中，系统根据输入行In[n]的数字类型对计算结果做出相应的处理。如果有一些特殊的要求，就要进行数据类型转换。

<b>N[x]</b>	将x转换成实数
<b>N[x,n]</b>	将x转换成近似实数，精度为n
<b>Rationalize[x]</b>	给出x的有理数近似值
<b>Rationalize[x,dx]</b>	给出x的有理数近似值，误差小于dx

例如：

```
In[1]=N[5.3,20]
Out[1]=1.6666666666666666666667
In[2]:=N[%,10]
Out[2]=1.66666667
```

二行输出是把上面计算的结果变为10位精度的数字。%表示上一输出结果。

```
In[3]=Rationalize[%]
Out[3]=5/3
```

## 第2章： Mathematica的基本量

### 3.数学常数

Mathematica 中定义了一些常见的数学常数，这些数学常数都是精确数，例如表示圆周率。

Pi	表示 $\pi = 3.14159\cdots$
E	自然对数的底, $e=2.71828\cdots$ .
Degree	$\pi / 180$
i	虚数单位
Infinity	无穷大 $\infty$
-infinity	负的无穷大 $-\infty$
GondenRatio	黄金分割数0.61803

数学常数可用在公式推导和数值计算中。在数值计算中表示精确值：如：

```
n[1]:=Pi^2
Out[1]=  $\pi^2$ 
In[2]:=Pi^2//N
Out[2]=9.86961
```

## 第2章： Mathematica的基本量

### 4.数的输出形式

在数的输出中可以使用转换函数进行不同数据类型和精度的转换。另外对一些特殊要求的格式还可以使用如下的格式函数：

`NumberForm[expr, n]` 以n位精度的实数形式输出实数expr

`ScientificFormat[expr]` 以科学记数法输出实数expr

`EngineergForm[expr]` 以工程记数法输出实数expr

```
In[1]:=N[Pi^30, 30]
```

```
Out[2]:=8.21289330402749581586503585434 x 1014
```

```
n[2]:=NumberForm[%, 10]
```

```
Out[2]//NumberForm=8.212893304 x 1014
```

## 第2章： Mathematica的基本量

### 2.2 变量

#### 1. 变量的命名

Mathematica中内部函数和命令都是以大写字母开始的标示符。为了不会与它混淆，我们自定义的变量应该是以小写字母开始，后跟数字和字母的组合，长度不限。例如：`a12`,`ast`,`aST`都是合法的，而`12a`，`z*a`是非法的。另外在Mathematica中的变量是区分大小写的 在Mathematica中，变量不仅可以存放一个数值，还可以存放表达式或复杂的算式。

#### 2. 给变量赋值

在Mathematica中用等号=为变量赋值。同一个变量可以表示一个数值，一个数组，一个表达式，甚至一个图形。如：

```
In[1]:=x=3
```

```
Out[1]=3
```

```
In[2]:=x^2+2x
```

```
Out[2]=15
```

```
In[3]:=x=%+1
```

```
Out[3]=16
```

对不同的变量可同时赋不同的值例如：

```
In[4]:={u,v,w}={1,2,3}
```

```
Out[4]={1,2,3}
```

```
In[5]:=2u+3v+w
```

```
Out[5]=11
```

对于已定义的变量，当你不再使用它是，为防止变量值的混淆，可以随时用`=.`清除他的值，如果变量本身也要清除用函数`Clear[x]`例如

```
In[6]:=u=.
```

```
In[7]:=2u+v
```

```
Out[7]=2+2u
```

## 第2章： Mathematica的基本量

### 3.变量的替换

在给定一个表达式时其中的变量可能取不同的值，这是可用变量替换来计算表达式的不同值。方法为用`expr/.`例如：

```
In[1]:=f=x/2+1
```

```
Out[1]= 1 +  $\frac{x}{2}$ 
```

```
In[2]:=f/.x->1
```

```
Out[2]=  $\frac{3}{2}$ 
```

```
In[3]:=f/.->2
```

```
Out[3]=3
```

如果表达式中有多个变量也可以同时替换方法为例如有两个：

```
expr/.{x->xval,y->val}
```

```
Ln[4]:= (x+y) (x-y)^2/.{x->3,y->1-a}
```

```
Out[4]= (4-a)(2+a)^2
```

## 第2章： Mathematica的基本量

### 2.3函数

1. **系统函数** 在Mathmatic中定义了大量的数学函数可以直接调用，这些函数其名称一般表达了一定的意义，可以帮助我们理解。下面是几个常用的函数：

<b>Floor[x]</b>	不比x大的最大整数
<b>Ceiling[x]</b>	不比x小的最小整数
<b>Sign[x]</b>	符号函数
<b>Round[x]</b>	接近x的整数
<b>Abs[x]</b>	x绝对值
<b>Max[x1,x2,x3.....]</b>	x1 ,x2,x3.....中的最大值
<b>Min[x1,x2,x3.....]</b>	x1,x2,x3.....中的最小值
<b>Random[ ]</b>	0~1之间的随机函数
<b>Random[Real,xmax]</b>	0~xmax之间的随机函数
<b>Exp[x]</b>	指数函数
<b>Log[x]</b>	自然对数函数lnx
<b>Log[b,x]</b>	以b为底的对数函数
<b>Sin[x],Cos[x],Tan[x],Csc[x],Sec[x],Cot[x]</b>	三角函数（变量是以弧度为单位的）
<b>Sinh[x],Cosh[x],Tanhx[x],Csch[x],Sech[x],Coth[x]</b>	双曲函数
<b>Mod[m,n]</b>	m被n整除的余数，余数与n的符相同

。 。 。 。 。 。

## 第2章： Mathematica的基本量

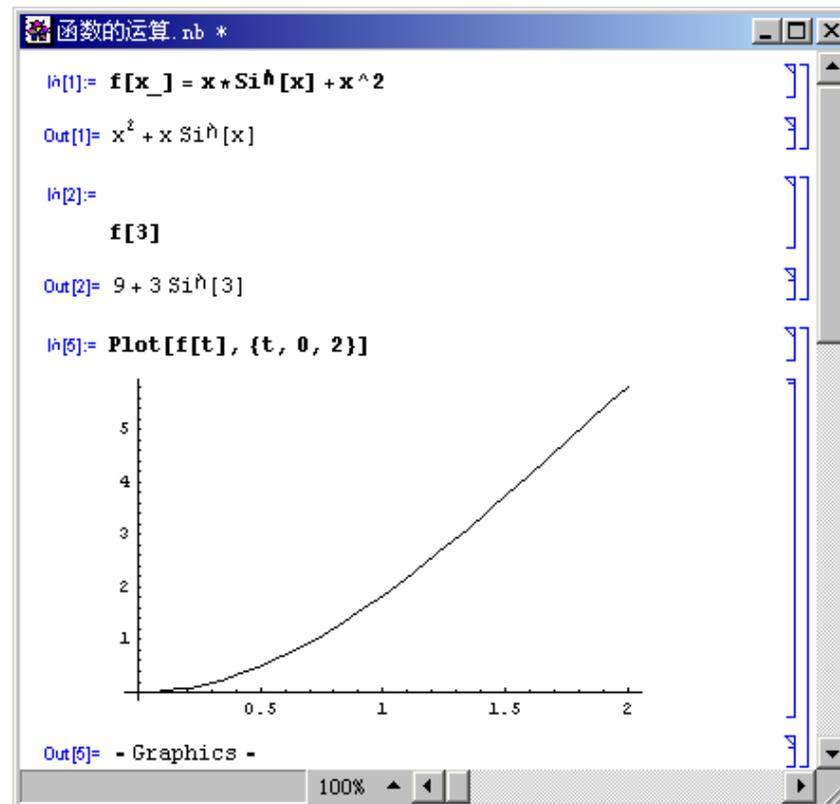
### 2. 函数的定义

#### (1) 函数的立即定义

立即定义函数的语法如下`f[x_]=expr`函数名为f,自变量为x, `expr`是表达式。在执行时会把`expr`中的x都换为f的自变量x(不是x\_)。函数的自变量具有局部性,只对所在的函数起作用。函数执行结束后也就没有了,不会改变其它全局定义的同名变量的值。请看下面的例子

定义函数 $f(x)=x*\text{Sin}x+x^2$ 对定义的函数我们可以求函数值,也可绘制它的图形。

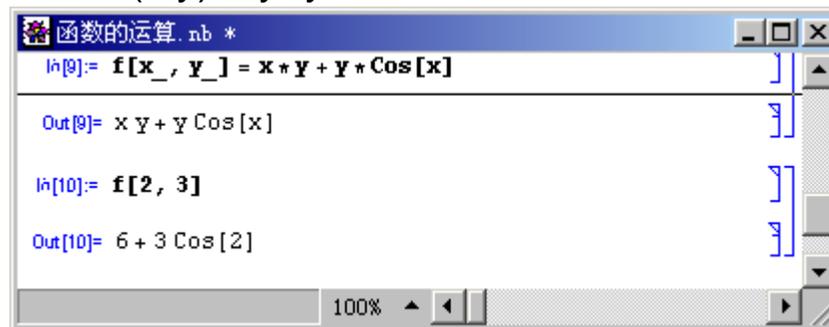
对于定义的函数我们可以使用命令`Clear[f]`清除掉  
而`Remove[f]`则从系统中删除该函数。



## 第2章： Mathematica的基本量

### (2) 多变量函数的定义

也可以定义多个变量的函数，格式为 $f[x_, y_, z_, \dots] = \text{expr}$ 自变量为 $x, y, z, \dots$ ，相应的 $\text{expr}$ 中的自变量会被替换。例如定义数  $f(x, y) = xy + y \cos x$



```
函数的运算.nb *
In[9]:= f[x_, y_] = x*y + y*Cos[x]
Out[9]= x y + y Cos[x]
In[10]:= f[2, 3]
Out[10]= 6 + 3 Cos[2]
```

### (3) 延迟定义函数

延迟定义函数从定义方法上与即时定义的区别为“=”与“:=”延迟定义的格式为 $f[x_] := \text{expr}$ 其他操作基本相同。那么延迟定义和即时定义的主要区别是什么？即时定义函数在输入函数后立即定义函数并存放在内存中并可直接调用。延迟定义只是在调用函数时才真正定义函数。

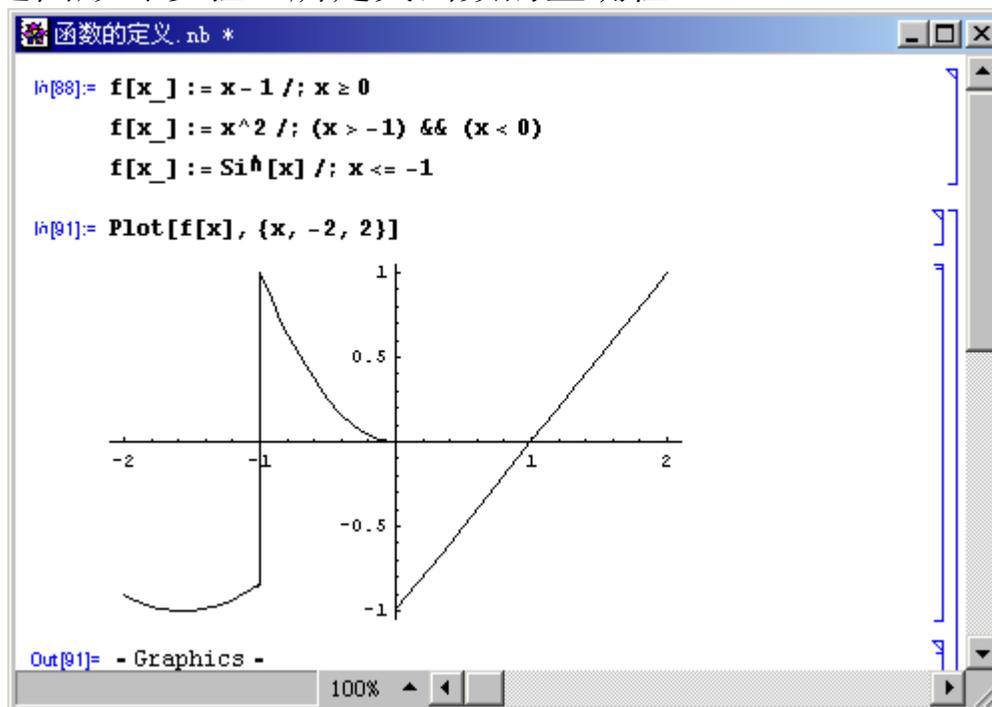
## 第2章： Mathematica的基本量

### (4) 使用条件运算符定义和If命令定义函数

如果要定义如：

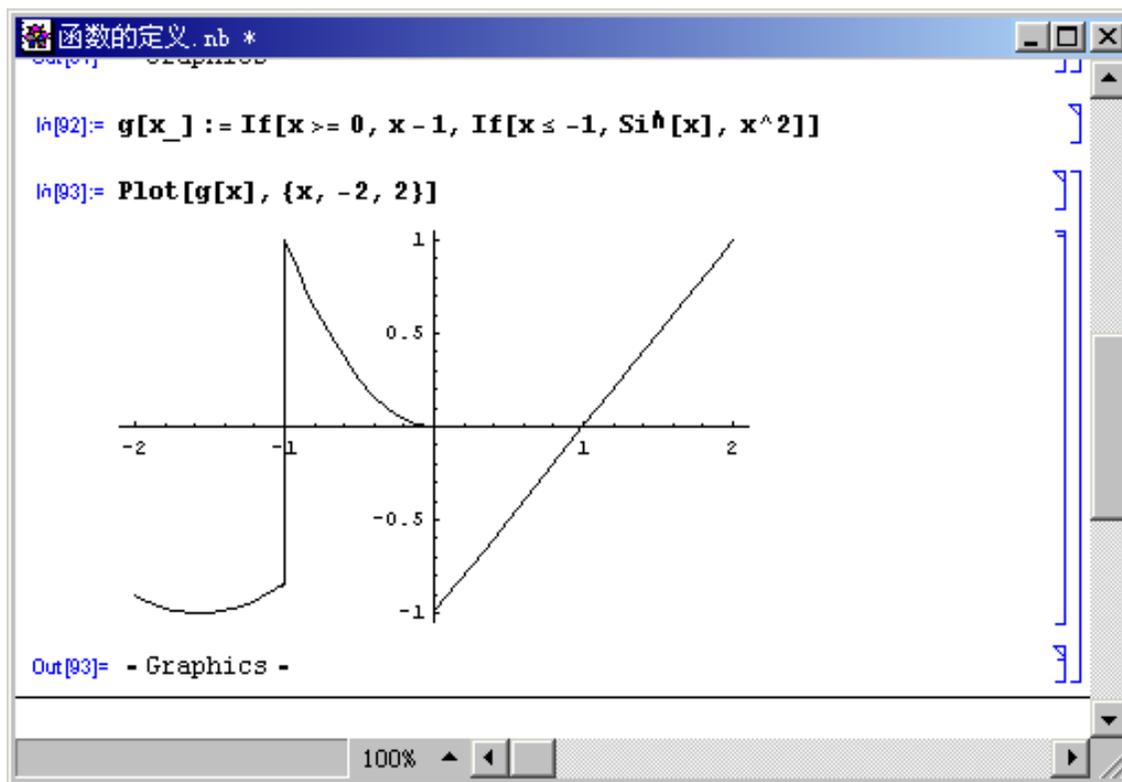
$$f(x) = \begin{cases} x-1 & x > 0 \\ x^2 & 0 \geq x > -1 \\ \text{Sin}x & x \leq -1 \end{cases}$$

这样的分段函数应该如何定义，显然要根据x的不同值给出不同的表达式。一种办法是使用条件运算符，基本格式为f[x\_]:=expr/condition，当condition条件满足时才把expr赋给f。下面定义方法，通过图形可以验证所定义函数的正确性



## 第2章： Mathematica的基本量

当然使用If命令也可以定义上面的函数，If语句的格式为If[条件, 值1, 值2]如果条件成立取“值1”，否则取“值2”，下面用If语句的定义结果



可以看出用If定义的函数 $g(x)$ 和前面函数 $f(x)$ 相同，这里使用了两个If嵌套。

## 第2章： Mathematica的基本量

### 2.4 表

将一些相互关联的元素放在一起，使它们成为一个整体。既可以对整体操作，也可以对整体中的一个元素单独进行操作。在Mathematica中这样的数据结构就称作表（List）。表 {a,b,c} 可以表示一个向量；表 {{a,b},{c,d}}可表示一个矩阵。

#### 1. 建表

在表中元素较少时，可以采取直接列表的方式列出表中的元素，如 {1,2,3} .请看下面的操作

```
In[1]:={1,2,3}
Out[1]={1,2,3}
```

下面是符号表达式的列表

```
Ln[2]:=1+%x+x^%
Out[2]={1+2x,1+2x+x2,1+3x+x2}
```

对列表中的表达式对x求导

```
Ln[3]:=D[%,x]
Out[3]={2,2+2x,3+2x}
Ln[4]:=%/ .x->1
Out[4]={2,4,5}
```

## 第2章: Mathematica的基本量

如果表中的元素较多时, 可以用建表函数进行建表。

<b>Table[f,{l,min,max,step}]</b>	以 <b>step</b> 为步长给出 <b>f</b> 的数值表, <b>i</b> 由 <b>min</b> 变到 <b>max</b> ,
<b>Table[f,{min,max}]</b>	给出 <b>f</b> 的数值表, <b>l</b> 由 <b>min</b> 变到 <b>max</b> 步长为 <b>1</b>
<b>Table[f,max]</b>	给出 <b>max</b> 个 <b>f</b> 的表
<b>Table[f,{l,imin,imax},{j,jmin,jmax},.....]</b>	生成一个多维表
<b>TableForm[list]</b>	以表格格式显示一个表
<b>Range[n]</b>	生成一个 { <b>1,2,.....</b> } 的列表
<b>Range[n1,n2,d]</b>	生成 { <b>n1,n1+d,n1+d,.....,n2</b> } 的列表

下面给出x乘i的值的表, i的变化范围为[2,6]:

```
In[1]:=Table[x*i,{i,2,6}]  
Out[1]={2x,3x,4x,5x,6x}  
In[2]:=Table[x^2,{4}]  
Out[2]={x2,x2,x2,x2}
```

用Range函数生成一个以步长为2, 范围从8到20序列数

```
In[4]:=Range[8,20,2]  
Out[4]={8,10,12,14,16,18,20}
```

## 第2章： Mathematica的基本量

多个参数的表，下面生成一个多维表：

```
In[5]:=Table[2i+j,{i,1,3},{j,3,5]}
Out[5]={{5,6,7},{7,8,9},{9,10,11}}
```

使用函数TableForm可以以表格的方式输出

```
Ln[6]:=%/TableForm
Out[6]//TableForm=5 6 7
7 8 9
9 10 11
```

### 2. 表的元素的操作

当t表示一个表时，t[[i]]表示t中的第i个子表。如果t={1,2,a,b}那么t[[3]]表示“a”。如：

```
In[1]:=t=Table[l+2j, {l,1,3}, {j,3,5}]
Out[1]={{7,9,11},{8,10,12},{9,11,13}}
In[2]:=t[[2]]
Out[2]={8,10,12}
```

Mathematica提供了表操作的丰富的函数

## 第2章： Mathematica的基本量

### 2. 5 表达式

#### 1. 表达式的含义

Mathematica 能处理数学公式，表以及图形等多种数据形式。尽管他们从形式上看起来不一样，但在Mathematica内部都被看成同种类型，即都把他們当作表达式的形式。

Mathematica 中的表达式是由常量、变量、函数、命令、运算符和括号等组成，他最典型的形式是f[x,y]

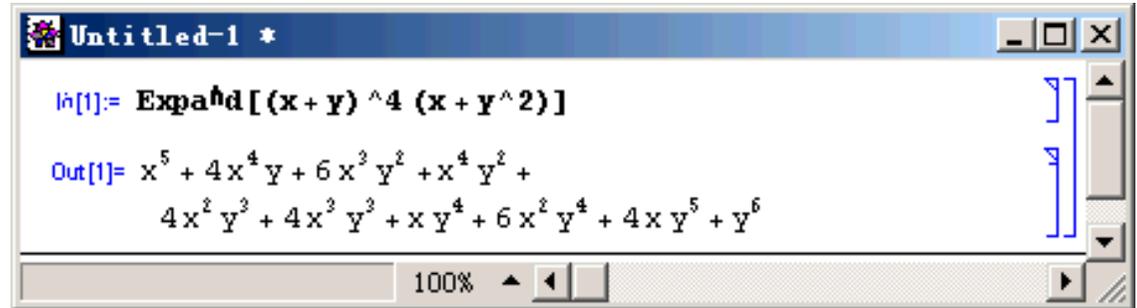
#### 2. 表达式的表示形式

在显示表达式时，由于需要的不同，有时我们需要表达式的展开形式，有时又需要其因子乘积的形式。在我们计算过程中可能得到很复杂的表达式，这时我们又需要对它们进行化简。常用的处理这种情况的函数。变换表达式表示形式函数

表达式表示形式函数	意义
Expand (expr)	按幂次升高的顺序展开表达式
Factor (expr)	以因子乘积的形式表示表达式
Simplify (expr)	进行最佳的代数运算，并给出表达式的最少项形式

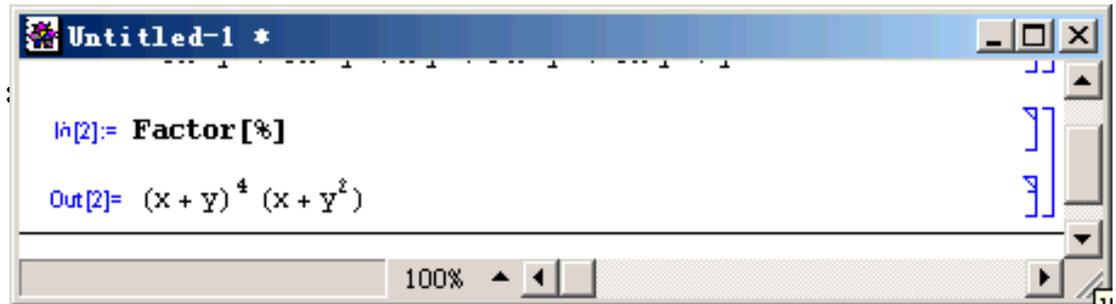
## 第2章： Mathematica的基本量

表达式  $(x + y)^4 (x + y^2)$  展开：



```
Untitled-1 *
In[1]:= Expand[(x + y)^4 (x + y^2)]
Out[1]= x^5 + 4x^4y + 6x^3y^2 + x^4y^2 +
        4x^2y^3 + 4x^3y^3 + xy^4 + 6x^2y^4 + 4xy^5 + y^6
```

还原上面的表达式为因子乘积的形式：



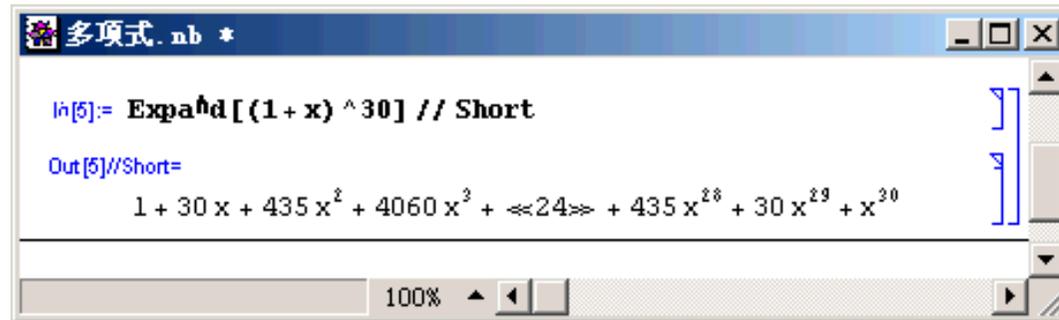
```
Untitled-1 *
In[2]:= Factor[%]
Out[2]= (x + y)^4 (x + y^2)
```

多项式表达式的项数较多，比较复杂，在显示时显得比较杂乱，而且在计算过程中没有必要知道全部的内容；或表达式的项很有规律，没有必要打印全部的表达式的结果，Mathematica提供了一些命令，可将它缩短输出或不输出

命令	意义
command	执行命令command，屏幕上不显示结果
expr/Short	显示表达式的一行形式
Short (expr, n)	显示表达式的n行形式命令后加一分号“;” 不打印结果

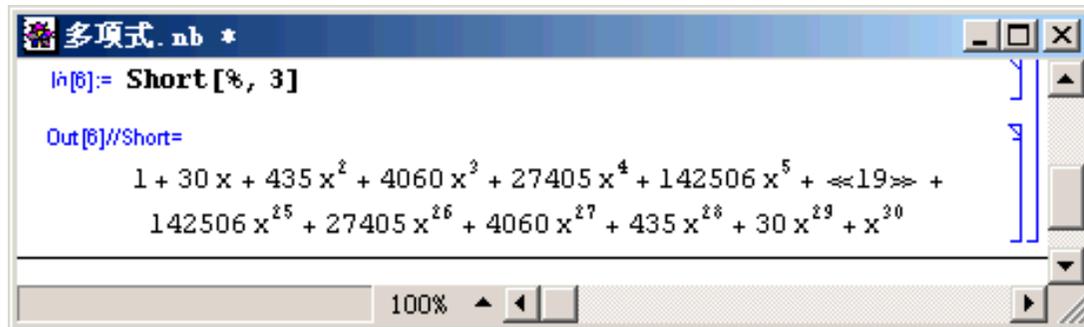
## 第2章： Mathematica的基本量

将表达式  $(1+x)^{30}$  展开，并仅显示一行有代表项的式子：



```
多项式.nb *
In[5]:= Expand[(1 + x) ^ 30] // Short
Out[5]//Short=
1 + 30 x + 435 x2 + 4060 x3 + <<24>> + 435 x28 + 30 x29 + x30
```

将上式分成三行的形式展开：



```
多项式.nb *
In[6]:= Short[%, 3]
Out[6]//Short=
1 + 30 x + 435 x2 + 4060 x3 + 27405 x4 + 142506 x5 + <<19>> +
142506 x25 + 27405 x26 + 4060 x27 + 435 x28 + 30 x29 + x30
```

把代数表达式变换到你所需要的形式没有一种固定的模式，一般情况下，最好的办法是进行多次实验，尝试不同的变换并观察其结果，再挑出你满意的表示形式。

## 第2章： Mathematica的基本量

### 3. 关系表达式与逻辑表达式

我们已经知道“=”表示给变量赋值。现在我们来学习一些其它的逻辑与关系算子。关系表达式是最简单的逻辑表达式，我们常用关系表达式表示一个判别条件。例如： $x>0,y=0$ 。关系表达式的一般形式是：表达式+关系算子+表达式。其中表达式可为数字表达式、字符表达式或意义更广泛的表达式，如一个图形表达式等。在我们实际运用中，这儿的表达式常常是数字表达式或字符表达式。下面出Mathematica中的各种关系算子。

$x==y$	相等
$x!=y$	不相等
$x>y$	大于
$x>=y$	大于或等于
$x<y$	小于
$x<=y$	小于等于
$x==y==z$	都相等
$x!=y!=z$	都不相等
$x>y>z, etc$	严格递减

## 第2章： Mathematica的基本量

给变量x,y赋值，输出后以变量的值，如：

```
In[1]:=x=2;y=9
Out[1]=9;
In[2]:=x>y
Out[2]=false;
```

比较两个表达式的大小

```
Ln[3]:=3^2>y+1
Out[3]=True
```

用一个关系式只能表示一个判定条件，要表示几个判定条件的组合，必须用逻辑运算符将关系表达式组织在一起，我们称表示判定条件的表达式为**逻辑表达式**。

常用的逻辑运算和它们的意义

!	非
&&	并
	或
Xor	异或
If	条件

例如

```
In[4]:=3x^2<Y+1&&3^2==y
Out[4]=false
In[5]:=3x^2+1||3^2==y
Out[5]=True
```

## 第2章： Mathematica的基本量

### 2. 6 常用的符号

(term)	圆括号用于组合运算
f[x]	方括号用于函数
{ }	花括号用于列表
[[i]]	双括号用于排序
%	代表最后产生的结果
%%	倒数第二次的算结果
%%% (k)	倒数第k次的计算结果
%n	例出行Out[n])的结果 (用时要小心)

## 第3章： Mathematica的基本运算

### 3.1 多项式的表示形式

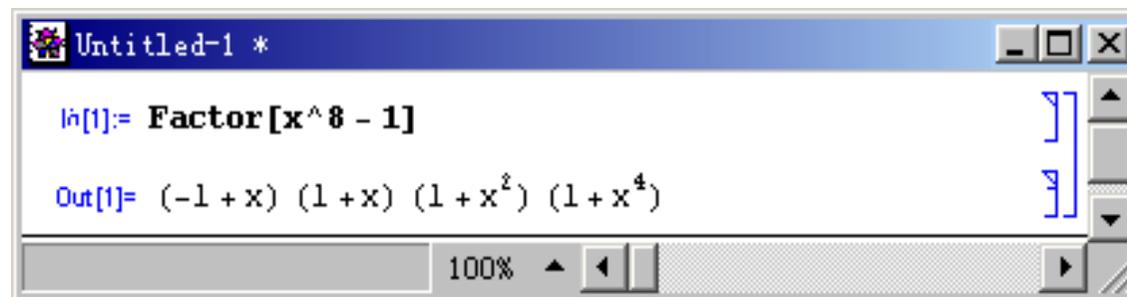
可认为多项式是表达式的一种特殊的形式，所以多项式的运算与表达式的运算基本一样，表达式中的各种输出形式也可用于多项式的输出。**Mathematica**提供一组按不同形式表示代数式的函数。

一些常用指令：

Expand[poly]	按幂次展开多项式poly
Expand[poly]	全部展开多项式poly
ExpandAll[poly]	全部展开多项式poly
Factor[poly]	对多项式poly 进行因式分解
FactorTerms[poly, {x, y, ...}]	按变量 x, y, ...进行分解
Simplify[poly]	把多项式化为最简形式
FullSimplify[poly]	把多项式展开并化简
Collect[poly, x]	把多项式poly按x幂展开
Collect[poly, {x, y...}]	把多项式poly按x, y.... 的幂次展开

例子

对 $x^8-1$  进行分解



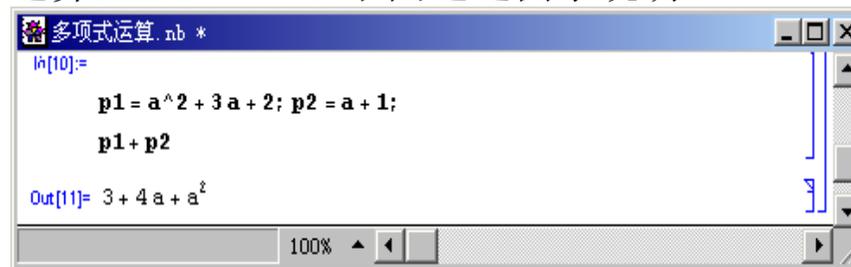
```
Untitled-1 *
In[1]:= Factor[x^8 - 1]
Out[1]= (-1 + x) (1 + x) (1 + x^2) (1 + x^4)
```

# 第3章： Mathematica的基本运算

## 2.多项式的代数运算

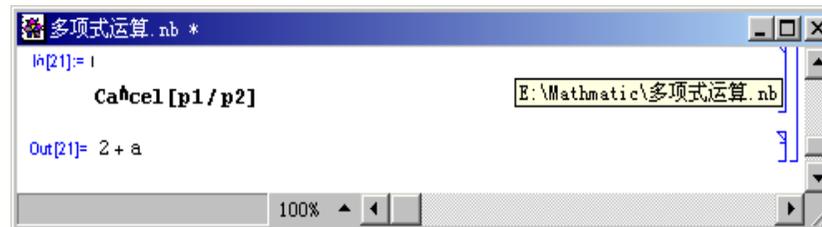
多项式的运算有加、减、乘、除运算： $+$ ， $-$ ， $*$ ， $/$ 下面通过例子说明。

多项式的加运算 $a^2+3a+2$ 与 $a+1$ 相加



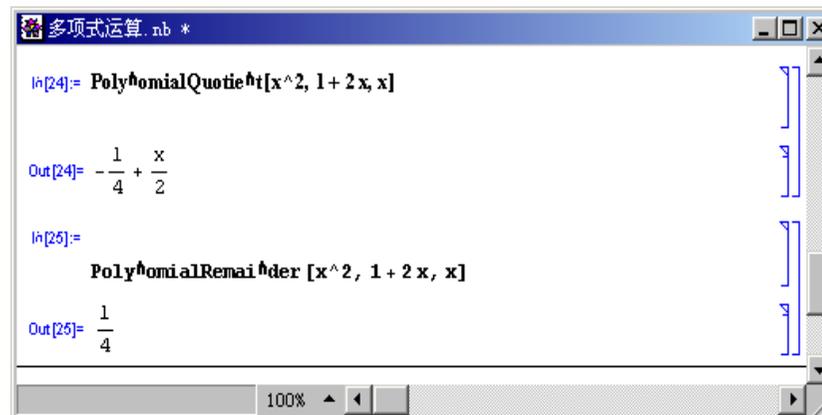
```
多项式运算.nb *
In[10]:=
p1 = a^2 + 3 a + 2; p2 = a + 1;
p1 + p2
Out[11]= 3 + 4 a + a^2
```

使用Cancel函数可以约去公因式



```
多项式运算.nb *
In[21]:= 1
Cancel[p1/p2]
Out[21]= 2 + a
```

两个多项式相除，总能写成一个多项式和一个有理式相加Mathematic中提供两个函数PolynomialQuotient和PolynomialRemainder分别返商式和余式。例如：



```
多项式运算.nb *
In[24]:= PolynomialQuotient[x^2, 1 + 2 x, x]
Out[24]= -1/4 + x/2
In[25]:= PolynomialRemainder[x^2, 1 + 2 x, x]
Out[25]= 1/4
```

## 第3章： Mathematica的基本运算

### 3. 2 方程及其根的表达

Mathematica把方程看作逻辑语句。在数学方程式表示为形如“ $x^2-2x+1=0$ ”的形式。在Mathematica中“=”用作赋值语句，这样在Mathematica中用“**==**”表示逻辑等号，则方程应表示为“ $x^2-2x+1==0$ ”。方程的解同原方程一样被看作是逻辑语句。

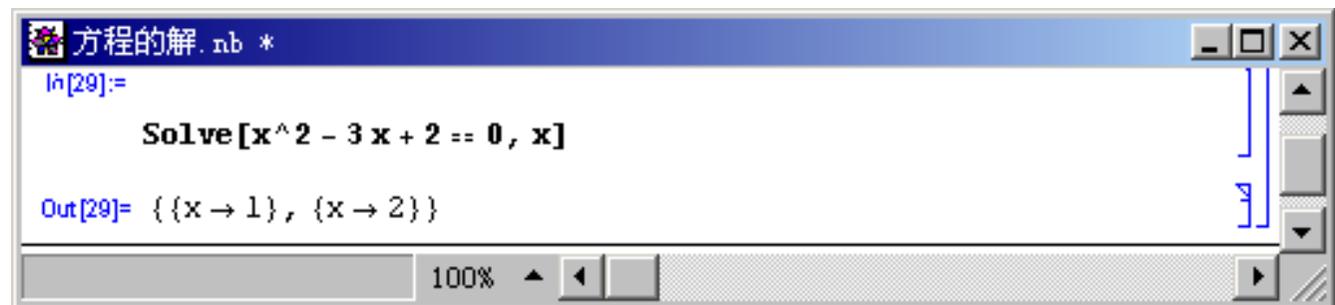
用**Roots**求方程 $x^2-3x+2$ 的根

这种表示形式说明 $x$ 取1或2均可



```
方程的解.nb *
In[28]:= Roots[x^2 - 3 x + 2 == 0, x]
Out[28]= x == 1 || x == 2
```

用**Solve[ ]**可得解集形式



```
方程的解.nb *
In[29]:= Solve[x^2 - 3 x + 2 == 0, x]
Out[29]= {{x -> 1}, {x -> 2}}
```

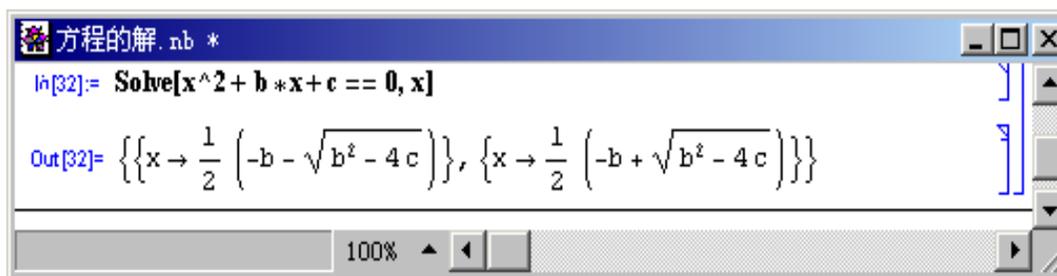
## 第3章： Mathematica的基本运算

### 1.求解一元代数方程

常用的一些方程求解函数

<code>Solve[lhs==rhs, vars]</code>	给出方程的解集
<code>NSolve[lhs==rhs, vars]</code>	直接给出方程的数值解集
<code>Roots[lhs==rhs, vars]</code>	求表达式的根
<code>FindRoot[lhs==rhs, {x, x0}]</code>	求 $x=x_0$ 时，方程的解值

Solve函数例子



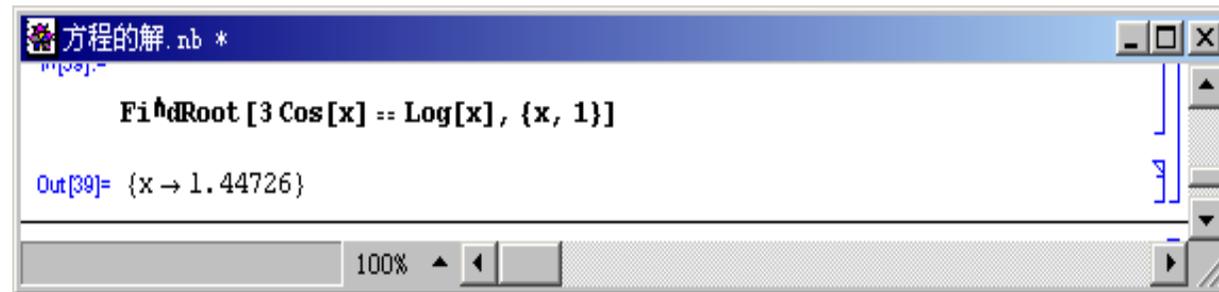
```
方程的解.nb *
In[32]:= Solve[x^2 + b * x + c == 0, x]
Out[32]= {{x -> 1/2 (-b - Sqrt[b^2 - 4 c])}, {x -> 1/2 (-b + Sqrt[b^2 - 4 c])}}
```

Solve函数可处理的主要方程是多项式方程。Mathematica总能对不高于四次的方程进行精确求解，对于三次或四次方程，解的形式可能很复杂。

## 第3章： Mathematica的基本运算

当方程中有一些复杂的函数时，Mathematica可能无法直接给出解来。在这种情况下我们可用 `FindRoot[]` 来求解，但要给出起始条件。

求  $3\cos x = \log x$  的解

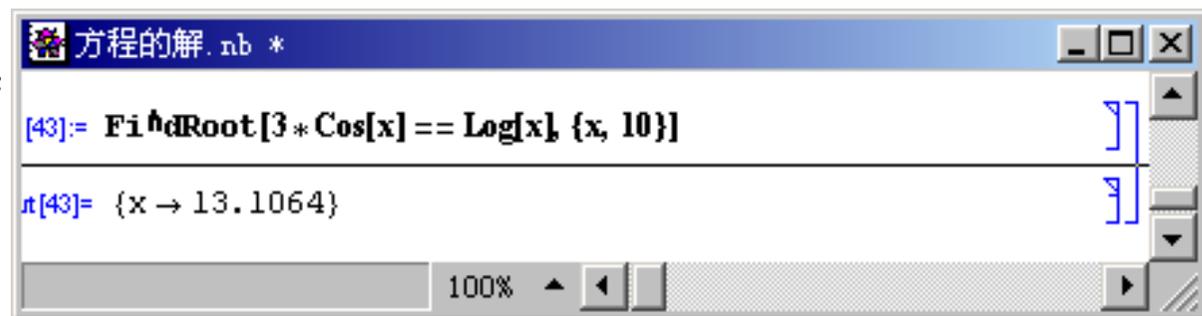


```
FindRoot[3 Cos[x] == Log[x], {x, 1}]
```

Out[39]= {x → 1.44726}

只能求出  $x=1$  附近的解，如果方程有几个不同的解，当给定不同的条件时，将给出不同的解。

上例若求  $x=10$  附近的解命令为：



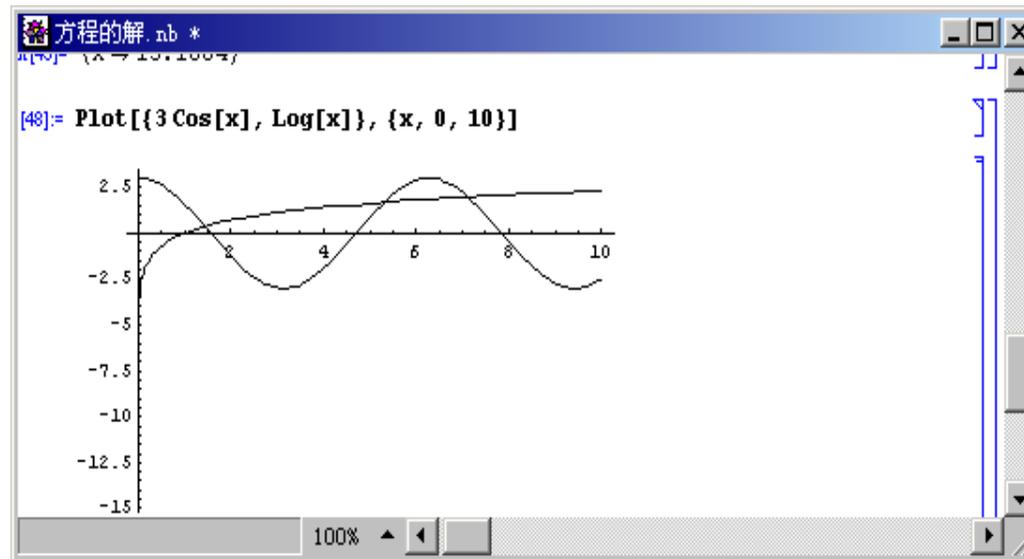
```
[43]:= FindRoot[3 * Cos[x] == Log[x], {x, 10}]
```

Out[43]= {x → 13.1064}

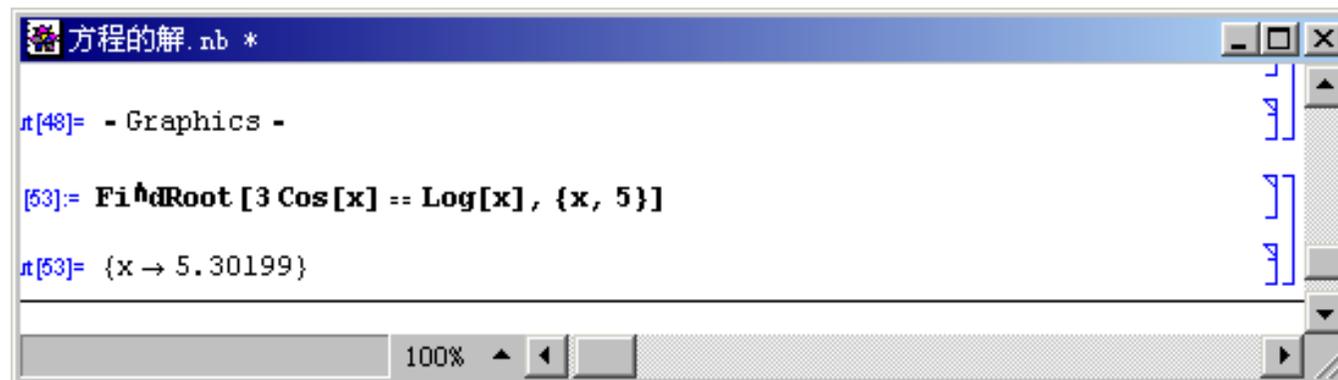
因此确定解的起始位置是比较关键

## 第3章： Mathematica的基本运算

一种常用的方法是，先绘制图形观察后再解



如上例通过图形可断定在 $x=5$ 附近有另一根



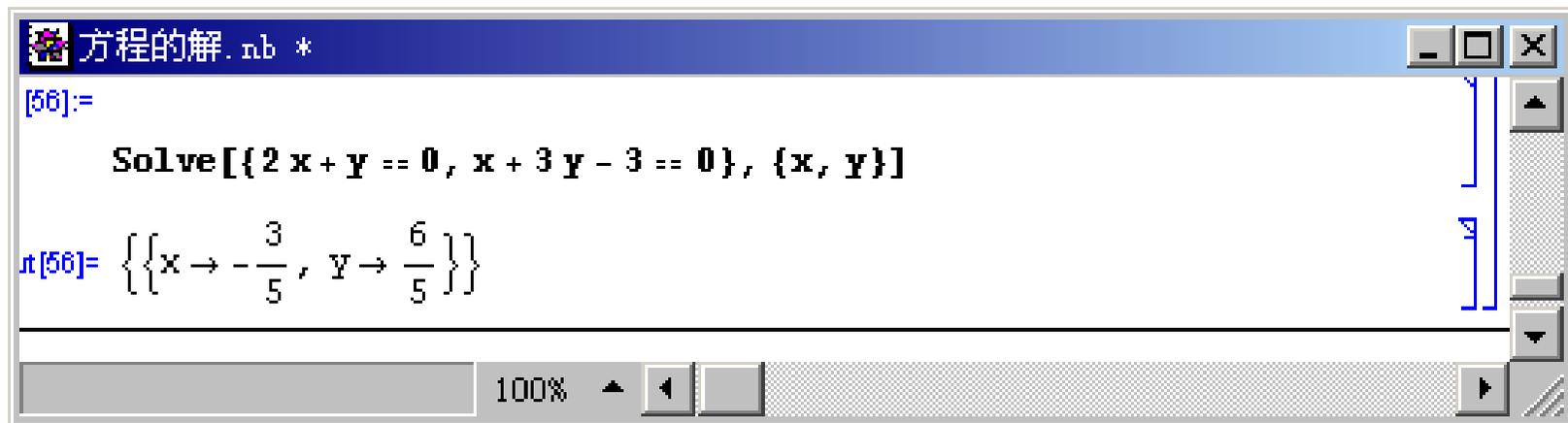
## 第3章： Mathematica的基本运算

### 2.求方程组的根

使用Solve和NSolve， FindRoot也可求方程组的解，只是使用时格式略有不同下面给出一个Solve函数的例子：

求解：

$$\begin{cases} 2x + y = 0 \\ x + 3y - 3 = 0 \end{cases}$$

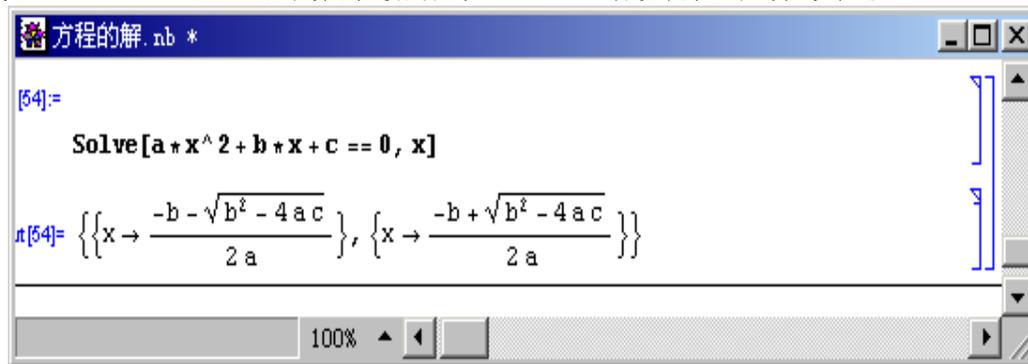


The screenshot shows a Mathematica notebook window titled "方程的解.nb \*". The input cell contains the command `Solve[{2 x + y == 0, x + 3 y - 3 == 0}, {x, y}]`. The output cell shows the result: `{{x -> -3/5, y -> 6/5}}`. The notebook interface includes standard window controls (minimize, maximize, close) and a status bar at the bottom showing "100%" zoom level.

## 第3章： Mathematica的基本运算

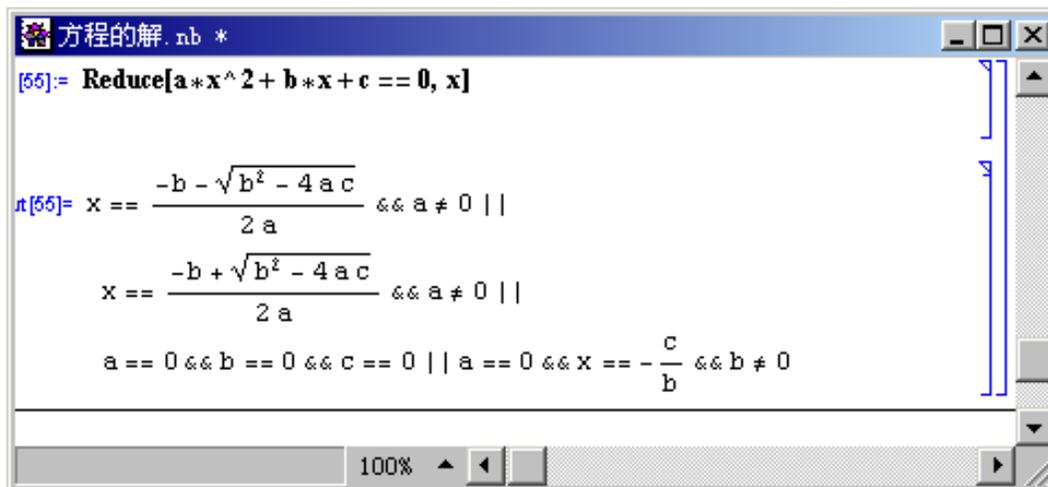
### 3.求方程的全解， Reduce命令

如果我们求 $ax^2+bx+c=0$ 的根我们用Solve函数解的结果是：



```
方程的解.nb *
[54]:=
Solve[a*x^2 + b*x + c == 0, x]
Out[54]= {{x -> (-b - Sqrt[b^2 - 4ac])/2a}, {x -> (-b + Sqrt[b^2 - 4ac])/2a}}
```

这显然是不合理的，因为对不同的 $a,b,c$ 方程的解有不同的情况，而上面只是给出部分解如果要解决这个问题可用Reduce命令，它可根据 $a,b,c$ 的取值给出全部值。



```
方程的解.nb *
[55]:= Reduce[a*x^2 + b*x + c == 0, x]
Out[55]= x == (-b - Sqrt[b^2 - 4ac])/2a && a != 0 ||
x == (-b + Sqrt[b^2 - 4ac])/2a && a != 0 ||
a == 0 && b == 0 && c == 0 || a == 0 && x == -c/b && b != 0
```

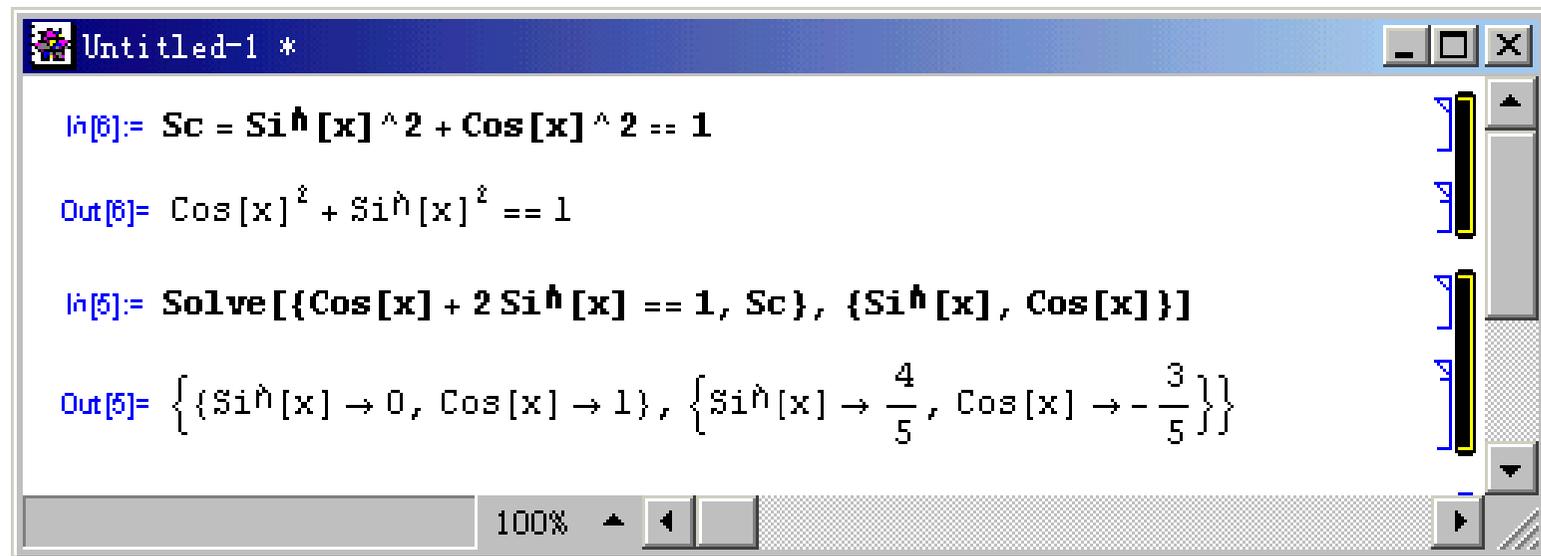
因此Solve,Roots只给出方程的一般解，而Reduce函数数可以给出方程的全部可能解。

## 第3章： Mathematica的基本运算

### 4.解辅助条件方程

在作方程计算时，可以把一个方程看作你要处理的主要方程，而把其他方程作为必须满足的辅助条件，你将会发现这样处理很方便。譬如在求解像  $x^4+bx^2+c=0$  这样的方程时，通常我们采用  $X^2=y$  的代换方法使求解方程得到简化。在Mathematica中，我们通常是首先命名辅助条件组，然后用名字把辅助条件包含在你要用函数Solve[] 求解的方程组中。

例：用Sc定义方程  $\sin^2x+\cos^2x=1$ ，在这种条件下，求解方程。



```
Untitled-1 *  
  
In[8]:= Sc = Sin[x]^2 + Cos[x]^2 == 1  
  
Out[8]= Cos[x]^2 + Sin[x]^2 == 1  
  
In[9]:= Solve[{Cos[x] + 2 Sin[x] == 1, Sc}, {Sin[x], Cos[x]}]  
  
Out[9]= {{Sin[x] -> 0, Cos[x] -> 1}, {Sin[x] -> 4/5, Cos[x] -> -3/5}}
```

## 第3章： Mathematica的基本运算

### 3.3 求和与求积

在Mathematica中，数学上的各式符号  $\sum$  用Sum表示，连乘  $\prod$  用Product表示。下面列出求各与求积函数的形式和意义：

Sum[f, {i, imin, imax}] 求和  $\sum_{i=imin}^{imax} f$

Sum[f, {i, imin, imax, di}] 以步长di增加i求和

Sum[f, {i, imin, imax}, {j, jmin, jmax}] 嵌套求和  $\sum_{i=imin}^{imax} \sum_{j=jmin}^{jmax} f$

Product[f, {i, imin, imax}] 求积  $\prod_{i=imin}^{imax} f$

Product [f, {i, imin, imax, di}] 以步长di增加i求和

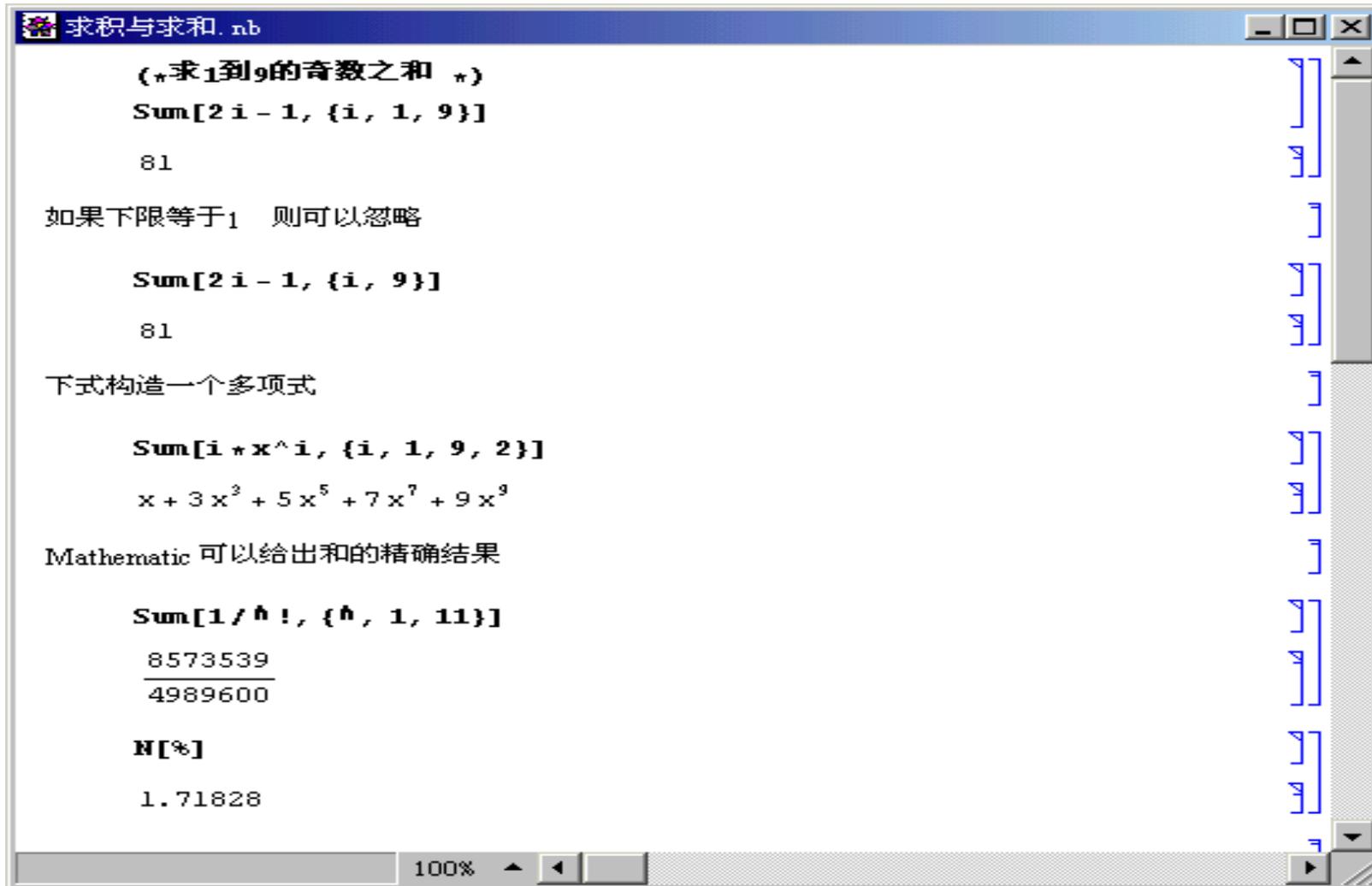
Product[f, {I, imin, imax}, {j, jmin, jmax}] 嵌套求积  $\prod_{i=imin}^{imax} \prod_{j=jmin}^{jmax} f$

Nsum[f, {i, imin, Infinity}] 求  $\sum_{i=imin}^{\infty} f$  近似值

NProduct[f, {i, imin, Infinity}] 求  $\prod_{i=imin}^{\infty} f$  近似值

## 第3章： Mathematica的基本运算

一些例子



```
求积与求和.nb

(*求1到9的奇数之和 *)
Sum[2 i - 1, {i, 1, 9}]

81

如果下限等于1 则可以忽略

Sum[2 i - 1, {i, 9}]

81

下式构造一个多项式

Sum[i * x^i, {i, 1, 9, 2}]

x + 3 x2 + 5 x5 + 7 x7 + 9 x9

Mathematic 可以给出和的精确结果

Sum[1 / h!, {h, 1, 11}]

8573539
-----
4989600

N[%]

1.71828
```

## 第4章： Mathematica的函数作图

### 4.1 基本的二维图形

Mathematica在直角坐标系中作一元函数图形用下列基本命令。

**Plot[f, {x, xmin, xmax}, option->value]**

在指定区间上按选项定义值画出函数在直角坐标系中的图形。

**Plot[{f1,f2,f3,...},{x,xmin,xmax},option->value]**

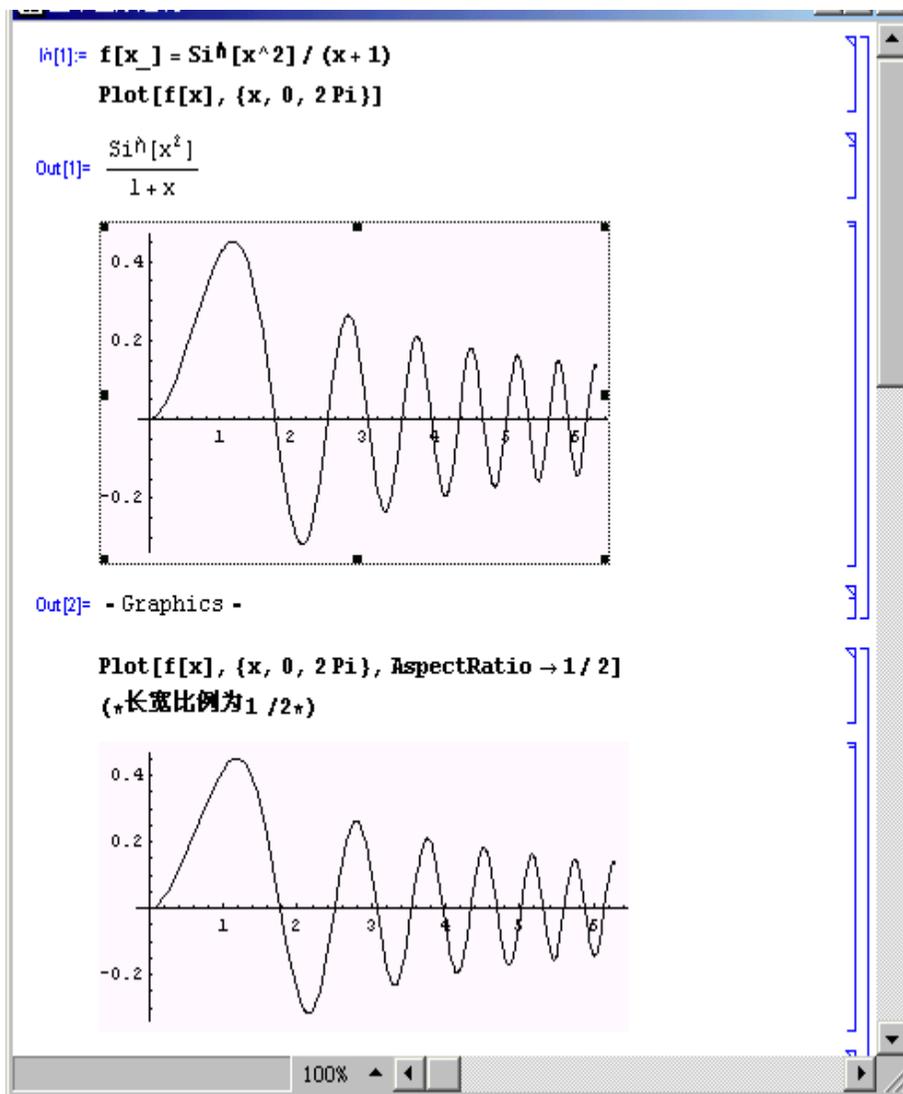
在指定区间上按选项定义值同时画出多个函数在直角坐标系中的图形。

Mathematica绘图时允许用户设置选项值对绘制图形的细节提出各种要求。例如：要设置图形的高宽比，给图形加标题等。每个选项都有一个确定的名字，以“选项名->选项值”的形式放在Plot中的最右边位置，一次可设置多个选项，选项依次排列，用逗号隔开，也可以不设置选项，采用系统的默认值。

选项	默认值	说明
AspectRatio	1/0.618	图形的高、宽比
AxesLabel	不加	给坐标轴加上名字
PlotLabel	不加	给图形加上标题
PlotRange	计算的结果	指定函数因变量的区间
PlotStyle	值是一个表	用什么样方式作图（颜色，粗细等）
PlotPoint	25	画图时计算的点数

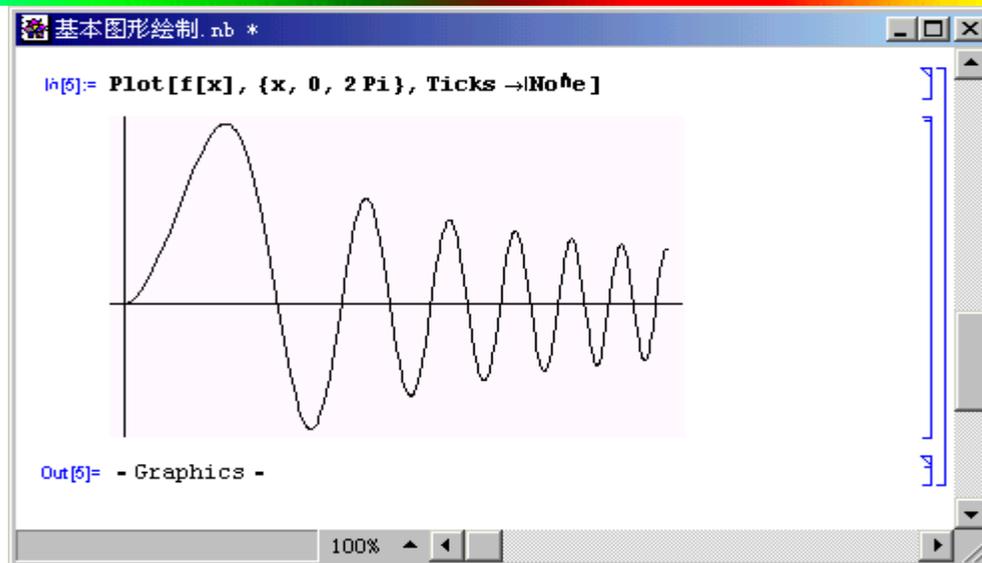
## 第4章： Mathematica的函数作图

(1) 例如绘制  $f(x) = \frac{\sin x^2}{x+1}$  的图形。

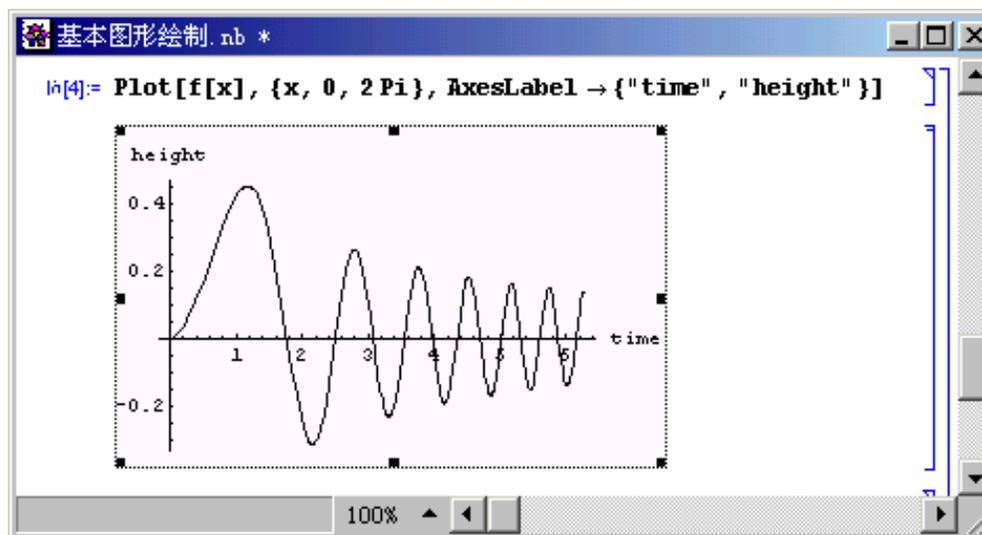


## 第4章： Mathematica的函数作图

(2). 如果要取消刻度可以使用**Ticks**选项

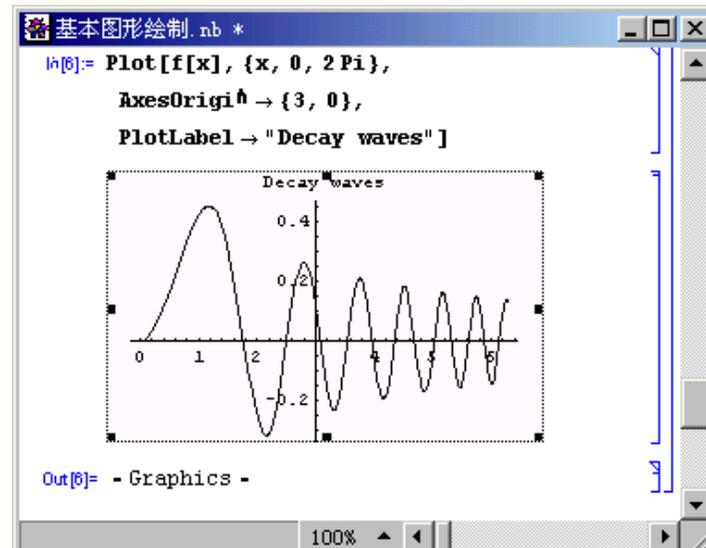


(3). 如果要标注坐标名称 x轴为“Time”,y轴为“Height”

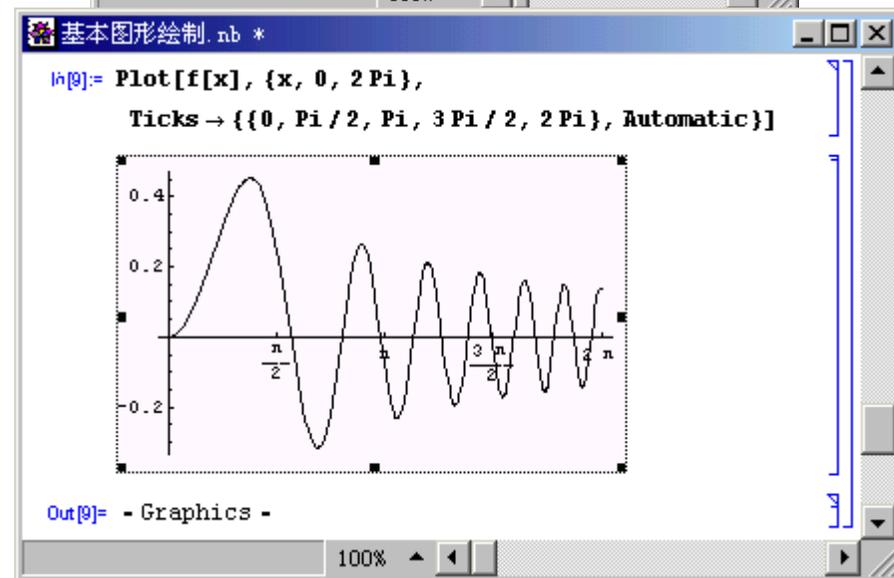


## 第4章： Mathematica的函数作图

(4). 将坐标原点移至  $(3, 0)$ ，并标注图形名称。

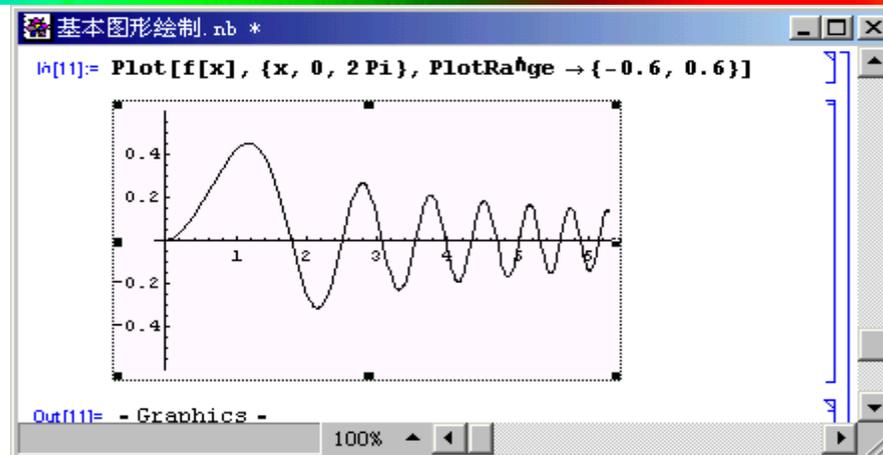


(5). 修改x方向的刻度，y轴方向的刻度  
则用默认值。

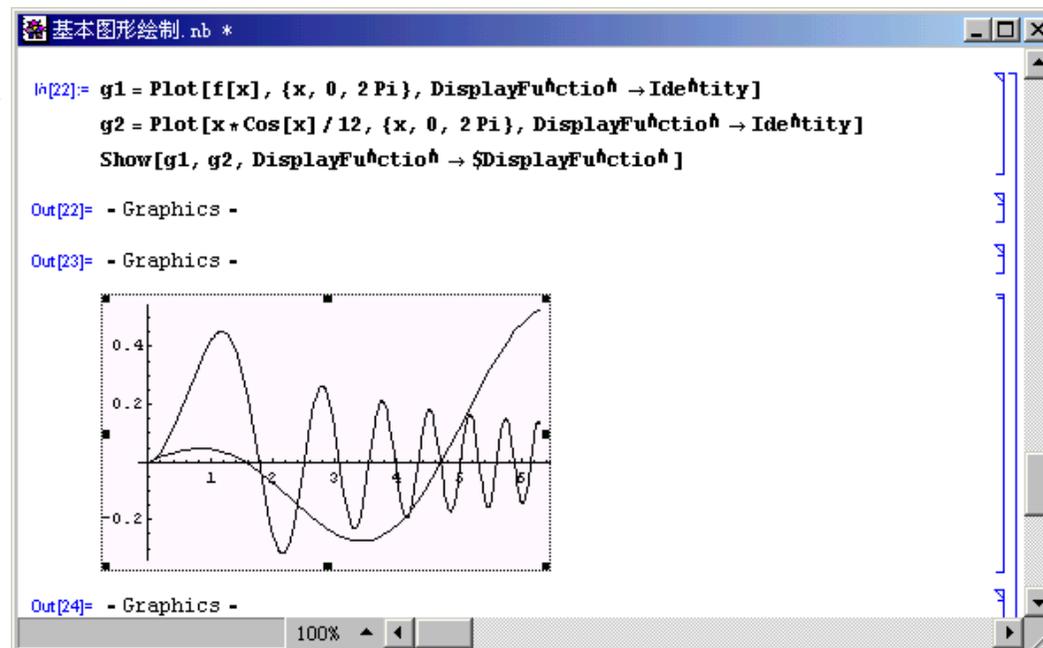


## 第4章： Mathematica的函数作图

(6). 定义y轴的绘图范围



(7). 另外我们也可以将图形结果定义给变量，但不显示图形，后用Show命令显示。



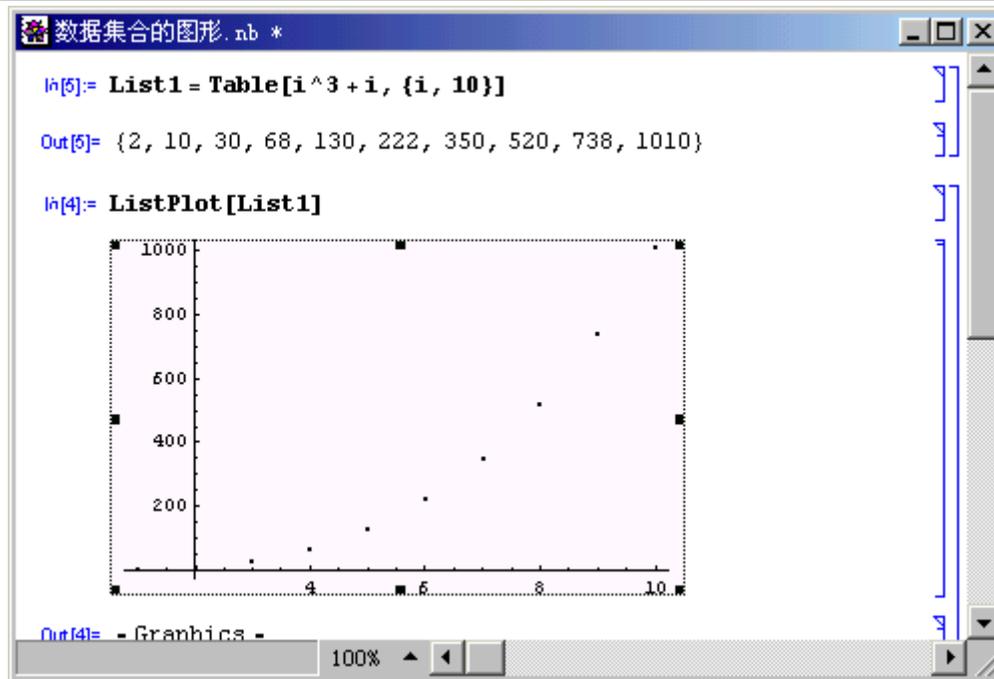
## 第4章： Mathematica的函数作图

### 2.数据集合的图形

Mathematica用于绘数字集合的图形的命令与前而介绍的绘函数图形的命令是相似的。如下：

<code>ListPlot[{y1, y2, .....}]</code>	绘出在x的值为1, 2...时y1, y2, ...的图形
<code>ListPlot[{{x1, y1}, {x2, y2}, .....}]</code>	绘出离散点 (xi, yi)
<code>ListPlot[List, PlotJoined-&gt;True]</code>	把离散点连成曲线

举例说明下面是一个离散数据的集合的图形



# 第4章： Mathematica的函数作图

## 3.二维参数作图 ParametricPlot

`ParametricPlot[{fx, fy}, {t, tmin, tmax}]`

绘出参数图

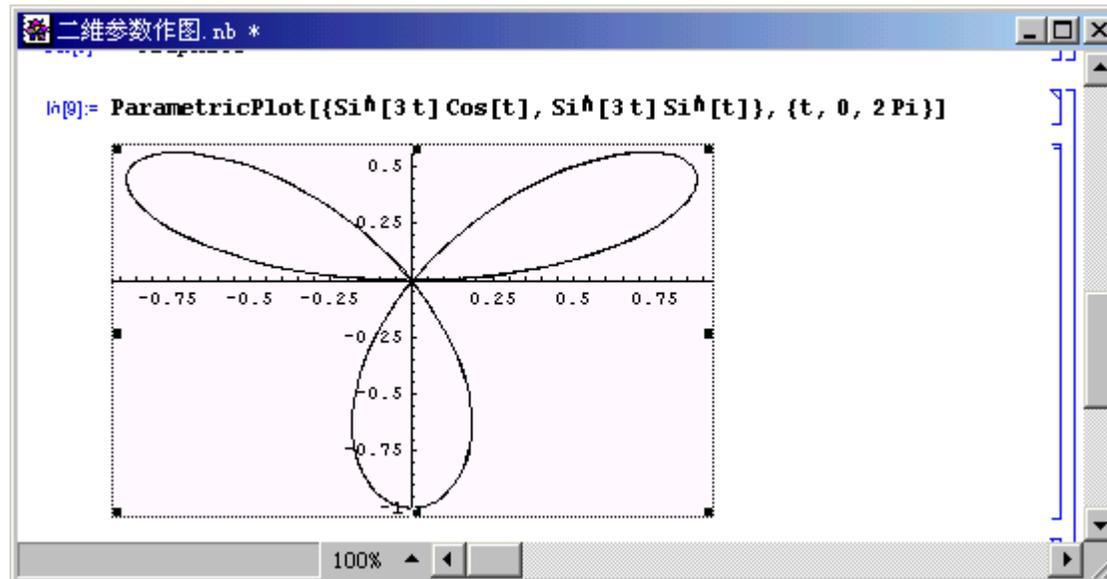
`ParametricPlot[{fx, fy}, {gx, gy}, ..., {t, tmin, tmax}]`

绘出一组参数图

`ParametricPlot[{fx, fy}, {t, tmin, tmax}, AspectRatio->Automatic]`

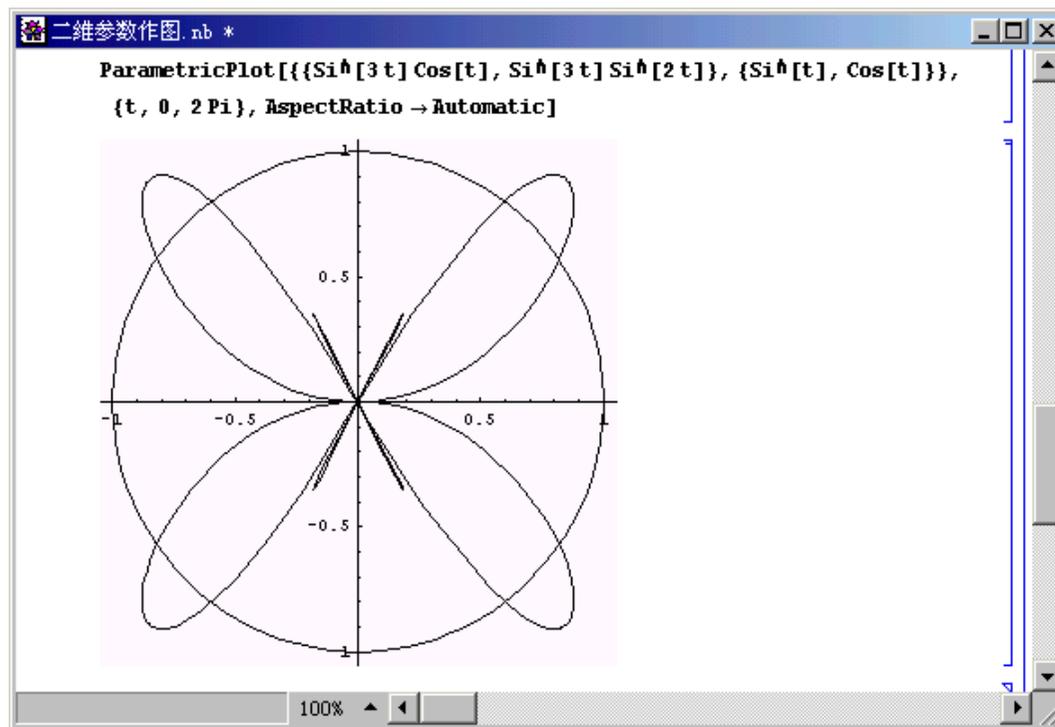
设法保持曲线的形

(1). 绘制参数方程  $\begin{cases} x = \sin 3t \cos t \\ y = \sin 3t \sin t \end{cases}$  的图形



## 第4章： Mathematica的函数作图

(2). 下面将一个园与上面参数绘在同一个坐标下，并保证图形的形状正确。



## 第4章： Mathematica的函数作图

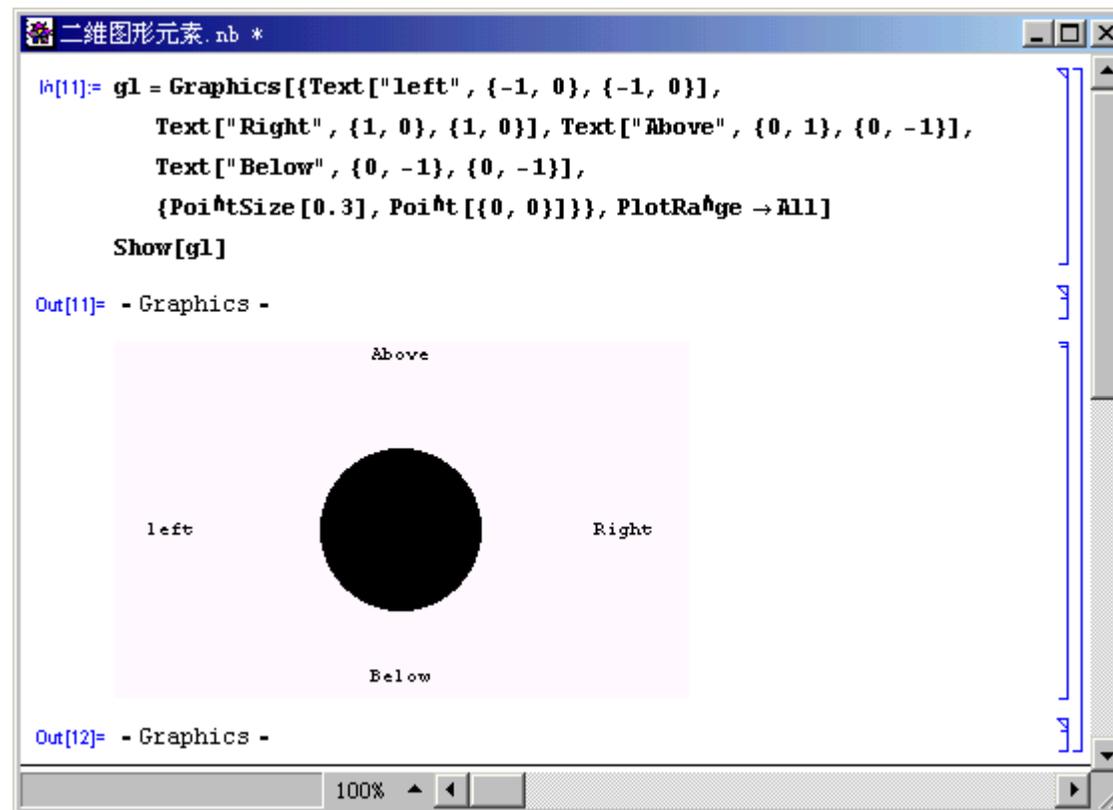
### 4. 2 二维图形元素

用图形元素绘图适合于绘制结构复杂的图形。Mathematica中还提供了各种如绘制点、线段、圆弧等函数。同样我们可先用Graphics作出平面图形的表达式，再用Show显示守成的图形。

<code>Point[{x, y}]</code>	点
<code>Line[{{x1, y1}, {x2, y2}, ...}]</code>	线段
<code>Rectangle[{xmin, ymin}, {xmax, ymax}]</code>	填充矩阵
<code>Polygon</code>	<code>[{{x1, y1}, {x2, y2}, .....}]</code> 填充多边形
<code>Circle[{x, y}, r]</code>	圆
<code>Circle[{x, y}, {rx, ry}]</code>	半轴分别为rx, ry的椭圆
<code>Circle[{x, y}, r, {theta1, theta2}]</code>	圆弧
<code>Circle[{x, y}, {rx, ry}, {theta1, theta2}]</code>	椭圆弧
<code>Disk[{x, y}, r]</code>	填充圆
<code>Raster[{{a11, a12, ....}, {a21, .....}, ....}]</code>	灰度在0到1之间的灰层组
<code>Text[Expr, {x, y}]</code>	文本大小

## 第4章： Mathematica的函数作图

下图绘出一个有颜色和大小 的点，且在图形四周插入文本

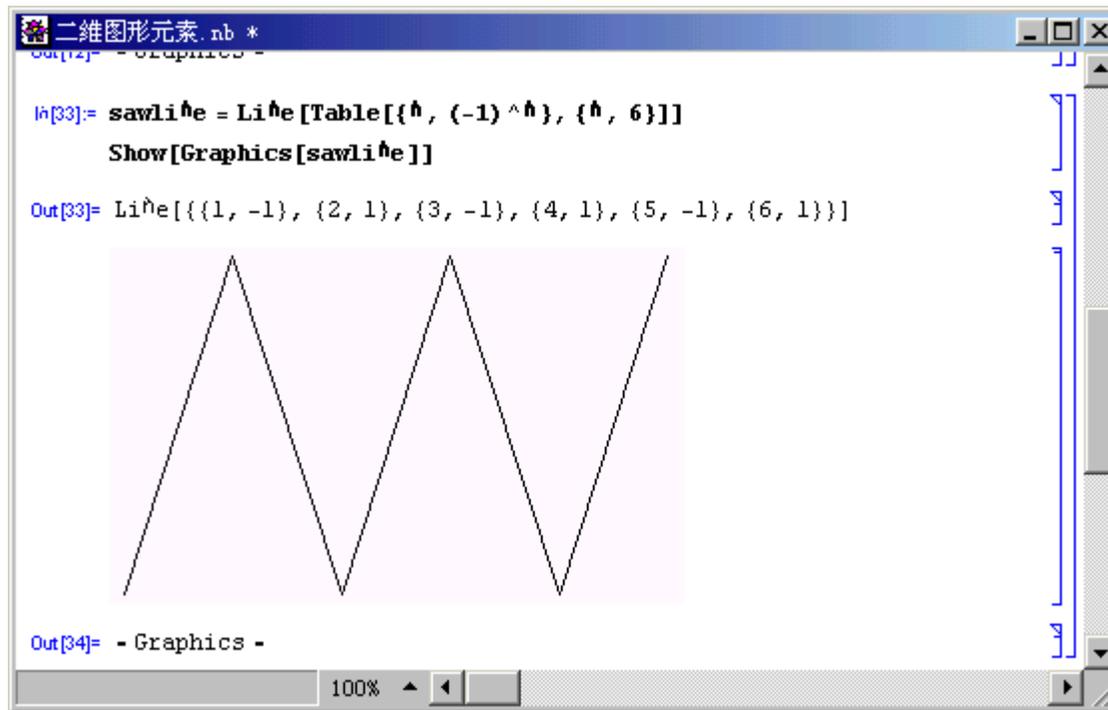


```
二維图形元素.nb *
In[11]:= g1 = Graphics[{Text["left", {-1, 0}, {-1, 0}],
    Text["Right", {1, 0}, {1, 0}], Text["Above", {0, 1}, {0, -1}],
    Text["Below", {0, -1}, {0, -1}],
    {PointSize[0.3], Point[{0, 0}]}], PlotRange -> All]
Show[g1]
Out[11]= - Graphics -
Out[12]= - Graphics -
```

The screenshot shows a Mathematica notebook window titled "二維图形元素.nb \*". The input cell contains the following code: `In[11]:= g1 = Graphics[{Text["left", {-1, 0}, {-1, 0}], Text["Right", {1, 0}, {1, 0}], Text["Above", {0, 1}, {0, -1}], Text["Below", {0, -1}, {0, -1}], {PointSize[0.3], Point[{0, 0}]}], PlotRange -> All] Show[g1]`. The output cell shows `Out[11]= - Graphics -` and a plot of a black circle centered at the origin. The plot has a light pink background and the words "left", "right", "above", and "below" are placed around the circle. The plot range is set to "All". The notebook interface includes a scroll bar on the right and a status bar at the bottom showing "100%" zoom.

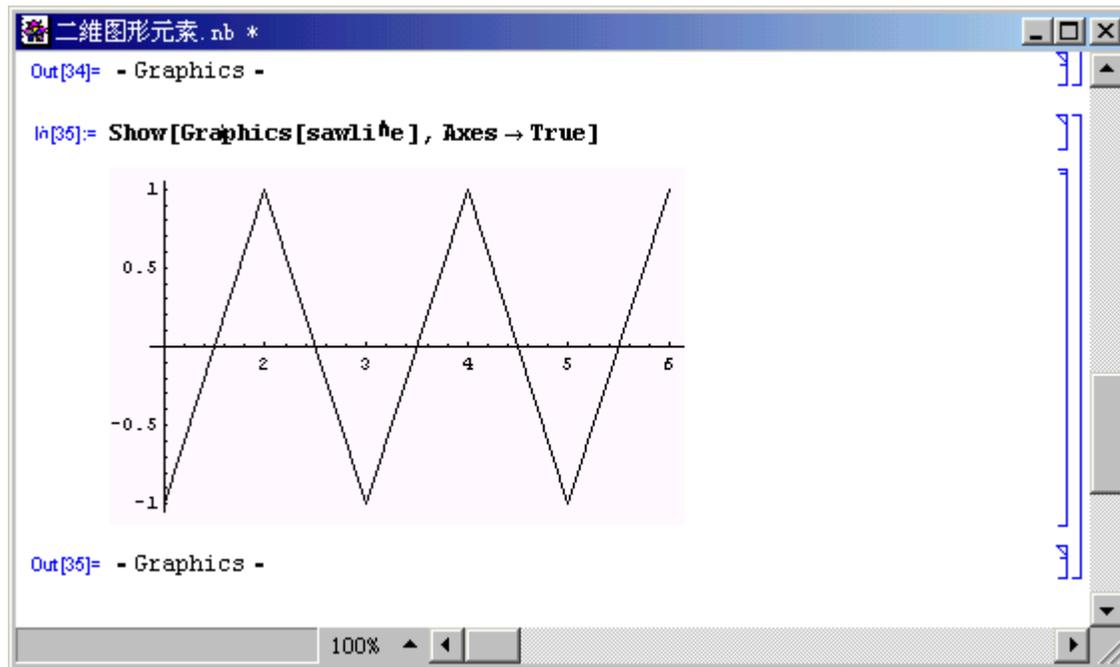
## 第4章： Mathematica的函数作图

下面绘制一些有线条组成的图形



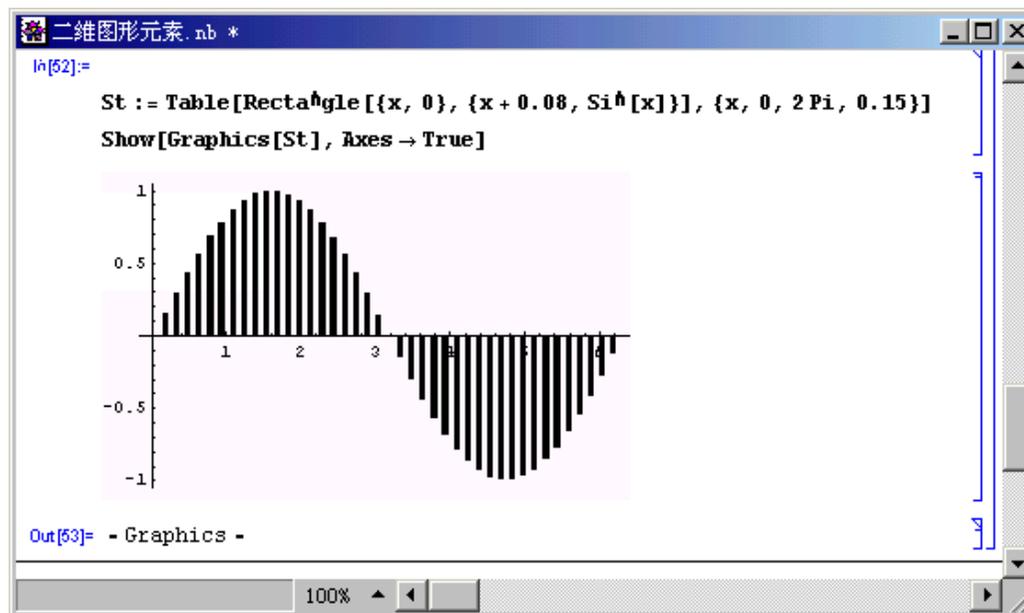
## 第4章： Mathematica的函数作图

当然也可以添加坐标轴下面的例子，说明了这一点。



## 第4章： Mathematica的函数作图

下面的例子，是说明了 `Rectangle` 的图形绘制，例子中用一些小矩形逼近正弦曲线与  $x$  轴所成面积。程序中生成一个图形集合并显示出来。



## 第4章： Mathematica的函数作图

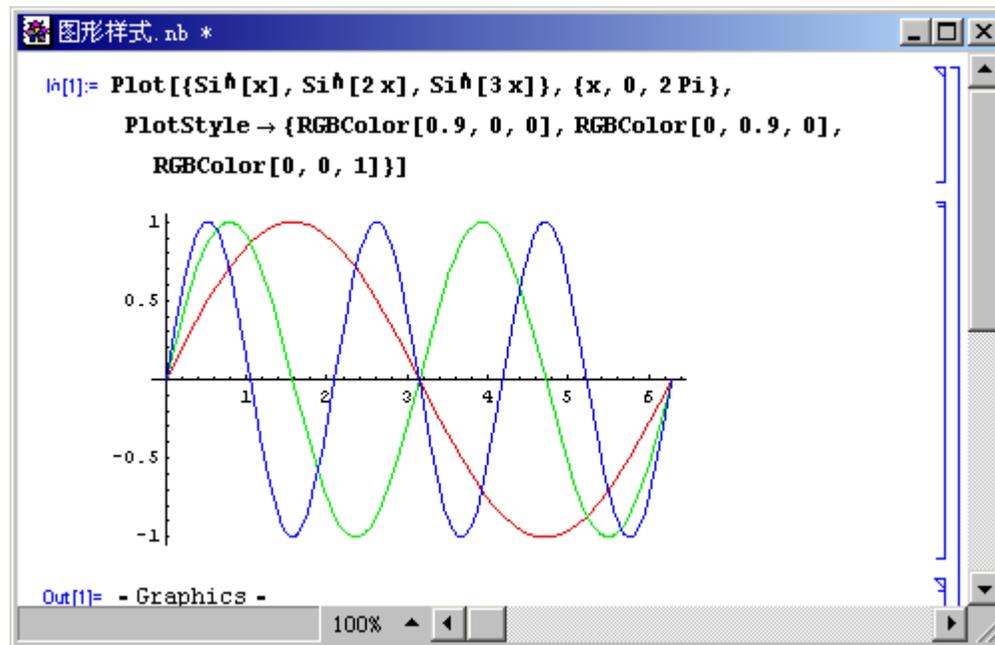
### 4. 3 图形的样式

<code>GrayLevel[]</code>	灰度介于0(黑)到1(白)之间
<code>RGBColor[r, g, b]</code>	由红、绿、蓝组成的颜色，每种色彩取0到1之间的数
<code>Hue[A]</code>	取0到1之间的色彩
<code>Hue[h, s, b]</code>	指定色调，位置和亮度的颜色，每项介于0到1之间
<code>PointSize[d]</code>	给出半径为d的点，单位是Plot的一个分数
<code>AbsolutePointSize[d]</code>	给出半径为d的点(以绝对单位量取)
<code>Thickness[w]</code>	给所有线的宽度w，单位是Plot的分数
<code>AbsoluteThickness[w]</code>	给所有线的宽度w，(以绝对单位量取)
<code>Dashing[w1, w2, .... ]</code>	给所有线为一系列虚线，虚线段的长度为w1, w2, ...
<code>Absolutedashing[ {w1, w2, .... } ]</code>	以绝对单位给出虚线长度
<code>PlotStyle-&gt;style</code>	设立Plot中所有曲线的风格
<code>PlotStyle-&gt;{{Style1}, {Style2}..... }</code>	设立Plot中一些列曲线的风格
<code>MeshStyle-&gt;Style</code>	设立宽度和表面网格的风格

## 第4章： Mathematica的函数作图

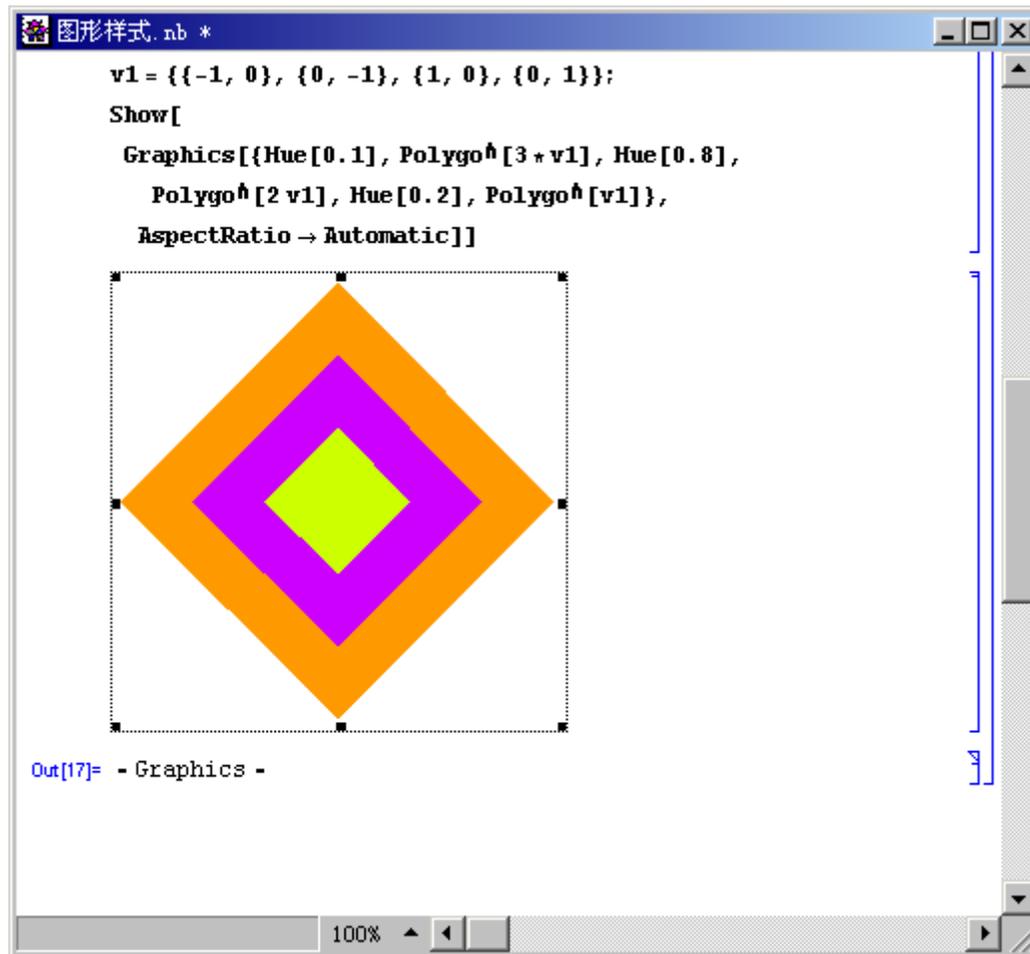
### 1.图形颜色的设置

在Mathematica提供各种图形指令中，对图形元素颜色的设置是一个很重要的设置。下面给出三条不同颜色的正弦曲线，此处以灰度表示，即颜色深浅不同。



## 第4章： Mathematica的函数作图

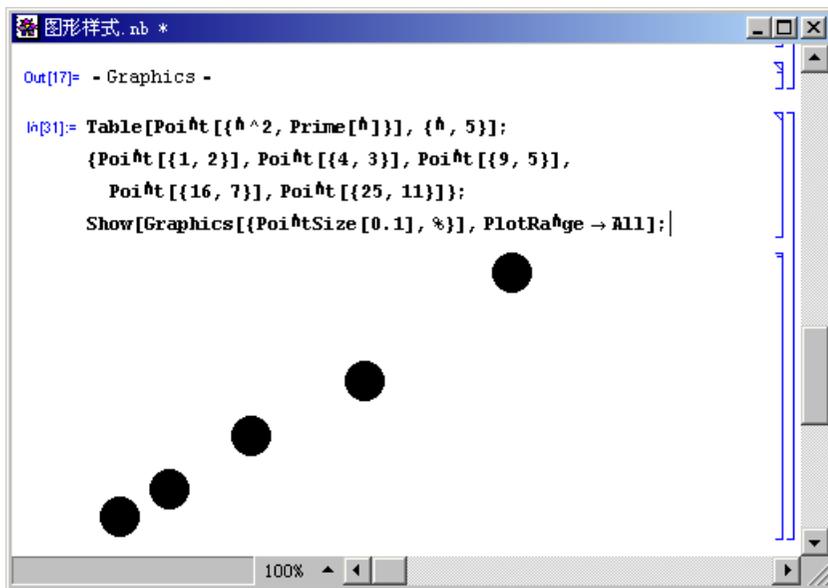
下面用不同的色调对三个菱形进行着色。



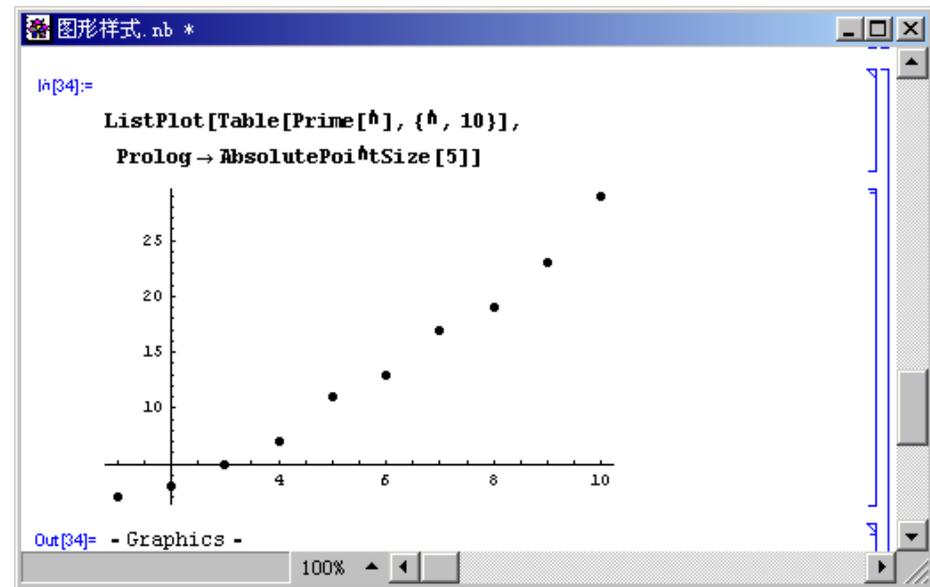
## 第4章： Mathematica的函数作图

### 2.图形大小

下面是一些点，注意点大小的控制。



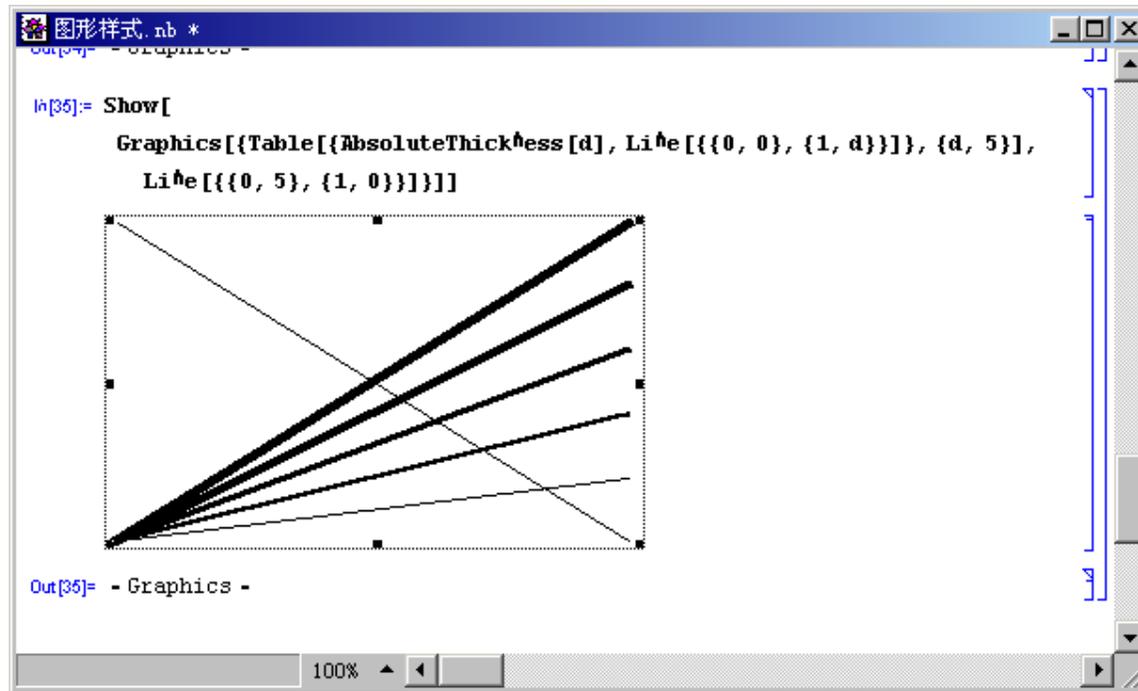
下面的点的控制是用绝对单位



## 第4章： Mathematica的函数作图

### 3.线段的控制

下面的例子是控制线段的宽度，使用的是绝对控制。



Mathematica提供的虚线指令可生成多种不同的复杂虚线。

## 第4章： Mathematica的函数作图

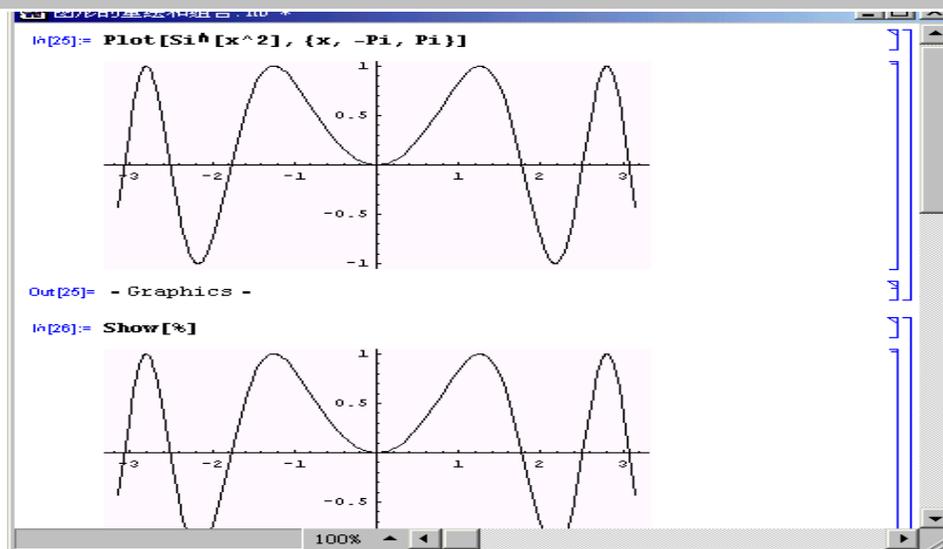
### 4.4 图形的重绘和组合

每次绘制图形后，**Mathematica**保存了图形的所有信息，所以用户可以重绘 这些图形。我们在重绘图形的时候，还可以改变一些使用。下面是常用重绘图形的函数。

<code>Show[plot]</code>	重绘图形
<code>Show[plot, option-&gt;value]</code>	改变方案重绘图形
<code>Show[plot1, plot2, plot3...]</code>	多个图形的绘制
<code>Show[GraphicsArray[{{plot1, plot2, ...}...}]]</code>	绘制图形矩阵
<code>InputForm[plot]</code>	给出所有的图形信息

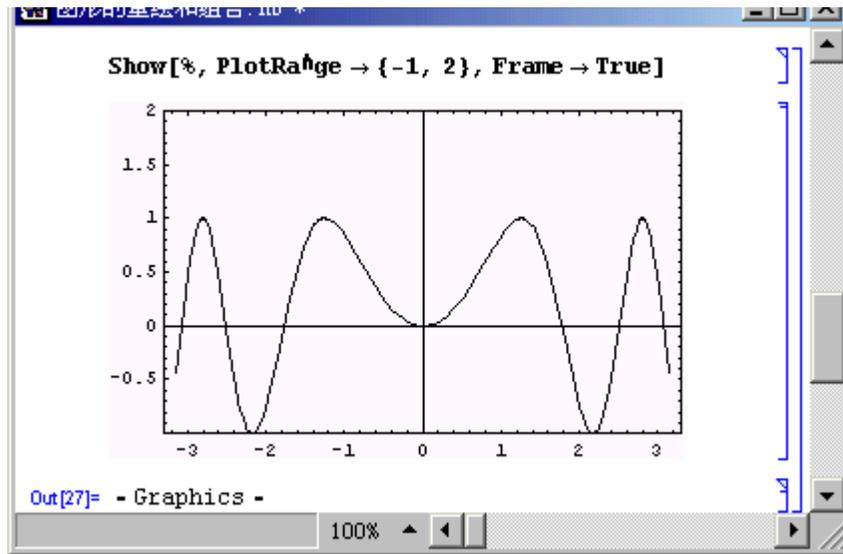
#### 1.使用Show显示图形

绘制函数 $\text{Sin}[x^2]$  的图形。



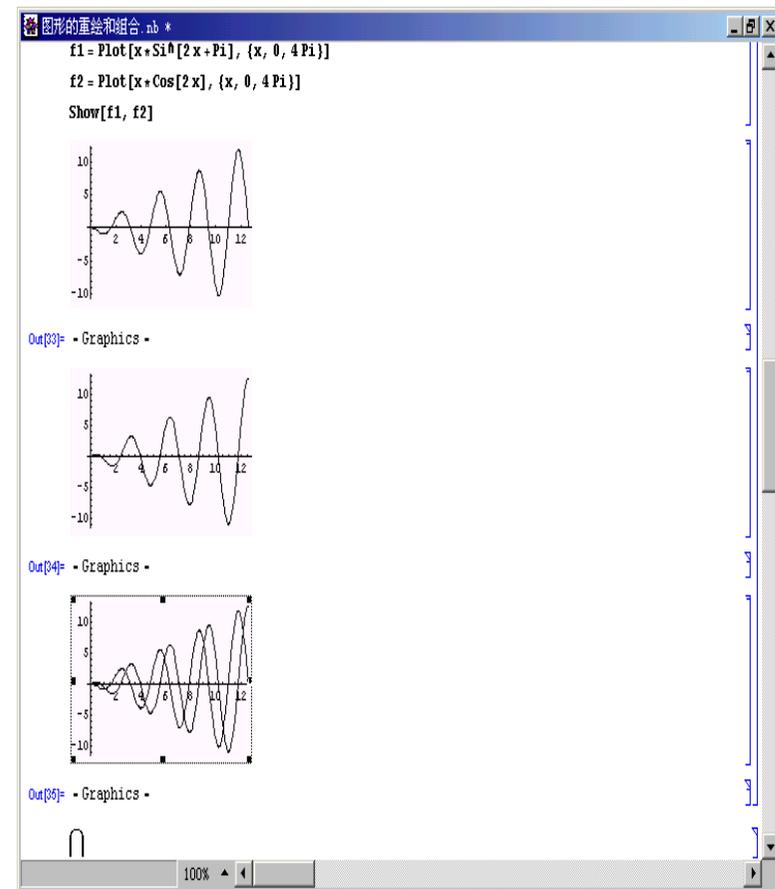
## 第4章： Mathematica的函数作图

重绘图形时，可以改变命令的设置，下面改变y的比例同时给图边框



### 2.使用Show命令进行组合

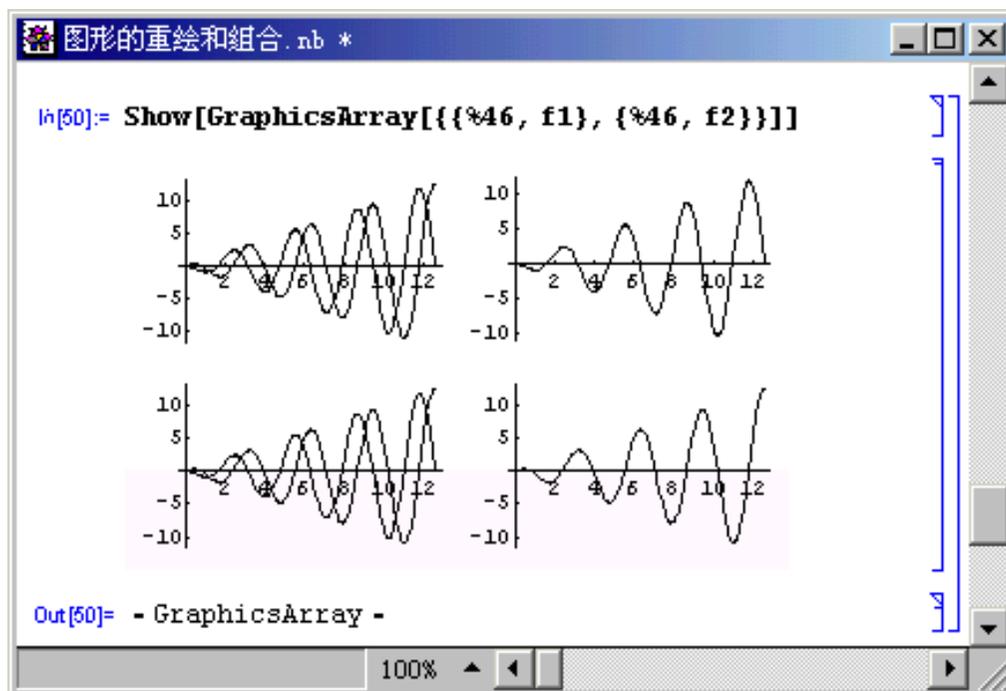
也可使用Show进行图形组合。图形组合与图形是否有相同的比例无关，这是Mathematica会自动选择新的比例来绘制图形。下面绘制函数 $-x\sin(2x+\pi)$ 的图形和 $x\cos(2x)$ 然后绘制在一张图时。



## 第4章： Mathematica的函数作图

### 3. 将多个图形组合为一个图形

可把图形组合为一个图形，我们还可以用`GraphicsArray`把多个图形绘制在一个图形矩阵中。



## 第4章： Mathematica的函数作图

### 4. 5 基本三维图形

`Plot3D[f ,(x,xmin,xmax), (y,ymin,ymax)]`

绘制以x和y为变量的三维函数  $f$  的图形

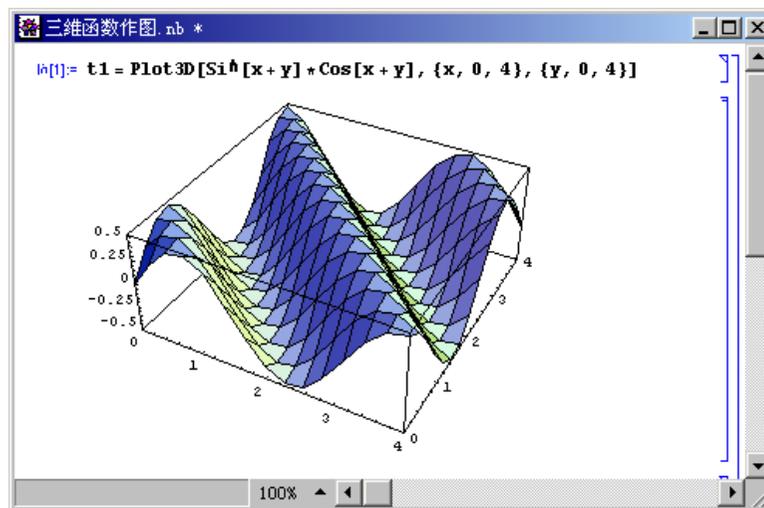
`ListPlot3D[{Z11,Z12,...}, {Z21,Z22,...},.....]`

绘出高度为Zvx数组的三维图形

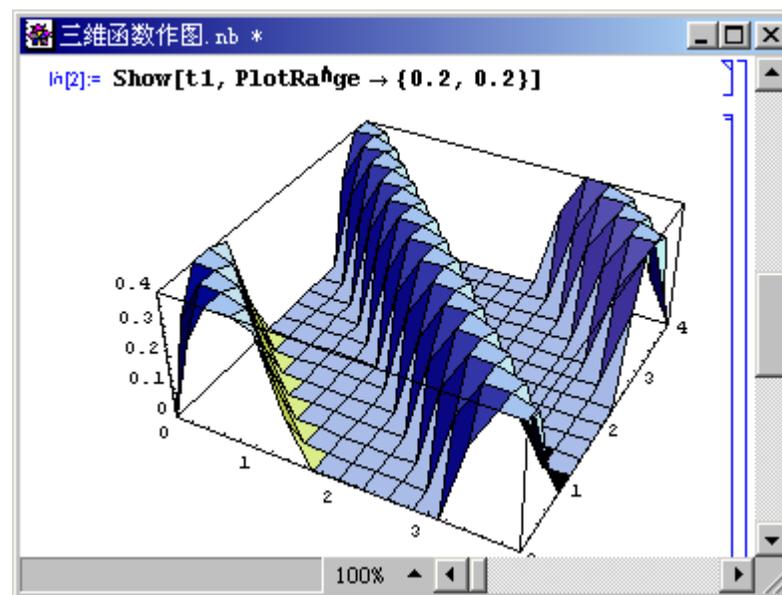
选项	取值	意义
Axes	True	是否包括坐标轴
AxesLabel	None	在轴上加上标志：zlabel规定z轴的标志， {xlabel,ylabel,zlabel}规定所有轴的标志
Boxed	True	是否在曲面周围加上立方体
ColorFunction	Automatic	使用什么颜色的明暗度；Hue表示使用一系列颜色
TextStyle	STextStyle	用于图形文本的缺省类型
FormatType	StandardForm	用于图形文本的缺省格式类型
DisplayFunction	SdisplayFunction	如何绘制图形，Identity表示不显示
FaceGrids	None	如何在立体界面上绘上网格；All表示在每个界面上绘上网格
HiddenSurface	True	是否以立体的形式绘出曲面
Lighting	True	是否用明暗分布来给表面上色
Mesh	True	是否在表面上绘出xy网格
PlotRange	Automatic	图中坐标的范围；可以规定为All, {zmin,zmax}或 {xmin,xmax}, {ymin,ymax}, {zmin,zmax}
Shading	True	表面是用阴影还是留空白
ViewPoint	{1.3, -2.4, 2}	表面的空间观察点

## 第4章： Mathematica的函数作图

(1). 函数 $\sin(x+y)\cos(x+y)$ 的立体图

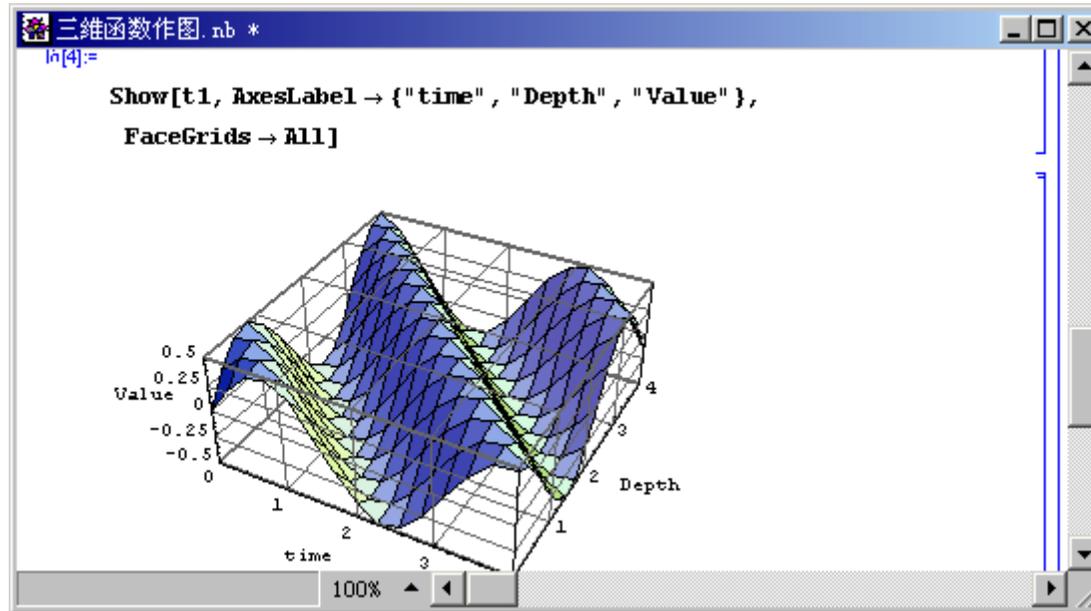


(2). 对于三维图形中Axes、Axeslabel、Boxed等操作同二维图形的一些操作很相似。用PlotRange设定曲线的表面的变化范围。



## 第4章： Mathematica的函数作图

(3). 图形轴上加上标记， 且在每个平面上画上网格。

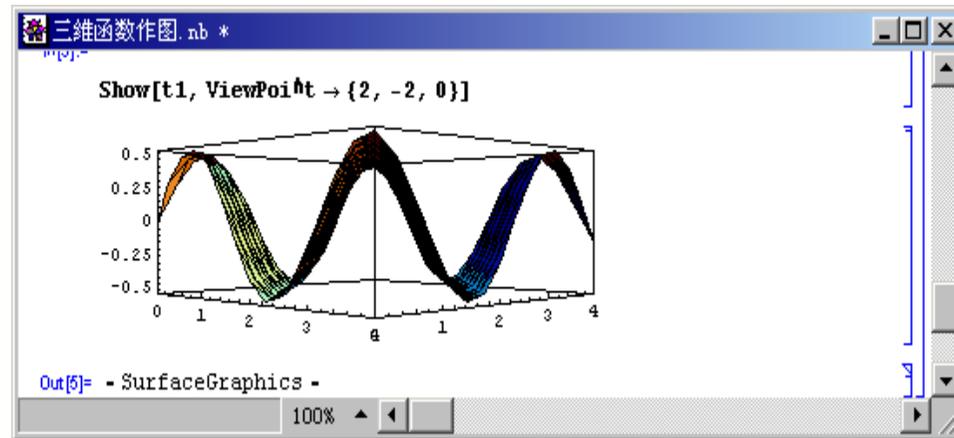


(4). 视图的改变

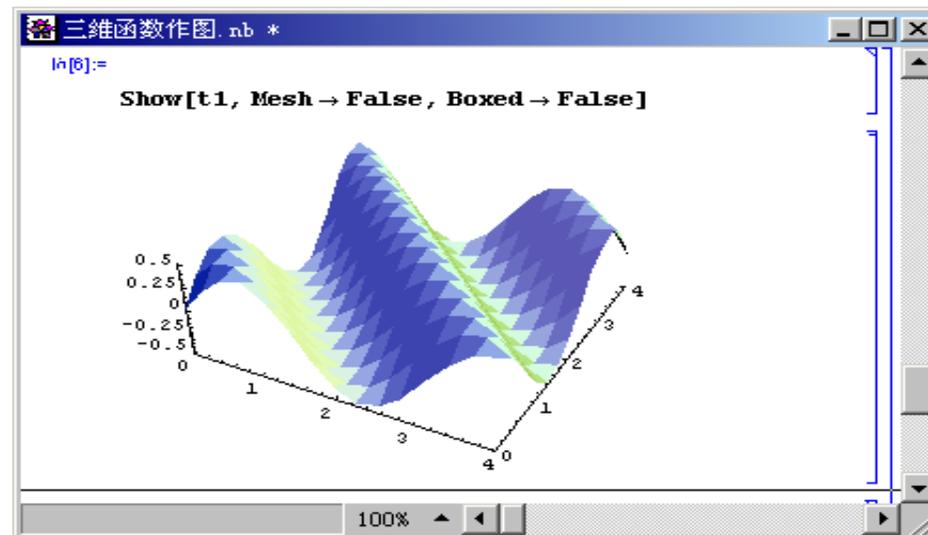
制图时通常用三视图来表示一个物体的具体形状特性。我们在生活中也知道从不同观察点观察物体，其效果是很不一样的。**Mathematica**在绘制立体图形时，在系统默认的情况下，观察点在(1. 3,-2. 4,2)处。这个参考点选择是具有一般性的，因此偶尔把图形的不同部分重在一起也不会发生视觉混乱。

## 第4章： Mathematica的函数作图

改变观察视点



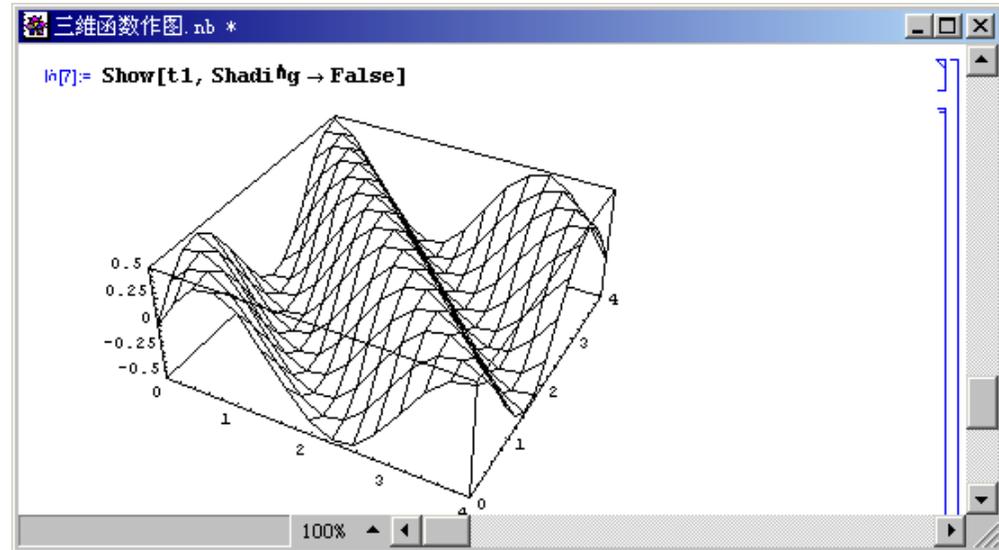
(6). 下面是没有网格和立体盒子的曲面图，它看起来就不如前面的图形清晰明了了。



## 第4章： Mathematica的函数作图

(7). 下图给出没有阴影的曲面

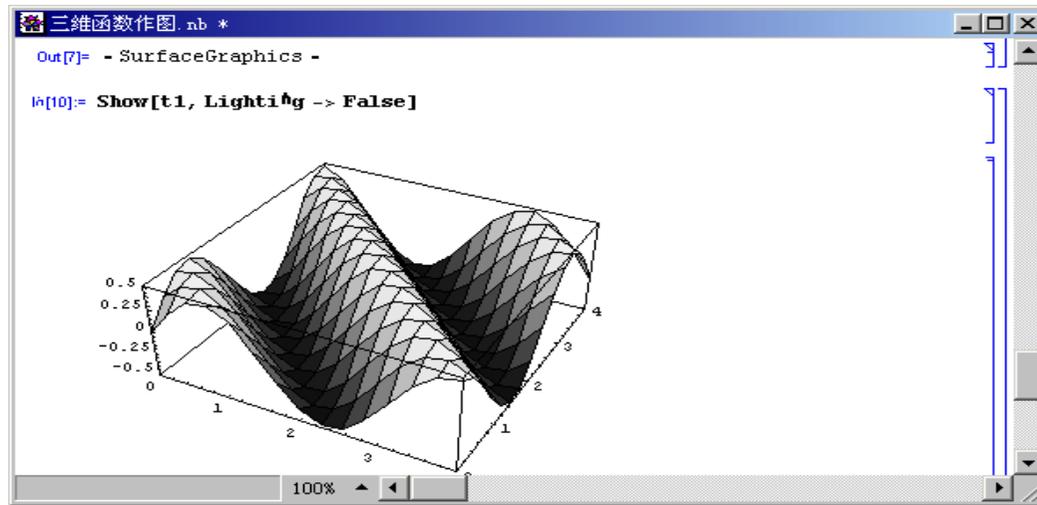
带有阴影和网格的图形对于理解曲面的形状是很有好处的。在有些矢量图形的输出装置中，你可能得不到阴影，但是当有阴影时，输出装置可能要花很长时间来输出它。



(8).给空间立体曲面着色

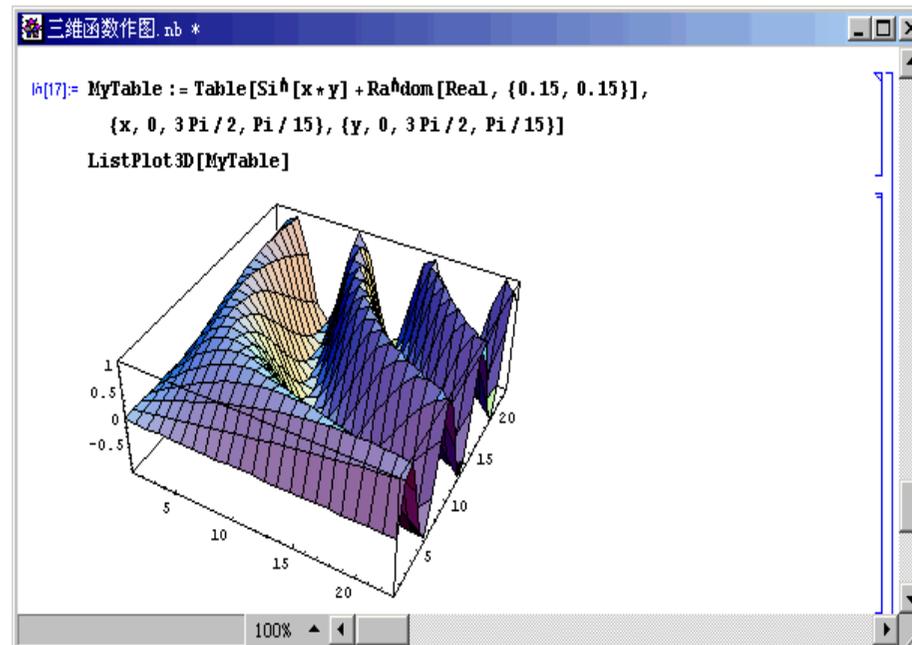
通常情况下，Mathematica为了使图形更加逼真而用明暗分布的形式给空间立体曲面着色。在这种情况下，Mathematica假定在图形的右上方有三种光源照在物体上。但有时这种方法会造成混乱，此时你可用`Lighting->False`来采取根据高度在表面上涂以不同灰度的阴影的方法。

## 第4章： Mathematica的函数作图



### 2.用数据来进行绘图

三维图形也可用数据来进行绘图



## 第4章： Mathematica的函数作图

### 3.三维空间的参数方程绘图

三维空间中的参数绘图函数`ParametricPlot3D[{fx,fv,fz},{t,tmin,tmax}]`和二维空间中的`ParametricPlot`很相仿。在这种情况下，Mathematica实际上都是根据参数`t`来产生系列点，然后再连接起来。

`ParametricPlot3D[{fx,fv,fz},{t,tmin,tmax}]`

给出空间曲线的参数图

`ParametricPlot3D[{fx, fv, fz}, {t, tmin, tmax}, {u, umin, umax}]`

给出空间曲面的参数图

`ParametricPlot3D[{fx, fv, fz, s}.....]`

按照函数关系`s`绘出参数图的阴影

`ParametricPlot3D[{fx, fv, fz}, {gx, gy, gz}.....]`

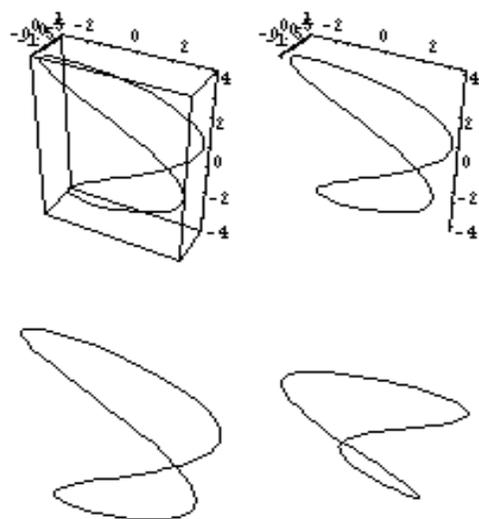
把一些图形绘制在一起

## 第4章： Mathematica的函数作图

一些空间曲线的例子

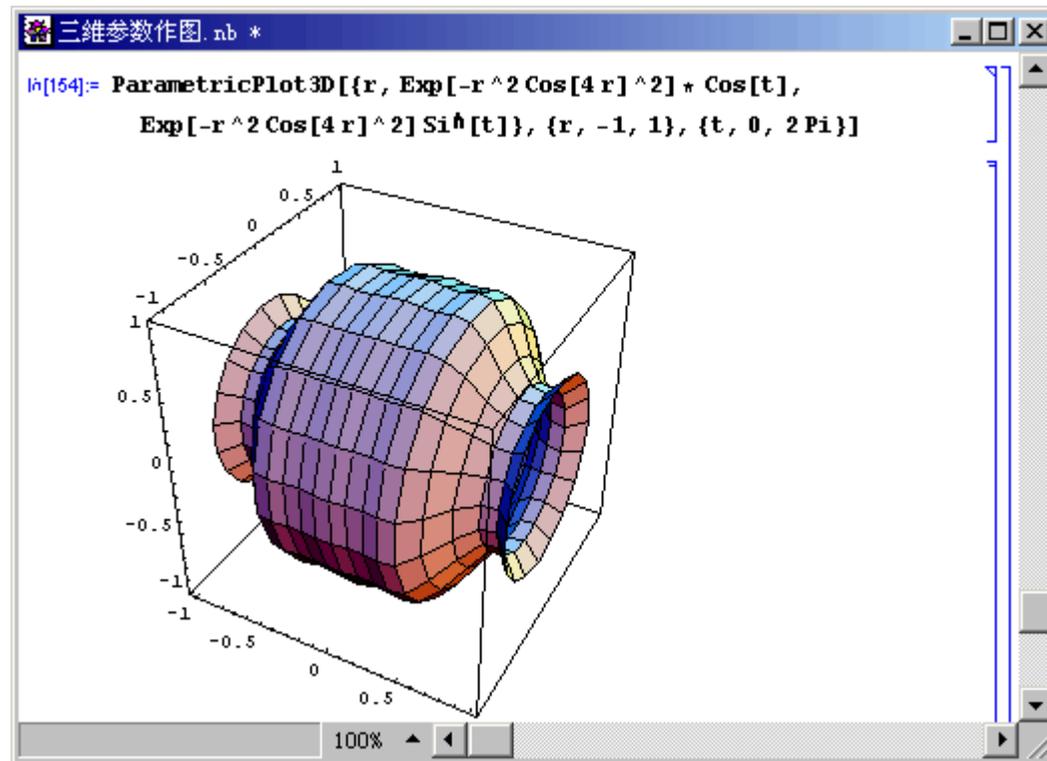
```
pp1 := ParametricPlot3D[{3 * Cos[4 * t + 1], Cos[2 * t + 3], 4 Cos[2 * t + 5]}, {t, 0, Pi}];  
pp2 := ParametricPlot3D[{3 Cos[4 t + 1], Cos[2 t + 3], 4 Cos[2 t + 5]}, {t, 0, Pi}, Boxed → False];  
pp3 := ParametricPlot3D[{3 Cos[4 t + 1], Cos[2 t + 3], 4 Cos[2 t + 5]}, {t, 0, Pi}, Boxed → False,  
  Axes → False];  
pp4 := ParametricPlot3D[{3 Cos[4 t + 1], Cos[2 t + 3], 4 Cos[2 t + 5]}, {t, 0, Pi}, Boxed → False,  
  Axes → False, BoxRatios → {1, 1, 1}];  
Show[GraphicsArray[{{pp1, pp2}, {pp3, pp4}}]]
```

结果为



## 第4章： Mathematica的函数作图

命令`ParametricPlot3D[{fx,fv,fz},{t,tmin,tmax}, {u,umin,umax}]` 产生一个曲面而不是一条曲线曲面是由四边形组成。



# 第5章：微积分的基本操作

## 5.1 极限

Mathematica计算极限的命令是Limit它的使用方法主要有

Limit[expr, x->x0]	当x趋向于x0时求expr的极限
Limit[expr, x->x0, Direction->1]	当x趋向于x0时求expr的左极限
Limit[expr, x->x0, Direction->-1]	当x趋向于x0时求expr的右极限

趋向的点可以是常数，也可以是 $+\infty$ ， $-\infty$  例如

1. 求  $\lim_{x \rightarrow \infty} \frac{\sqrt{x^2 + 2}}{3x - 6}$

```
In[1]:= Limit[Sqrt[x^2 + 2] / (3 x - 6), x -> Infinity]
Out[1]= 1/3
```

2. 求  $\lim_{x \rightarrow 0} \frac{\sin^2 x}{x^2}$

```
In[2]:= Limit[Sin[x]^2 / x^2, x -> 0]
Out[2]= 1
```

# 第5章：微积分的基本操作

## 5.2 微分

在Mathematica中，计算函数的微分或是非常方便的，命令为D[f,x],表示对x求函数f的导数或偏导数。该函数的常用格式有以下几种

D[f, x]	计算微分	$\frac{\partial f}{\partial x}$
D[f, x1, x2, ...]	计算多重偏微分	$\frac{\partial}{\partial x_1} \frac{\partial}{\partial x_2} f$
D[f, {x, n}]	计算n阶微分	$\frac{\partial^n}{\partial x^n} f$
D[f, x, NonConstants->{v1, v2, ...}]	计算微分	$\frac{\partial f}{\partial x}$ 其中v1, v2...依赖于x

1.求函数sinx的导数

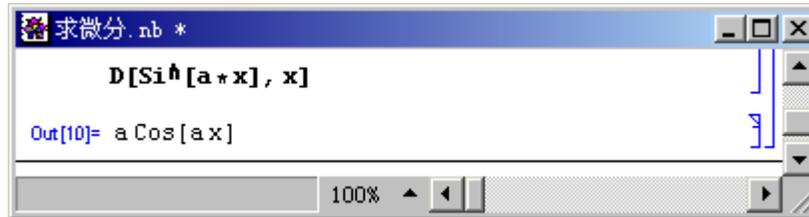
```
In[4]:= D[Exp[x] * Sin[x], x]
Out[4]= e^x Cos[x] + e^x Sin[x]
```

2.求函数e^xsinx的2阶导数

```
In[9]:= D[Exp[x] * Sin[x], {x, 2}]
Out[9]= 2 e^x Cos[x]
```

## 第5章：微积分的基本操作

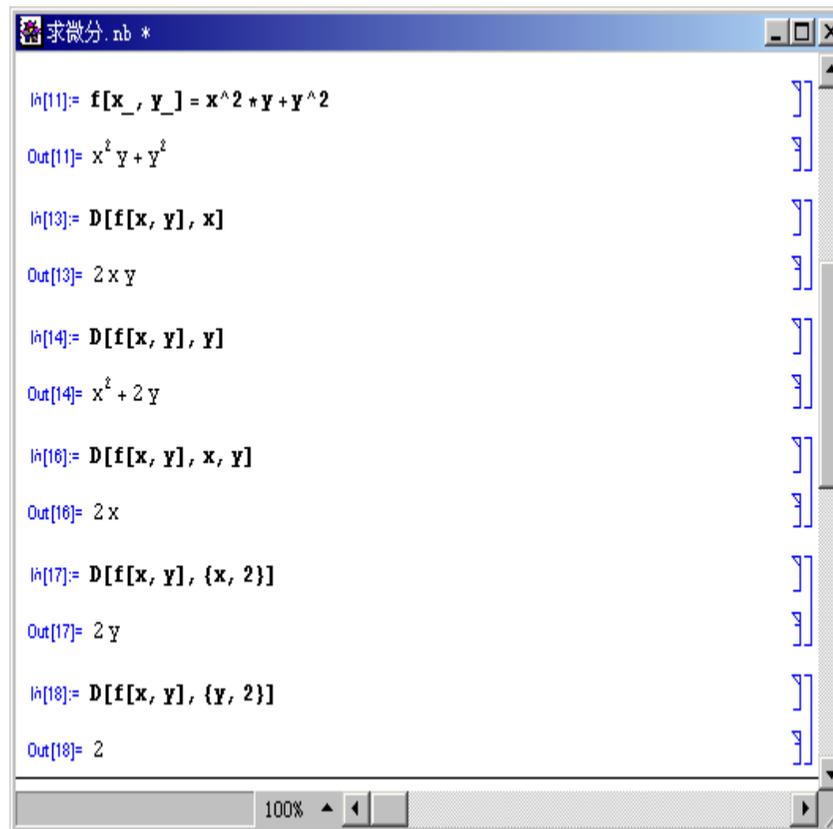
3. 假设 $a$ 是常数可以对 $\sin(ax)$ 求导



```
求微分.nb *
D[Sin[a x], x]
Out[10]= a Cos[a x]
100%
```

4. 对二元函数

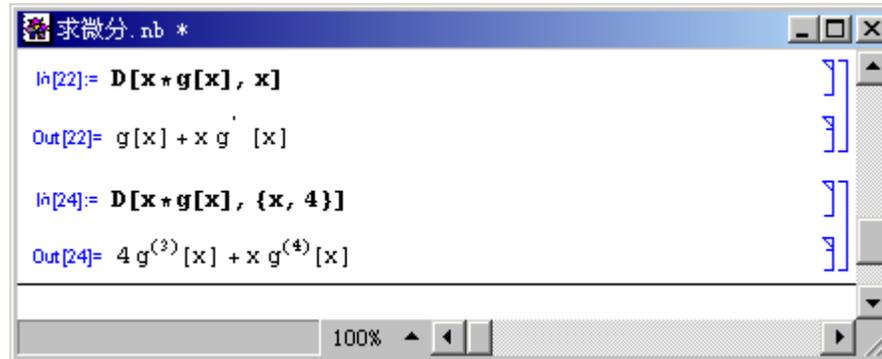
$f(x,y)=x^2*y+y^2$ 求对 $x,y$   
求一阶和二阶偏导



```
求微分.nb *
In[11]= f[x_, y_] = x^2 * y + y^2
Out[11]= x^2 y + y^2
In[13]= D[f[x, y], x]
Out[13]= 2 x y
In[14]= D[f[x, y], y]
Out[14]= x^2 + 2 y
In[16]= D[f[x, y], x, y]
Out[16]= 2 x
In[17]= D[f[x, y], {x, 2}]
Out[17]= 2 y
In[18]= D[f[x, y], {y, 2}]
Out[18]= 2
100%
```

## 第5章：微积分的基本操作

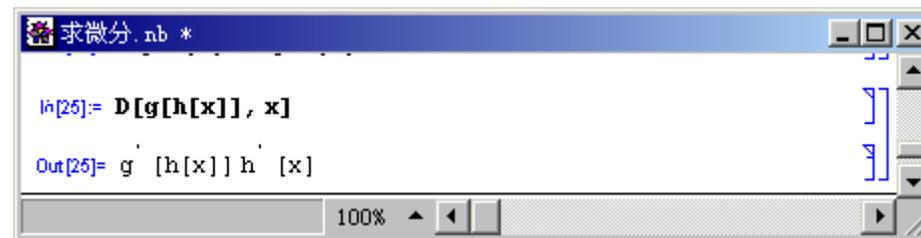
Mathematica可以求函数式未知的函数微分，通常结果使用数学上的表示法



```
求微分.nb *
In[22]:= D[x + g[x], x]
Out[22]= g[x] + x g'[x]

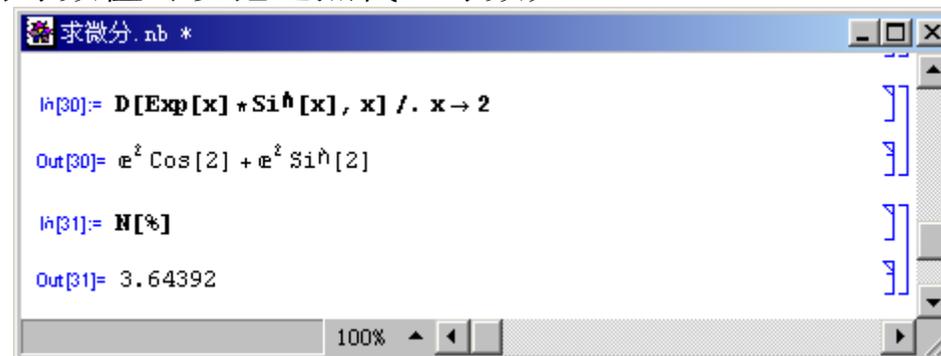
In[24]:= D[x + g[x], {x, 4}]
Out[24]= 4 g(3)[x] + x g(4)[x]
```

对链导法则同样可用



```
求微分.nb *
In[25]:= D[g[h[x]], x]
Out[25]= g'[h[x]] h'[x]
```

如果要得到函数在某一点的导数值可以把这点代入导数如



```
求微分.nb *
In[30]:= D[Exp[x] + Sin[x], x] /. x -> 2
Out[30]= e2 Cos[2] + e2 Sin[2]

In[31]:= N[%]
Out[31]= 3.64392
```

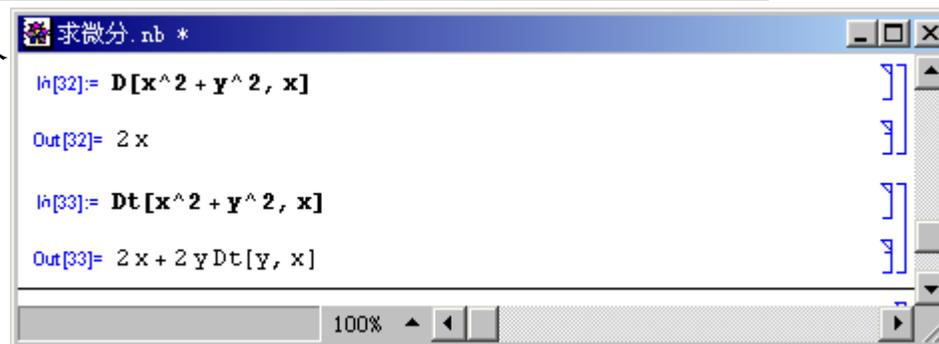
# 第5章：微积分的基本操作

## 2.全微分

在Mathematica中， $D[f,x]$ 给出 $f$ 的偏导数，其中假定 $f$ 中的其他变量与 $x$ 无关。当 $f$ 为单变量时， $D[f,x]$ 计算 $f$ 对 $x$ 的导数。函数 $Dt[f,x]$ 给出 $f$ 的全微分形式，并假定 $f$ 中所有变量依赖于 $x$ 。下面是 $Dt$ 命令的常用形及意义

$Dt[f]$	求全微分 $df$
$Dt[f, x]$	求全微分 $\frac{df}{dx}$
$Dt[f, x_1, x_2, \dots]$	求多重全微分 $\frac{d}{dx_1} \frac{d}{dx_2} f$
$Dt[f, x, Constants \rightarrow \{c_1, c_2, \dots\}]$	求全微分其中 $c_1, c_2, \dots$ 是常数

求 $x^2+y^2$ 的偏微分和全微分



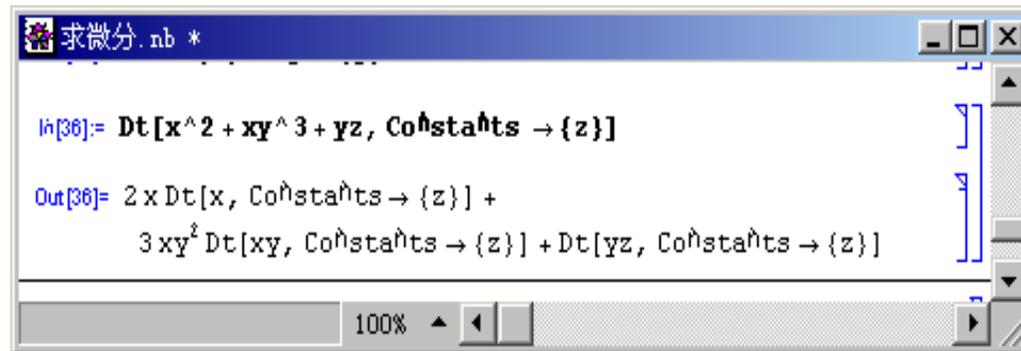
```
求微分.nb *
In[32]:= D[x^2 + y^2, x]
Out[32]= 2 x

In[33]:= Dt[x^2 + y^2, x]
Out[33]= 2 x + 2 y Dt[y, x]
```

可以看出第一种情况 $y$ 与 $x$ 没有关系，第二种情况 $y$ 是 $x$ 的函数

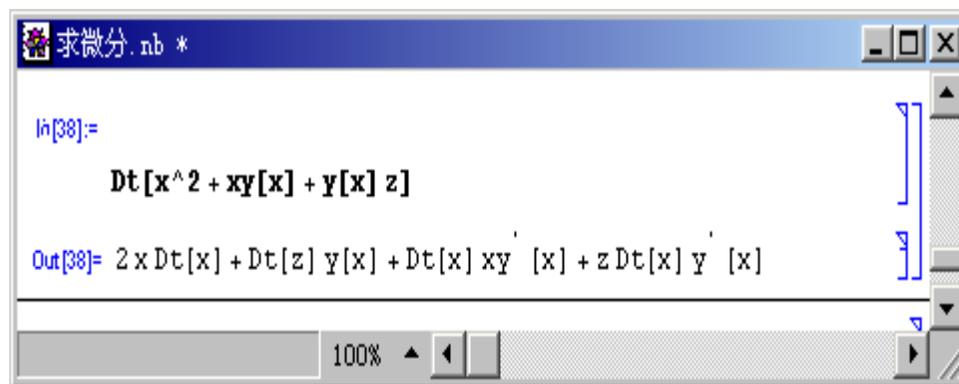
## 第5章：微积分的基本操作

求多项式 $x^2+xy^3+yz$ 的全微分并假定 $z$ 保持不变是常数



```
求微分.nb *  
  
In[36]:= Dt[x^2 + xy^3 + yz, Constants -> {z}]  
  
Out[36]= 2 x Dt[x, Constants -> {z}] +  
         3 xy^2 Dt[xy, Constants -> {z}] + Dt[yz, Constants -> {z}]
```

如果 $y$ 是 $x$ 的函数那么， $y$ 被看成是常数



```
求微分.nb *  
  
In[38]:= Dt[x^2 + xy[x] + y[x] z]  
  
Out[38]= 2 x Dt[x] + Dt[z] y[x] + Dt[x] xy'[x] + z Dt[x] y'[x]
```

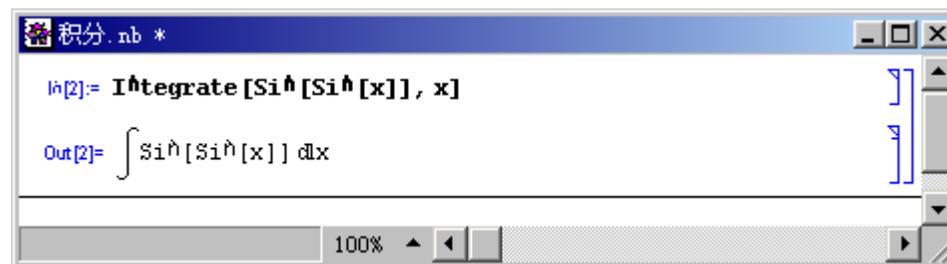
# 第5章：微积分的基本操作

## 5.3 计算积分

### 1. 不定积分

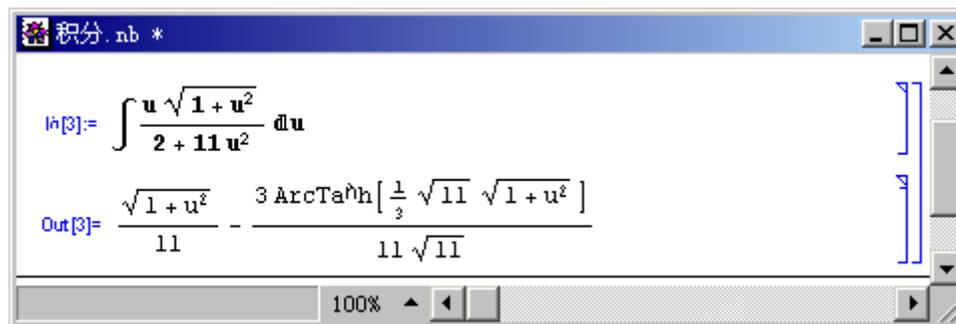
在Mathematica中计算不定积分命令为`Integrate[f, x]`当然也可使用工具栏直接输入不定积分式。来求函数的不定积分。当然并不是所有的不定积分都能求出来。

例如若求  $\int \sin x \sin x dx$  Mathematica就无能为力。



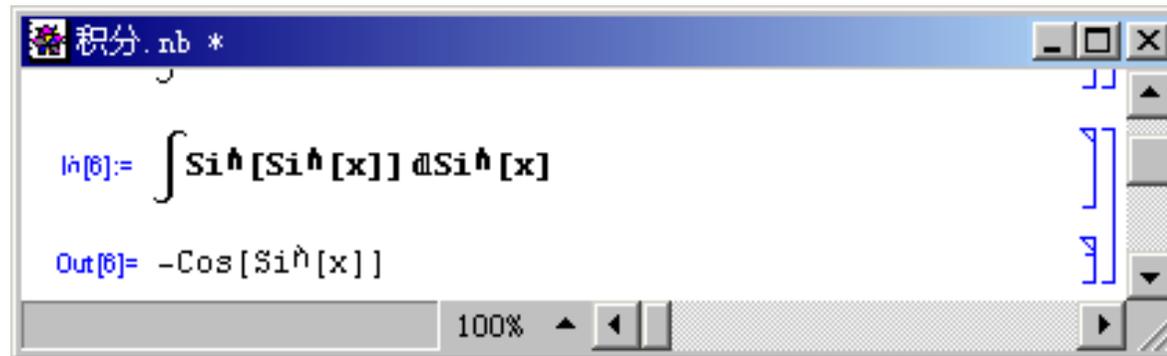
对于一些手工计算相当复杂的不定积分，Mathematica还是能轻易求得，例如求

$$\int \frac{u\sqrt{1+u^2}}{2+11u^2} du$$



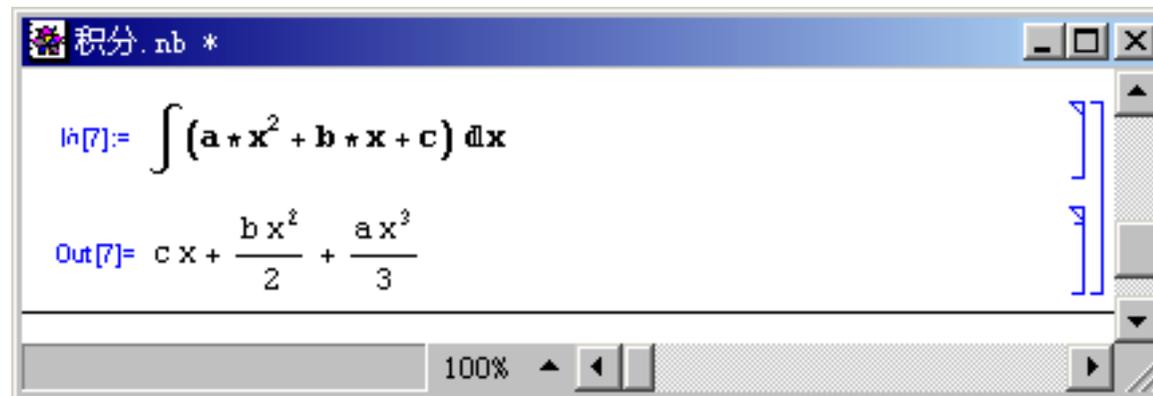
## 第5章：微积分的基本操作

积分变量的形式也可以是一函数，例如



```
积分.nb *  
  
In[6]:= Integrate[Sinh[Sinh[x]], Sinh[x]]  
  
Out[6]:= -Cos[Sinh[x]]  
  
100%
```

对于在函数中出现的除积分变量外的函数，统统当作常数处理，请看下面例子。



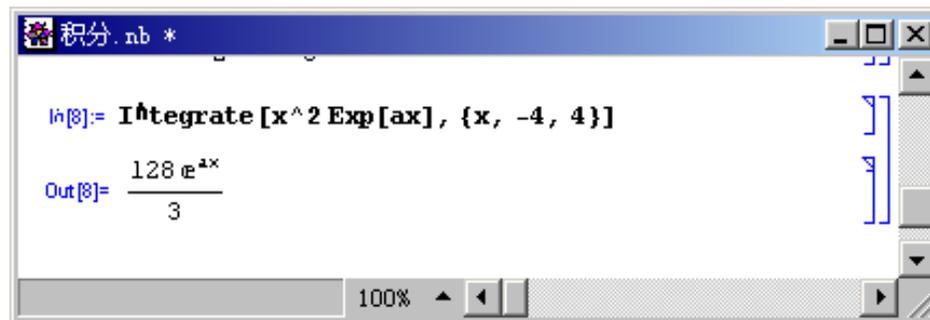
```
积分.nb *  
  
In[7]:= Integrate[a*x^2 + b*x + c, x]  
  
Out[7]:= c*x + (b*x^2)/2 + (a*x^3)/3  
  
100%
```

# 第5章：微积分的基本操作

## 2. 定积分

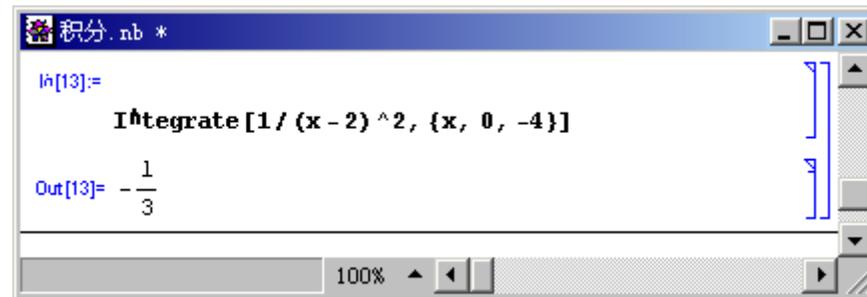
定积分的求解主要命令也是用Integrate只是要在命令中加入积分限Integrate[f,{x,min,max}]

求  $\int_{-4}^4 x^2 e^{ax} dx$



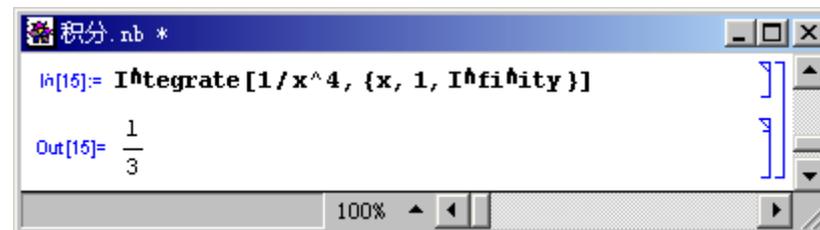
```
In[8]:= Integrate[x^2 Exp[ax], {x, -4, 4}]  
Out[8]=  $\frac{128 e^{4x}}{3}$ 
```

求  $\int_0^{-4} \frac{1}{(x-2)^2} dx$



```
In[13]:= Integrate[1/(x-2)^2, {x, 0, -4}]  
Out[13]=  $-\frac{1}{3}$ 
```

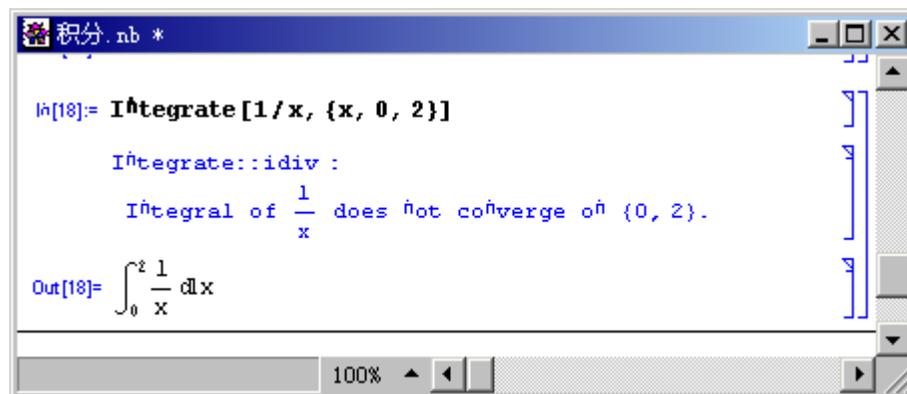
求无穷积例如  $\int_1^{+\infty} \frac{1}{x^4} dx$



```
In[15]:= Integrate[1/x^4, {x, 1, Infinity}]  
Out[15]=  $\frac{1}{3}$ 
```

## 第5章：微积分的基本操作

如果无法判定敛散性，就用给出一个提示，例如

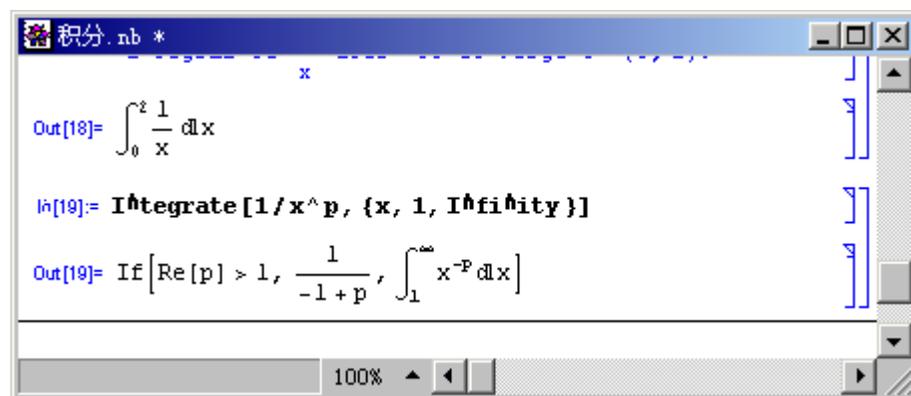


```
In[18]:= Integrate[1/x, {x, 0, 2}]

Integrate::idiv:
Integral of 1/x does not converge on {0, 2}.

Out[18]= \int_0^2 1/x dx
```

如果广义积分敛散性与某个符号的取值有关，它也能给出在不同情况下的积分结果例如  $\int_1^{+\infty} \frac{1}{x^p} dx$



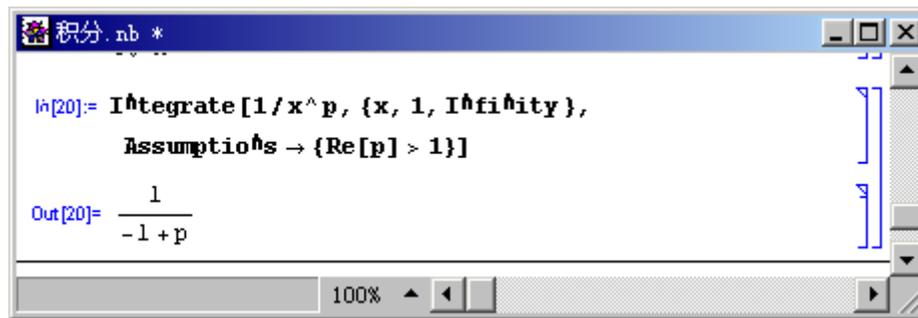
```
In[19]:= Integrate[1/x^p, {x, 1, Infinity}]

Out[19]= If[Re[p] > 1, 1/(-1+p), \int_1^{+\infty} x^{-p} dx]
```

结果的意义是当 $|p|>1$ 时，积分值为 $1/1-p$ ，否则不收敛。

## 第5章：微积分的基本操作

在Integrate中可加两个参数**Assumptions** 和 **GenerateConditions**例如上例中，  
只要用**Assumptions->{Re[p]>1}**就可以得到收敛情况的解



```
In[20]:= Integrate[1/x^p, {x, 1, Infinity},  
Assumptions -> {Re[p] > 1}]  
  
Out[20]=  $\frac{1}{-1+p}$ 
```

### 3.数值积分

数值积分是解决求定积分的另一种有效的方法，它可以给出一个近似解。特别是对于用Integrate命令无法求出的定积分，数值积分更是可以发挥巨大作用。  
它的命令格式为

**NIntegrate[f,{x,a,b}]**

在[a,b]上求f数值积分

**NIntegrate[f,{x,a,x1,x2,...,b}]**

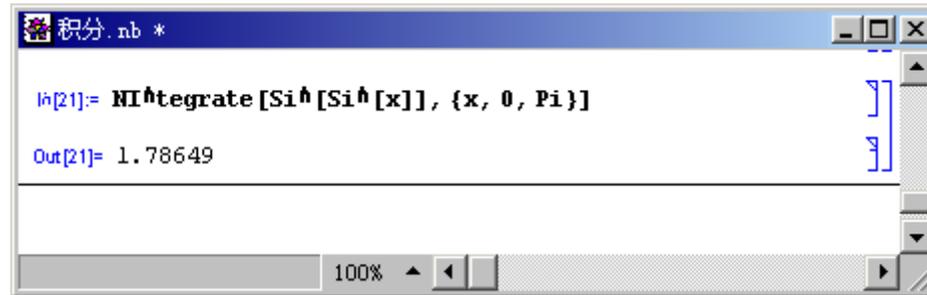
以x1,x2....为分割求[a,b]上的数值积分

**NIntegrate[f,{x,a,b},MaxRecursion->n]**

求数值积分时指定迭代次数n.

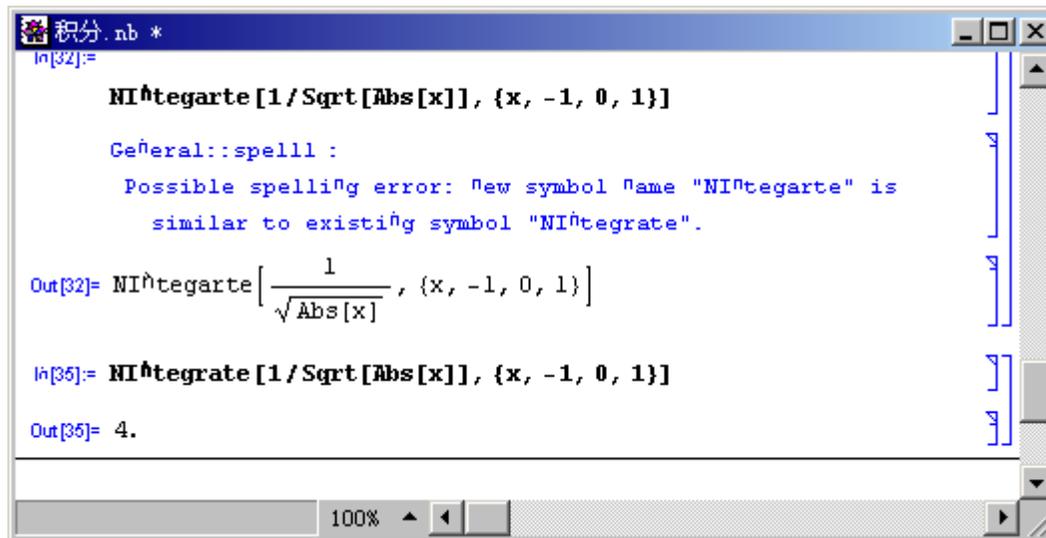
## 第5章：微积分的基本操作

下面我们求  $\sin x$  在  $[0, \pi]$  上的积分值，由于这个函数的不定积分求不出，因此使用 `Integrate` 命令无法得到具体结果，但可以用数值积分求



```
积分.nb *
In[21]:= NIntegrate[Sin[x], {x, 0, Pi}]
Out[21]= 1.78649
100%
```

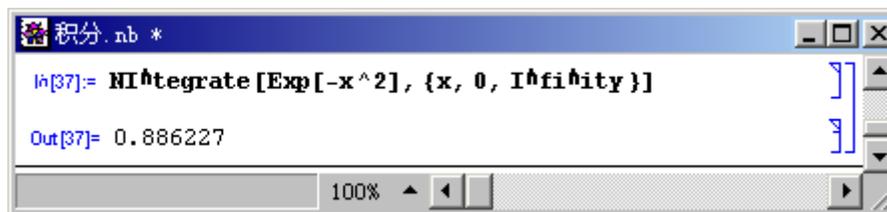
如果积分函数存在不连续点，或存在奇点我们可对积分进行分段求解。例如函数  $\frac{1}{\sqrt{|x|}}$  在  $[-1, 1]$  上，显然  $x=0$  点是一个无穷间断点。因此若要求其数值积分，必须在其中插入点 0



```
积分.nb *
In[32]:= NIntegrate[1/Sqrt[Abs[x]], {x, -1, 0, 1}]
General::spell1: Possible spelling error: New symbol name "NIntegarte" is similar to existing symbol "NIntegrate".
Out[32]= NIntegrate[1/Sqrt[Abs[x]], {x, -1, 0, 1}]
In[35]:= NIntegrate[1/Sqrt[Abs[x]], {x, -1, 0, 1}]
Out[35]= 4.
100%
```

## 第5章：微积分的基本操作

对无穷积分，也可求数值积分，例如



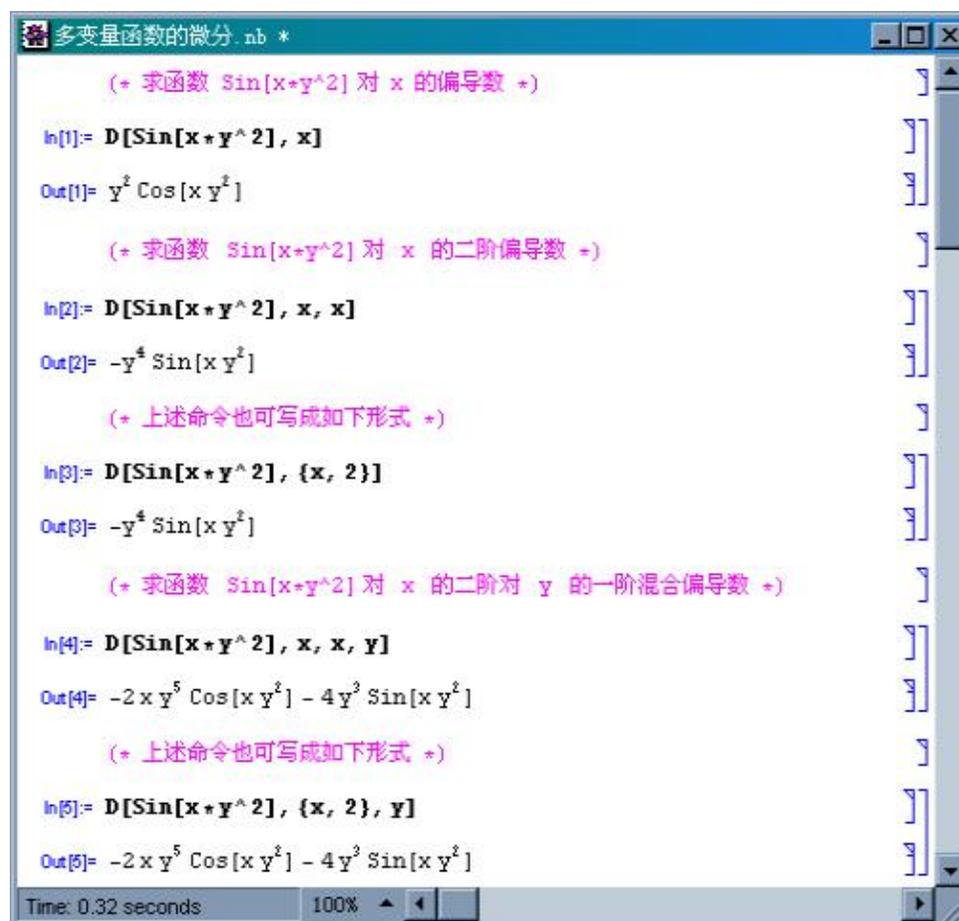
```
积分.nb *
In[37]= NIntegrate[Exp[-x^2], {x, 0, Infinity}]
Out[37]= 0.886227
100%
```

### 5.4 多变量函数的微分

( I )  $D[f, x_1, x_2, \dots, x_n]$

计算偏导数  $\frac{\partial^n f}{\partial x_1 \partial x_2 \dots \partial x_n}$

下面是实际的例子：



```
多变量函数的微分.nb *
(* 求函数 Sin[x+y^2] 对 x 的偏导数 *)
In[1]= D[Sin[x+y^2], x]
Out[1]= y^2 Cos[x y^2]
(* 求函数 Sin[x+y^2] 对 x 的二阶偏导数 *)
In[2]= D[Sin[x+y^2], x, x]
Out[2]= -y^4 Sin[x y^2]
(* 上述命令也可写成如下形式 *)
In[3]= D[Sin[x+y^2], {x, 2}]
Out[3]= -y^4 Sin[x y^2]
(* 求函数 Sin[x+y^2] 对 x 的二阶对 y 的一阶混合偏导数 *)
In[4]= D[Sin[x+y^2], x, x, y]
Out[4]= -2 x y^5 Cos[x y^2] - 4 y^3 Sin[x y^2]
(* 上述命令也可写成如下形式 *)
In[5]= D[Sin[x+y^2], {x, 2}, y]
Out[5]= -2 x y^5 Cos[x y^2] - 4 y^3 Sin[x y^2]
Time: 0.32 seconds 100%
```

## 第5章：微积分的基本操作

( II )  $D[f, x, \text{NonConstants} \rightarrow \{v_1, v_2, \dots\}]$

其中  $\frac{\partial f}{\partial x}$  里变量  $v_i$  依赖于  $x$  .

右面是实际的例子:

```

多变量函数的微分.nb *
In[3]:= D[x^2 + y^2, x, NonConstants -> {y}]
Out[3]= 2 x + 2 y D[y, x, NonConstants -> {y}]

(* 注: D[y, x, NonConstants -> {y}] 表示  $\frac{dy}{dx}$ , 其中 y 是 x 的函数 *)

Time: 0.03 seconds 100%
    
```

( III )  $Dt[f]$  计算全微分  $df$  .

下面是实际的例子:

```

多变量函数的微分.nb *
In[8]:= (* 计算 d(x^2 y^3), 其中 Dt[x] 为 dx, Dt[y] 为 dy *)
Dt[x^2 y^3]
Out[8]= 2 x y^3 Dt[x] + 3 x^2 y^2 Dt[y]

In[9]:= (* 定义 z 为一个二元函数, 求 z 的全微分, 并提出 Dt[x] 和 Dt[y] *)
z = x^3 y + x^2 y^2 - 3 x + y^2;
Collect[Dt[z], {Dt[x], Dt[y]}]
Out[9]= (3 x^2 y - 3 y^2 + 2 x y^2) Dt[x] + (x^3 - 6 x y + 2 x^2 y) Dt[y]

(* 将上式表示成 f_x dx + f_y dy 的标准形式 *)

In[10]:= % /. {Dt[x] -> dx, Dt[y] -> dy}
Out[10]= dy (x^3 - 6 x y + 2 x^2 y) + dx (3 x^2 y - 3 y^2 + 2 x y^2)

(* 求 f_x, 因为 Mathematica 不知道 y 是否为 x 的函数, 所以保留 Dt[y, x] *)

In[11]:= Dt[z, x]
Out[11]= 3 x^2 y - 3 y^2 + 2 x y^2 + x^3 Dt[y, x] - 6 x y Dt[y, x] + 2 x^2 y Dt[y, x]

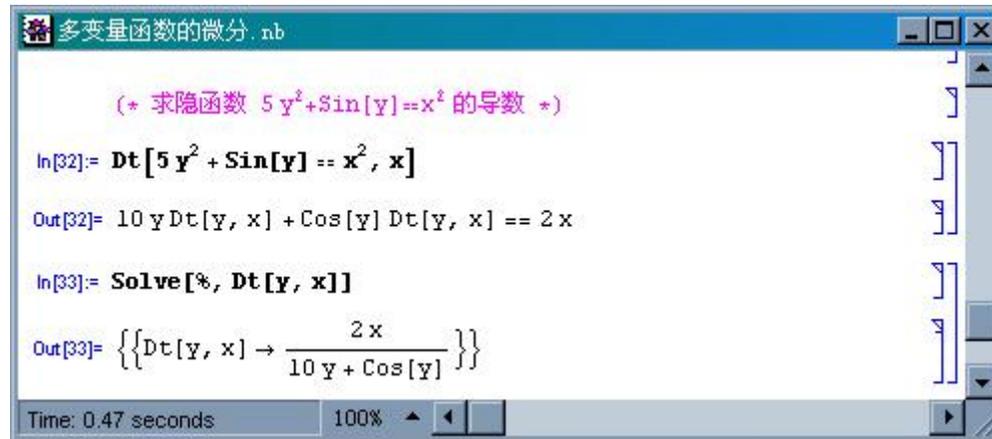
(* 用置换运算将 Dt[y, x] 置换成 0 即可求得 f_x *)

In[12]:= Dt[z, x] /. Dt[y, x] -> 0
Out[12]= 3 x^2 y - 3 y^2 + 2 x y^2

Time: 0.03 seconds 100%
    
```

## 第5章：微积分的基本操作

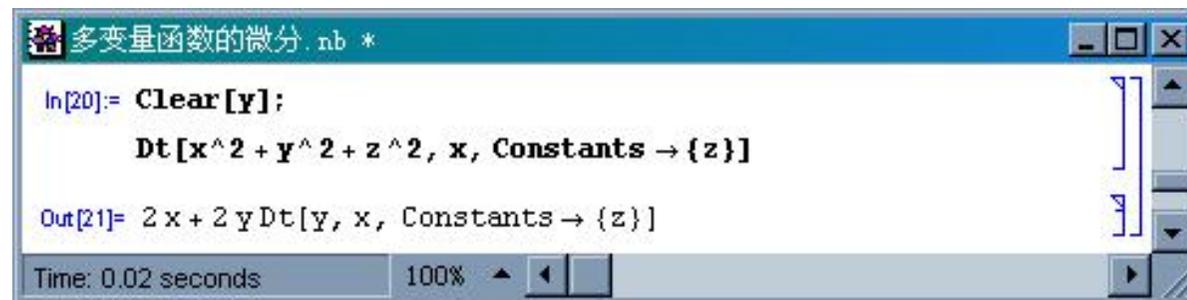
(IV) 求隐函数的导数  
下面是实际的例子：



```
多变量函数的微分.nb  
  
(* 求隐函数  $5y^2 + \sin[y] = x^2$  的导数 *)  
  
In[32]:= Dt[5 y^2 + Sin[y] == x^2, x]  
  
Out[32]= 10 y Dt[y, x] + Cos[y] Dt[y, x] == 2 x  
  
In[33]:= Solve[%, Dt[y, x]]  
  
Out[33]= {{Dt[y, x] ->  $\frac{2 x}{10 y + \cos[y]}$ }}
```

Time: 0.47 seconds    100%

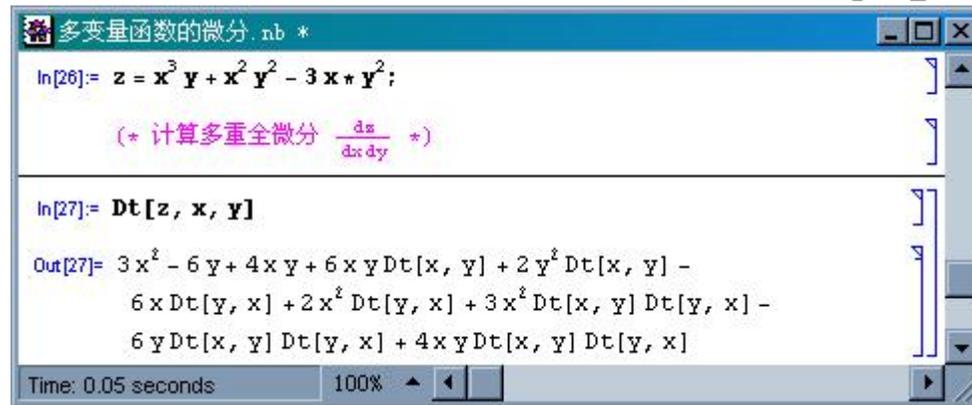
(V)  $\text{Dt}[f, x, \text{Constants} \rightarrow \{c_1, c_2, \dots\}]$  计算全微分  $df$ ，其中  $c_i$  是常数。  
下面是实际的例子：



```
多变量函数的微分.nb *  
  
In[20]:= Clear[y];  
          Dt[x^2 + y^2 + z^2, x, Constants -> {z}]  
  
Out[21]= 2 x + 2 y Dt[y, x, Constants -> {z}]  
  
Time: 0.02 seconds    100%
```

## 第5章：微积分的基本操作

( VI )  $\text{Dt}[f, x_1, x_2, \dots, x_n]$  计算多重全微分  $\frac{d^n f}{dx_1 dx_2 \dots dx_n}$ .



```
多变量函数的微分.nb *
In[26]:= z = x^3 y + x^2 y^2 - 3 x * y^2;
(* 计算多重全微分  $\frac{dz}{dx dy}$  *)

In[27]:= Dt[z, x, y]

Out[27]= 3 x^2 - 6 y + 4 x y + 6 x y Dt[x, y] + 2 y^2 Dt[x, y] -
6 x Dt[y, x] + 2 x^2 Dt[y, x] + 3 x^2 Dt[x, y] Dt[y, x] -
6 y Dt[x, y] Dt[y, x] + 4 x y Dt[x, y] Dt[y, x]

Time: 0.05 seconds 100%
```

### 5.5 多变量函数的积分 (重积分)

多变量函数的积分类似于一元函数的积分, 可以利用Integrate函数来完成. 命令如下:

$\text{Integrate}[f, \{x, a, b\}, \{y, c, d\}, \dots, \{z, m, n\}]$  计算重积分  $\int_a^b \int_c^d \dots \int_m^n f(x, y, \dots, z) dz \dots dy dx$  .

$\text{NIntegrate}[f, \{x, a, b\}, \{y, c, d\}, \dots, \{z, m, n\}]$  数值积分或重积分的数值解 .

## 第5章：微积分的基本操作

```

多变量函数的积分.nb *
(* 计算重积分  $\iint \frac{1}{x^2+y+1} dx dy$  *)

In[35]=  $\int \int \frac{1}{x^2+y+1} dx dy$ 
Out[35]=  $2\sqrt{1+y} \operatorname{ArcTan}\left[\frac{x}{\sqrt{1+y}}\right] + x \operatorname{Log}[1+x^2+y]$ 

(* 我们也可以直接输入Integrate命令进行积分, 但要注意x与y的顺序 *)

In[36]= Integrate[1/(x^2+y+1), y, x]
Out[36]=  $2\sqrt{1+y} \operatorname{ArcTan}\left[\frac{x}{\sqrt{1+y}}\right] + x \operatorname{Log}[1+x^2+y]$ 

(* 计算二重积分  $\int_0^a \int_0^b (x^2+y^2) dx dy$  *)

In[37]= Integrate[x^2+y^2, {x, 0, a}, {y, 0, b}]
Out[37]=  $\frac{a^3 b}{3} + \frac{a b^3}{3}$ 

(* y的积分限也可以是x的函数 *)

In[38]= Integrate[x^2+y^2, {x, 0, a}, {y, 0, x^2}]
Out[38]=  $\frac{a^5}{5} + \frac{a^7}{21}$ 

Time: 0.09 seconds
    
```

```

多变量函数的积分.nb *
(* 在重积分中, 无法求出某个变量的积分值, Mathematica
   会求出可积分的部分, 再输出运算的结果 *)

In[52]= Integrate[Sqrt[x+y], {x, 0, 2}, {y, 0, Sqrt[x+2]}]
Out[52]=  $\int_0^2 \left(-\frac{2x^{3/2}}{3} + \frac{2}{3}(x+\sqrt{2+x})^{3/2}\right) dx$ 

In[53]= (* 将上式转换成数值解 *)
N[%]
Out[53]= 4.65557

(* 直接利用NIntegrate命令求解, 也可以得到相同的答案 *)

In[54]= NIntegrate[Sqrt[x+y], {x, 0, 2}, {y, 0, Sqrt[x+2]}]
Out[54]= 4.65557

(* 以下是一个三重积分  $\int_{-2}^2 \int_{\ln^2}^4 \int_{-\sqrt{y-x^2}}^{\sqrt{y-x^2}} \sqrt{x^2+z^2} dz dy dx$  *)

In[55]= Off[NIntegrate::slwcon]
NIntegrate[Sqrt[x^2+z^2], {x, -2, 2}, {y, x^2, 4},
           {z, -Sqrt[y-x^2], Sqrt[y-x^2]}]
Out[56]= 26.8083

(* 注: 命令 Off[NIntegrate::slwcon]的作用是不显示提示信息 *)

Time: 13.07 seconds
    
```

# 第6章：微分方程的求解

## 6. 1 微分方程解

在Mathematica中使用**Dsolve[ ]**可以求解线性和非线性微分方程，以及联立的微分方程组。在没有给定方程的初值条件下，我们所得到的解包括C[1],C[2]是待定系数。求解微分方程就是寻找未知的函数的表达式，未稳中有降函数用y[x]表示，其微分用y'[x],y''[x]等表示。

Dsolve[eqn, y[x], x]

求解微分方程y[x]

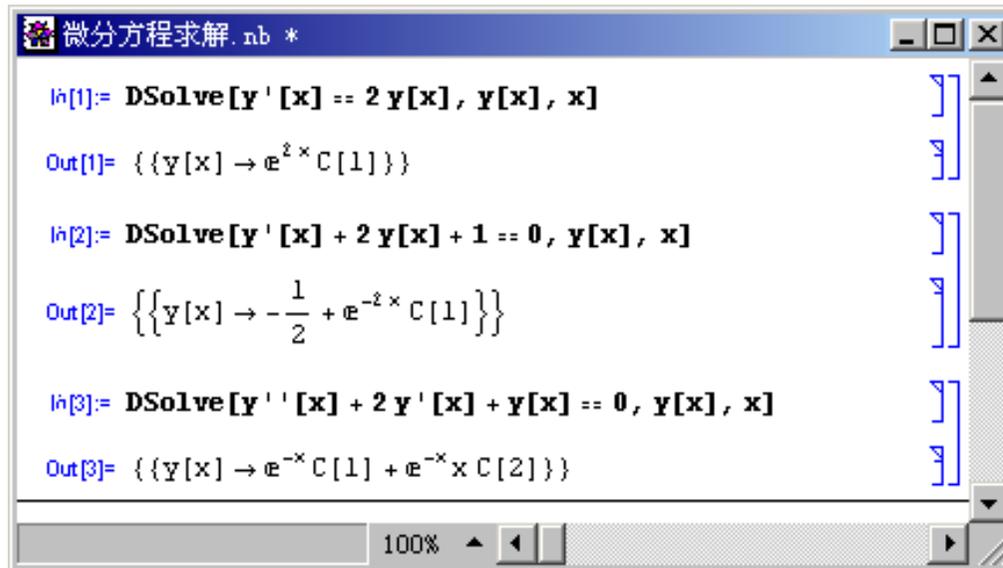
Dsolve[eqn, y, x]

求解微分方程函数y

Dsolve[{eqn1, eqn2, ...}, {y1, y2, ...}, x]

求解微分方程组

### 1. 用Dsolve求解微分方程y[x]



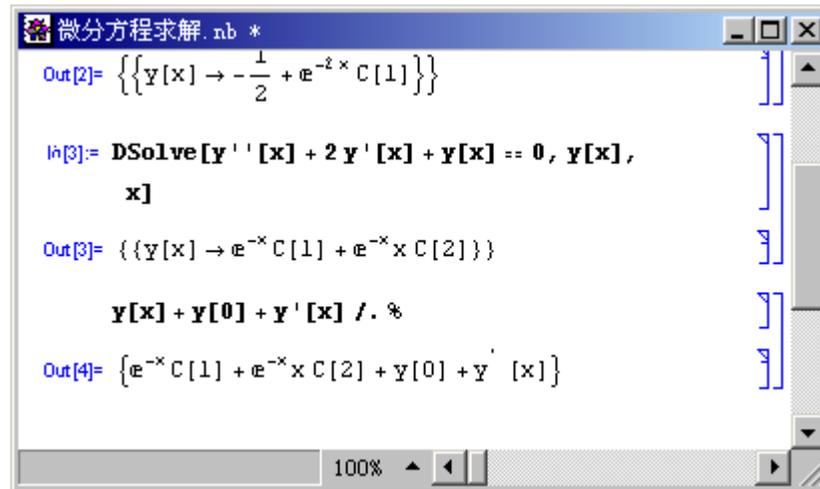
```
微分方程求解.nb *
In[1]:= DSolve[y' [x] == 2 y[x], y[x], x]
Out[1]= {{y[x] -> e2 x C[1]}}

In[2]:= DSolve[y' [x] + 2 y[x] + 1 == 0, y[x], x]
Out[2]= {{y[x] -> -1/2 + e-2 x C[1]}}

In[3]:= DSolve[y' ' [x] + 2 y' [x] + y[x] == 0, y[x], x]
Out[3]= {{y[x] -> e-x C[1] + e-x x C[2]}}
```

## 第6章：微分方程的求解

解 $y[x]$ 仅适合其本身，并不适合于 $y[x]$ 的其它形式，如 $y'[x]$ ， $y[0]$ 等，也就是说 $y[x]$ 不是函数，例如我们如果有如下操作， $y'[x]$ ， $y[0]$ 并没有发生变化。



```
微分方程求解.nb *
Out[2]= {{Y[x] -> -1/2 + e^{-2*x} C[1]}}

In[3]:= DSolve[Y''[x] + 2 Y'[x] + Y[x] == 0, Y[x],
             x]
Out[3]= {{Y[x] -> e^{-x} C[1] + e^{-x} x C[2]}}

Y[x] + Y[0] + Y'[x] /. %
Out[4]= {e^{-x} C[1] + e^{-x} x C[2] + Y[0] + Y'[x]}
```

### 2. 解的纯函数形式

使用Dsolve命令可以给出解的纯函数形式，

## 第6章：微分方程的求解

例如

```
微分方程求解.nb *  
In[4]:= DSolve[y' [x] == 2 y[x], y[x], x]  
Out[4]:= {{Y[x] -> e2 x C[1]}}
```

```
In[5]:= DSolve[y' [x] + 2 y[x] + 1 == 0, y, x]  
Out[5]:= {{Y -> (-1/2 + e-2 #1 C[1] & )}}
```

```
In[6]:= DSolve[y' [x] == 2 y[x], y, x]  
Out[6]:= {{Y -> (e2 #1 C[1] & )}}
```

```
In[7]:= DSolve[y' ' [x] + 2 y' [x] + y[x] == 0, y, x]  
Out[7]:= {{Y -> (e-#1 C[1] + e-#1 C[2] #1 & )}}
```

这里y适合y的所有情况下面的例子可以说明这一点

```
微分方程求解.nb *  
In[13]:= DSolve[y' ' [x] + 2 y' [x] + y[x] == 0, y, x]  
Out[13]:= {{Y -> (e-#1 C[1] + e-#1 C[2] #1 & )}}
```

```
In[14]:= y[x] + y' [x] + y[0] /. %  
Out[14]:= {C[1] + e-x C[2]}
```

## 第6章：微分方程的求解

### 3. 求微分方程组

```
In[16]:= DSolve[{y[x] == -z'[x], z[x] == -y'[x]}, {y[x], z[x]}, x]

Out[16]= {{y[x] -> 1/2 e^{-x} (C[1] + e^{2x} C[1] + C[2] - e^{2x} C[2]),
z[x] -> -1/2 e^{-x} (-C[1] + e^{2x} C[1] - C[2] - e^{2x} C[2])}}
```

### 4. 带初始条件的微分方程的解

```
In[18]:= DSolve[{y'[x] == y[x], y[0] == 5}, y[x], x]

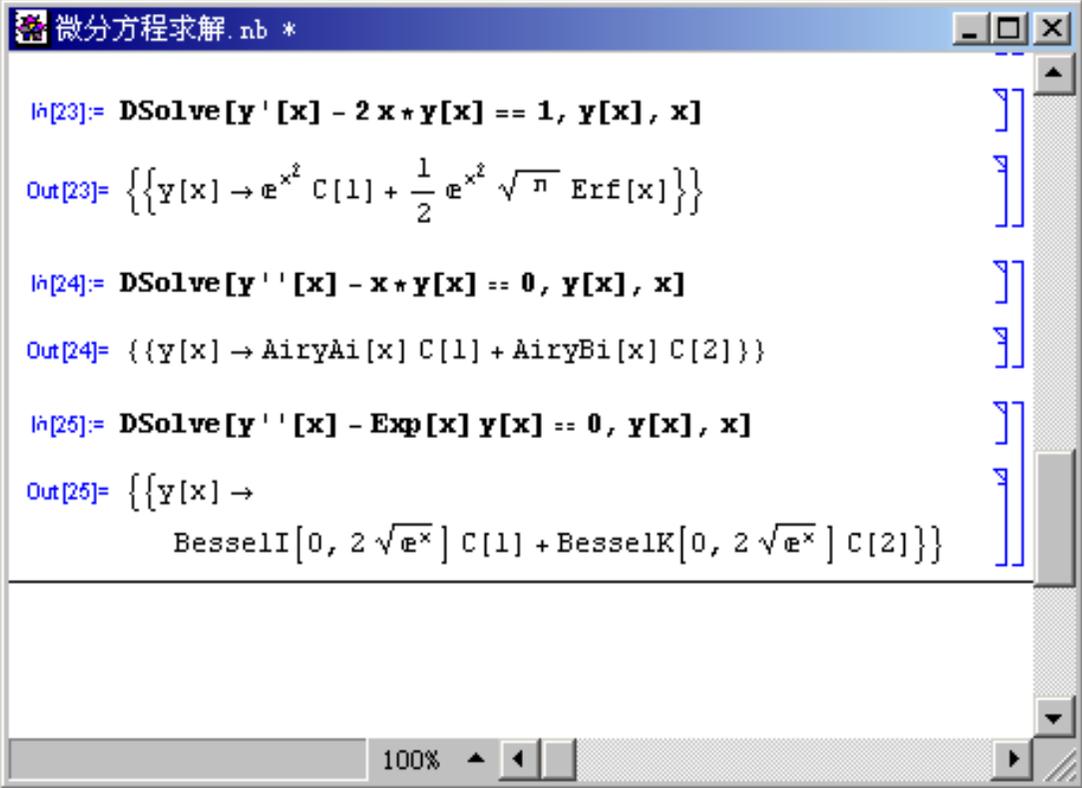
Out[18]= {{y[x] -> 5 e^x}}

In[19]:= DSolve[{y''[x] == y[x], y'[0] == 0}, y[x], x]

Out[19]= {{y[x] -> e^{-x} (C[2] + e^{2x} C[2])}}
```

## 第6章：微分方程的求解

对于简单的微分方程的解比较简单，对一些微分方程它的解就复杂的多。特别是对一些微分方程组或高阶微分方程，不一定能得具体的解，其解中可能含有一些特殊函数。并且很多特殊函数的提出就是为了解这些方程的如：



```
微分方程求解.nb *

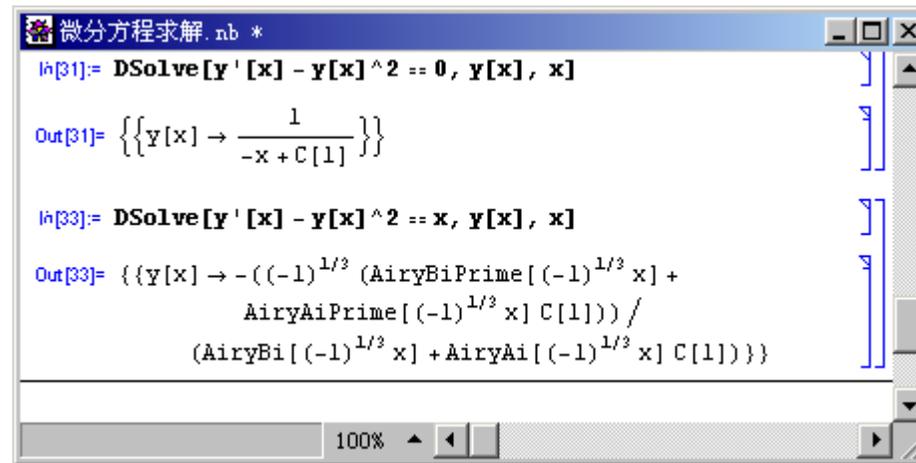
In[23]:= DSolve[y' [x] - 2 x * y[x] == 1, y[x], x]
Out[23]= {{y[x] -> e^{x^2} C[1] + \frac{1}{2} e^{x^2} \sqrt{\pi} Erfi[x]}}

In[24]:= DSolve[y'' [x] - x * y[x] == 0, y[x], x]
Out[24]= {{y[x] -> AiryAi[x] C[1] + AiryBi[x] C[2]}}

In[25]:= DSolve[y'' [x] - Exp[x] y[x] == 0, y[x], x]
Out[25]= {{y[x] ->
          BesselI[0, 2 \sqrt{e^x}] C[1] + BesselK[0, 2 \sqrt{e^x}] C[2]}}
```

## 第6章：微分方程的求解

对于非线性微分方程，仅有一些特殊的情况可用标准数学函数得到解。Dsolve能够处理所有在标准数学手册有解的非线性微分方程。



```
微分方程求解.nb *
In[31]:= DSolve[y'[x] - y[x]^2 == 0, y[x], x]
Out[31]= {{Y[x] -> 1/(-x + C[1])}}

In[33]:= DSolve[y'[x] - y[x]^2 == x, y[x], x]
Out[33]= {{Y[x] -> -((-1)^(1/3) (AiryBiPrime[(-1)^(1/3] x] +
    AiryAiPrime[(-1)^(1/3] x] C[1])) /
    (AiryBi[(-1)^(1/3] x] + AiryAi[(-1)^(1/3] x] C[1])}}
```

### 6.2 微分方程的数值解

在Mathematica中用函数DSolve[]得到微分方程的准确解，用函数NDSolve得到微分方程的数值解，当然在此处要给出求解区间（x,xmin,xmax）。

NDSolve也是既能计算单个的微分方程，也能计算联立微分方程组。它可对大多数的常微分方程和部分偏微分方程求解。在常微分可能有一些未知函数 $y_i$ ，但这些未知函数都依赖于一个单变量 $x$ 。

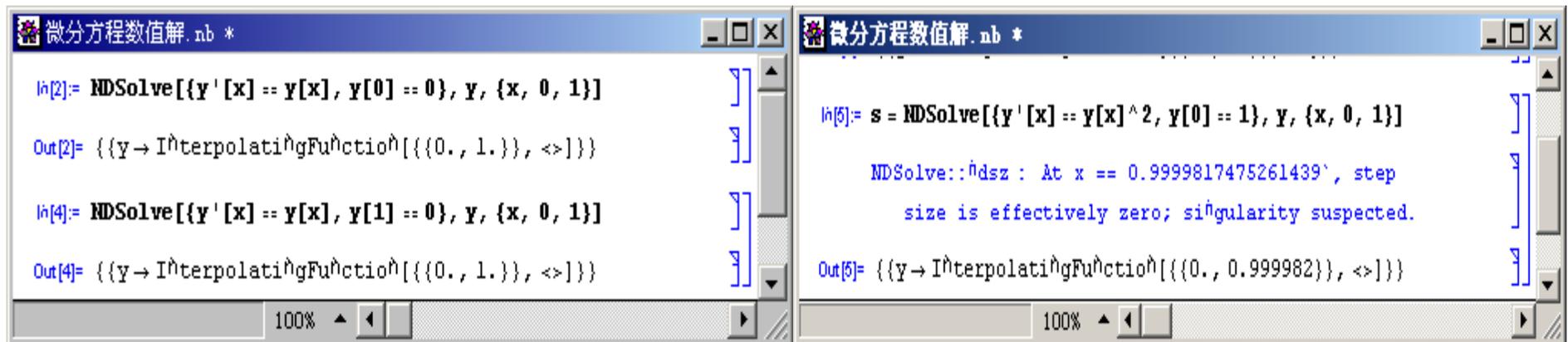
## 第6章：微分方程的求解

**NDSolve[{eqn1,eqn2,...},y,{x,xmin,xmax}]** 求函数y的数值解，x属于[xmin,xmax]

**NDSolve[{eqn1,eqn2,...}, {y1,y2,...}{x,xmin,xmax}]** 求多个函数yi的数值解

DSolve以InterpolatingFunction 目标生成函数yi的解，InterpolatingFunction目标提供在独立变量x的xmin到xmax范围内求出的近似值。NDSolve用迭代法求解，它以某一个x值开始，尽可能覆盖从xmin到xmax的全区间。

为使迭代开始，NDSolve指定yi及其导数为初始条件。初始条件给定某定点x处的yi[x]及尽可能的导数y'i[x]，一般情况下，初始条件可在任意x处，NDSolve将以此为起点自动覆盖xmin到xmax的全区域。下面对初始条件y[0]=0和y[1]=0分别求出x从0到1的范围内y'[x]=y[x]的解。



```
In[2]:= NDSolve[{y'[x] == y[x], y[0] == 0}, y, {x, 0, 1}]
Out[2]:= {{y -> InterpolatingFunction[{{0., 1.}}, <>]}}

In[4]:= NDSolve[{y'[x] == y[x], y[1] == 0}, y, {x, 0, 1}]
Out[4]:= {{y -> InterpolatingFunction[{{0., 1.}}, <>]}}

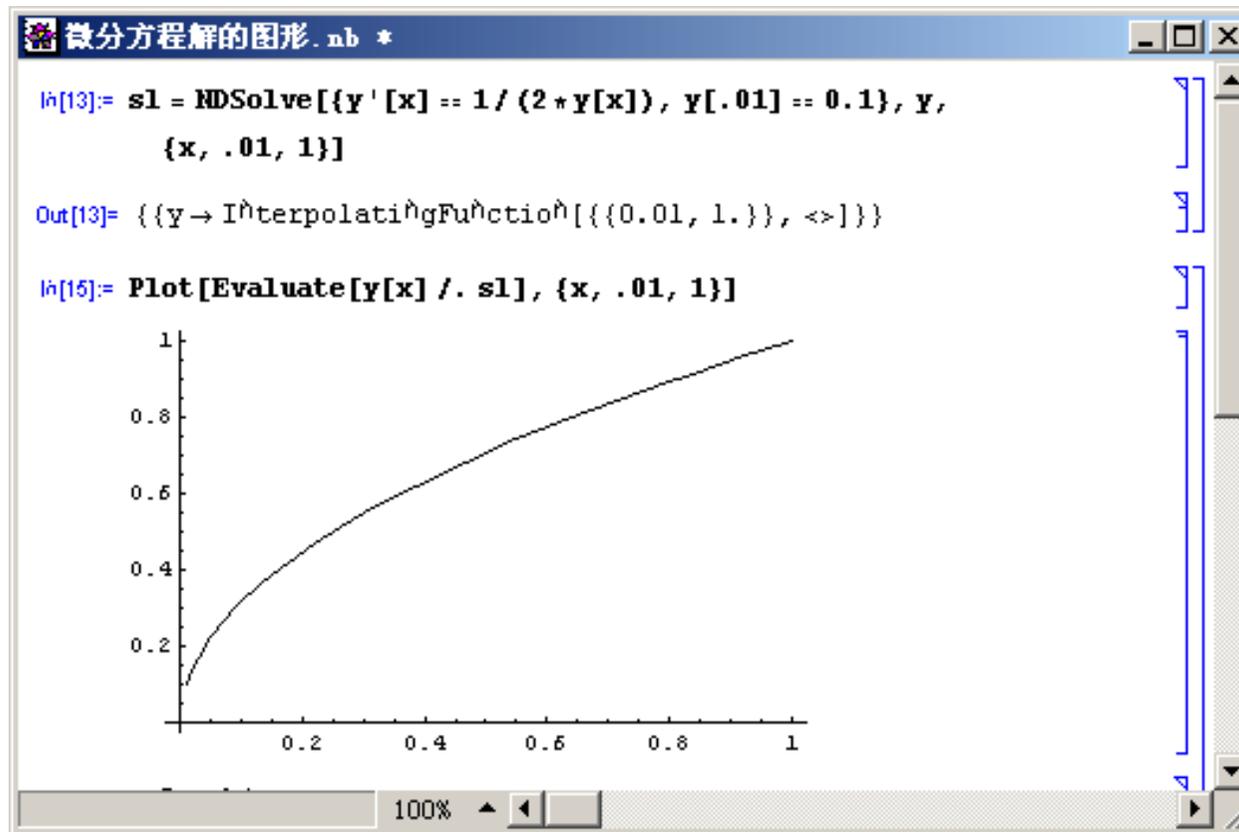
In[5]:= s = NDSolve[{y'[x] == y[x]^2, y[0] == 1}, y, {x, 0, 1}]
NDSolve::npsz: At x == 0.9999817475261439, step size is effectively zero; singularity suspected.
Out[5]:= {{y -> InterpolatingFunction[{{0., 0.999982}}, <>]}}
```

## 第6章：微分方程的求解

使用Mathematica可以很容易的得到解的图形。这儿给出如何观察微商的逆函数的近似值图形。

我们使用命令**Evaluate**代替**InterpolatingFunction**能够节省时间。

例如：



# 第7章：Mathematica程序设计

## 7.1 模块和块中的变量

为了使Mathematica更有效的工作,我们可对Mathematica进行模块化运算。在模块内部通过编写一系列表达式语句,使其实现一定的功能。在Mathematica内部也提供了很多程序包,我们将学习如何调用它们。

一般情况下,Mathematica假设所有变量都为全局变量。也就是说无论何时你使用一个你定义的变量,Mathematica都假设你指的是同一个目标。然而在编制程序时,你则不会想把所有的变量当作全局变量,因为如果这样程序可能就不具有通用性,你也可能在调用程序时陷入混乱状态。给出定义模块或块和局部变量的常用形式:

<code>Module[{x, y, ...}, body]</code>	具有局部变量x, y...的模块
<code>Module[{x=x0, y=y0, ...}, body]</code>	具有初始值的局部变量的模块
<code>lhs:=Module[vars, rhs/:cond]</code>	rhs和cond共享局部变量
<code>Block[{x, y, ... }, body]</code>	运用局部值x, y, ...计算body
<code>Block[{x=x0, y=y0, ...}, body]</code>	给x, y,.. 赋初始值

## 第7章：Mathematica程序设计



Mathematica中的模块工作很简单,每当使用模块时,就产生一个新的符号来表示它的每一个局部变量。产生的新符号具有唯一的名字,互不冲突,有效的保护了模块内外的每个变量的作用范围。

首先我们来看**Module**函数,这个函数的第一部分参数,里说明的变量只在Module内起作用,body执行体,包含合法的Mathematica语句,多个语句之间可用“;”分割下面定义有初值的变量t,Mathematica默认它为全局变量:

```
In[1]:=t=10  
Out[1]=10
```

模块中的t为局部变量,因此它独立于全局变量t

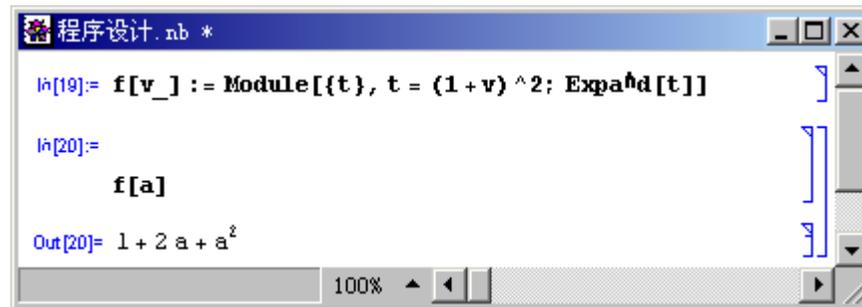
```
In[2]:=Module[{t},t=8;Print[t]]
```

全局变量t的值仍为10

```
In[3]:=t=10  
Out[3]=10
```

## 第7章：Mathematica程序设计

下面定义函数中的中间变量t为局部变量并调用f:



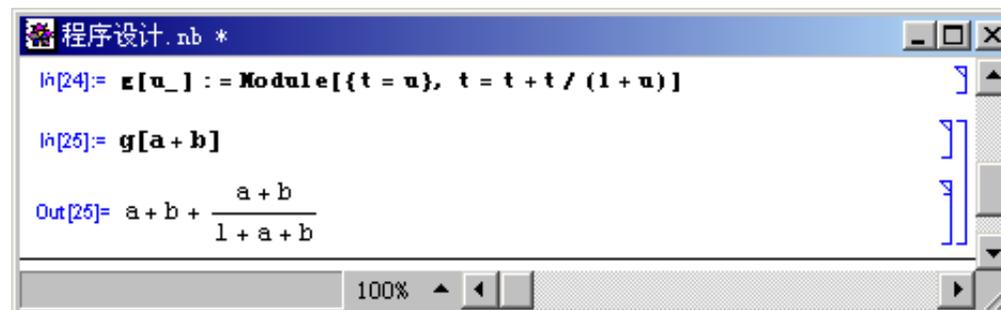
```
程序设计.nb *  
In[19]:= f[v_] := Module[{t}, t = (1+v)^2; Expand[t]  
In[20]:= f[a]  
Out[20]= 1 + 2 a + a^2
```

全局变量t的值仍为10

**In[6]:=t=10**

**Out[6]=10**

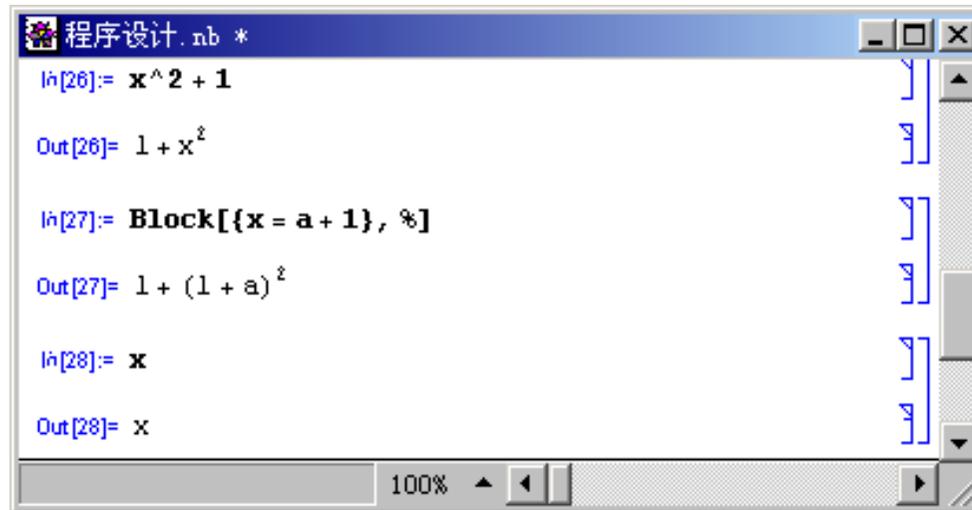
我们可以对模块中的任意局部变量进行初始化, 这些初始值总是在模块执行前就被计算出来。下面给局部变量t赋初值u:调用函数g:



```
程序设计.nb *  
In[24]:= g[u_] := Module[{t = u}, t = t + t / (1 + u)]  
In[25]:= g[a + b]  
Out[25]= a + b +  $\frac{a + b}{1 + a + b}$ 
```

## 第7章：Mathematica程序设计

Mathematica中的模块允许你把某变量名看作局部变量名。然而又存在有时你又希望它们为全局变量时,但变量值为局部的矛盾,这时我们可以用Block[]函数。下面是一个含有全局变量x表达式,使用x的局部值计算上面的表达式:



```
程序设计.nb *
In[26]:= x^2 + 1
Out[26]= 1 + x^2

In[27]:= Block[{x = a + 1}, %]
Out[27]= 1 + (1 + a)^2

In[28]:= x
Out[28]= x
```

在Mathematica中编制程序时,必须使程序中的各个部分尽可能的独立,这样程序才便于读懂、维护和修改。确保程序各部分不相干的主要方法是设置具有一定作用域的变量。在Mathematica中有两种限制变量作用域的基本方法:模块(Module)和块(Block)。我们在书写实际程序中,模块比块更具普遍性。然而在交互式计算中需要定义作用域时,块更实用。

## 第7章：Mathematica程序设计

**Module[vars,body]**所要做的是把执行模块时表达式**body**的形式看成Mathematica程序的“代码”。然而当“代码”中直接出现变量**vats**时，这些**vars**都将被看作局部的。**Block[vats,body]**并不查看表达式**body**的形式，而在整个计算**Body**的过程中，实用**vars**的局部值。

下例中我们根据*i*定义*m*:

```
In[12]:=m=i^2
```

```
Out[12]=i2
```

在计算*i+m*的整个过程中使用块中*i*的局部值:

```
In[13]:=Block[{i=a}, i+m]
```

```
Out[13]=a+a2
```

而对于下面的例子，只有直接出现在*i+m*中的*i*，才被看作局部变量:

```
In[14]:=Module[{i=a}, i+m]
```

```
Out[14]=a+i2
```

# 第7章：Mathematica程序设计

## 7.2 条件结构

我们在用计算机语言进行编程时，常用到条件语句。在Mathematica中也提供了多种设置条件的方法，并规定只有在该条件满足时才计算表达式。

下面条件结构的常用形式。

<code>lhs:=rhs1/:test</code>	当test为真时使用定义
<code>If[test, then, else]</code>	如test为真计算then，反之计算else
<code>which[test1, value1, test2, ...]</code>	依次计算test1，给出对应的第一个为真的值
<code>Switch[expr, form1, value1, form2, ...]</code>	expr与每一个formi相比较，给出第一个相匹配的值
<code>Switch[expr, form1, value1, form2, ..., _, def]</code>	用def为系统默认值

### 1.If命令

下面的test为真,故返回第一表达式的值:

```
In[1]:=If [1>0,1+2,2+3]  
Out[1]=3
```

## 第7章：Mathematica程序设计

用Mathematica编程时,不可避免的要在单个或多个定义之间进行选择。单个定义的右边包含多个由If函数控制的分支,多个定义是用/;condition来表示的。运用多个定义进行编程你常能得到结构很好的程序。

下面定义了一个阶跃函数,即当 $x>0$ 时值为1,反之值为-1:

```
In[2]:=If[x>0,1,-1]
```

运用/; condition形式分别定义阶跃函数的正数和负数部分:

```
In[3]:g=1/:x>0
```

```
In[4]:g=-1/:x<0
```

用“?”显示用/:condition定义的函数g的完整信息:

```
In[5]:=?g
```

```
Global`g
```

```
g[x_]:=1/:x>0
```

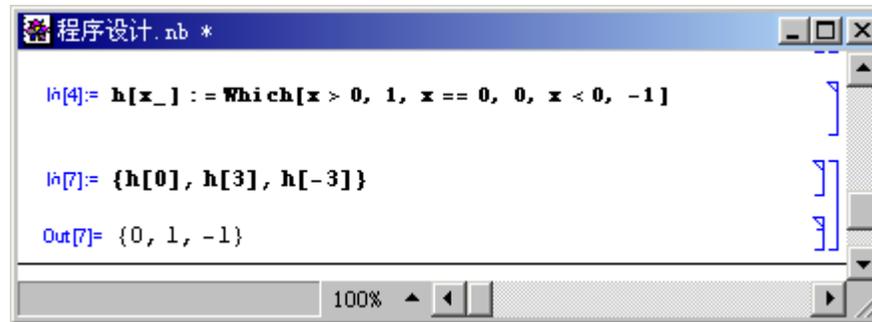
```
g[x_]:=-1 / :x<0
```

我们用函数If时,还可以用if(test,expr)结构,即当test真时,计算表达式expr,表达式expr的值就是整个If结构的值,反之返回空值。

## 第7章：Mathematica程序设计

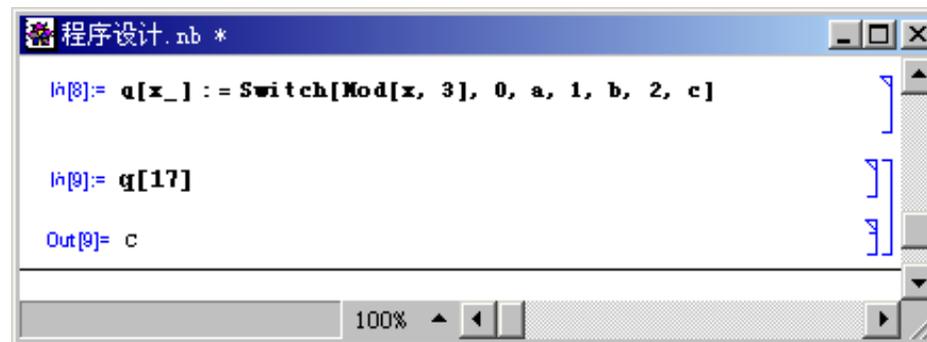
### 2. Which命令

对于一般情况函数If提供一个两者择一的方法。然而,有时条件多于两个,在这种情况下可用If函数的嵌套方式来处理,但在这种情况下使用Which或Switch函数将更合适。下面用Which定义具有三个条件的函数,调用这个函数:



```
程序设计.nb *  
  
In[4]:= h[x_] := Which[x > 0, 1, x == 0, 0, x < 0, -1]  
  
In[7]:= {h[0], h[3], h[-3]}  
  
Out[7]= {0, 1, -1}
```

用Switch定义一个与模的余数有关的函数:



```
程序设计.nb *  
  
In[8]:= a[x_] := Switch[Mod[x, 3], 0, a, 1, b, 2, c]  
  
In[9]:= a[17]  
  
Out[9]= c
```

Mod[17,3]=2,因此运用了Switch中的第三种情况

## 第7章：Mathematica程序设计

### 3.符号条件

在Mathematica中,有一种可能的情况就是你给出的条件结果既不是真也不为假。下面测试的结果既不是真也不是假,因此If的两个分支保持不变:

```
In[1]:=If[x==y,a,b]
Out[1]:If[x==y,a,b]
```

可以给If加上第三个条件结果,这允许你测试的结果既不是真也不是假的情况下使用它:

```
In[2]:=If[x==y,a,b,c]
Out[2]=c
```

但Mathematica在下面情况下以符号等式输出:

```
In[4]:=x==y
Out[4]:=x==y
```

除非表达式能得出真,否则都被假设为假:

```
In[5]:=TrueQ[x==x]
Out[5]=True
In[6]:=TrueQ[x==y]
Out[6]=false
```

## 第7章：Mathematica程序设计

我们用“**===**”可直接测试两个表达式的等同性:

```
In[7]:x===y
Out[7]:=False
```

一般情况下,“**===**”返回值为真(True)或假(False),而“**==**”为符号形式输出,表示一个符号等式。在特殊情况下可用“**===**”测试一个表达式的结构,而用“**==**”测试数学上的等同性。

### 4.是逻辑表达式的运算形式

```
expr1&&expr2&&expr3
```

计算expr<sub>i</sub>,直到其中有一个为假为止

```
expr1||expr2||expr3
```

计算expr<sub>i</sub>,直到其中有一个为真为止

下面的函数包括两个组合条件:

```
In[10]:=t[x_]:= (x!=0&&1/x<3)
```

对这两个测试条件进行计算,下面的第一次测试得出为假,因此不进行第二个条件的测试,第二测试结果可能为1或0,因此 输出结果为假:

```
In[12]:=t[0]
Out[12]=False
```

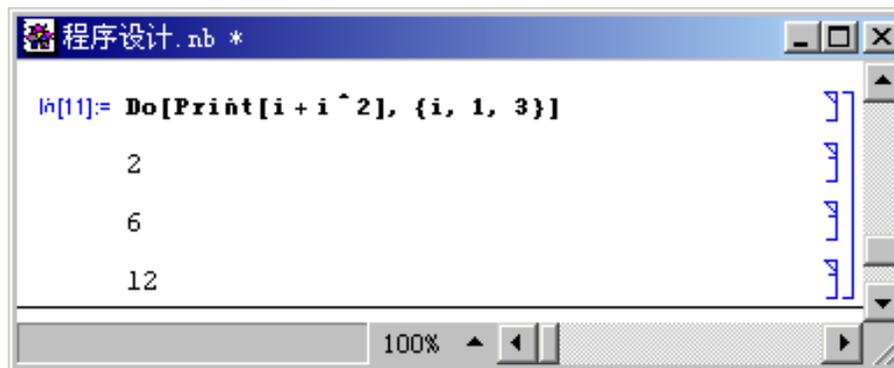
## 第7章：Mathematica程序设计

### 7.3 循环结构

简单地Do循环结构形式:

<code>Do[expr, {i, imax}]</code>	循环计算expr, 以步长1, i从1增加到imax
<code>Do[expr, {i, imin, imax, di}]</code>	循环计算expr, 以步长di, i从imin增加到imax
<code>Do[expr, {n}]</code>	循环计算expr n次

计算Print[i+i<sup>2</sup>],i从1增加到3:



```
In[11]:= Do[Print[i + i^2], {i, 1, 3}]
```

```
2
```

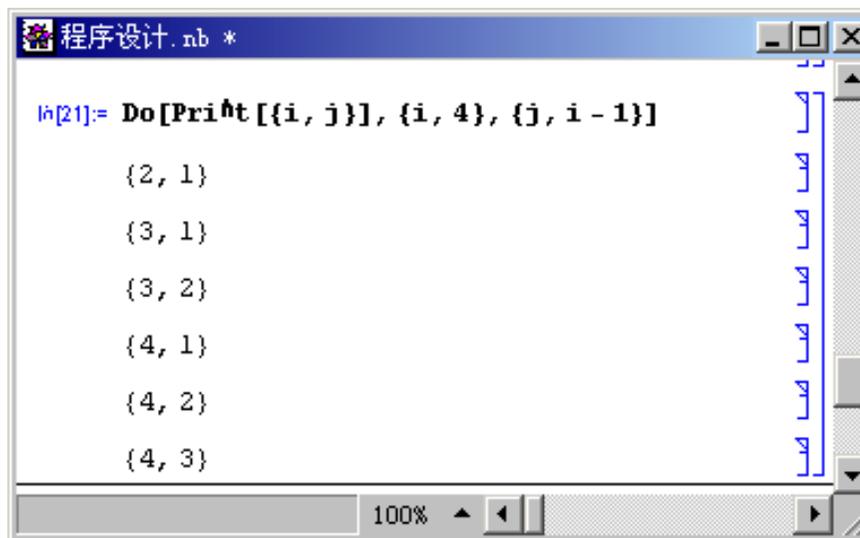
```
6
```

```
12
```

Do中的定义的循环方式与函数Table和Sum中的定义一样。在函数Do中, 你同样能建立重循环。

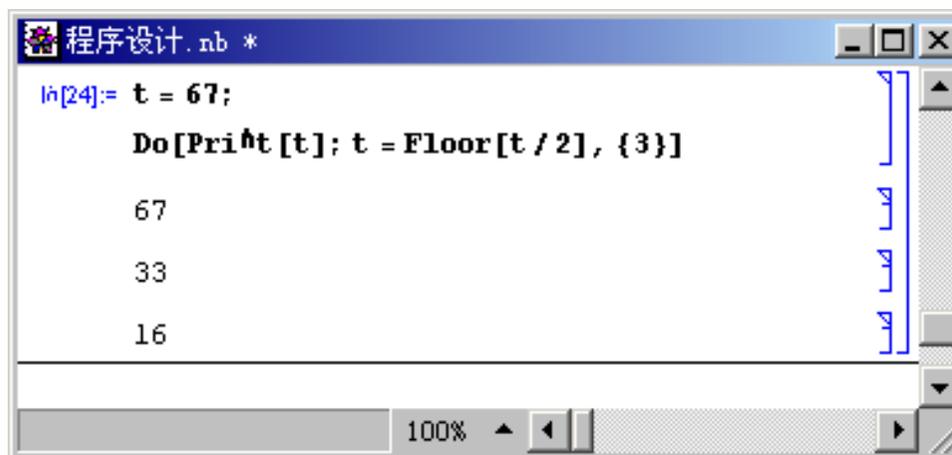
## 第7章：Mathematica程序设计

下面给出的 $i$ 从1到4进行循环，  
而对于每个 $i$ ， $j$ 又从1到 $i-1$ 进行循环：



```
程序设计.nb *  
  
In[21]:= Do[Print[{i, j}], {i, 4}, {j, i - 1}]  
  
{2, 1}  
  
{3, 1}  
  
{3, 2}  
  
{4, 1}  
  
{4, 2}  
  
{4, 3}
```

我们还可把一个过程放入Do函数中：



```
程序设计.nb *  
  
In[24]:= t = 67;  
Do[Print[t]; t = Floor[t / 2], {3}]  
  
67  
  
33  
  
16
```

# 第7章：Mathematica程序设计

## 2.While与For结构

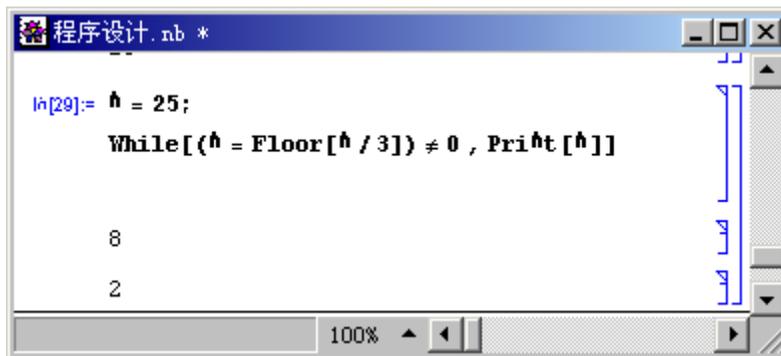
While[test, body]

只要test为真，就重复计算body

For[start, test, incr, body]

以为start起始值，重复计算body和incr，直到test为假为止

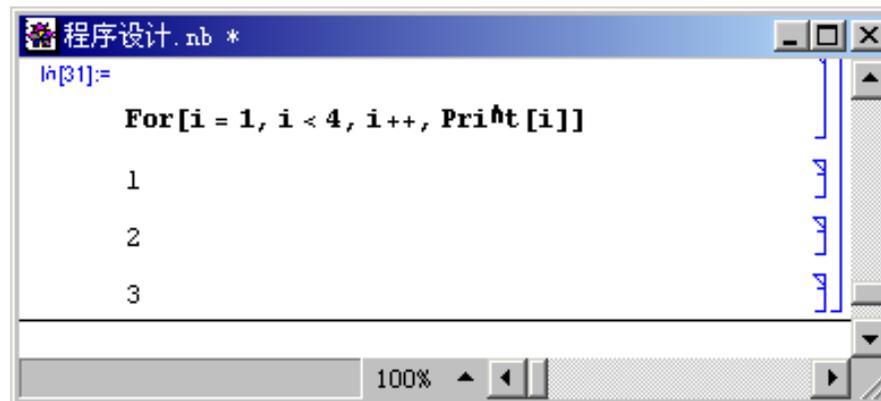
当条件满足时，While循环一直进行,因此为了防止死循环，在While 中应包括命令能改变test的值。



```
程序设计.nb *
In[29]:= h = 25;
         While[{h = Floor[h / 3]} ≠ 0, Print[h]]

8
2
```

下面给出For循环的例子，  
i++表示i的值加1

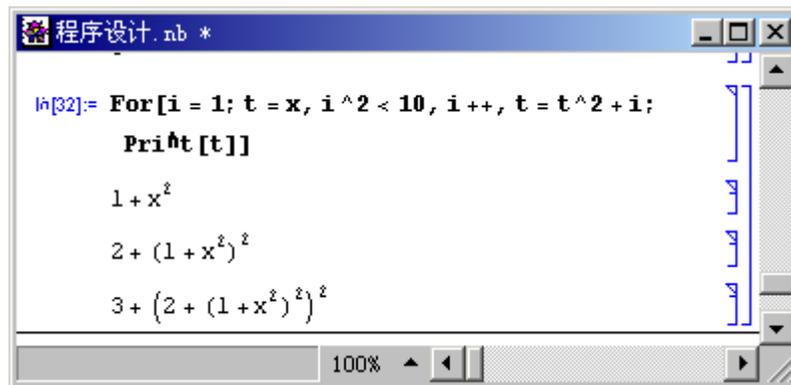


```
程序设计.nb *
In[31]:=
         For[i = 1, i < 4, i++, Print[i]]

1
2
3
```

## 第7章：Mathematica程序设计

下面再给出一个较复杂的For循环的例子，一旦 $i^2 < 10$ 不成立，就中止循环：



```
程序设计.nb *  
In[32]:= For[i = 1; t = x, i^2 < 10, i++, t = t^2 + i;  
Print[t]]  
1 + x2  
2 + (1 + x2)2  
3 + (2 + (1 + x2)2)2  
100%
```

### 3. 一些特殊的赋值方式

<code>i++</code>	变量i加1
<code>i--</code>	变量i减1
<code>++i,</code>	变量i先加1
<code>--i</code>	变量i先减1
<code>i+=di</code>	i加di
<code>i-=di</code>	i减di
<code>x*=C</code>	x乘以C
<code>x /=c</code>	x除以c
<code>{x, y} = {y, x}</code>	交换x和y值

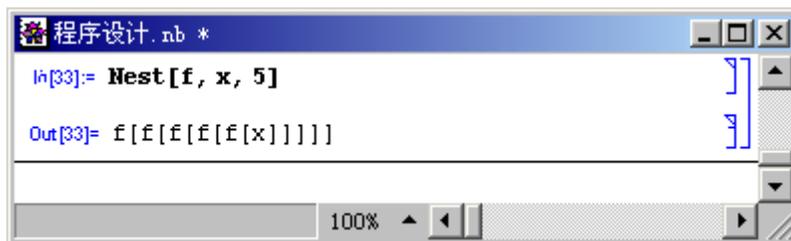
## 第7章：Mathematica程序设计

### 4. 重复运用函数

我们除了可用Do、While、For等进行循环计算外，我们还可以运用函数进行编程。运用函数编程结构你能得出非常有效的程序。例如Nest[f,x,n]允许你对某一表达式重复运用函数f

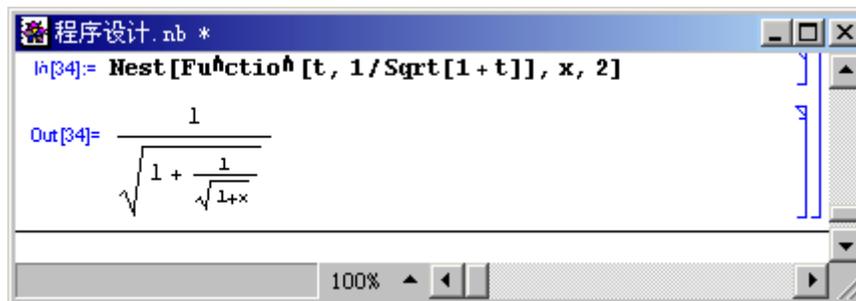
Nest[expr, n]	对表达式expr重复调用函数fn次
FixedPoint[y, expr]	对表达式expr重复调用函数fn次，直到结果不变为止
NestWhile[f, expr, test]	对表达式expr重复调用函数f，直到产生的结果为假时为止

对函数f迭代5次：



```
程序设计.nb *
In[33]:= Nest[f, x, 5]
Out[33]= f[f[f[f[f[x]]]]]
```

对纯函数进行迭代，  
能得出与运用Do函数得出的结果一样：

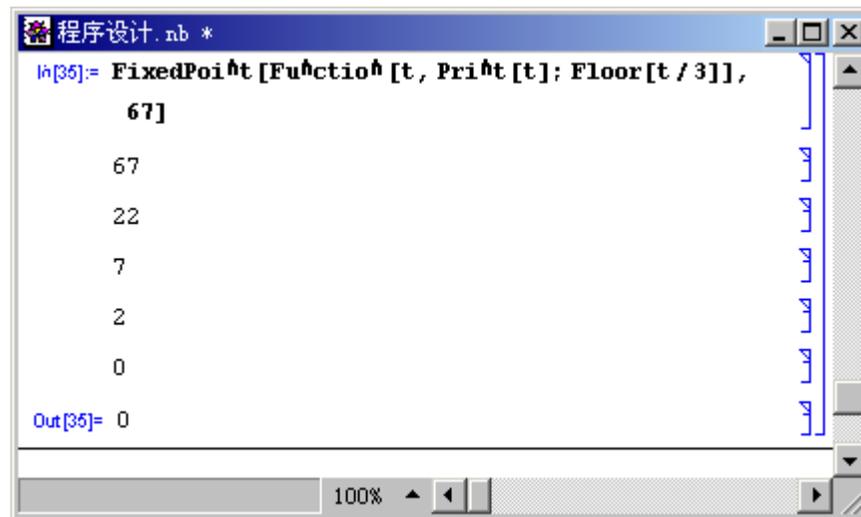


```
程序设计.nb *
In[34]:= Nest[Function[t, 1/Sqrt[1+t]], x, 2]
Out[34]= 
$$\frac{1}{\sqrt{1 + \frac{1}{\sqrt{1+x}}}}$$

```

## 第7章：Mathematica程序设计

**Nest**函数允许你重复运用某函数。然而，有时你想在结果不再发生变化的情况下就中止对函数的使用，此时立刻使用函数**FixedPoint[f,x]**。**FixedPoint**函数重复运用某一函数直到结果不再发生变化：



The screenshot shows a Mathematica notebook window titled "程序设计.nb \*". The input cell contains the code: `In[35]= FixedPoint[Function[t, Print[t]; Floor[t/3]], 67]`. The output cell shows the sequence of values printed: `67`, `67`, `22`, `7`, `2`, `0`, and `0`. The final output is `Out[35]= 0`. The notebook interface includes a scroll bar on the right and a status bar at the bottom showing "100%" zoom.

### 7.4 流程控制

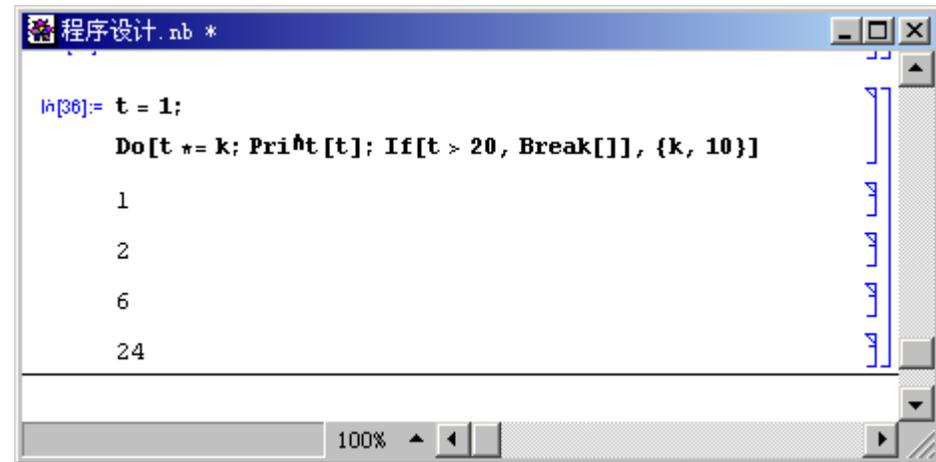
函数程序结构的流程控制一般来说比较简单，但是在应用**While**或**For**等循环时就比较复杂了，这是因为他们的流程控制依赖于表达式的值。而且在这样的循环中，流程的控制并不依赖于循环体中表达式的值。有时你在编制**Mathematica**程序时，在该程序中，流程控制受某一过程或循环体执行结果的影响。这时，我们可用**Mathematica**提供的流程控制函数来控制流程。这些函数的工作过程与**C**语言中的很相似。

## 第7章：Mathematica程序设计

常用的流程控制函数

Break[]	退出本层的循环
Continue[]	转入当前循环的下一步
Return[expr]	退出函数中的所有过程及循环，并返回expr值
Goto[name]	转入当前过程中的元素Label[name]
Throw[value]	返回expr值

当 $t > 20$ 时，Break[]就引起循环体的中断：

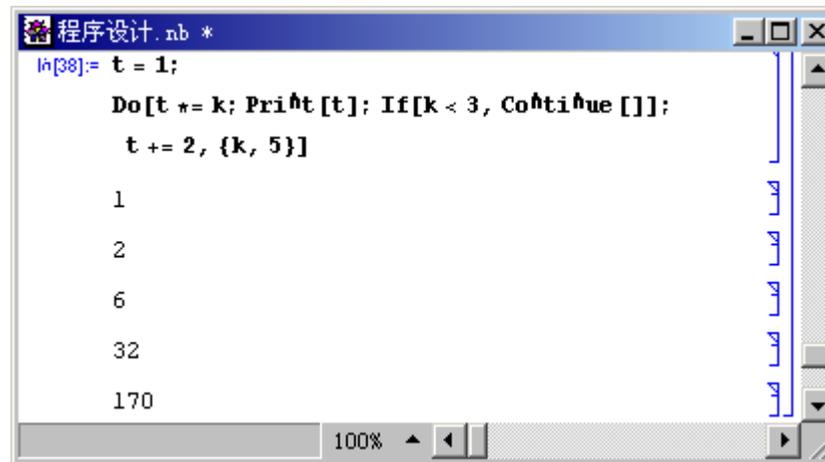


```
In[36]:= t = 1;
Do[t *= k; Print[t]; If[t > 20, Break[]], {k, 10}]

1
2
6
24
```

## 第7章：Mathematica程序设计

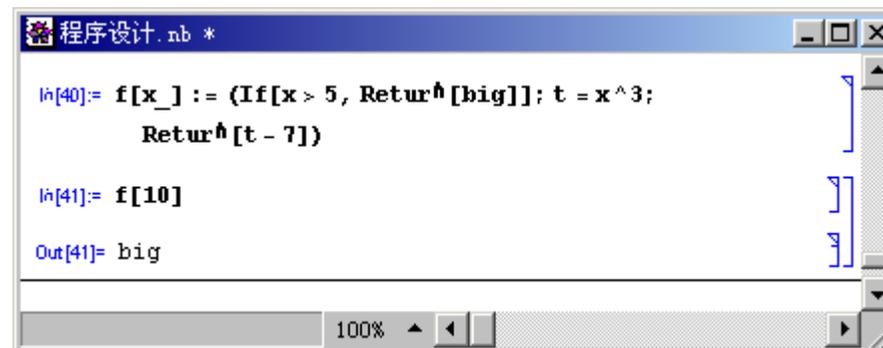
当 $k < 3$ 时，Continue[]继续执行循环：



```
程序设计.nb *
In[38]:= t = 1;
Do[t *= k; Print[t]; If[k < 3, Continue[]];
  t += 2, {k, 5}]

1
2
6
32
170
```

给出Return的一个例子：

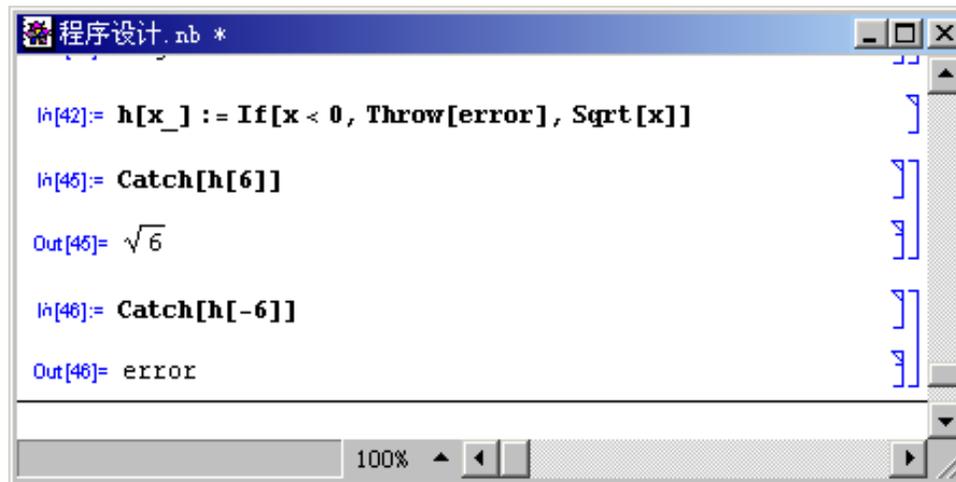


```
程序设计.nb *
In[40]:= f[x_] := (If[x > 5, Return[big]]; t = x^3;
  Return[t - 7])
In[41]:= f[10]
Out[41]= big
```

Return[]允许你退出一函数，并返回一个值。Mathematica可以进行局部返回，这可允许你退出一列迭代函数。非局部返回在错误处理时是很有用的。

## 第7章：Mathematica程序设计

给出的例子中如函数变量小于0则输出error~



```
程序设计.nb *  
  
In[42]:= h[x_] := If[x < 0, Throw[error], Sqrt[x]]  
  
In[45]:= Catch[h[6]]  
Out[45]=  $\sqrt{6}$   
  
In[46]:= Catch[h[-6]]  
Out[46]= error
```

In[6]不产生error，且出示Catch的结果无效：当变量小于0时输出error

# 第8章：Mathematica中的常用函数

## 8.1 运算符及特殊符号和系统常量

### 1.运算符及特殊符号

Line1	执行Line, 不显示结果
Line1, line2	顺次执行Line1, Line2, 并显示结果
?name	关于系统变量name的信息
??name	关于系统变量name的全部信息
!command	执行Dos命令
N!	N的阶乘
!!filename	显示文件内容
<<filename	读入文件并执行
Expr: >>filename	打开文件写
Expr>>>filename	打开文件从文件末写
()	结合率
[]	函数
{}	一个表
<*MathFun*>	在c语言中使用math的函数

## 第8章：Mathematica中的常用函数

### 运算符及特殊符号（继续）

<code>(*Note*)</code>	程序的注释
<code>#n</code>	第n个参数
<code>##</code>	所有参数
<code>Rule&amp;</code>	把rule作用于后面的式子
<code>%</code>	前一次的输出
<code>%%</code>	倒数第二次的输出
<code>Var::mote</code>	变量var的注释
<code>"Astring"</code>	字符串
<code>Context</code>	上下文
<code>A+b</code>	加
<code>a-b</code>	减
<code>A*b</code> 或 <code>ab</code>	乘
<code>A/b</code>	除

## 第8章：Mathematica中的常用函数

### 2.系统常量

Pi	3.1415的无限精度数值
E	2.71828的无限精度数值
Catalan	0.915966Catalan常数
EulerGamma	0.5772Euler常数
Khinchin	2.68545Khinchin
Glaisher	0.915966Glaisher
GoldenRatio	1.61803黄金分割数
Degree	Pi/180角度弧度换算
I	复数单位
Infinity	无穷大
-Infinity	负无穷大
ComplexInfinity	复无穷大
Indeterminate	不定式

# 第8章: Mathematica中的常用函数

## 8.2 代数计算

<code>Expand[expr]</code>	展开表达式
<code>Factor[expr]</code>	展开表达式
<code>Simplify[expr]</code>	化简表达式
<code>FullSimplify[expr]</code>	将特殊函数也进行化简
<code>PowerExpand[expr]</code>	展开所有的幂次形式
<code>ComplexExpand[expr, {x1, x2...}]</code>	按复数实部虚部展开
<code>FunctionExpand[expr]</code>	化简表达式中的特殊函数
<code>Collect[expr, x]</code>	合并同次项
<code>Collect[expr, {x1, x2, ...}]</code>	合并 $x_1, x_2, \dots$ 的同次项
<code>Together[expr]</code>	通分
<code>Apart[expr]</code>	部分分式展开
<code>Apart[expr, var]</code>	对var的部分分式展开
<code>Cancel[expr]</code>	约分
<code>xpandAll[expr]</code>	展开表达式
<code>ExpandAll[expr, patt]</code>	展开表达式
<code>FactorTerms[poly]</code>	提出共有的数字因子
<code>FactorTerms[poly, x]</code>	提出与x无关的数字因子
<code>FactorTerms[poly, {x1, x2...}]</code>	提出与 $x_i$ 无关的数字因子
<code>Coefficient[expr, form]</code>	多项式expr中form的系数
<code>Coefficient[expr, form, n]</code>	多项式expr中 $\text{form}^n$ 的系数
<code>Exponent[expr, form]</code>	表达式expr中form的最高指数
<code>Numerator[expr]</code>	表达式expr的分子
<code>Denominator[expr]</code>	表达式expr的分母
<code>ExpandNumerator[expr]</code>	展开expr的分子部分

## 第8章：Mathematica中的常用函数

### 8.3 解方程

<code>Solve[eqns, vats]</code>	从方程组eqns中解出Vats
<code>Solve[eqns, vats, elims]</code>	从方程组eqns中削去变量elims, 解出vats
<code>DSolve[eqn, y, x]</code>	解微分方程, 其中、y是x的函数
<code>DSolve[{eqn1, eqn2, ...}, {y1, y2...},]</code>	解微分方程组, 其中yi是x的函数
<code>DSolve[eqn, y, {x1, x2...}]</code>	解偏微分方程
<code>Eliminate[eqns, Vats]</code>	把方程组eqns中变量vars约去
<code>SolveAlways[eqns, vars]</code>	给出等式成立的所有参数满足的条件
<code>Reduce[eqns, Vats]</code>	化简并给出所有可能解的条件
<code>LogicalExpand[expr]</code>	用&&和, , 将逻辑表达式展开
<code>InverseFunctionI刀</code>	求函数f的反函数
<code>Root[f, k1]</code>	求多项式函数的第k个根
<code>Roots[lhs==rhs, var]</code>	得到多项式方程的所有根

## 第8章：Mathematica中的常用函数

### 8.4 微积分

<code>D[f, x]</code>	求 $f[x]$ 的微分
<code>D[f, {x, n}]</code>	求 $f[x]$ 的 $n$ 阶微分
<code>D[f, x1, x2...]</code>	求 $f[x]$ 对 $x_1, x_2, \dots$ 的偏微分
<code>Dt[f, x]</code>	求 $f[x]$ 的全微分 $df/dx$
<code>Dt(f)</code>	求 $f[x]$ 的全微分 $df$
<code>Dt[f, {x, n}]</code>	$n$ 阶全微分 $df^n/dx^n$
<code>Dt[f, x1, x2...]</code>	对 $x_1, x_2, \dots$ 的偏微分
<code>Integrate[f, x]</code>	$f[x]$ 对 $x$ 在的不定积分
<code>Integrate[f, {x, xmin, xmax}]</code>	$f[x]$ 对 $x$ 在区间 $(xmin, xmax)$ 的定积分
<code>Integrate[f, {x, xmin, xmax}, {y, ymin, ymax}]</code>	$f[x, y]$ 的二重积分
<code>Limit[expr, x-&gt;x0]</code>	$x$ 趋近于 $x_0$ 时 $expr$ 的极限
<code>Residue[expr, {x, x0}]</code>	$expr$ 在 $x_0$ 处的留数
<code>Series[f, {x, x0, n}]</code>	给出 $f[x]$ 在 $x_0$ 处的幂级数展开
<code>Series[f, {x, x0, nx}, {y, y0, ny}]</code>	先对 $y$ 幂级数展开，再对 $x$ 幂级数展开
<code>Normal[expr]</code>	化简并给出最常见的表达式
<code>SeriesCoefficient[series, n]</code>	给出级数中第 $n$ 次项的系数
<code>SeriesCoefficient[series, {n1, n2...}]</code>	一阶导数
<code>InverseSeries[s, x]</code>	给出逆函数的级数
<code>ComposeSeries[serie1, serie2...]</code>	给出两个基数的组合
<code>SeriesData[x, x0, {a0, a1, ...}, nmin, nmax, den]</code>	表示一个 $x_0$ 处 $x$ 的幂级数
<code>O[x]^n</code>	$n$ 阶小量 $x^n$

# 第8章: Mathematica中的常用函数

## 8.5 多项式函数

<code>Variables[poly]</code>	给出多项式poly中独立变量的列表
<code>CoefficientList[poly, var]</code>	给出多项式poly中变量var的系数
<code>CoefficientList[poly, {var1, var2...}]</code>	给出多项式poly中变量var(i)的系数列
<code>PolynomialMod[poly, m]</code>	poly中各系数mod m同余后得到的多项式, m可为整式
<code>PolynomialQuotient[p, q, x]</code>	以x为自变量的两个多项式之商式p/
<code>PolynomialRemainder[p, q, x]</code>	以x为自变量的两个多项式之余式
<code>PolynomialGCD[poly1, poly2, ...]</code>	poly(i)的最大公因式
<code>PolynomialLCM[poly1, poly2, ...]</code>	poly(i)的最小公倍式
<code>PolynomialReduce[poly, {poly1, Poly2, ...}, {x1, x2...}]</code>	得到一个表I(a1, a2, ...), b)其中SumIa <sub>i</sub> *poly <sub>i</sub> +b=poly
<code>Resultant[poly1, poly2, var]</code>	约去poly1, poly2中的var
<code>Factor[poly]</code>	因式分解(在整式范围内)
<code>FactorTerms[poly]</code>	提出poly中的数字公因子
<code>FactorTerms[poly, {x1, x2...}]</code>	提出poly中与xi无关项的数字公因子
<code>FactorList[poly], FactorSquareFreeList[poly], FactorTermsList[poly, {x1, x2...}]</code>	给出各个因式列表
<code>Cyclotomic[n, x]</code>	n阶柱函数
<code>Decompose[poly, x]</code>	迭代分解, 给出{p1, p2, ...}, 其中P1(p2(...))=poly
<code>InterpolatingPolynomial[data, Var]</code>	在数据data上的插值多项式
<code>RootSum[f, form]</code>	得到f[x]=0的所有根, 并求得Sum[form[xi]]

## 第8章：Mathematica中的常用函数

### 8.6 随机函数

<code>RandomCtype, range]</code>	产生type类型且在range范围内的均匀分布随机数
<code>Random[]</code>	0-1上的随机实数
<code>SeedRandom[n]</code>	以n为seed产生伪随机数
<code>RandomDistribution]</code>	可以产生各种分布

## 第8章：Mathematica中的常用函数

### 8.7 数值函数

<code>N[expr]</code>	表达式的机器精度近似值
<code>N[expr, n]</code>	表达式的n位近似值, n为任意正整数
<code>NSolve[lhs==rhs, val]</code>	求方程数值解
<code>NSolve[eqn, Var, n]</code>	求方程数值解, 结果精度到n位
<code>NDSolve[eqns, y, {x, xmin, xmax}]</code>	微分方程数值解
<code>NDSolve[eqns, {y1, y2, .. 1}, {x, xmin, xmax}]</code>	微分方程组数值解
<code>FindRoot[lhs==rhs, {x, x0}]</code>	以x0为初值, 寻找方程数值解
<code>FindRoot[lhs==rhs, {x, xstart, xmin, xmax}]</code>	以xstart为初值, 在[xmin, xmax]范围内寻找方程数值解
<code>NSum[f, {imin, imax, di}]</code>	数值求和, 出为步长
<code>NSum[f, {imin, imax, di}, {j, ..}, ..]</code>	多维函数求和
<code>NProduct[f, {i, imin, imax, di}]</code>	函数求积
<code>NIntegrate[f, {x, xmin, xmax}]</code>	函数数值积分
<code>FindMinimum[f, {x, x0}]</code>	以x0为初值, 寻找函数最小值
<code>FindMinimum[f, {x, xstart, xmin, xmax}]</code>	以xstart为初值, 在[xmin, xmax]范围内寻找方程解

## 第8章：Mathematica中的常用函数

### 数值函数

<code>ConstrainedMin[f, {inequ}, {x, y, ..}]</code>	inequ为线性不等式组, f为x, y, . 之线性函数, 得到最小值及此时的x, y. . 取值
<code>ConstrainedMax[f, {inequ}, {x, y, ..}]</code>	同上
<code>LinearProgramming[C, m, b]</code>	解线性组合c. x在m. $x \geq b$ & $x \geq 0$ 约束下的最小值, x, b, c为向量, m为矩阵
<code>LatticeReduce[{v1, v2...}]</code>	向量组Vi的极小无关组
<code>Fit[data, funs, vats]</code>	用指定函数组对数据进行最小二乘拟合
<code>Interpolation[data]</code>	对数据进行插值
<code>Lisfinterpolation[array]</code>	对离散数据插值, array可为n维
<code>ListInterpolafion[array, {{xmin, xmax}, {min, ymax}, ..}]</code>	在特定网格上进行插值
<code>FunctionInterpolation[expr, {x, xmin, xmax}, {y, ymin, ymax}, ..]</code>	以对应expr[xi, yi]的数值为数据进行插值
<code>Fourier[list]</code>	对复数数据进行傅氏变换
<code>InverseFourieroiSt)</code>	对复数数据进行傅氏逆变换

# 第8章：Mathematica中的常用函数

## 8.8 表相关函数

### 1.制表函数

<code>{e1, e2, ...}</code>	一个表，元素可以为任意表达式，无穷嵌套
<code>Table[expr, {imax}]</code>	生成一个表，共imax个元素
<code>Table[expr, {i, imax}]</code>	生成一个表，共imax个元素expr间
<code>Table(expr, {i, imin, imax}, {j, jmin, jmax}, ...]</code>	多维表
<code>Range[imax]</code>	简单数表f1, 2+, imax)
<code>Range[imin, imax, di]</code>	以di为步长的数表
<code>Array[f, n]</code>	一维表，元素为fI'' (i从1到n)
<code>Array[f, {n1, n2...}]</code>	多维表，元素为玊i小. 1 (各自从1到ni)
<code>IdentityMatrix[n]</code>	n阶单位阵
<code>DiagonalMatrix[list]</code>	对角阵

# 第8章: Mathematica中的常用函数

## 2.元素操作

<code>Part[expr,i]或expr[[i]]</code>	第i个元素
<code>expr[[-i]]</code>	倒数第i个元素
<code>expr[{i,j,..}]</code>	多维表的元素
<code>expr[{i1,i2,..}]</code>	返回由第i(n)的元素组成的子表
<code>FirstCexpr]</code>	第一个元素
<code>Last[expr]</code>	最后一个元素
<code>Head[expr]</code>	函数头, 等于 <code>expr[[0]]</code>
<code>Extract[expr,list]</code>	取出由表list指定位置上expr的元素值
<code>Take[list,n]</code>	取出表list前n个元素组成的表
<code>Take[list, {m,n}]</code>	取出表list从m到n的元素组成的表
<code>Drop[list,n]</code>	去掉表list前n个元素组下的表
<code>Rest[expr]</code>	去掉表list第一个元素剩下的表
<code>Select[USt, crit]</code>	把crit作用到每一个list的元素上, 为True的所有元素组成的表
<code>Length[expr]</code>	expr第一层元素的个数
<code>Dimensions[expr]</code>	表的维数返回(n1,n2..),expr为一个n1*n2...的阵
<code>TensorRank[expr)</code>	秩
<code>Depth[expr]</code>	expr最大深度
<code>Level[expr,n]</code>	给出expr中第n层子表达式的列表
<code>Count[USt, paUem]</code>	满足模式的list中元素的个数
<code>MemberQ[list, form]</code>	list中是否有匹配form的元素
<code>FreeQ[expr,form]</code>	MemberQ的反函数
<code>FreeQ[expr,form]</code>	表中匹配模式pattern的元素的位置列表
<code>Cases[{e1,e2...}, pattem}</code>	匹配模式pattem的所有元素ei的表

## 第8章：Mathematica中的常用函数

### 3.表的操作

<code>Append[exp,elem]</code>	返回在表 <code>expr</code> 的最后追加 <code>elem</code> 元素后的表
<code>Prepend[expr,elem]</code>	返回在表 <code>expr</code> 的最前添加 <code>elem</code> 元素后的表
<code>Insert[1list, elem, n]</code>	在第 <code>n</code> 元素前插入 <code>elem</code>
<code>Insert[expr,elem,{i,j,...}]</code>	在元素 <code>expr[[{i,j,...}]]</code> 前插入 <code>elem</code>
<code>Delete[expr,{i,j,...}]</code>	删除元素 <code>expr[[{i,j,...}]]</code> 后剩下的表
<code>DeleteCases[expr,pattern]</code>	删除匹配 <code>pattern</code> 的所有元素后剩下的表
<code>ReplacePart[expr,new,n]</code>	将 <code>expr</code> 的第 <code>n</code> 元素替换为 <code>new</code>
<code>Sort[list]</code>	返回 <code>list</code> 按顺序排列的表
<code>Reverse[expr]</code>	把表 <code>expr</code> 倒过来
<code>RotateLeft[expr,n]</code>	把表 <code>expr</code> 循环左移 <code>n</code> 次
<code>RotateRight[expr,n]</code>	把表 <code>expr</code> 循环右移 <code>n</code> 次
<code>Partition[list,n]</code>	把 <code>list</code> 按每 <code>n</code> 个元素为一个子表分割后再组成的大表
<code>Flatten[1list]</code>	抹平所有子表后得到的一维大表
<code>Flatten[1list,n]</code>	抹平到第 <code>n</code> 层
<code>Split[1list]</code>	把相同的元素组成一个子表，再合成的大表

# 第8章：Mathematica中的常用函数

## 8.9 绘图函数

### 1. 二维绘图

`Plot[f, {x, xmin, xmax}]`

一维函数 $f[x]$ 在区间 $[xmin, xmax]$ 上的函数曲线

`Plot[{f1, f2. .}, {x, xmin, xmax}]`

在同一图形上画几条曲线

`ListPlot[{y1, y2, . .}]`

绘出由离散点对 $(n, y_n)$ 组成的图

`ListPlot[{x1, y1}, {x2, y2}, . .]`

绘出由离散点对 $(x_{r1}, y_{r1})$ 组成的图

`ParametricPlot[{fx, fy}, {t, tmin, tmax}]`

由参数方程在参数变化范围内产生的曲线

`ParametricPlot[{fx, fy}, {gx, gy}, . . .], {t, tmin, truax}]`

# 第8章：Mathematica中的常用函数

## 2.二维设置

<code>PlotRange-&gt;{0, 1}</code>	作图显示的值域范围
<code>AspectRatio-&gt;1/GoldenRatio</code>	生成图形的纵横比
<code>PlotLabel-&gt;label</code>	标题文字
<code>Axes-&gt;{false, True}</code>	分别制定是否画x, y轴
<code>AxesLabel-&gt;{xlabel, ylabel}</code>	x, y轴上的说明文字
<code>Ticks-&gt;None, Automatic, fun</code>	用什么方式画轴的刻度
<code>AxesOrigin-&gt;{x, y}</code>	坐标轴原点位置
<code>AxesStyle-&gt;{{xstyle}, {ystyle}}</code>	设置轴线的线性颜色等属性
<code>Frame-&gt;True, False</code>	是否画边框
<code>FrameLabel-&gt;{xlabel, ylabel, xlabel, ylabel}</code>	边框四边上的文字
<code>FrameTicks</code> 同Ticks	边框上是否画刻度
<code>GridLines</code> 同Ticks	图上是否画栅格线
<code>Framestyle-&gt;{{xmstyle}, {ymstyle}}</code>	设置边框线的线性颜色等属性
<code>ListPlot[data, PlotJoined-&gt;True]</code>	把离散点按顺序连线
<code>Plotsyle-&gt;{{style1}, {style2},...}</code>	曲线的线性颜色等属性
<code>PlotPoints-&gt;15</code>	曲线取样点, 越大越细致

## 第8章：Mathematica中的常用函数

### 3. 三维绘图

<code>Plot3D[f, {x, xmin, xmax}, {y, ymin, ymax}]</code>	二维函数 $f(x, y)$ 的空间曲面
<code>Plot3D[{f, s}, {x, xmin, xmax}, {y, ymin, ymax}]</code>	同上，曲面的染色由 $s(x, y)$ 值决定
<code>ListPlot3D[array]</code>	二维数据阵 <code>array</code> 的立体高度图
<code>ListPlot3D[array, shades]</code>	同上，曲面的染色由 <code>shades[数据]</code> 值决定
<code>ParametricPlot3D[{fx, fy, fz}, {t, tmin, tmax}]</code>	三维参数图形
<code>ContourPlot[f, {x, xmin, xmax}, {y, ymin, ymax}]</code>	二维函数 $f(x, y)$ 在指定区间上的等高线图
<code>ListContourPlot[array]</code>	二维函数 $f(x, y)$ 在指定区间上的等高线图

### 4. 三维设置

<code>Contours-&gt;n</code>	画 $n$ 条等高线
<code>Contours-&gt;{z1, z2, ...}</code>	在 $z_i$ 处画等高线
<code>ContourShading-&gt;False</code>	是否用深浅染色
<code>ContourLines-&gt;True</code>	是否画等高线
<code>ContourStyle-&gt;{{style1}, {style2}, ...}</code>	等高线线性颜色等属性

## 第8章：Mathematica中的常用函数

### 5.1 密度图

<code>DensityPlot[f, {x, xmin, xmax}, {y, ymin, ymax}]</code>	二维函数 $f[x, y]$ 在指定区间上的密度图
<code>ListDensityPlot[array]</code>	二维函数 $f[x, y]$ 在指定区间上的密度图

### 5.2 图形显示

<code>Show[graphics, options]</code>	显示一组图形对象，options为选项设置
<code>Show[g1, g2...]</code>	在一个图上叠加显示一组图形对象
<code>GraphicsArray[{g1, g2, ...}]</code>	在一个图上分块显示一组图形对象
<code>SelectionAnimate[notebook, t]</code>	把选中的notebook中的图画循环放映

## 第8章：Mathematica中的常用函数

### 6. 图元函数

Graphics[prim, options]	prim为下面各种函数组成的表，表示一个二维图形对象
Graphics3D[prim, options]	prim为下面各种函数组成的表，表示一个三维图形对象
SurfaceGraphics[array, shades]	表示一个由array和shade决定的曲面对象
ContourGraphics[array]	表示一个由array决定的等高线图对象
DensityGraphics[array]	表示一个由array决定的密度图对象
Point[p]	p={x, y}或{x, y, 2}，在指定位置画点
Line[{p1, p2, ..}]	经由Pi点连线
Rectangle[{xmin, ymin}, {xmax, ymax}]	画矩形
Cuboid[{xmin, ymin, zmin}, {xmax, ymax, zmax}]	由对角线指定的长方体
Polygon[{p1, p2, ...}]	封闭多边形
Circle[{x, y}, r]	画圆
Circle[{x, y}, {rx, ry}]	画椭圆，rx, ry为半长短轴
Circle[{x, y}, r, {a1, a2}]	从角度a1-a2的圆弧
Disk[{x, y}, r]	填充的圆、椭圆、圆弧等参数同上
Raster[array, ColorFunction->f]	颜色栅格
Text[expr, coords]	在坐标coords上输出表达式
PostScript["string"]	直接用Postscript图元语言写
Scaled[{x, y, ...}]	返回点的坐标，且均大于0小于1

## 第8章：Mathematica中的常用函数

### 7. 着色及其他

<code>GrayLevel[level]</code>	灰度level为0~1间的实数
<code>RGBColor[red, green, blue]</code>	RGB颜色，均为0~1间的实数
<code>Hue[h, s, b]</code>	亮度，饱和度等，均为0~1间的实数
<code>CMYKColor[cyan, magenta, yellow, black]</code>	CMYK颜色
<code>Thickness[r]</code>	设置线宽为r
<code>PointSize[d]</code>	设置绘点的大小
<code>Dashing[{r1, r2, ...}]</code>	画一个单元的间隔长度的虚线
<code>ImageSize-&gt;{x, y}</code>	显示图形大小(单位为像素)

## 第8章：Mathematica中的常用函数

### 8.10 流程控制

<code>If[condition, t, f]</code>	如果condition为True, 执行t, 否则执行f段
<code>if[condition, t, f, u]</code>	如果condition为True, 执行t, 为False执行f, 既非True 又非False, 则执行u段
<code>Which[test1, block1, test2, block2...]</code>	执行第一为True的testfi对应的blocki
<code>Switch[expr, form1, block1, form2, block2...]</code>	重复执行expr imax次
<code>Do[expr, {imax}]</code>	重复执行expr imax次

<code>Do[expr, {i, imin, imax}, {j, jmin, jmax}]</code>	多重循环
<code>While[test, body]</code>	循环执行body直到test为False
<code>For[start, test, incr, body]</code>	循环执行body直到test为False
<code>Throw[value]</code>	停止计算, 把value返回给最近一个Catch处理
<code>Throw[value, tag]</code>	停止计算, 把value返回给最近一个Catch处理
<code>Catch[expr]</code>	计算expr, 遇到Throw返回的值则停止
<code>Catch[expr, form]</code>	当Throw[value, tag]中Tag匹配form时停止

## 第8章：Mathematica中的常用函数

### 流程控制

<code>Return[expr]</code>	从函数返回，返回值为 <code>expr</code>
<code>Return[]</code>	返回值Null
<code>Break1[]</code>	结束最近的一重循环
<code>Continuel[]</code>	停止本次循环，进行下一次循环
<code>Goto[tag]</code>	无条件转向Label[Tag]处
<code>Label[tag]</code>	设置一个断点
<code>Check[expr, failexpr]</code>	计算 <code>expr</code> ，如果有出错信息产生，则返回 <code>failexpr</code> 的值
<code>Check[expr, failexpr, s1::t1, s2::t2, ...]</code>	当特定信息产生时则返回 <code>failexpr</code>
<code>CheckAbort[expr, failexpr]</code>	当产生abort信息时返回 <code>failexpr</code>
<code>Interrupt[]</code>	中断运行
<code>Abort[]</code>	中断运行
<code>TimeConstrained[expr, t]</code>	计算 <code>expr</code> ，当耗时超过 <code>t</code> 秒时终止
<code>MemoryConstrained[expr, b]</code>	计算 <code>expr</code> ，当耗用内存超过 <code>b</code> 字节时终止运算
<code>Print[expr1, expr2, ...]</code>	顺次输出 <code>expr<sub>i</sub></code> 的值
<code>Input[]</code>	产生一个输入对话框，返回所输入的任意表达式
<code>Input["prompt"]</code>	同上， <code>prompt</code> 为对话框的提示
<code>Pause[n]</code>	运行暂停 <code>n</code> 秒