列表操作

列表的简单操作

```
In[∘]:= data = {a, b, c, d}
        data[[2]]
        Clear[data]
        清除

Out[∘]= {a, b, c, d}

Out[∘]= b
```

```
In[∘]:= s = Solve[x^2 - 3 x + 2 == 0, x]
              解方程

        2 x - 1 /. s[[1]]
        x^2 - 3 x + 2 /. s[[2]]
        Clear["Global`*"]
        清除

Out[∘]= {{x → 1}, {x → 2}}

Out[∘]= 1

Out[∘]= 0
```

相比于“=”直接赋值,替换规则可以实现在部分区域的赋值能力.

```
In[∘]:= data = {{a, b}, {c, d}};
        data[[1]] = data[[1]] * 2;
        data
        data[[1, 2]] = data[[1, 2]] + 2
        data
        Clear[data]
        清除

Out[∘]= {{2 a, 2 b}, {c, d}}

Out[∘]= 2 + 2 b

Out[∘]= {{2 a, 2 + 2 b}, {c, d}}
```

```
In[∘]:= data = {{a, b}, {c, d}}
        TableForm[data]
        表格形式
        MatrixForm[data]
        矩阵格式
        Clear[data]
        清除

Out[∘]= {{a, b}, {c, d}}

Out[∘]//TableForm=
        a    b
        c    d

Out[∘]//MatrixForm=
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

针对个人喜好还有呈现的理论体系要求可以进行形式上的改变, 只是单单改变最后输出的呈现效果, 并不会改变过程中的计算机理.

```
In[∘]:= data = {{a, b}, {c, d}}
        data = Prepend[data, {"frequency", "energy"}]
               加在前面
        data
        TableForm[data]
        表格形式
        Clear[data]
        清除

Out[∘]= {{a, b}, {c, d}}

Out[∘]= {{frequency, energy}, {a, b}, {c, d}}

Out[∘]= {{frequency, energy}, {a, b}, {c, d}}

Out[∘]//TableForm=
        frequency    energy
        a            b
        c            d
```

表的制造

```
In[∘]:= Range[5]
        范围

Out[∘]= {1, 2, 3, 4, 5}

        list = Range[5]
               范围
        Table[Sin[π / 2 * i], {i, list}] #后面的花括号内的文件是用来描述和限制最终变量的取值范围的
        表格   正弦
        Clear[list]
        清除

Out[∘]= {1, 2, 3, 4, 5}

Out[∘]= {1, 0, -1, 0, 1}
```

```
In[•]:= list = Array[fun, 3]
                 ⌊数组
        ConstantArray[0, {3, 3}] // MatrixForm
        ⌊常量数组                    ⌊矩阵格式
        DiagonalMatrix[list] // MatrixForm
        ⌊对角矩阵                 ⌊矩阵格式
        IdentityMatrix[3] // MatrixForm
        ⌊单位矩阵              ⌊矩阵格式
        Clear[list]
        ⌊清除

Out[•]= {fun[1], fun[2], fun[3]}

Out[•]//MatrixForm=
        ( 0  0  0 )
        ( 0  0  0 )
        ( 0  0  0 )

Out[•]//MatrixForm=
        ( fun[1]    0       0    )
        (   0     fun[2]    0    )
        (   0       0     fun[3] )

Out[•]//MatrixForm=
        ( 1  0  0 )
        ( 0  1  0 )
        ( 0  0  1 )

In[•]:= n = 5;
        m = SparseArray[{{i_, i_} → -2, {i_, j_} /; Abs[i - j] == 1 → 1}, {n, n}]
            ⌊稀疏数组                                  ⌊绝对值
        MatrixForm[m]
        ⌊矩阵格式
        Normal[m]
        ⌊转换为普通表达式
        Length[m]
        ⌊长度
        Dimensions[m]
        ⌊维数
        Clear[n, m]
        ⌊清除

Out[•]= SparseArray[ ⊞  Specified elements: 13
                         Dimensions: {5, 5} ]

Out[•]//MatrixForm=
        ( -2   1   0   0   0 )
        (  1  -2   1   0   0 )
        (  0   1  -2   1   0 )
        (  0   0   1  -2   1 )
        (  0   0   0   1  -2 )

Out[•]= {{-2, 1, 0, 0, 0}, {1, -2, 1, 0, 0},
         {0, 1, -2, 1, 0}, {0, 0, 1, -2, 1}, {0, 0, 0, 1, -2}}

Out[•]= 5

Out[•]= {5, 5}
```

表的操作函数

对元素的操作

```
In[∘]:= x = {{a, c}, {d, f}, {g, k}, {l, q}};
        Append[x, {p, o}]
        追加
        x
        AppendTo[x, {p, o}]
        附加
        x
        Prepend[x, {b, e}];
        加在前面
        x
        Clear[x]
        清除

Out[∘]= {{a, c}, {d, f}, {g, k}, {l, q}, {p, o}}

Out[∘]= {{a, c}, {d, f}, {g, k}, {l, q}}

Out[∘]= {{a, c}, {d, f}, {g, k}, {l, q}, {p, o}}

Out[∘]= {{a, c}, {d, f}, {g, k}, {l, q}, {p, o}}

Out[∘]= {{a, c}, {d, f}, {g, k}, {l, q}, {p, o}}
```

```
In[∘]:= x = {{a, b, c}, {d, e, f}, {g, h, k}, {l, p, q}};
        x // TableForm
             表格形式
        Delete[x, 2]
        删除
        x
        Delete[x^T, 2] // TableForm
        删除                表格形式
        Clear[x]
        清除
Out[∘]//TableForm=
        a    b    c
        d    e    f
        g    h    k
        l    p    q

Out[∘]= {{a, b, c}, {g, h, k}, {l, p, q}}

Out[∘]= {{a, b, c}, {d, e, f}, {g, h, k}, {l, p, q}}
Out[∘]//TableForm=
        a    d    g    l
        c    f    k    q
```

```
In[∘]:= x1 = {{a, c}, {d, f}, {g, k}, {l, p}};
        x1 // TraditionalForm
              传统格式
        x2 = {b, e, h, q};
        Insert[x1, x2, -2]
        插入
        x1
        Insert[x1^T, x2, 2]^T // TableForm
        插入                     表格形式
        Clear[x1, x2]
        清除
```

Out[∘]//TraditionalForm=

$$\begin{pmatrix} a & c \\ d & f \\ g & k \\ l & p \end{pmatrix}$$

```
Out[∘]= {{a, c}, {d, f}, {g, k}, {b, e, h, q}, {l, p}}

Out[∘]= {{a, c}, {d, f}, {g, k}, {l, p}}
```

Out[∘]//TableForm=

```
a    b    c
d    e    f
g    h    k
l    q    p
```

```
In[∘]:= x = {{a, c}, {d,  f}, {g,  k}, {l, p}};
        Take[x, 3]
        选取
        x
        Take[x, -1]
        选取
        Drop[x, 3]
        去掉元素
        Drop[x, -1]
        去掉元素
        Clear[x]
        清除

Out[∘]= {{a, c}, {d, f}, {g, k}}

Out[∘]= {{a, c}, {d, f}, {g, k}, {l, p}}

Out[∘]= {{l, p}}

Out[∘]= {{l, p}}

Out[∘]= {{a, c}, {d, f}, {g, k}}
```

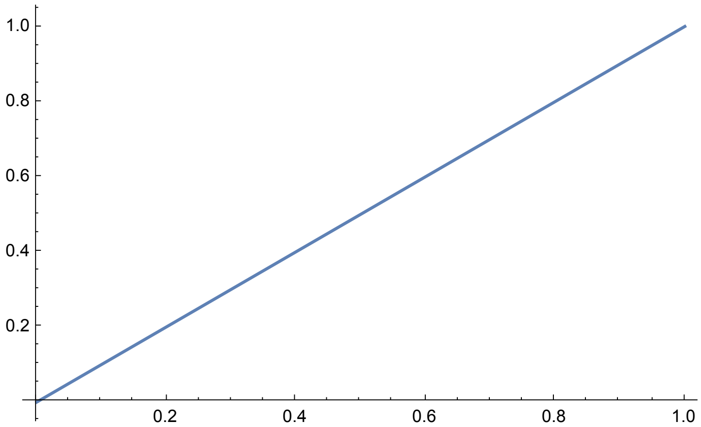
```
In[•]:= x = {1, 2, 4, 7, 6, 2}
        Select[x, EvenQ]
        选择         偶数判定
        Select[x, # > 2 &]
        选择
        Select[x,  # > 2 &, 1]
        选择
        Clear[x]
        清除

Out[•]= {1, 2, 4, 7, 6, 2}

Out[•]= {2, 4, 6, 2}

Out[•]= {4, 7, 6}

Out[•]= {4}
```

```
In[•]:= list = {1, 1, f[a], 2, 3, y, f[8], 9, f[10]};
        Cases[list, _Integer]
        模式匹配      虚数单位
        Cases[list, Except[_Integer]]
        模式匹配      除了     虚数单位
        Cases[list, f[y_] → y]
        模式匹配
        list = {{1, 2}, {2}, {3, 4, 1}, {5, a}, {3, 3}};
        Cases[list,  {a_, b_} → Total[{a, b}]]
        模式匹配                     总计
        Clear[list]
        清除

Out[•]= {1, 1, 2, 3, 9}

Out[•]= {f[a], y, f[8], f[10]}

Out[•]= {a, 8, 10}

Out[•]= {3, 5 + a, 6}
```

```
        "Cases[{f[{a,b}],f[{a}],g[{a}],f[{a,b,c,d}]},f{x_}⧴Length[x]]" #没有懂
         模式匹配                                                  长度

Out[•]= Cases[{f[{a,b}],f[{a}],g[{a}],f[{a,b,c,d}]},f{x_}⧴Length[x]]
```

```
In[•]:= Join[IdentityMatrix[3], {{1, 2, 3}}] // MatrixForm
        连接  单位矩阵                          矩阵格式
        Join[IdentityMatrix[3], Transpose[{{1, 2, 3}}], 2] // MatrixForm
        连接  单位矩阵              转置                       矩阵格式
        Join[{{a, b}, {c, d}}, {{1, 2}, {3, 4}}] // MatrixForm
        连接                                      矩阵格式
        Join[{{a, b}, {c, d}}, {{1, 2}, {3, 4}}, 2] // MatrixForm
        连接                                         矩阵格式
        Union[{a, b, a, c}, {d, a, e, b}, {c, a}]
        并集
Out[•]//MatrixForm=
        ( 1 0 0 )
        ( 0 1 0 )
        ( 0 0 1 )
        ( 1 2 3 )

Out[•]//MatrixForm=
        ( 1 0 0 1 )
        ( 0 1 0 2 )
        ( 0 0 1 3 )

Out[•]//MatrixForm=
        ( a b )
        ( c d )
        ( 1 2 )
        ( 3 4 )

Out[•]//MatrixForm=
        ( a b 1 2 )
        ( c d 3 4 )

Out[•]= {a, b, c, d, e}

In[•]:= {a, b, c} // FullForm
                     完全格式

Out[•]//FullForm= List[a, b, c]

In[•]:= Head[{c, a, b}]
        表达式的标头

Out[•]= List

In[•]:= list = {a, b, c}
        list[[2]]
        Clear[list]
        清除

Out[•]= {a, b, c}

Out[•]= b

In[•]:= Part[{a, b, c}, 2]
        部分

Out[•]= b

        Table[Part[{a, b, c}, i], {i, 0, 3}] #Part是提取列表中的第i个元素
        表格  部分

Out[•]= {List, a, b, c}
```

Out[∘]=



In[∘]:=
```
Length[x''[t] + 2 x[t] + y[x[t]] == 0]
长度
```

Out[∘]=
```
2
```

In[∘]:=
```
?? Length
```

Out[∘]=

Symbol

Length[*expr*] gives the number of elements in *expr*.

Documentation Local » | Web »

Attributes {Protected}

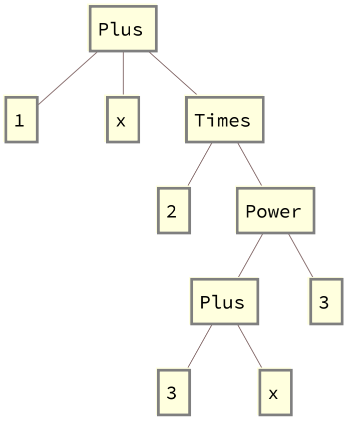
Full Name System`Length

In[∘]:=
```
x''[t] + 2 x[t] + y[x[t]] == 0 // FullForm
完全格式
```

Out[∘]//FullForm=
```
Equal[Plus[Times[2, x[t]], y[x[t]], Derivative[2][x][t]], 0]
```

In[∘]:=
```
1 + x + 2 (x + 3)^3 // TreeForm
树形式
```

Out[∘]//TreeForm=



In[∘]:=
```
Level[1 + x + 2 (x + 3)^3, {3}]
层
```

Out[∘]=
```
{3 + x, 3}
```

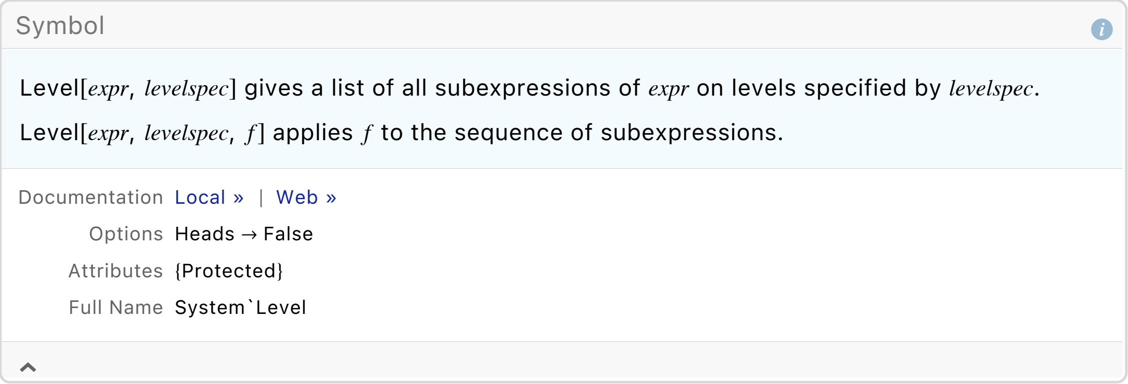
```
Level[1 + x + 2 (x + 3)^3, {3, 4}]
层
Level[1 + x + 2 (x + 3)^3, {3, 4}, Heads → True] #激活第三层和第四层的头部
层                                   标头      真
```

```
Out[∘]= {3, x, 3 + x, 3}
```

```
Out[∘]= {Power, Plus, 3, x, 3 + x, 3}
```

```
In[∘]:= ?? Level
```

Out[∘]=



Symbol

Level[*expr*, *levelspec*] gives a list of all subexpressions of *expr* on levels specified by *levelspec*.

Level[*expr*, *levelspec*, *f*] applies *f* to the sequence of subexpressions.

Documentation	Local » | Web »
Options	Heads → False
Attributes	{Protected}
Full Name	System`Level

模式

Blank(_)(任意表达式)
_, 任意表达式

x_, 任意表达式, 命名x

Pattern(:)(模式)
x:pattern, 名为x的任意模式

_h, 指定头部h的模式

patterntest(?)(模式检验)
p?test
是一个模式对象, 代表匹配p的任何表达式, 并且表达式应用test给出True.

Condition(/;)(条件)
patt/:test
是一个模式, 仅当test为True时才匹配.

patt:def或Optional[patt,def]
是一份个模式对象, 表示如果省略了形为patt的表达式, 应使用默认值def进行替换

Alternatives(|)(或)
Subscript[*P*, 1] | P_2|...
是一个模式对象, 用于代表任意模式p_1.

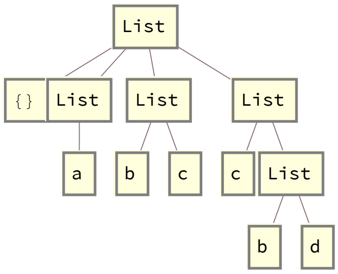
p..或Repeated[p]
是一个模式对象, 表示一个或多个表达式的序列, 每个表达式匹配p.

p..或RepeatedNull[p]
是一个模式对象, 表示一个由0或更多表达式(其中每个表达式斗鱼p匹配)构成的序列.

__(两个_字符)或者BlankSequence
一种模式对象, 可表示任意一个或多个Wolfram语言表达式序列.

___(三个_字符)或BlankNullSequence[]
一种模式对象, 可表示任意零个或者多个Wolfram语言表达式序列.

```
In[∘]:= {h[], h[x^2], g[x], h[b, c]} /. h[x_] → "MATCH"
Out[∘]= {h[], MATCH, g[x], h[b, c]}

In[∘]:= Cases[{{}, {a}, {b, c}, {c, {b, d}}}, {_, _}]
        模式匹配
Out[∘]= {{b, c}, {c, {b, d}}}

        Cases[{{}, {a}, {b, c}, {c, {b, d}}}, {_, _}, Infinity] #分解到最大层
        模式匹配                                        无穷大
Out[∘]= {{b, c}, {b, d}, {c, {b, d}}}

In[∘]:= {{}, {a}, {b, c}, {c, {b, d}}} // TreeForm
                                          树形式
Out[∘]//TreeForm=
```



```
In[∘]:= Position[{{}, {a}, {b, c}, {c, {b, d}}}, {_, _}]
        位置
Out[∘]= {{3}, {4, 2}, {4}}

In[∘]:= (h[a] + h[b, c] + h[a, a]) h[d, e, f] /. h[x_, y_] → x^y
```

Out[∘]= $\left(a^{a} + b^{c} + h[a]\right)\, h[d, e, f]$

```
In[∘]:= Sin[1 + a^2] /. h : Sin[x_ + y_] → {h, x, y}
        正弦             正弦
```

Out[∘]= $\left\{\mathrm{Sin}\left[1 + a^{2}\right], 1, a^{2}\right\}$

```
In[∘]:= Head /@ {x, "good", 3, 2 / 3, 5 / 7}
        表达式的标头

Out[∘]= {Symbol, String, Integer, Rational, Rational}

        {x, "good", 3, 2 / 3, 5 / 7} /. x_Rational → x ^ 2 #头部为有理数
```

Out[∘]= $\left\{x,\ \text{good},\ 3,\ \frac{4}{9},\ \frac{25}{49}\right\}$

```
In[∘]:= RandomInteger[100, 10]
        伪随机整数

Out[∘]= {15, 36, 74, 63, 71, 49, 15, 100, 73, 63}

In[∘]:= Count[%, _?EvenQ]
        计数      偶数判定

Out[∘]= 3

In[∘]:= {6, -7, 3, 2, -1, 2} /. (x_ /; x < 0) → Abs[x]
                                                 绝对值

Out[∘]= {6, 7, 3, 2, 1, 2}
```

In[∘]:= MatchQ[$a^2 - b^2$, x_^2 - y_^2]
匹配判定

```
Out[∘]= True

In[∘]:= MatchQ[(a - b) (a + b), x_ - y_]
        匹配判定

Out[∘]= False

In[∘]:= MatchQ[a - b, x_ - y_]
        匹配判定

Out[∘]= True

In[∘]:= MatchQ[a - 2 b, x_ - y_]
        匹配判定

Out[∘]= False

In[∘]:= a - 2 b // FullForm
                  完全格式

Out[∘]//FullForm= Plus[a, Times[-2, b]]
                  加       乘

In[∘]:= x_ - y_ // FullForm
                   完全格式

Out[∘]//FullForm= Plus[Pattern[x, Blank[]], Times[-1, Pattern[y, Blank[]]]]

In[∘]:= x → RandomReal[]
            伪随机实数

Out[∘]= x → 0.637489

In[∘]:= {x, x, x, x} /. x → RandomReal[]
                           伪随机实数

Out[∘]= {0.833765, 0.833765, 0.833765, 0.833765}
```

In[•]:= x :→ RandomReal[]
伪随机实数

Out[•]= x :→ RandomReal[]

{x, x, x, x} /. x :→ RandomReal[] #替换之后的赋值操作是不影响本身变量的
伪随机实数

Out[•]= {0.685317, 0.0992995, 0.143751, 0.722819}

In[•]:= {a, b} /. {x_, y_ : d} :→ {x^2, y^2}

Out[•]= {a^2, b^2}

{a} /. {x_, y_ : d} :→ {x^2, y^2} #如果没有找到就冒号后面的东西替代

Out[•]= {a^2, d^2}

In[•]:= {a, 2} /. {x_, y_Integer : 10} :→ {x^2, y^2}
输入行

Out[•]= {a^2, 4}

{a} /. {x_, y_Integer : 10} :→ {x^2, y^2} #Integer在这有啥用？
输入行

Out[•]= {a^2, 100}

In[•]:= a + b /. x_ + y_. :→ $x^2 + y^2$

Out[•]= $a^2 + b^2$

In[•]:= a /. x_ + y_. :→ $x^2 + y^2$ "找不到的话就用点来代替在加法中是0"

Out[•]= a^2

In[•]:= a b /. x_ y_. :→ x^2 + y^2

Out[•]= $a^2 + b^2$

In[•]:= a /. x_ y_. :→ $x^2 + y^2$

Out[•]= $1 + a^2$

In[•]:= a^2 /. x_ ^y_. :→ $x^2 + y^2$

Out[•]= $4 + a^2$

In[•]:= a /. x_^y_. :→ $x^2 + y^2$

Out[•]= $1 + a^2$

In[•]:= Cases[{3, x, 2 / 3, 2 + 3 I, Tan[x], Sqrt[x], 1.5}, _Integer | _Symbol | _Rational]
模式匹配 … 正切 平方根 虚数单位

Out[•]= {3, x, $\frac{2}{3}$}

In[•]:= {a, a, a, a, {a, b}, {a, b}} /. {x_Symbol .., y_List ..} :→ {x, y}

Out[•]= {a, {a, b}}

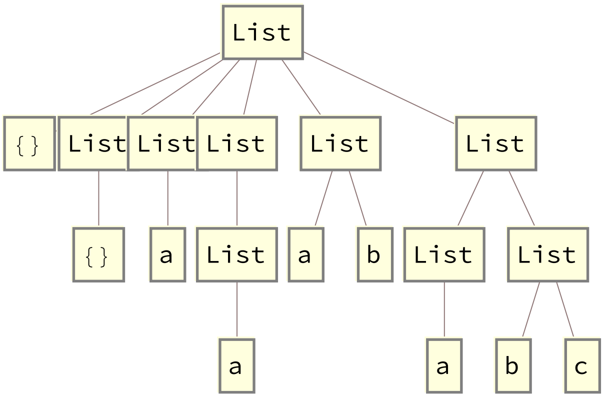
```
In[•]:= myfunc[x_ ] := {x}
        myfunc[a]
        Clear[myfunc]
        清除

Out[•]= {a}
```

```
In[•]:= ourfunc[x__] := {x}
        ourfunc[a]
        ourfunc[a, b, c]
        Clear[ourfunc]
        清除

Out[•]= {a}

Out[•]= {a, b, c}
```

```
In[•]:= Cases[{{}, {{}}, {a}, {{a}}, {a, b}, {{a}, {b, c}}}, {_List}]
        模式匹配

Out[•]= {{{}}, {{a}}}
```

```
In[•]:= {{}, {{}}, {a}, {{a}}, {a, b}, {{a}, {b, c}}} // TreeForm
                                                         树形式
```

Out[•]//TreeForm=



函数

```
In[•]:= x = 5;
        f[x_] := x^2;
        {f[2], f[3], x}
        g[x_] = x^2
        {g[2], g[3], x}

Out[•]= {4, 9, 5}

Out[•]= 25

Out[•]= {25, 25, 5}
```

In[•]:=
```
? f
? g
```

Out[•]=
```
Symbol
Global`f
Full Name  Global`f
```

Out[•]=
```
Symbol
Global`g
Full Name  Global`g
```

In[•]:=
```
Clear["Global`*"]
清除
```

In[•]:=
```
f[var_] := 3 var
f[y]
```

Out[•]= 3 y

In[•]:=
```
Function[var, 3 var][y]
纯函数
```

Out[•]= 3 y

In[•]:=
```
Function[3 #][y]
纯函数
```

Out[•]= 3 y

In[•]:=
```
3 # &[y]
```

Out[•]= 3 y

In[•]:=
```
Select[{1, a, x^2, 3, 5, 1 + x, 7}, # > 4 &]
选择
```

Out[•]= {5, 7}

In[•]:=
```
test1[expr_] := PolynomialQ[expr, x]
                多项式判定
Select[(1 + x + 2 x^2 + 3 x^3 + Sin[x]), test1]
选择                              正弦
```

Out[•]= $1 + x + 2 x^{2} + 3 x^{3}$

In[•]:=
```
Select[(1 + x + 2 x^2 + 3 x^3 + Sin[x]), Function[var, PolynomialQ[var, x]]]
选择                              正弦         纯函数          多项式判定
```

Out[•]= $1 + x + 2 x^{2} + 3 x^{3}$

In[•]:=
```
Select[(1 + x + 2 x^2 + 3 x^3 + Sin[x]), PolynomialQ[#, x] &]
选择                              正弦         多项式判定
```

Out[•]= $1 + x + 2 x^{2} + 3 x^{3}$

In[•]:=

```
myfunc[x_, y_] := x^2 + y^2
myfunc[a, b]
```

Out[•]= $a^2 + b^2$

In[•]:=

```
#1^2 + #2^2 &[a, b]
```

Out[•]= $a^2 + b^2$

In[•]:=

```
ourfunc[x__] := {x}
ourfunc[a]
ourfunc[a, b, c, d]
```

Out[•]= {a}

Out[•]= {a, b, c, d}

In[•]:=

```
{##} &[a]
```

Out[•]= {a}

In[•]:=

```
{##} &[a, b, c, d]
```

Out[•]= {a, b, c, d}