```
"version" /. Cases[j, HoldPattern["version" -> __], Infinity]
```

```
SplitBy[#,MatchQ[#,{0..}]&]/.{{0..}..}->Sequence[]
```

```
segmentGreenRed[i_]:=With[{grouping:=(#//Transpose//SplitBy[#,MatchQ[#,{2..}]&]&//SplitBy[#,MatchQ[#,{3..}]&]&/@#&//Flatten[#,1]&//Transpose/@#&)&},
   i//segmentByHorizon//labelWhite/@#&//grouping/@#&]
```

矩阵中连续的两个或三个连续的全零行替换成全2行

```
{{{0,0},{0,0}},{{1,1}},{{0,0}} } /. {{x:Repeated[{0..},{2,3}]}:>({x}/.{0->2}) }
  {{{2,2},{2,2}},{{1,1}},{{0,0}}}
```

x匹配的是一个Sequence，所以外面用{}括起来

:> 这里用的是延迟规则(RuleDelayed)，不要立既计算。注意不要被delayed和非delayed 坑了

```
{{2, 2}, {2, 2}} /. {x : 2 ..} :> ({x} /. {2 -> 3})
{{2, 2}, {2, 2}} /. x : {2 ..} :> (x /. _Integer?(# == 2 &) -> 3)
turnGreen=Function[{mt},mt/.x:{2..}:>(x/.{2->3})]
MatchQ[{1, 1, "中", "中天"}, {___, _?(# == "中天" &), ___}]
Cases[{1,2,"ab","cd",x,y},_String]
   {ab,cd}
MatchQ["中",_String?(#=="中"&)]
  True
```

```
StringMatchQ["中心","中心"]
  True
StringCases["abcadcacb","a"~~_~~"c"]
  {abc,adc}
StringPosition["中国国中国","中"]
  {{1,1},{4,4}}
StringTake["中国国中国",{{1,1},{4,4}}]
  {中,中}
StringCases["abcd", __, Overlaps -> False]
```

```
   {abcd}
StringCases["abcd", __, Overlaps -> All]
   {abcd,abc,ab,a,bcd,bc,b,cd,c,d}
```

Pick all the lines that contain a substring that matches the pattern:

```
grep[file_,patt_]:=With[{data=Import[file,"Lines"]},Pick[Transpose[{Range[Length[data]],data}],StringFreeQ[data,patt],False]]
```

Line numbers with corresponding texts that contain "noon" or "day of":

```
grep["ExampleData/USConstitution.txt", {"noon" ,"day of"}]//TableForm

data={"a","xnoona","","b","day","xxday oftt"};

Pick[data,StringFreeQ[data, {"noon" ,"day of"}],False]

   {xnoona,xxday oftt}

stringExistQ[str_,sub_]:=StringPosition[str,sub]=!={}
```

xx?AtomQ 原子表达式(不能在拆分成子表达式了)

```
 {{x1,x2},{x3,x4}}/.{x_?AtomQ,y_}->f[x,y]

    {f[x1,x2],f[x3,x4]}
```

Optional (:)

```
    f[x_, y_: 0] := {x, y}
```

        y 有一个默认值

OptionsPattern

OptionValue

    有点类似特定命名空间下的枚举值

```
MatchQ[#, {{x_?(# == 2 &) ..} ..}]
```

```
f[1, 2] /. x_f -> x^2 (*注意x匹配了f[1,2]*)
```

   _f 表示任何以f 为head 的表达式

   x称为模式变量pattern variable，是临时符号

```
"ExposureTime" /. Cases[Options[robot], HoldPattern["ExposureTime" -> __], Infinity]

   "ExposureTime" /. {"ExposureTime" -> 1/5}

      1/5
```

模式名称	符号	作用
`Blank[]`	`_`	匹配单个元素
`BlankSequence[]`	`__`	匹配一个或多个元素
`BlankNullSequence[]`	`___`	匹配零个、一个或多个元素

p... or RepeatedNull[p]

is a pattern object that represents a sequence of zero or more expressions, each matching p.

p.. or Repeated[p]

is a pattern object that represents a sequence of one or more expressions, each matching p.

Cases[{0,1,0},_Integer?Positive]

{1}

MatchQ[{1, 2, 3}, _?ListQ]

MatchQ[{1, 2, 3}, _List]

MatchQ[{a, b, c}, _List?(Length[#] > 2 &)]

__Number 一个或多个数字

n_?(IntegerQ && Positive)

x_List?(Length[#] > 2 &)

f[n_ /; IntegerQ[n] && Positive[n]] := f[n - 1] + f[n - 2] (*Fibonacci数列应该只对正整数有定义*)

Pattern (:)

s:obj

represents the pattern object obj, assigned the name s.

Cases[{{1,2,3},a,{4,5}},t:{__Integer}:>t^2]

{{1,4,9},{16,25}}

:> RuleDelayed represents a rule that transforms lhs to rhs, evaluating rhs only after the rule is used.

从有到无(结果就是其存在被抹消)

{{1, 2, 3}} /. {{_Integer ..} ..} -> Sequence[]

无模式

```
f[a|PatternSequence[]]:=x
  f[a]
    x
  f[]
    x
```

Match only the pattern x_:

```
  MatchQ[any[expression], Verbatim[x_]]
    False
  MatchQ[x_, Verbatim[x_]]
    True
```

删除矩阵中连续的两个全零行

```
mt={{1, 1, 1}, {0, 0, 0}, {1, 1, 1}, {0, 0, 0}, {0, 0, 0}, {1, 1, 1}};
mt /. {x__,PatternSequence[ Repeated[{0..},{2}]]..,y__} ->{x,y}
   {{1,1,1},{0,0,0},{1,1,1},{1,1,1}}
```

```
 {1,2, 3} /. {1->x, 2->y, _->z}
   {x,y,z}
```

```
In[3]:= Hold[{1, 2, 3, 4, 5}] /. n_Integer :> RuleCondition[n^2, OddQ[n]]Out[3]= Hold[{1, 2, 9, 4, 25}]
```

```
% /. {Repeated[0, {3}]} -> {x, x, x}
```

```
SetAttributes[test, HoldAll]test[f[_] | f[_]@_?test] = True;test[_] = False;f[a]@f[b]@f[c] // test
```

```
True
```

分割矩阵，条件为无素为全0 的列

```
MatrixForm /@ Transpose/@(SplitBy[mat\[Transpose],MatchQ[#,{0..}]&]/.{{0..}..}->Sequence[])
```

```
FullForm[ x:{{__}..} ]
```

```
 f @@ {"a", "b", "c", "d", "e"}
```

f[a,b,c,d,e]

f["a","b","c","d","e"] /. f[tt__] ->{tt} (*__ 表示任意符号，1或多，左边的tt 用于给它命名*)

{a,b,c,d,e}

```
dalist /.
  {x_?NumericQ, y_?NumericQ} :>
  {Which[y==1, COGCondition1, y==2, COGCondition2], y}
```

```
dalist /.
  {a:PatternSequence@@Array[_&,3], x_,
   b:PatternSequence@@Array[_&,5], y_Integer} :> {a, y /. conditions, b, y}
```

Now, `PatternSequence@@Array[_&,3]` could be written `PatternSequence[_, _, _]`, but by using `Array` it gives more flexibility.

来源： <http://stackoverflow.com/questions/7079244/conditional-list-values-replacement-in-mathematica>

```
Cases[Unevaluated[expr],s_Symbol:>HoldComplete[s],{0,Infinity},Heads->True]
```

Note that this will work for *any* Mathematica expression, including a piece of (perhaps unevaluated) Mathematica code.

Transpose /@ (SplitBy[m\[Transpose], MatchQ[#, {0 ..}] &] /. {{0 ..} ..} -> Sequence[]) (*more succinct 更短更清晰*)

I believe you're looking for `RepeatedNull`

```
Count[IdentityMatrix[10], {0 ..., 1, 0 ...}](* 10 *)
```

Count[IdentityMatrix[10],{0...,1,0...}]

IdentityMatrix[10] // MatrixForm

{0...,1,0...} 是一个pattern，匹配一个List，0或多个0开始，后接1个1，再接0或多个0

MatchQ[#, _Integer] & /@ {1, 2.}

{True,False}

函数定义的左边是模式匹配

```
f[x_, y_] /; x+y < 10 := x*y
```

```
Cases[Unevaluated[expr],s_Symbol:>HoldComplete[s],{0,Infinity},Heads->True]
```

Note that this will work for *any* Mathematica expression, including a piece of (perhaps unevaluated) Mathematica code.

I believe you're looking for `RepeatedNull`

```
Count[IdentityMatrix[10], {0 ..., 1, 0 ...}](* 10 *)
```

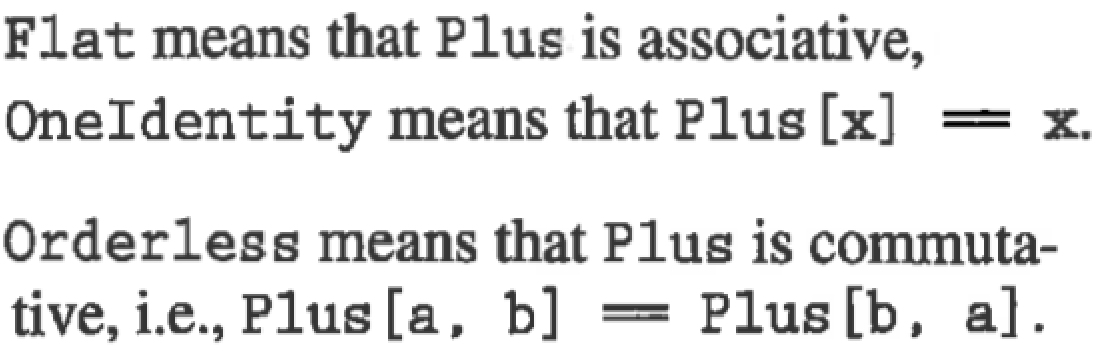
```
(* Replace all items in list that fall in the interval [0, 1] with 1
 * using a PatternTest *){1, 2, 3, 0.4, 0} /. x_?(0 <= # <= 1 &) -> 1
```

```
f[x_Integer?EvenQ] := x+1
```

And a `Condition` seems preferable when a more complex condition is required:

```
f[x_Integer /; EvenQ[x] && Positive[x]] := x+1
```

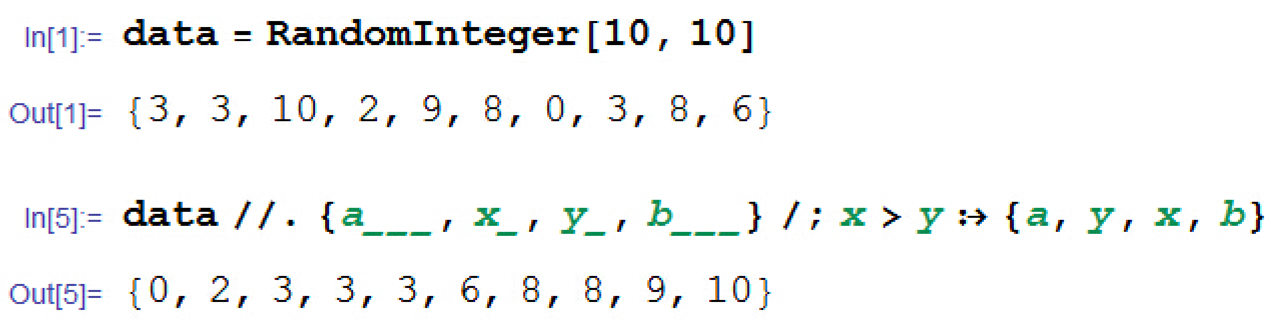
Case 捕获表达式集合的一个字集



`Flat` means that `Plus` is associative,
`OneIdentity` means that `Plus[x] == x`.

`Orderless` means that `Plus` is commutative, i.e., `Plus[a, b] == Plus[b, a]`.

冒泡程序。。利用模式匹配写的



```
In[1]:= data = RandomInteger[10, 10]

Out[1]= {3, 3, 10, 2, 9, 8, 0, 3, 8, 6}

In[5]:= data //. {a___, x_, y_, b___} /; x > y ⧴ {a, y, x, b}

Out[5]= {0, 2, 3, 3, 3, 6, 8, 8, 9, 10}
```

```
DeleteCases[L0, x_ /; MemberQ[x, z_ /; z > 1]]
```