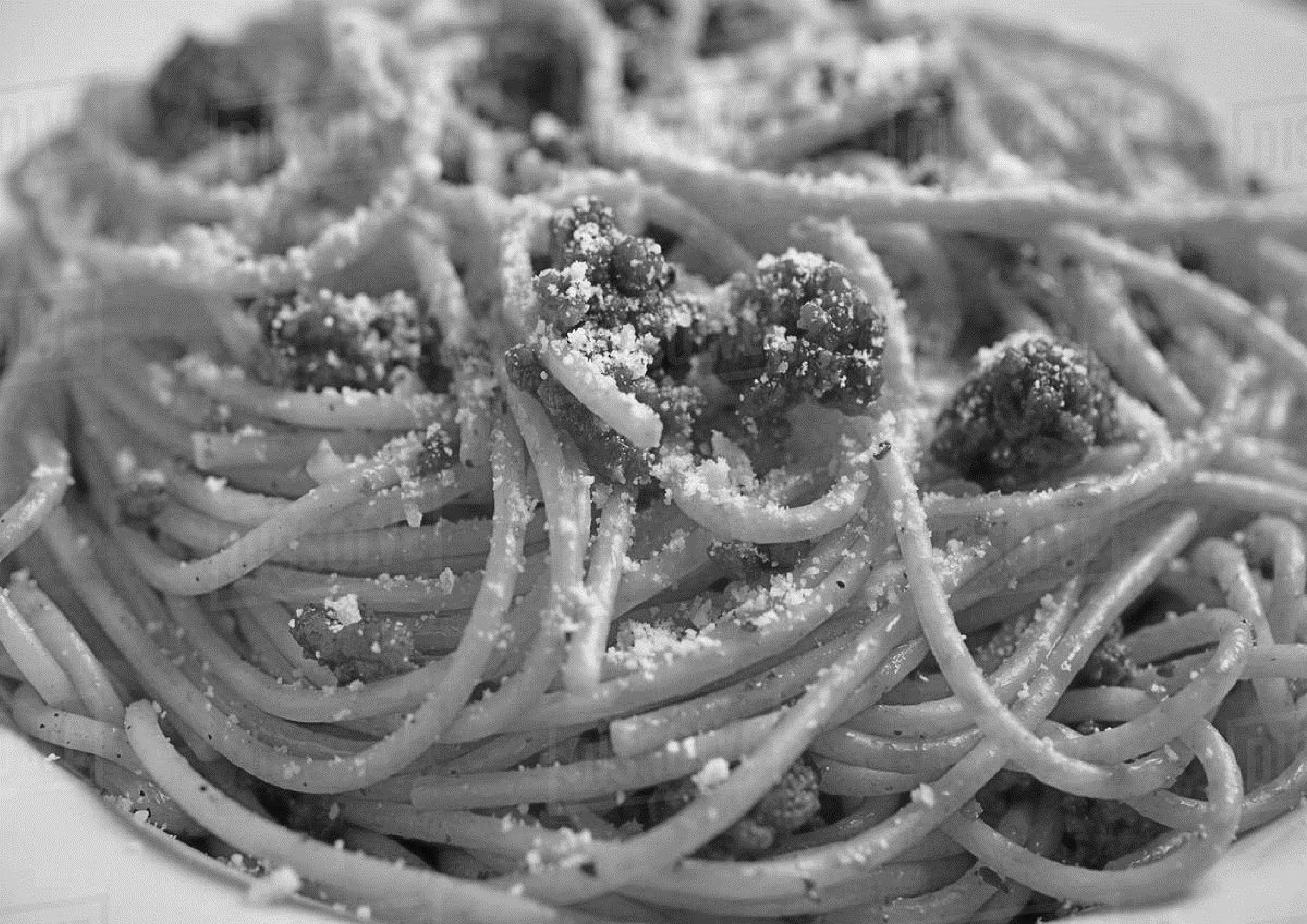
Dynamo Part 01

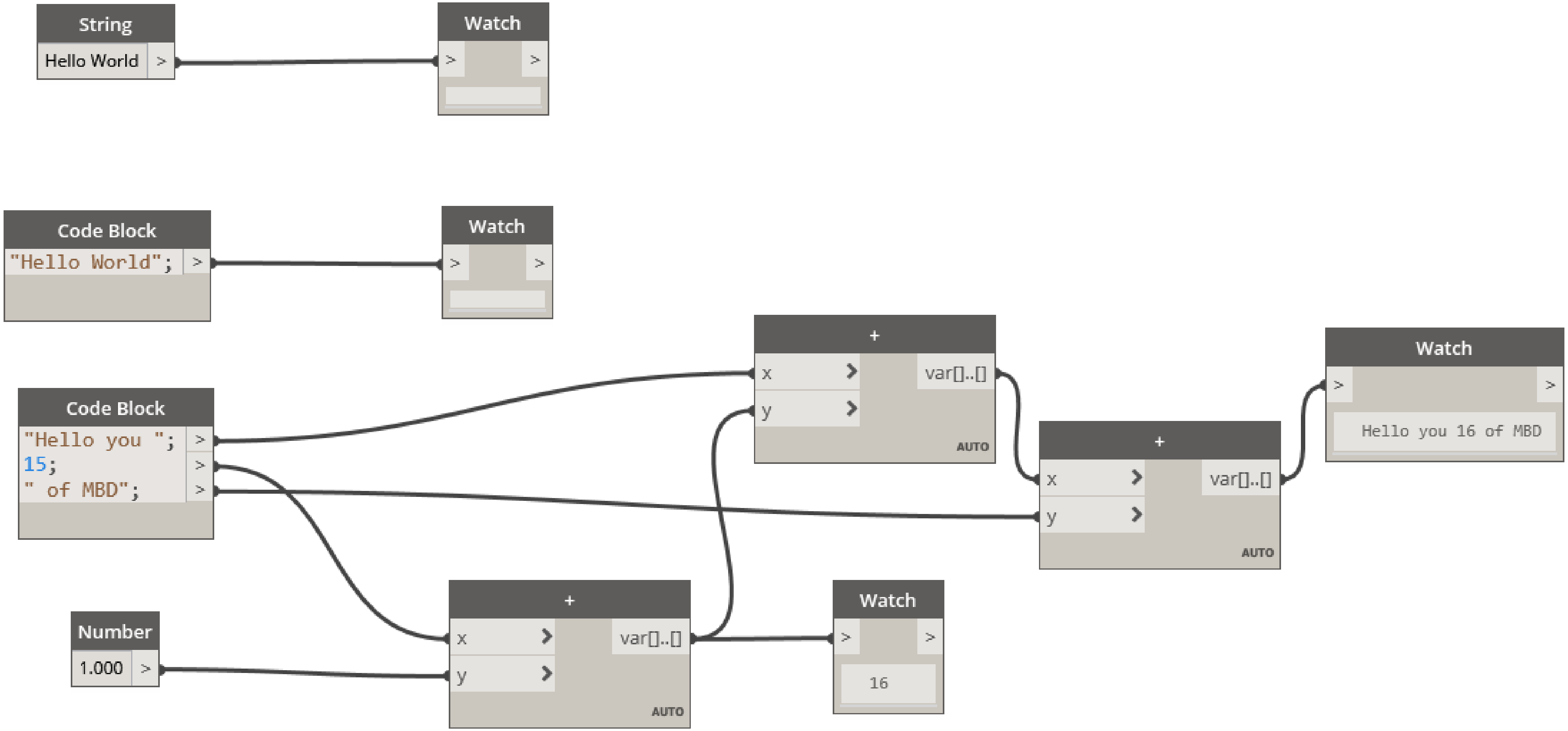
DIM_03

12.09.2024

Hanze



hello world



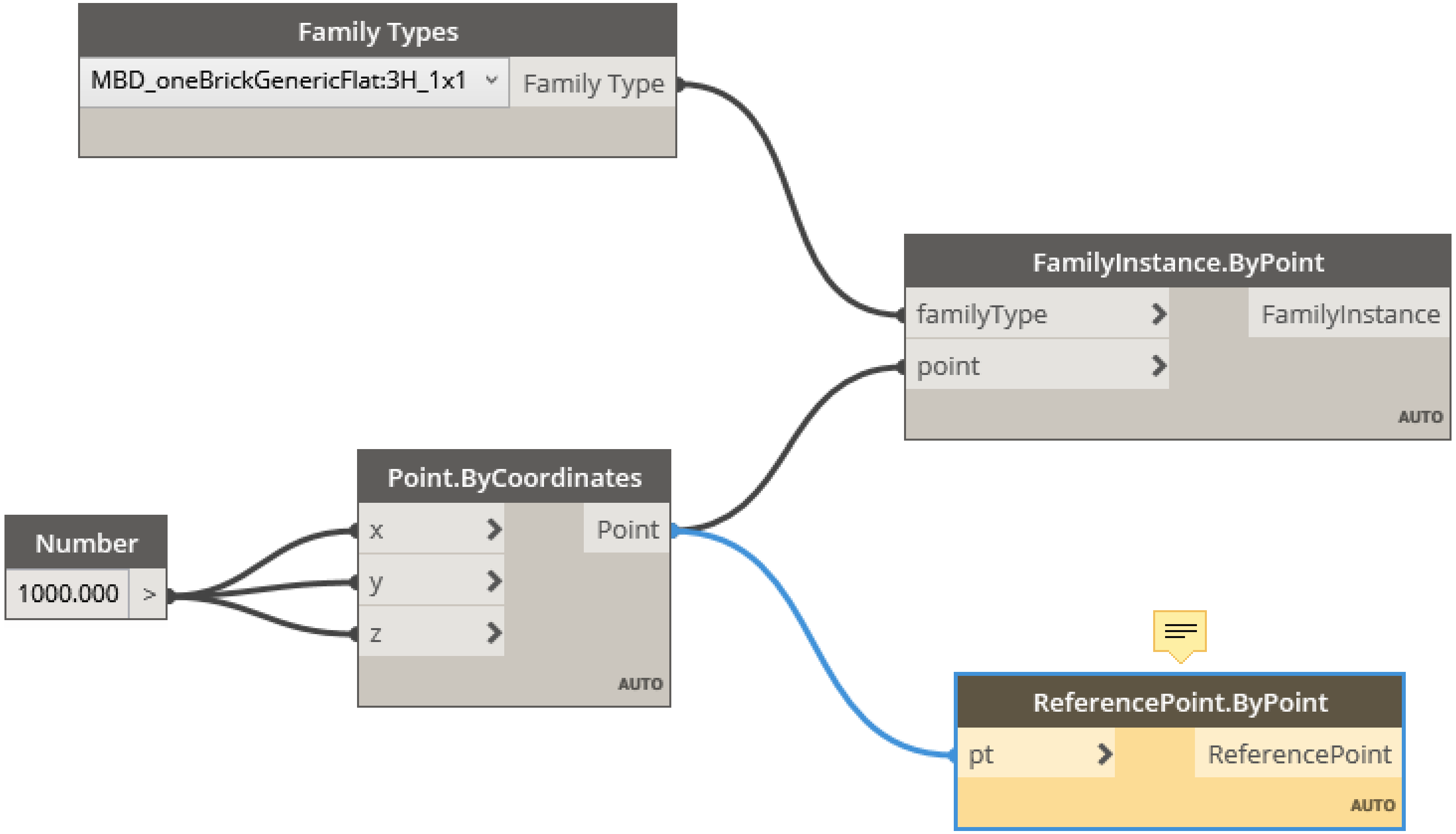
```
"Hello World";
```

```
"Hello you ";
15;
" of MBD";
```

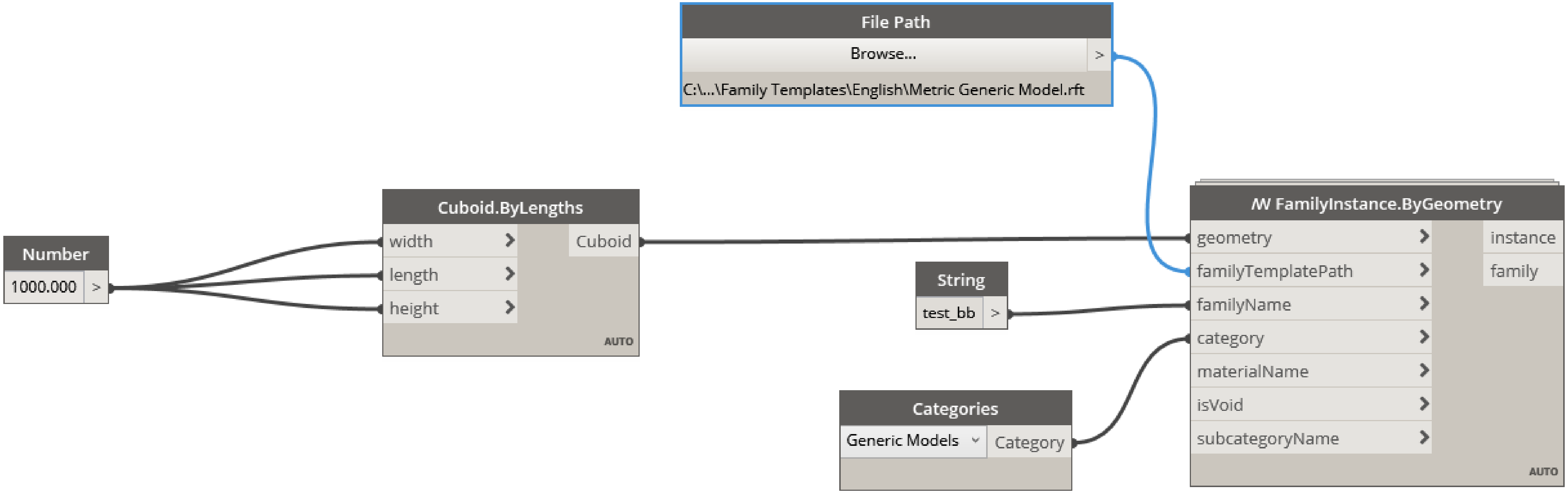
```
16
```

```
Hello you 16 of MBD
```

points


Family Types
MBD_oneBrickGenericFlat:3H_1x1
Family Type
FamilyInstance.ByPoint
familyType
point
FamilyInstance
AUTO
Number
1000.000
>
Point.ByCoordinates
x
y
z
Point
AUTO
ReferencePoint.ByPoint
pt
ReferencePoint
AUTO

CubeToRevit



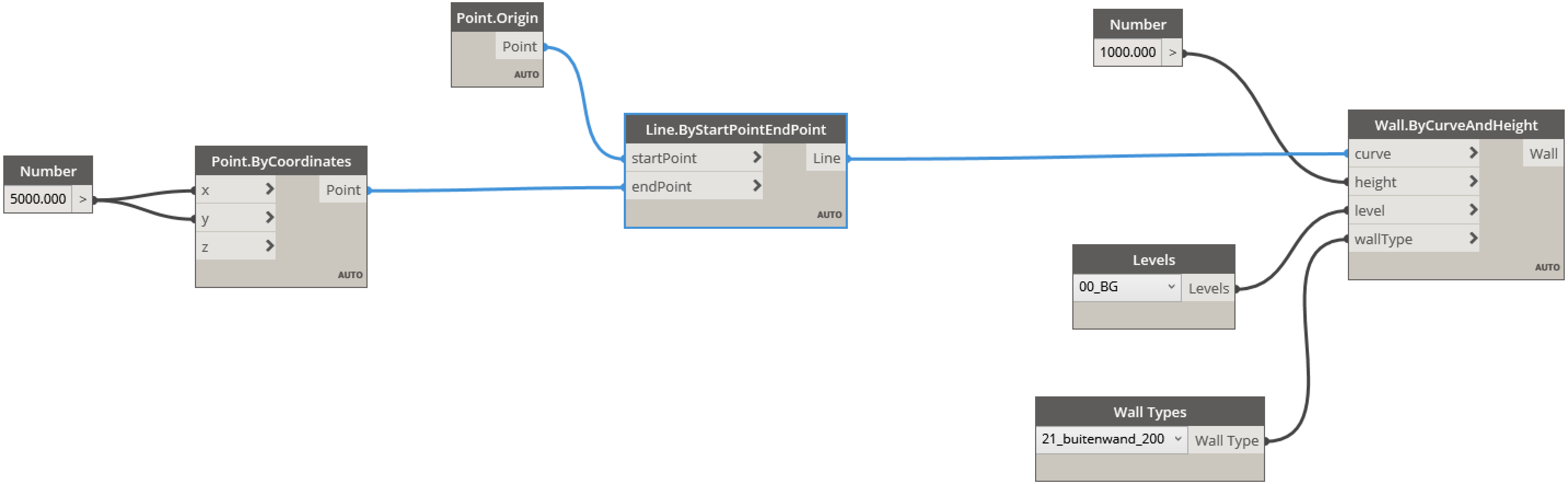
1000.000

C:\...\Family Templates\English\Metric Generic Model.rft

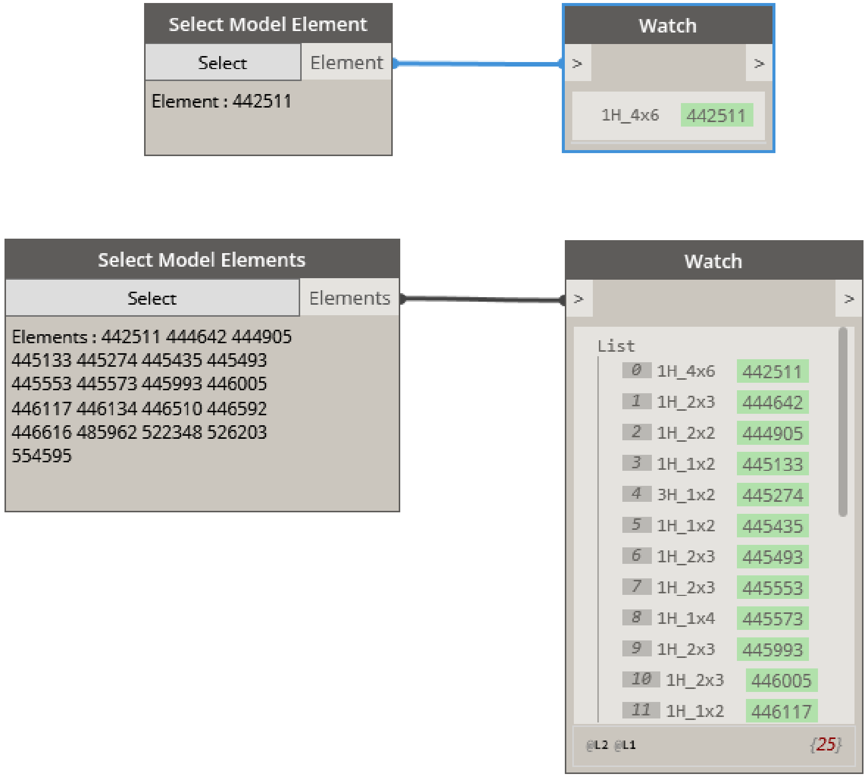
test_bb

Generic Models

WallByCurve



make your selection…



Element : 442511

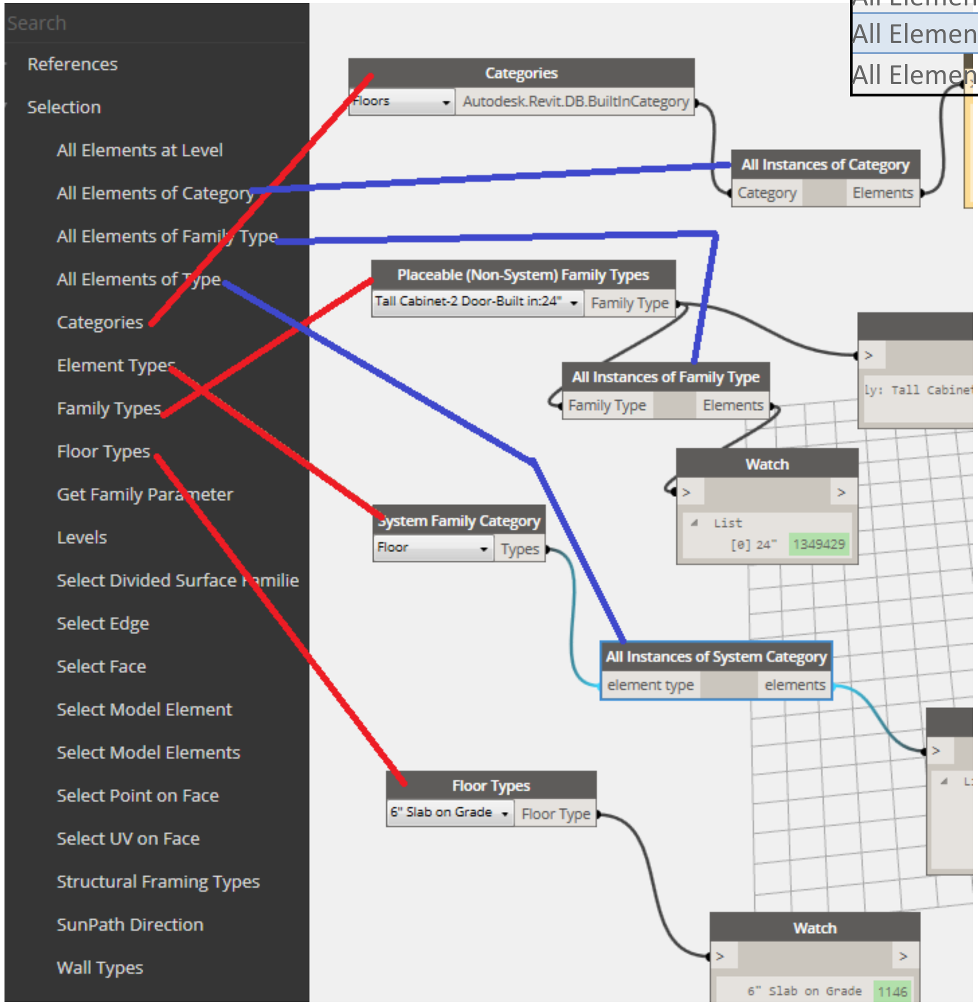
```
1H_4x6  442511
```

Elements : 442511 444642 444905 445133 445274 445435 445493 445553 445573 445993 446005 446117 446134 446510 446592 446616 485962 522348 526203 554595

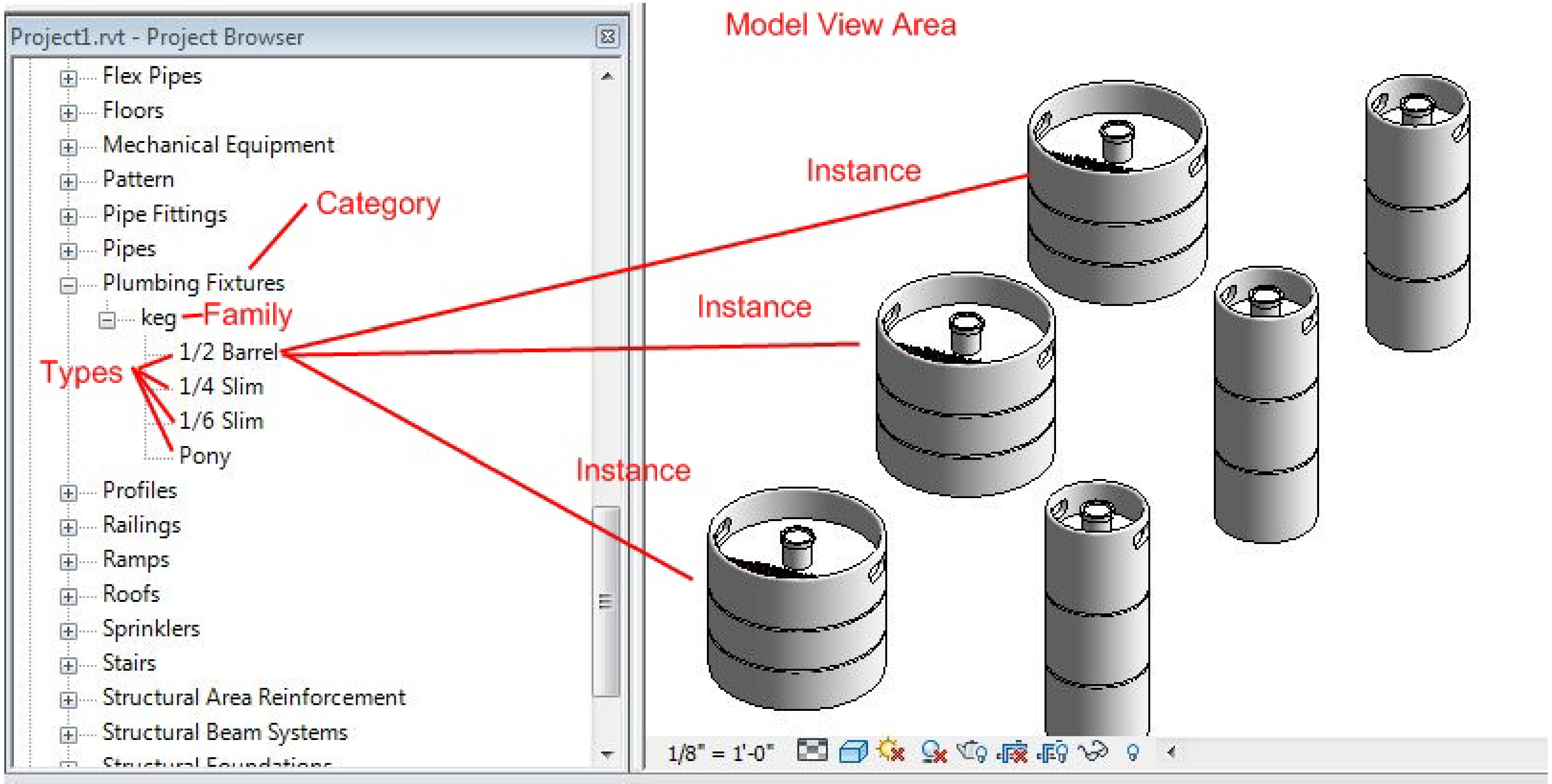
```
List
  0  1H_4x6  442511
  1  1H_2x3  444642
  2  1H_2x2  444905
  3  1H_1x2  445133
  4  3H_1x2  445274
  5  1H_1x2  445435
  6  1H_2x3  445493
  7  1H_2x3  445553
  8  1H_1x4  445573
  9  1H_2x3  445993
  10 1H_2x3  446005
  11 1H_1x2  446117
@L2 @L1                {25}
```

…from nodes

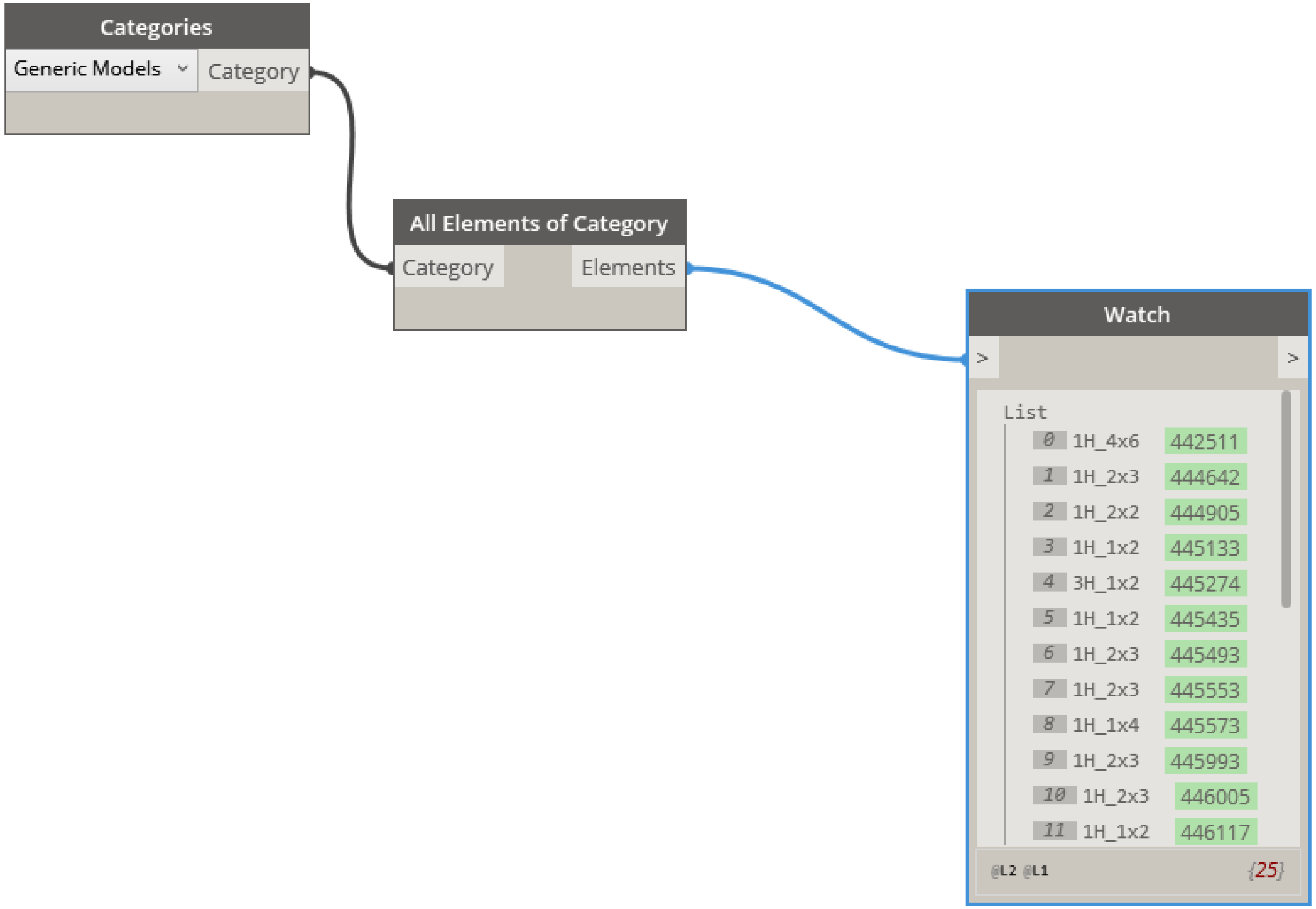
Current Node Name	What they Represent
Family Types	Placeable Family Types
Element Types	System Family Categories
All Elements of Family Type	All Instances of Family Type
All Elements of Type	All Instances of System Family Category
All Elements of Category	All Instances of Category



category, family, type, instance

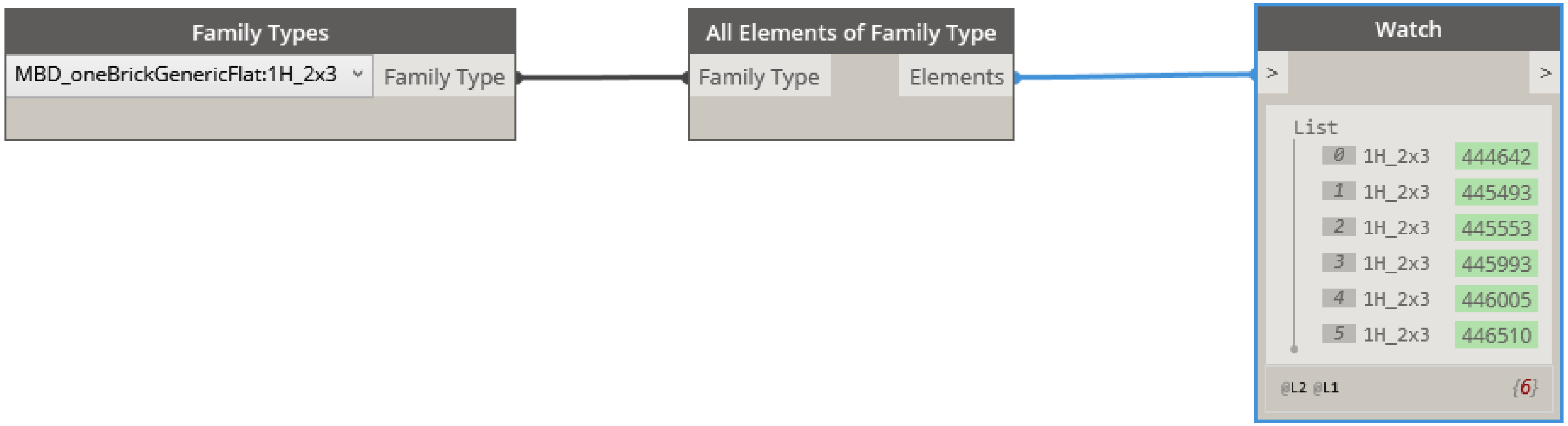


Categories



```
List
  0 1H_4x6   442511
  1 1H_2x3   444642
  2 1H_2x2   444905
  3 1H_1x2   445133
  4 3H_1x2   445274
  5 1H_1x2   445435
  6 1H_2x3   445493
  7 1H_2x3   445553
  8 1H_1x4   445573
  9 1H_2x3   445993
  10 1H_2x3   446005
  11 1H_1x2   446117
@L2 @L1                {25}
```

FamilyTypes



Family Types

MBD_oneBrickGenericFlat:1H_2x3 → Family Type

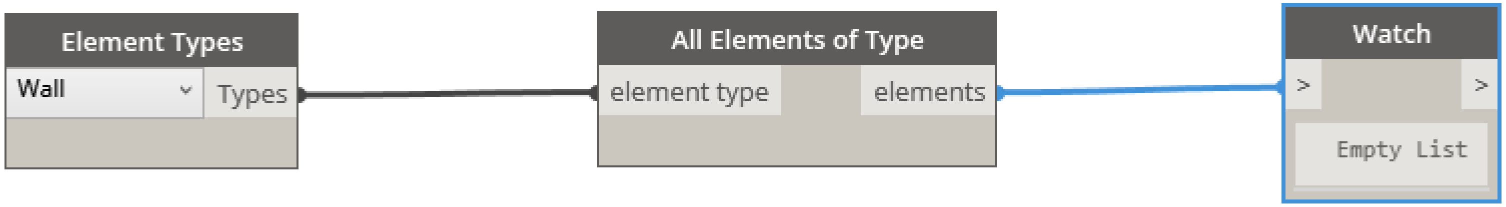
All Elements of Family Type

Family Type → Elements

Watch

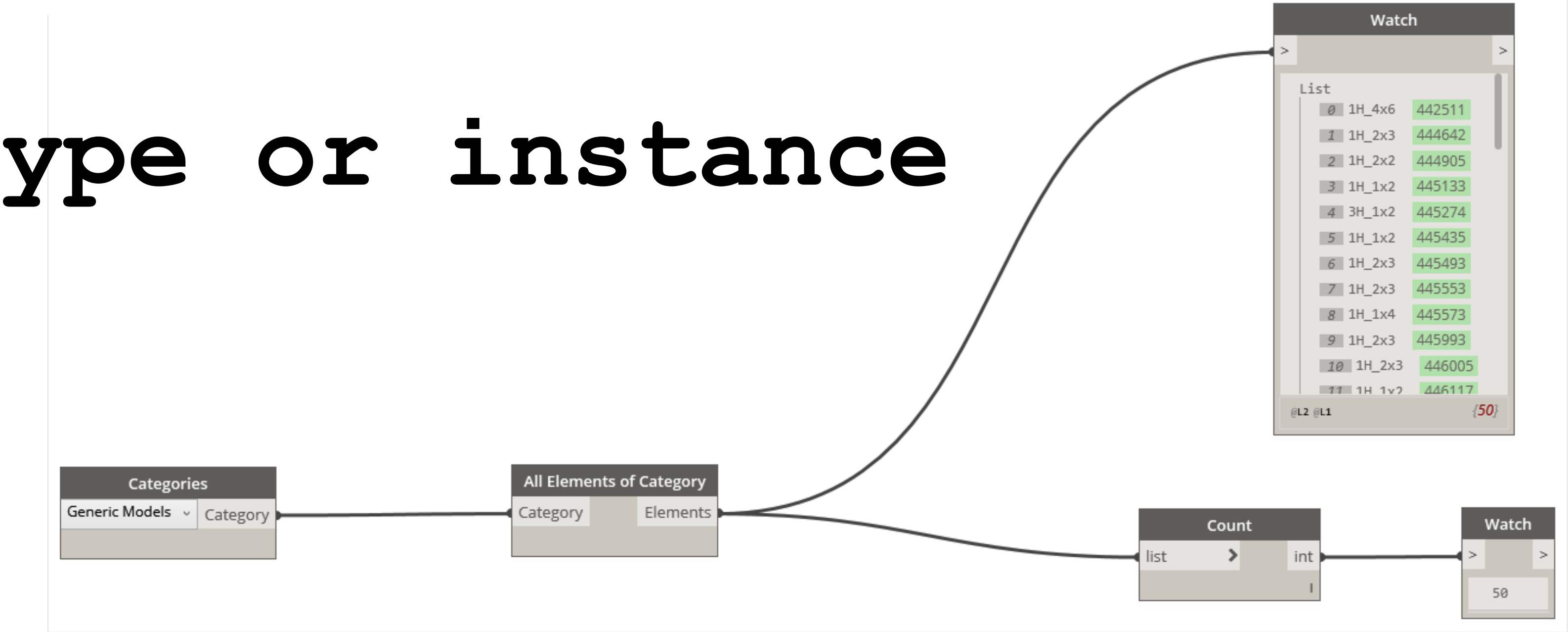
```
List
  0 1H_2x3 444642
  1 1H_2x3 445493
  2 1H_2x3 445553
  3 1H_2x3 445993
  4 1H_2x3 446005
  5 1H_2x3 446510
@L2 @L1                {6}
```

SystemFamilies



get type or instance

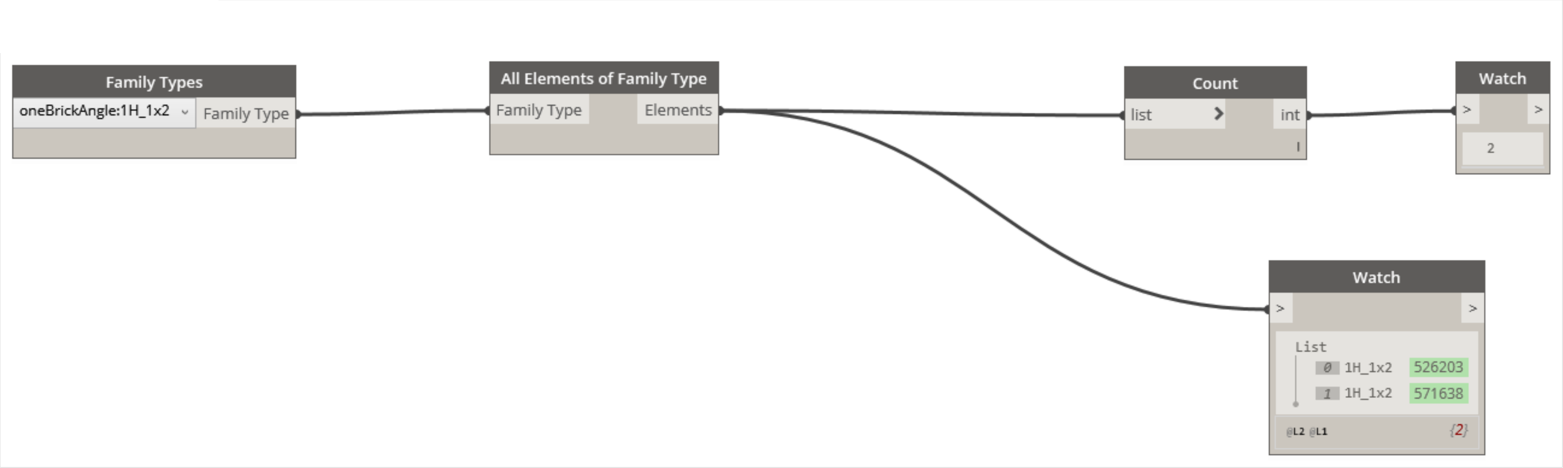
Get type:



```
List
  0 1H_4x6 442511
  1 1H_2x3 444642
  2 1H_2x2 444905
  3 1H_1x2 445133
  4 3H_1x2 445274
  5 1H_1x2 445435
  6 1H_2x3 445493
  7 1H_2x3 445553
  8 1H_1x4 445573
  9 1H_2x3 445993
  10 1H_2x3 446005
  11 1H_1x2 446117
@L2 @L1                {50}
```

```
50
```

Get instance:

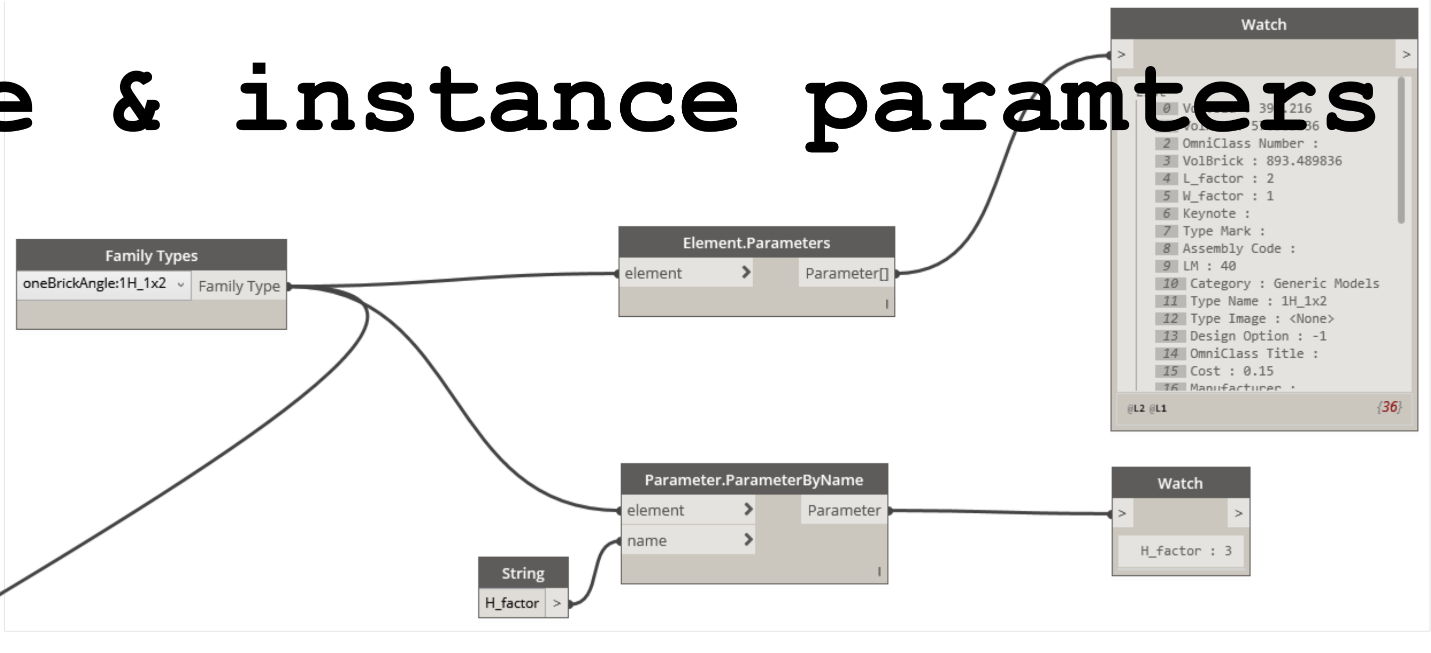


```
2
```

```
List
  0 1H_1x2 526203
  1 1H_1x2 571638
@L2 @L1                {2}
```

get type & instance paramters

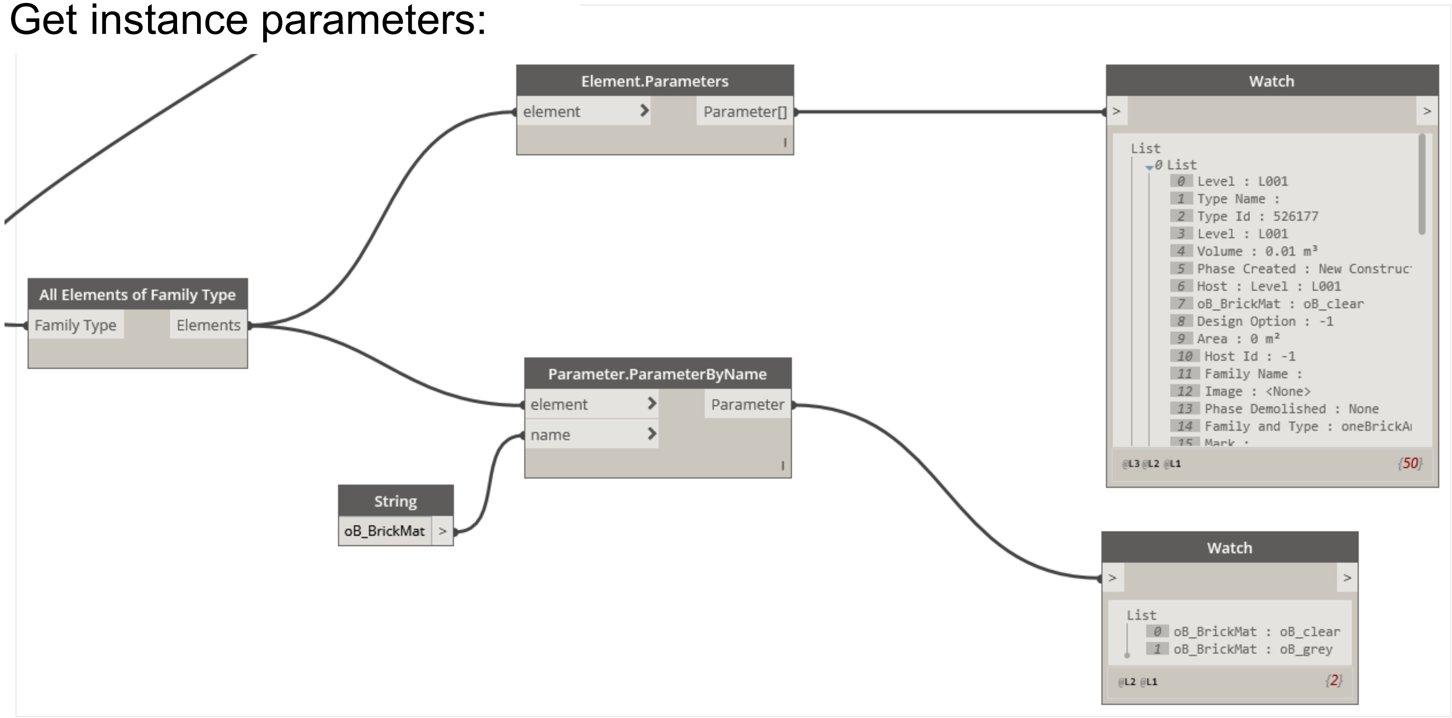
Get type parameters:



```
List
  0 Vo[illegible] 39[illegible].216
  1 Vol[illegible] 57[illegible]36
  2 OmniClass Number :
  3 VolBrick : 893.489836
  4 L_factor : 2
  5 W_factor : 1
  6 Keynote :
  7 Type Mark :
  8 Assembly Code :
  9 LM : 40
  10 Category : Generic Models
  11 Type Name : 1H_1x2
  12 Type Image : <None>
  13 Design Option : -1
  14 OmniClass Title :
  15 Cost : 0.15
  16 Manufacturer :
@L2 @L1                {36}
```

```
H_factor : 3
```

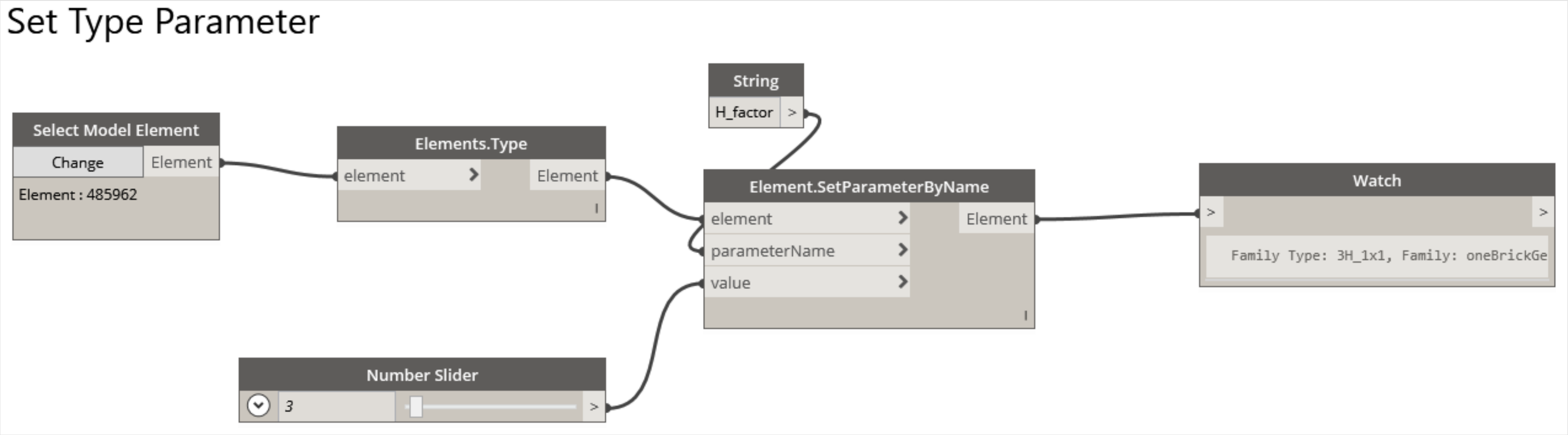
Get instance parameters:



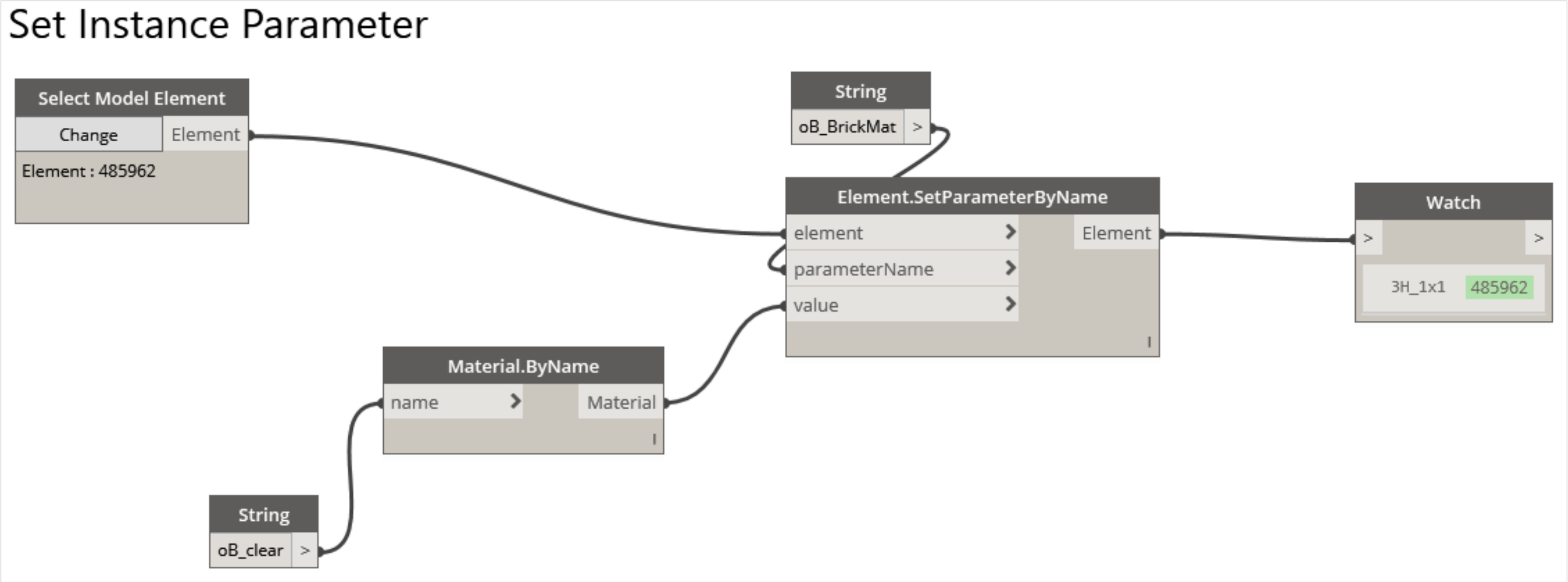
```
List
  0 List
    0 Level : L001
    1 Type Name :
    2 Type Id : 526177
    3 Level : L001
    4 Volume : 0.01 m³
    5 Phase Created : New Construc
    6 Host : Level : L001
    7 oB_BrickMat : oB_clear
    8 Design Option : -1
    9 Area : 0 m²
    10 Host Id : -1
    11 Family Name :
    12 Image : <None>
    13 Phase Demolished : None
    14 Family and Type : oneBrickA
    15 Mark :
@L3 @L2 @L1                {50}
```

```
List
  0 oB_BrickMat : oB_clear
  1 oB_BrickMat : oB_grey
@L2 @L1                {2}
```

set instance paramters

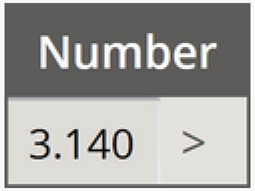
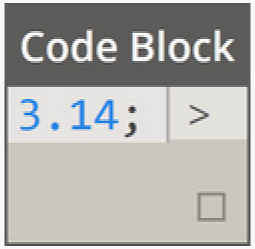
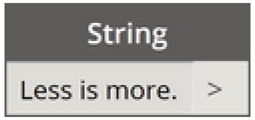
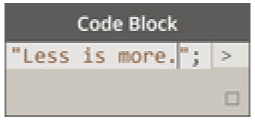
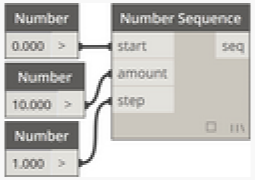
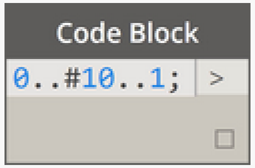
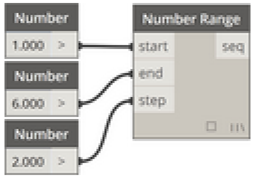
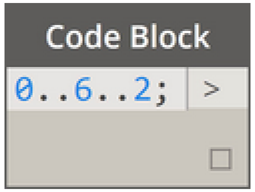


```
Family Type: 3H_1x1, Family: oneBrickGe
```



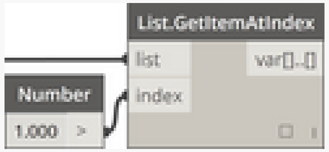
```
3H_1x1 485962
```

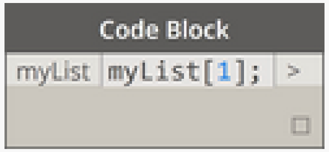
code block

Numbers	 Number 3.140	 Code Block `3.14;`
Strings	 String Less is more.	 Code Block `"Less is more.";`
Sequences	 Number 0.000 Number 10.000 Number 1.000 Number Sequence: start, amount, step → seq	 Code Block `0..#10..1;`
Ranges	 Number 1.000 Number 6.000 Number 2.000 Number Range: start, end, step → seq	 Code Block `0..6..2;`

code block

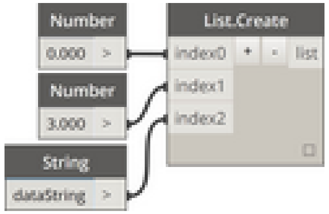
Get Item at Index

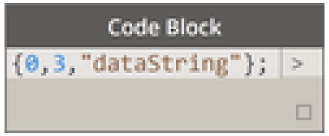




```
myList[1];
```

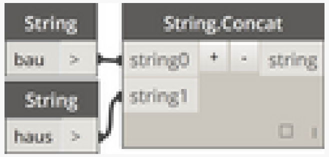
Create List

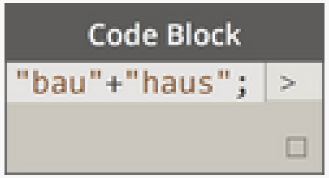




```
{0,3,"dataString"};
```

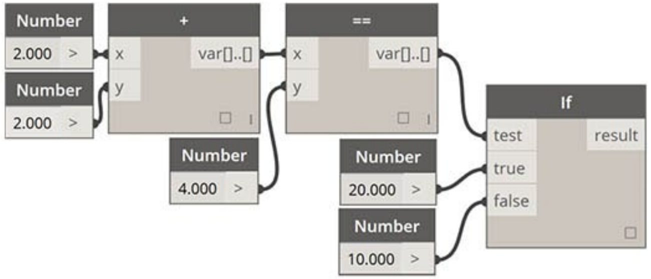
Concatenate Strings

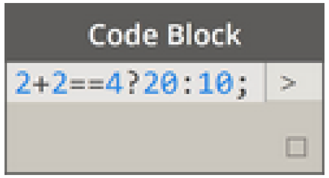




```
"bau"+"haus";
```

Conditional Statements



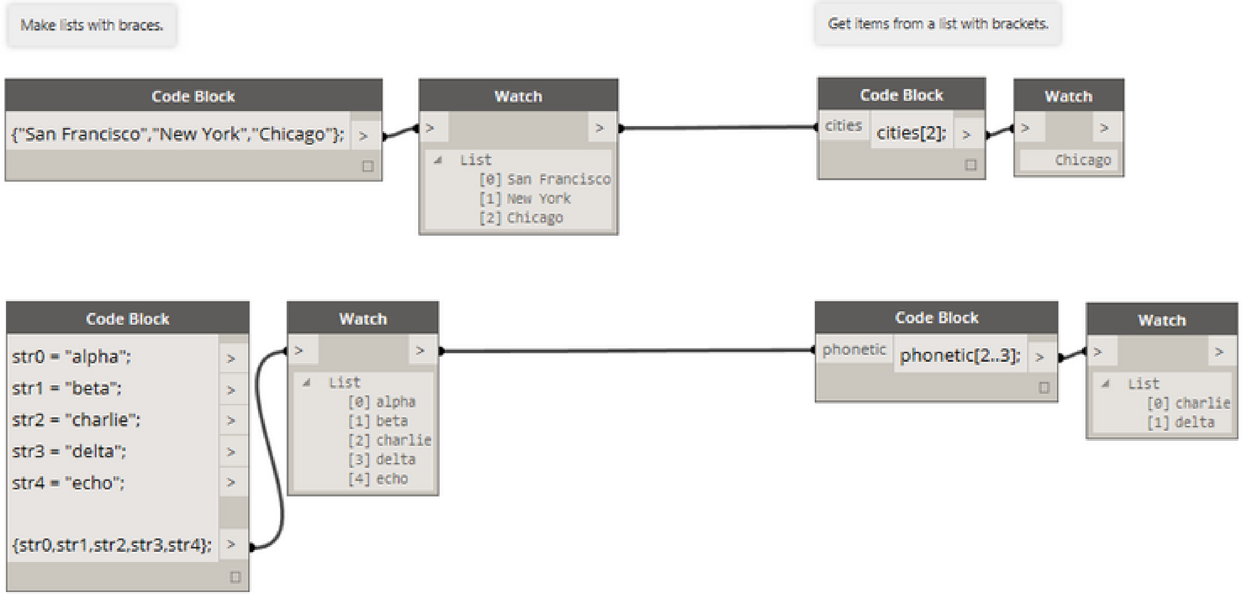


```
2+2==4?20:10;
```

lists

Make lists and get items from a list

Make lists with braces (a.k.a. “curly brackets”). Get items from a list with brackets (a.k.a. “square brackets”).



```
{"San Francisco","New York","Chicago"};
```

```
List
  [0] San Francisco
  [1] New York
  [2] Chicago
```

```
cities[2];
```

```
Chicago
```

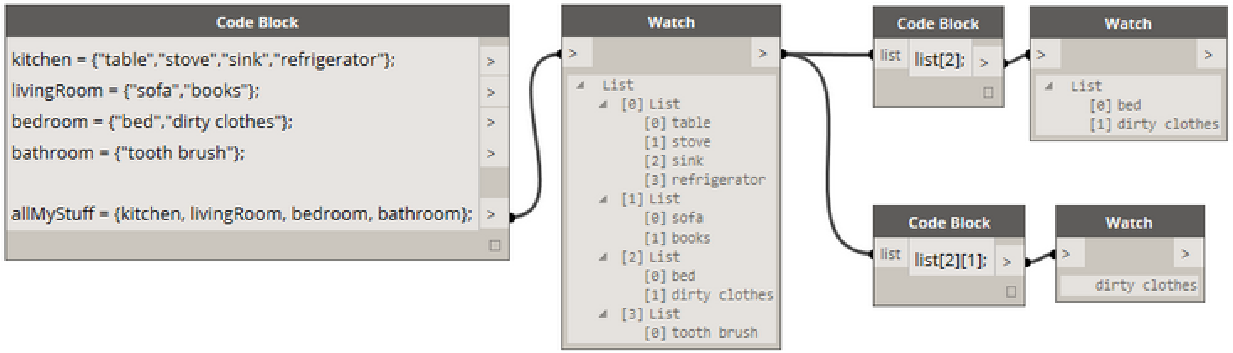
```
str0 = "alpha";
str1 = "beta";
str2 = "charlie";
str3 = "delta";
str4 = "echo";

{str0,str1,str2,str3,str4};
```

```
List
  [0] alpha
  [1] beta
  [2] charlie
  [3] delta
  [4] echo
```

```
phonetic[2..3];
```

```
List
  [0] charlie
  [1] delta
```

Working with nested lists is just a variation on the theme.



```
kitchen = {"table","stove","sink","refrigerator"};
livingRoom = {"sofa","books"};
bedroom = {"bed","dirty clothes"};
bathroom = {"tooth brush"};

allMyStuff = {kitchen, livingRoom, bedroom, bathroom};
```

```
List
  [0] List
    [0] table
    [1] stove
    [2] sink
    [3] refrigerator
  [1] List
    [0] sofa
    [1] books
  [2] List
    [0] bed
    [1] dirty clothes
  [3] List
    [0] tooth brush
```

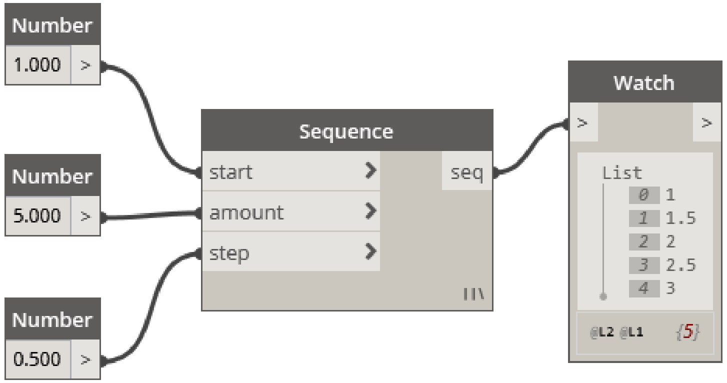
```
list[2];
```

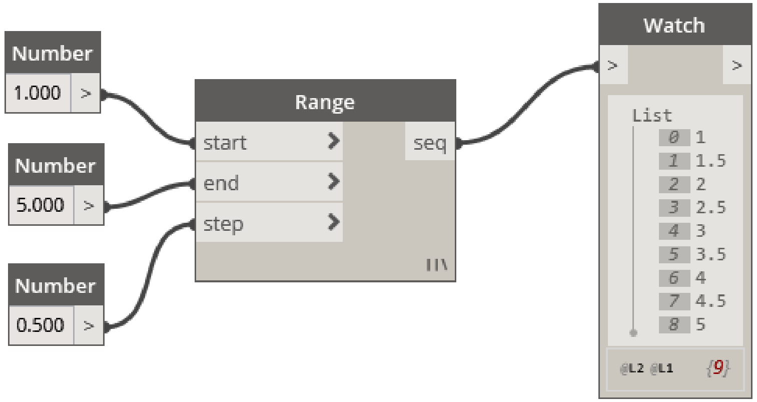
```
List
  [0] bed
  [1] dirty clothes
```

```
list[2][1];
```

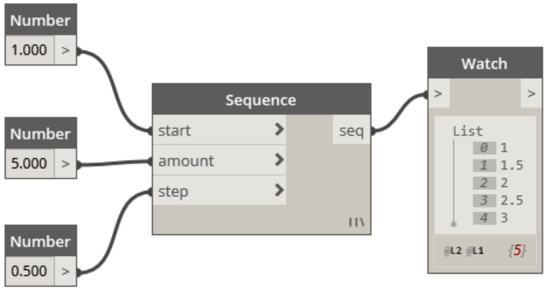
```
dirty clothes
```

lists…

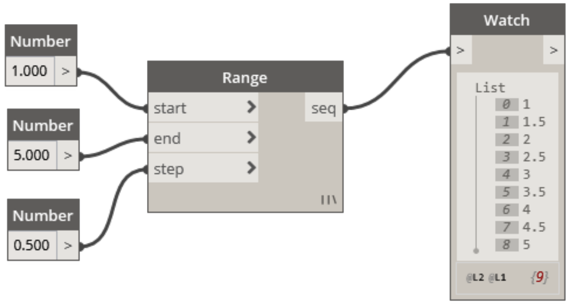

Number
1.000
Number
5.000
Number
0.500
Sequence
start
amount
step
seq
Watch
List
0 1
1 1.5
2 2
3 2.5
4 3
@L2 @L1
{5}


Number
1.000
Number
5.000
Number
0.500
Range
start
end
step
seq
Watch
List
0 1
1 1.5
2 2
3 2.5
4 3
5 3.5
6 4
7 4.5
8 5
@L2 @L1
{9}

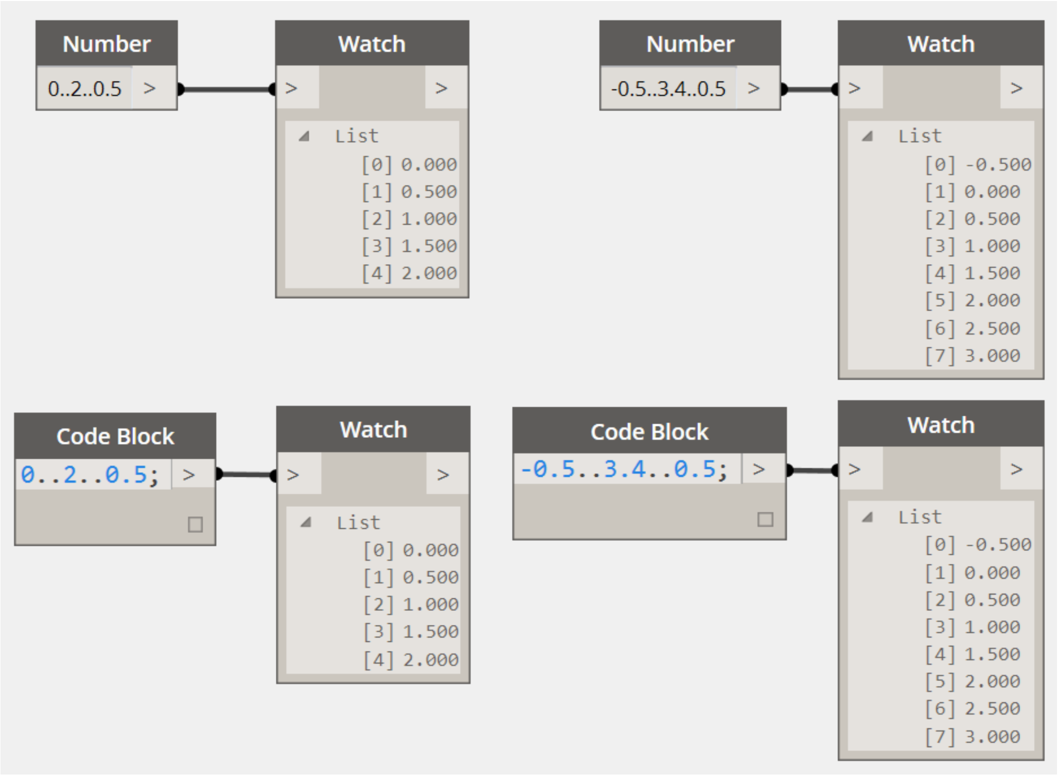
lists…



```
List
  0 1
  1 1.5
  2 2
  3 2.5
  4 3
@L2 @L1   {5}
```



```
List
  0 1
  1 1.5
  2 2
  3 2.5
  4 3
  5 3.5
  6 4
  7 4.5
  8 5
@L2 @L1   {9}
```



```
0..2..0.5;
```

```
List
  [0] 0.000
  [1] 0.500
  [2] 1.000
  [3] 1.500
  [4] 2.000
```

```
-0.5..3.4..0.5;
```

```
List
  [0] -0.500
  [1] 0.000
  [2] 0.500
  [3] 1.000
  [4] 1.500
  [5] 2.000
  [6] 2.500
  [7] 3.000
```

lists?

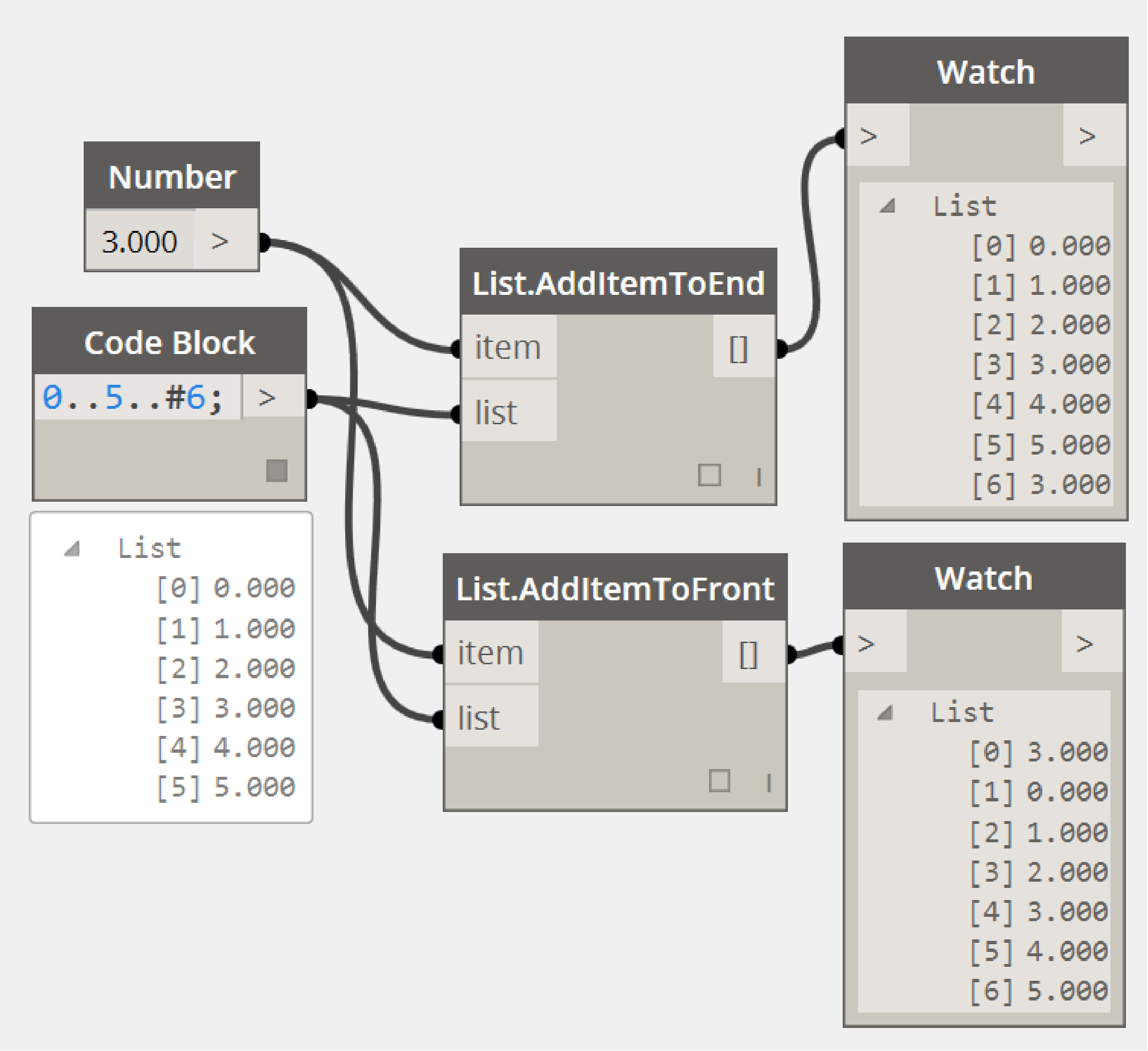
important?

list management
is the foundation of your
coding future…

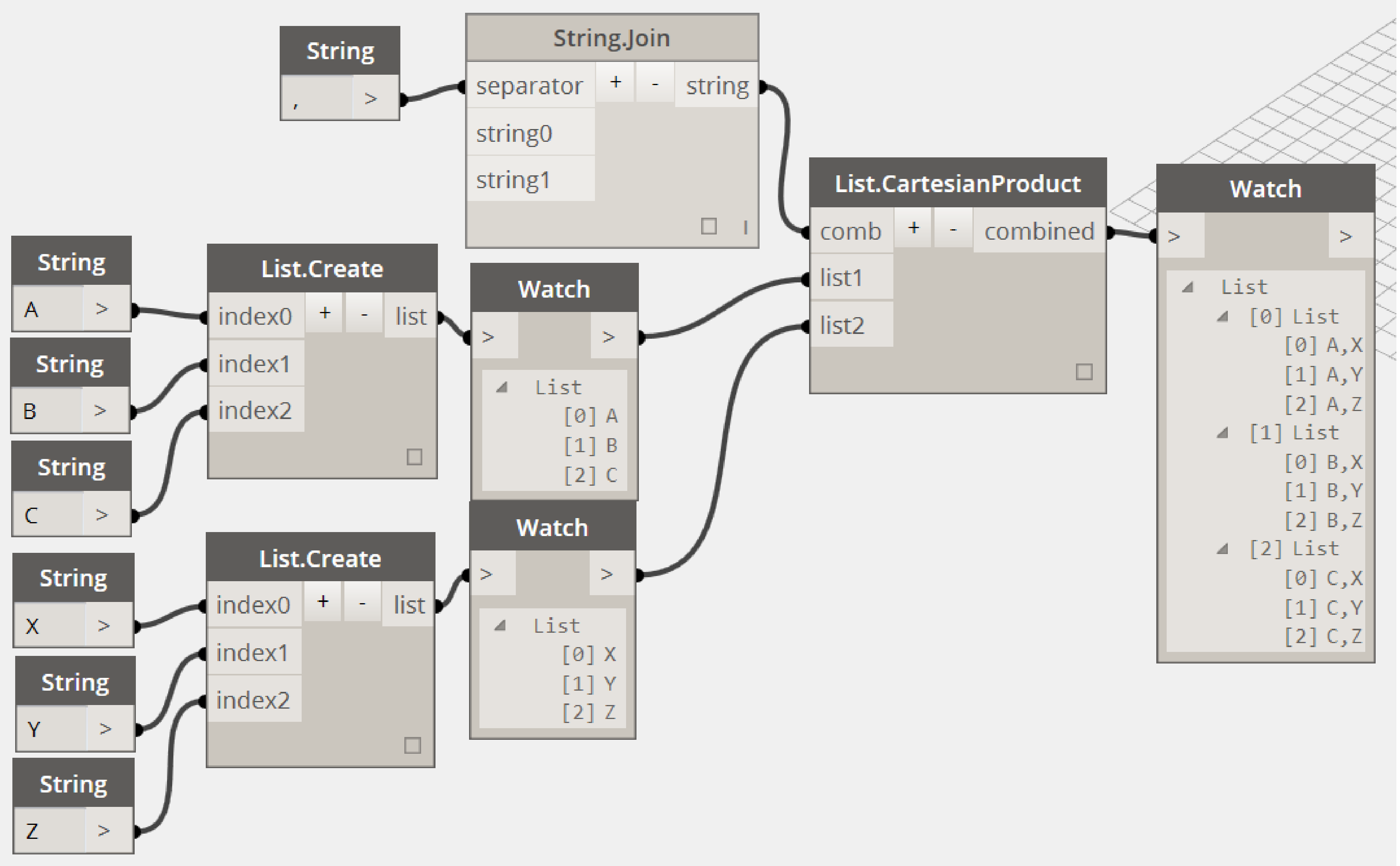
but step by step…

one list operation per day
keeps the doctor away

add item


Number
3.000
Code Block
0..5..#6;
List
[0] 0.000
[1] 1.000
[2] 2.000
[3] 3.000
[4] 4.000
[5] 5.000
List.AddItemToEnd
item
list
[]
List.AddItemToFront
item
list
[]
Watch
List
[0] 0.000
[1] 1.000
[2] 2.000
[3] 3.000
[4] 4.000
[5] 5.000
[6] 3.000
Watch
List
[0] 3.000
[1] 0.000
[2] 1.000
[3] 2.000
[4] 3.000
[5] 4.000
[6] 5.000

cartesian product

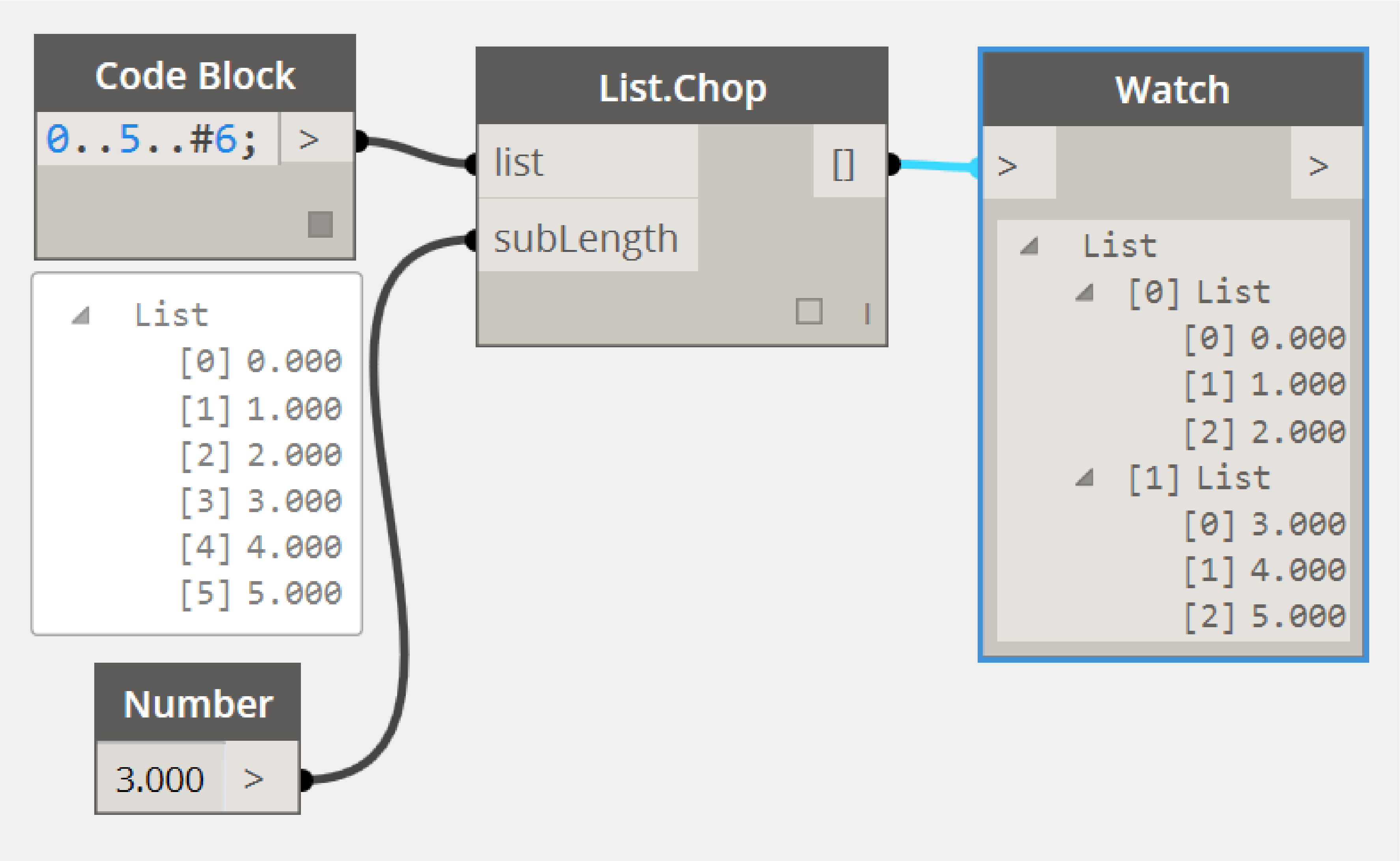


```
List
  [0] A
  [1] B
  [2] C
```

```
List
  [0] X
  [1] Y
  [2] Z
```

```
List
  [0] List
    [0] A,X
    [1] A,Y
    [2] A,Z
  [1] List
    [0] B,X
    [1] B,Y
    [2] B,Z
  [2] List
    [0] C,X
    [1] C,Y
    [2] C,Z
```

chop



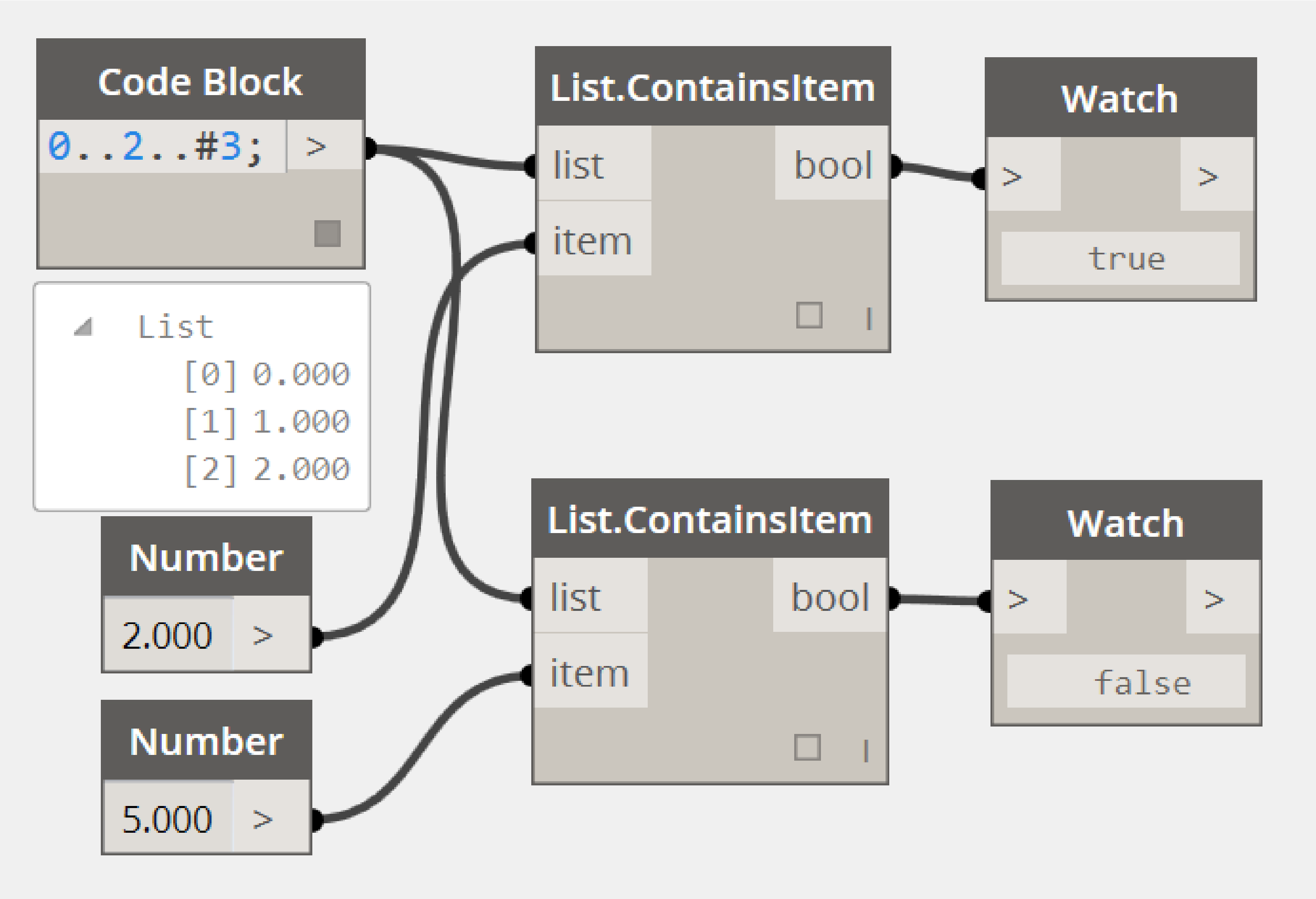
```
0..5..#6;
```

```
List
    [0] 0.000
    [1] 1.000
    [2] 2.000
    [3] 3.000
    [4] 4.000
    [5] 5.000
```

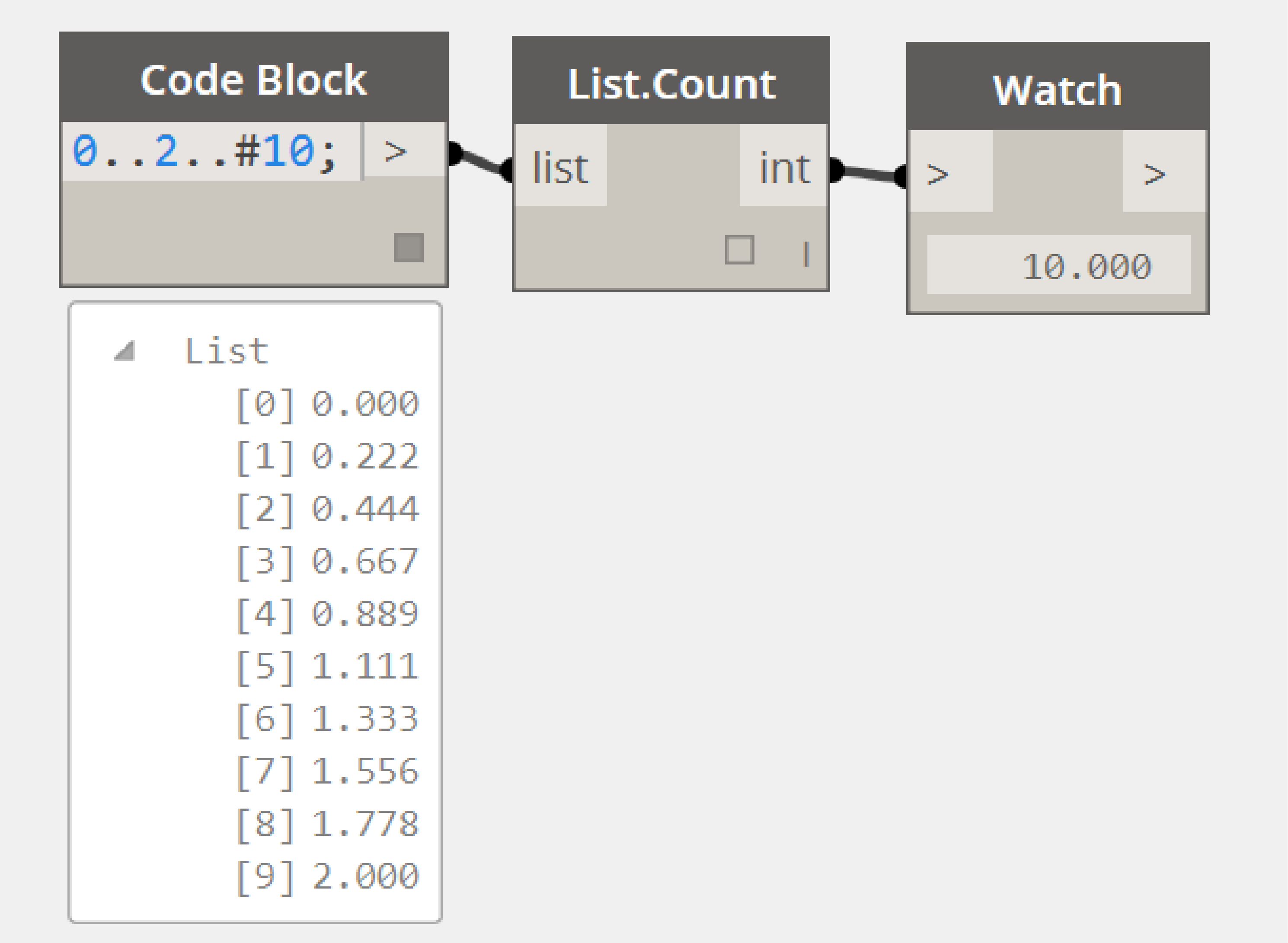
```
3.000
```

```
List
  [0] List
      [0] 0.000
      [1] 1.000
      [2] 2.000
  [1] List
      [0] 3.000
      [1] 4.000
      [2] 5.000
```

contain items

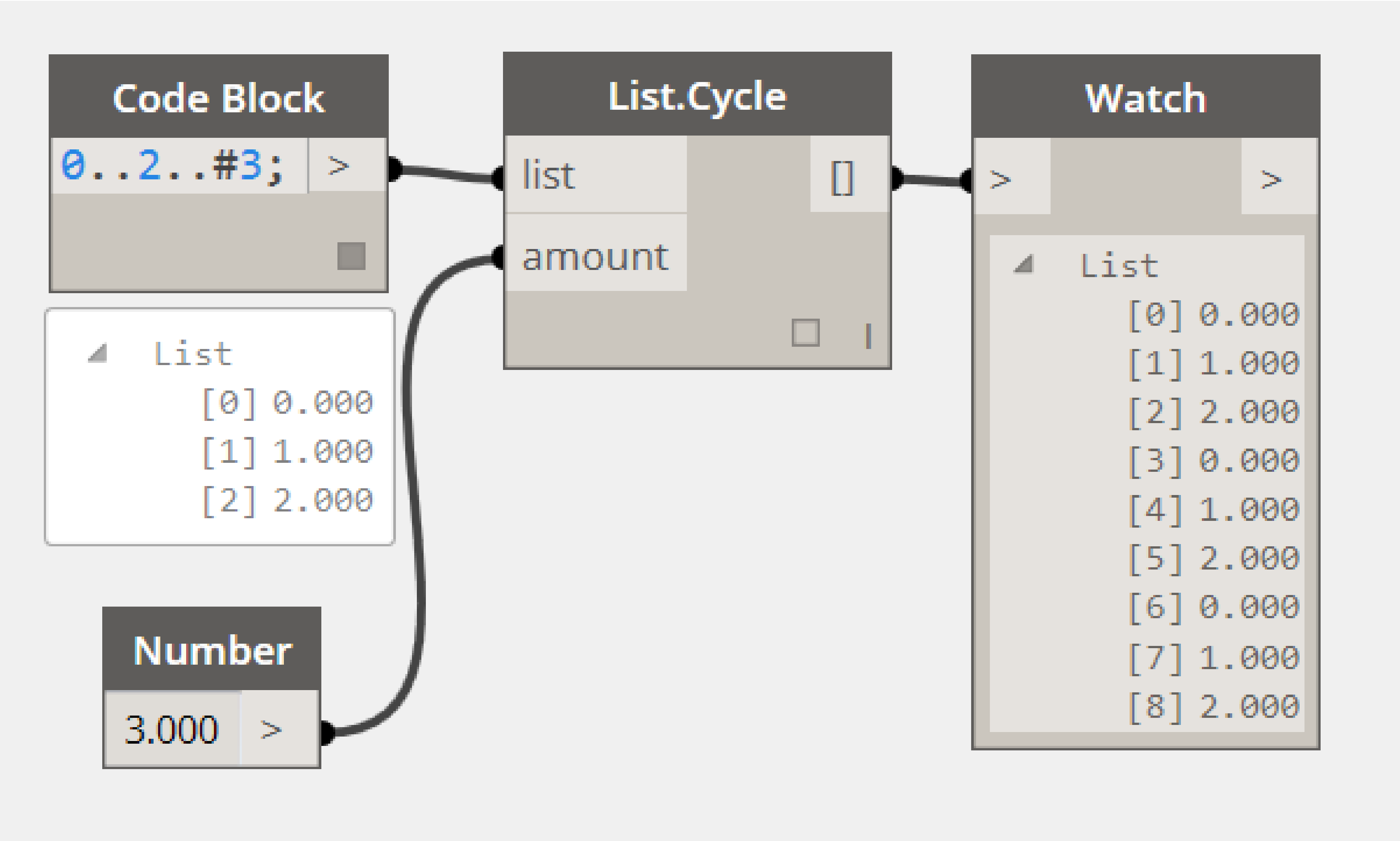


count



Code Block
0..2..#10;
>
List.Count
list
int
Watch
>
>
10.000
List
[0] 0.000
[1] 0.222
[2] 0.444
[3] 0.667
[4] 0.889
[5] 1.111
[6] 1.333
[7] 1.556
[8] 1.778
[9] 2.000

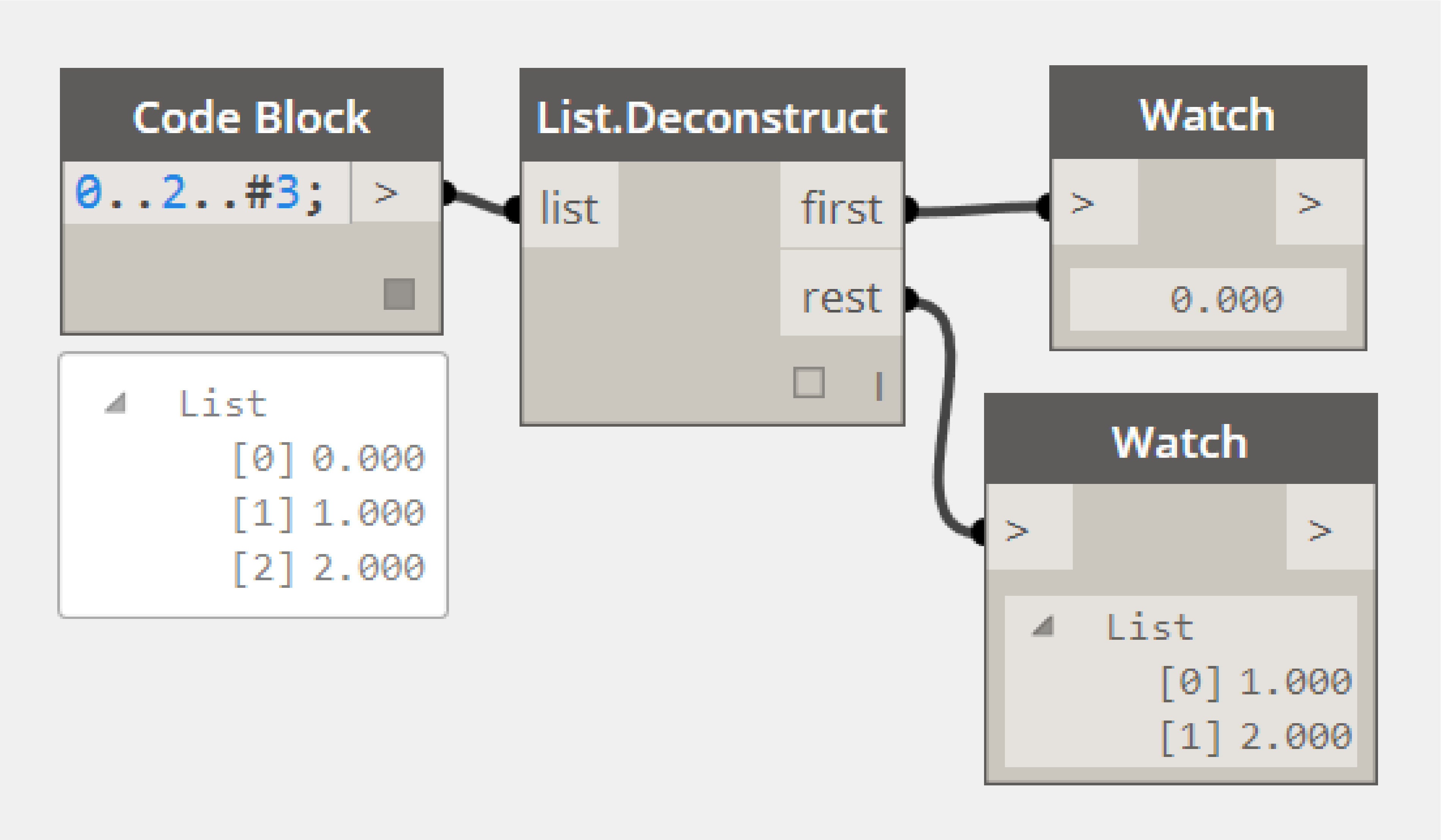
cycle



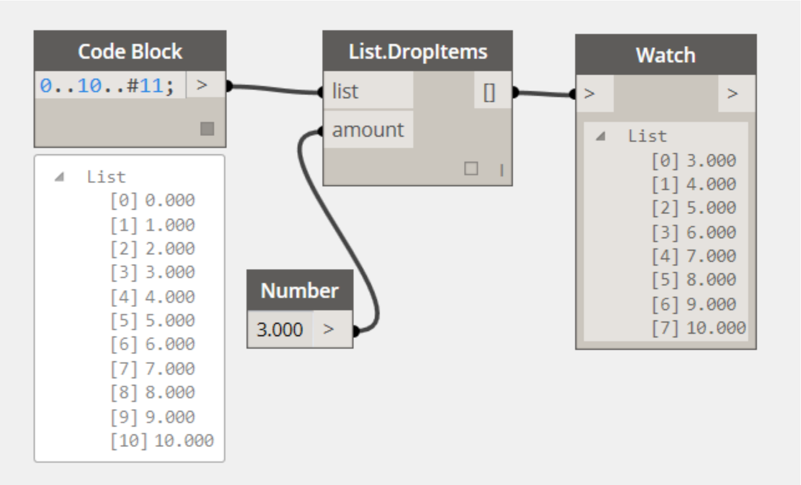
```
List
  [0] 0.000
  [1] 1.000
  [2] 2.000
```

```
List
  [0] 0.000
  [1] 1.000
  [2] 2.000
  [3] 0.000
  [4] 1.000
  [5] 2.000
  [6] 0.000
  [7] 1.000
  [8] 2.000
```

deconstruct



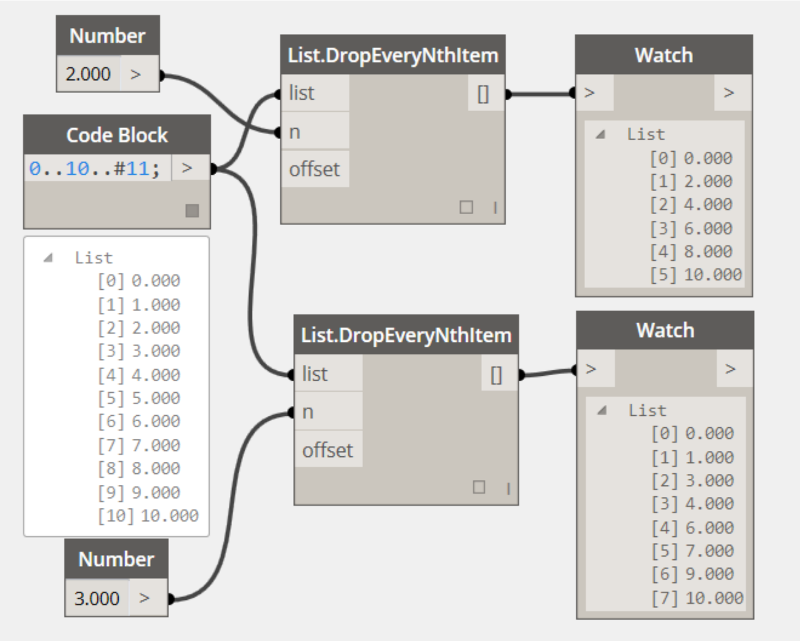
drop



```
0..10..#11;
```

```
List
  [0] 0.000
  [1] 1.000
  [2] 2.000
  [3] 3.000
  [4] 4.000
  [5] 5.000
  [6] 6.000
  [7] 7.000
  [8] 8.000
  [9] 9.000
  [10] 10.000
```

```
List
  [0] 3.000
  [1] 4.000
  [2] 5.000
  [3] 6.000
  [4] 7.000
  [5] 8.000
  [6] 9.000
  [7] 10.000
```



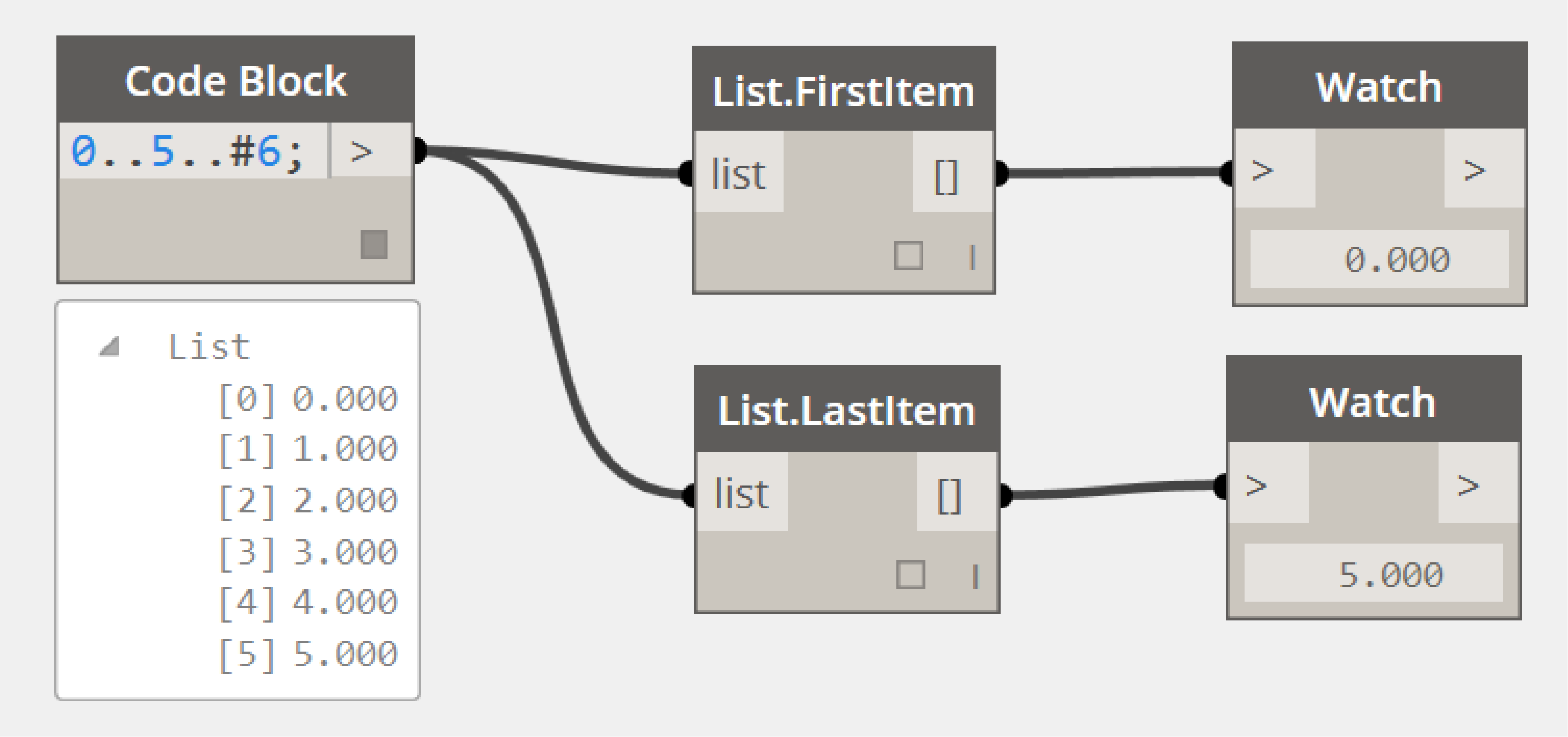
```
0..10..#11;
```

```
List
  [0] 0.000
  [1] 1.000
  [2] 2.000
  [3] 3.000
  [4] 4.000
  [5] 5.000
  [6] 6.000
  [7] 7.000
  [8] 8.000
  [9] 9.000
  [10] 10.000
```

```
List
  [0] 0.000
  [1] 2.000
  [2] 4.000
  [3] 6.000
  [4] 8.000
  [5] 10.000
```

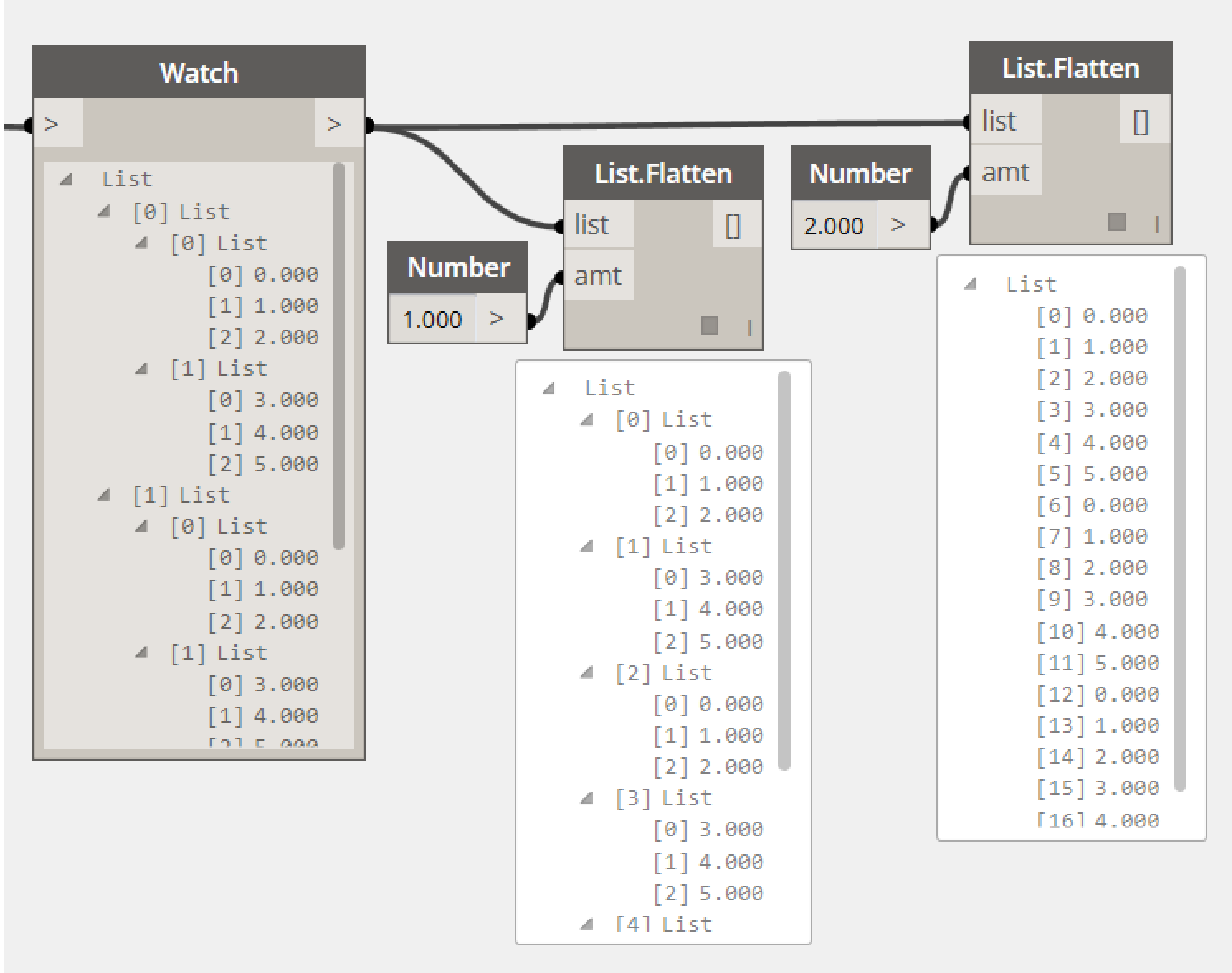
```
List
  [0] 0.000
  [1] 1.000
  [2] 3.000
  [3] 4.000
  [4] 6.000
  [5] 7.000
  [6] 9.000
  [7] 10.000
```

first/last



```
0..5..#6;
```

flatten

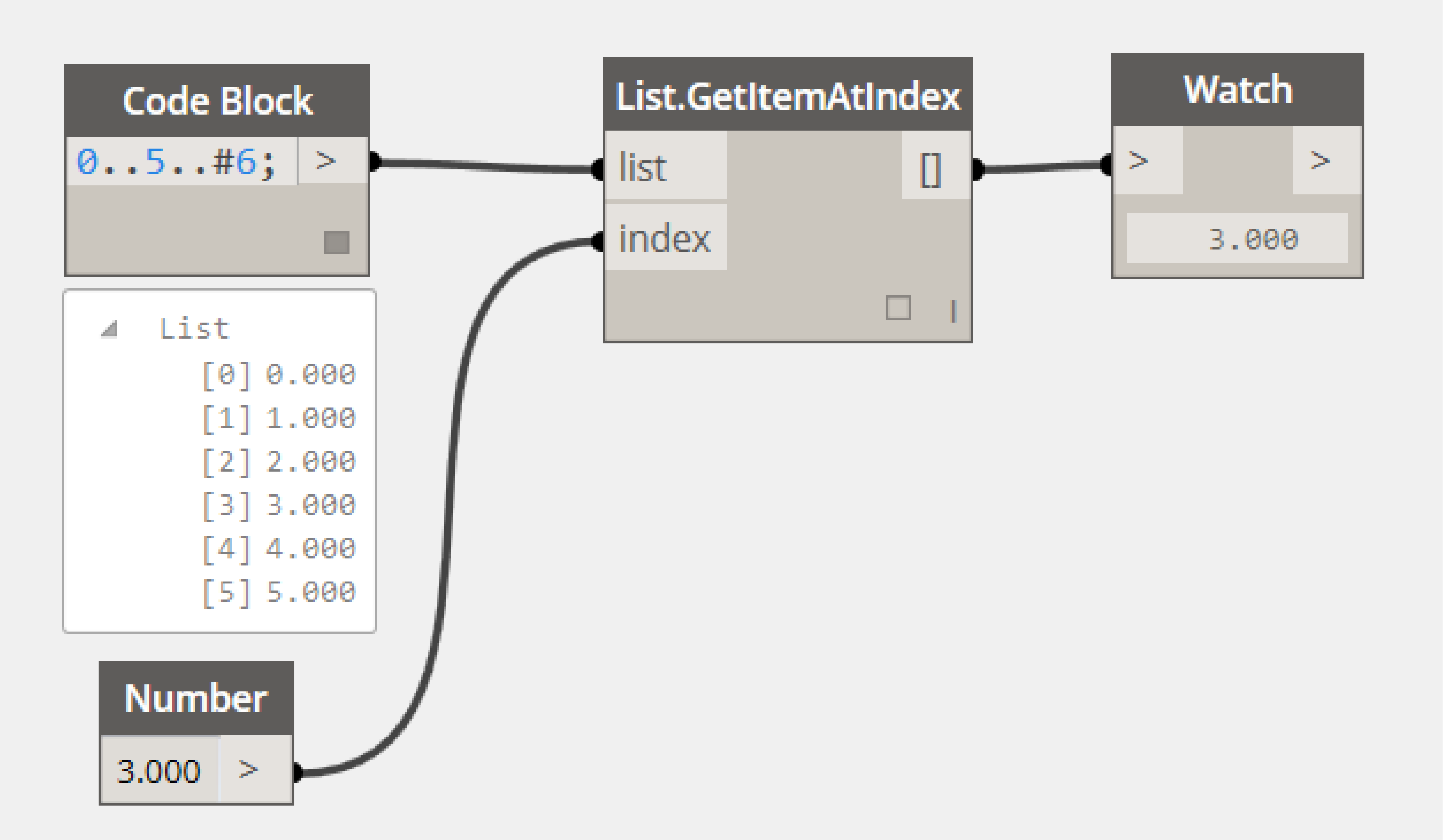


```
List
  [0] List
    [0] List
      [0] 0.000
      [1] 1.000
      [2] 2.000
    [1] List
      [0] 3.000
      [1] 4.000
      [2] 5.000
  [1] List
    [0] List
      [0] 0.000
      [1] 1.000
      [2] 2.000
    [1] List
      [0] 3.000
      [1] 4.000
      [2] 5.000
```

```
List
  [0] List
    [0] 0.000
    [1] 1.000
    [2] 2.000
  [1] List
    [0] 3.000
    [1] 4.000
    [2] 5.000
  [2] List
    [0] 0.000
    [1] 1.000
    [2] 2.000
  [3] List
    [0] 3.000
    [1] 4.000
    [2] 5.000
  [4] List
```

```
List
  [0] 0.000
  [1] 1.000
  [2] 2.000
  [3] 3.000
  [4] 4.000
  [5] 5.000
  [6] 0.000
  [7] 1.000
  [8] 2.000
  [9] 3.000
  [10] 4.000
  [11] 5.000
  [12] 0.000
  [13] 1.000
  [14] 2.000
  [15] 3.000
  [16] 4.000
```

get @ index



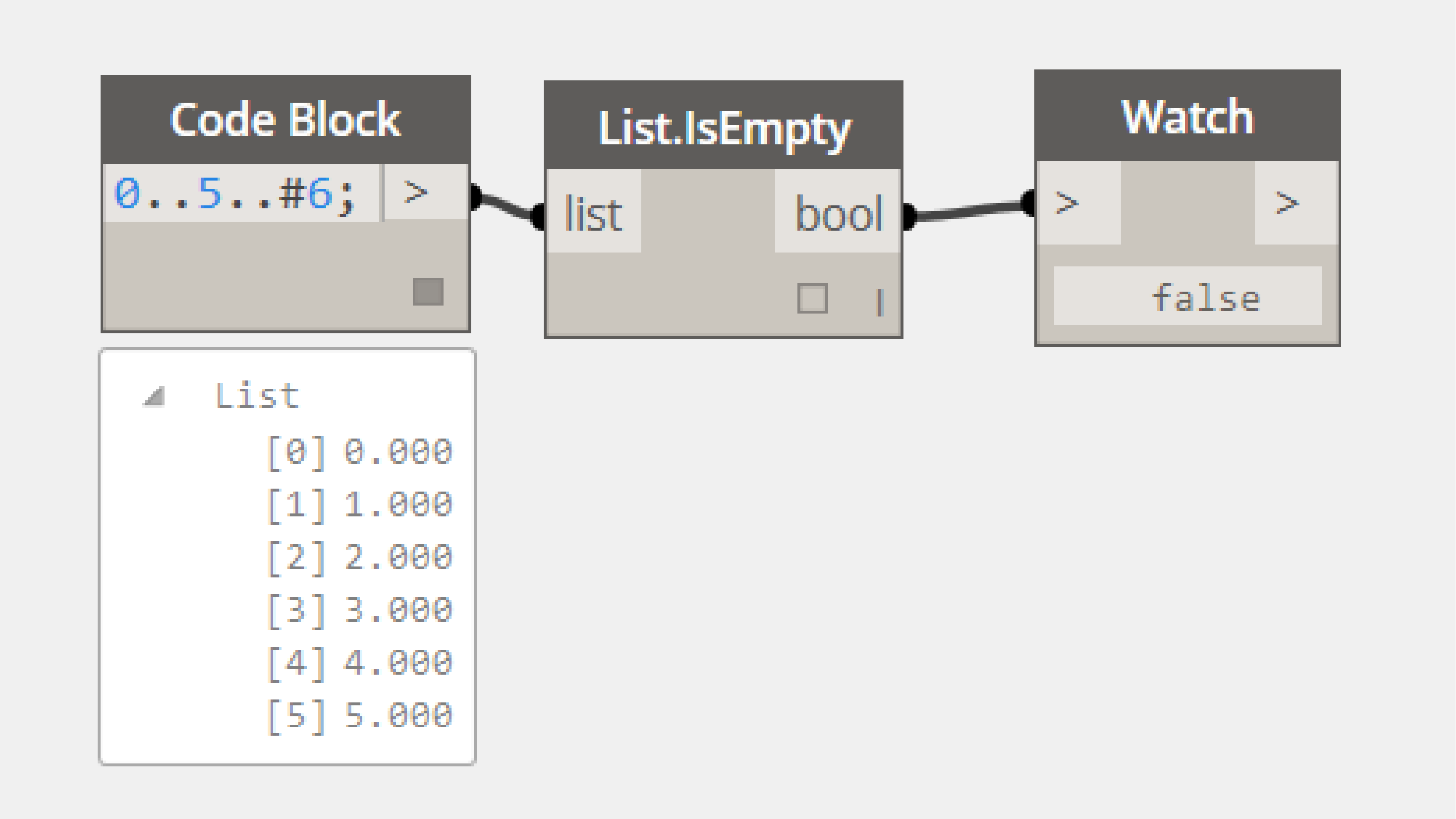
```
0..5..#6;
```

```
List
    [0] 0.000
    [1] 1.000
    [2] 2.000
    [3] 3.000
    [4] 4.000
    [5] 5.000
```

```
3.000
```

3.000

is empty

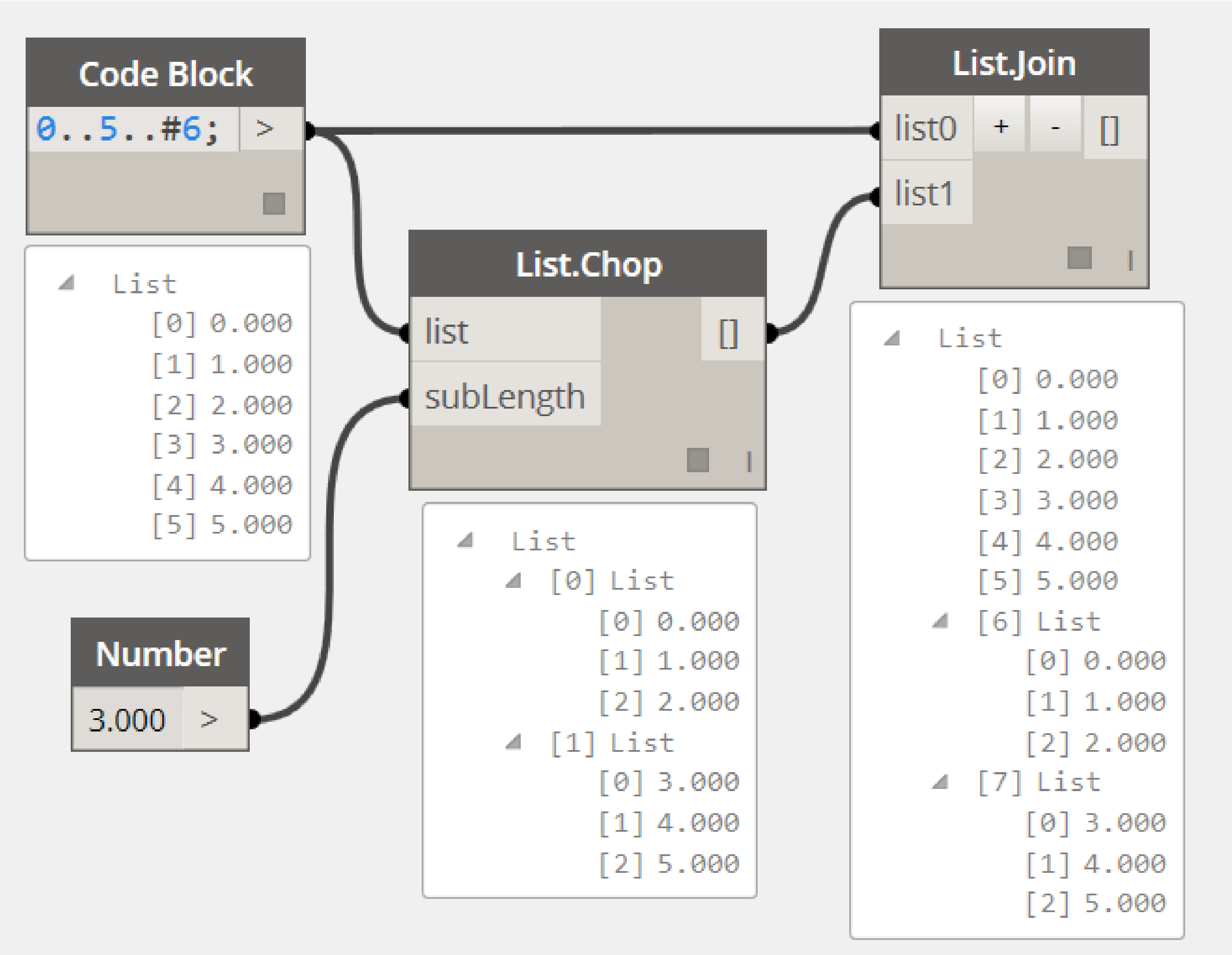


```
0..5..#6;
```

```
List
  [0] 0.000
  [1] 1.000
  [2] 2.000
  [3] 3.000
  [4] 4.000
  [5] 5.000
```

```
false
```

join

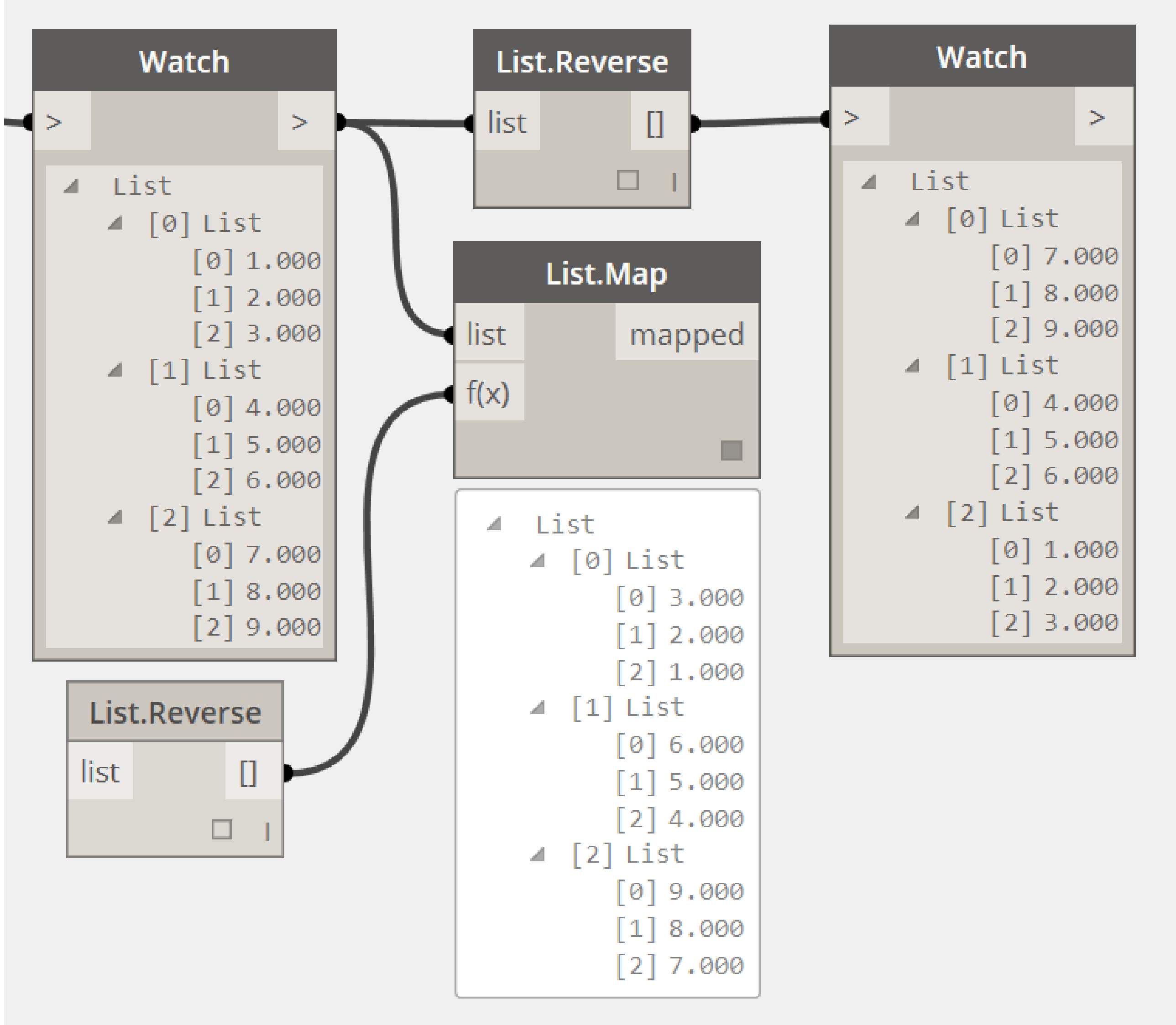


```
List
    [0] 0.000
    [1] 1.000
    [2] 2.000
    [3] 3.000
    [4] 4.000
    [5] 5.000
```

```
List
  [0] List
      [0] 0.000
      [1] 1.000
      [2] 2.000
  [1] List
      [0] 3.000
      [1] 4.000
      [2] 5.000
```

```
List
    [0] 0.000
    [1] 1.000
    [2] 2.000
    [3] 3.000
    [4] 4.000
    [5] 5.000
  [6] List
      [0] 0.000
      [1] 1.000
      [2] 2.000
  [7] List
      [0] 3.000
      [1] 4.000
      [2] 5.000
```

map/reverse

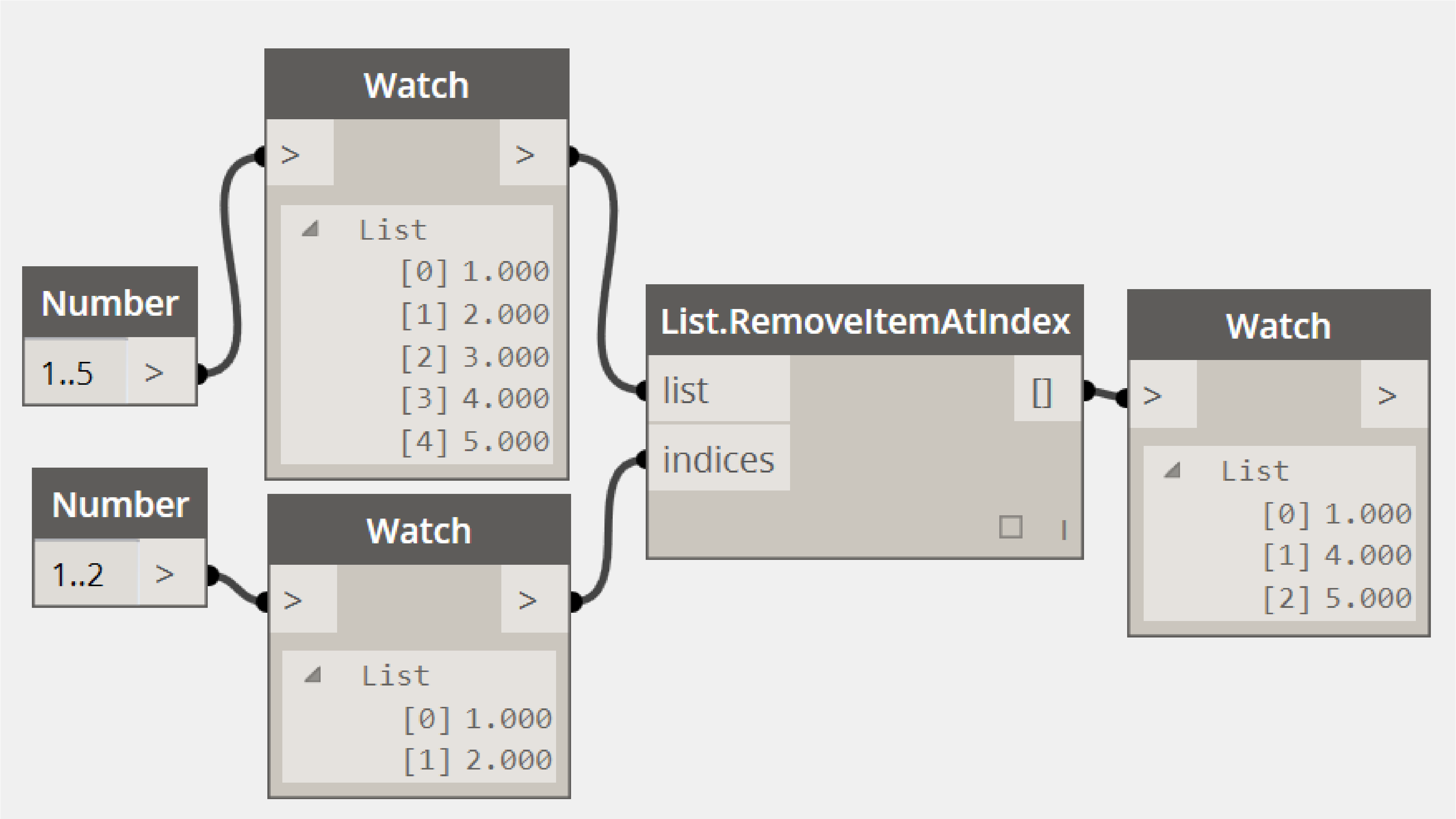


```
◢ List
  ◢ [0] List
      [0] 1.000
      [1] 2.000
      [2] 3.000
  ◢ [1] List
      [0] 4.000
      [1] 5.000
      [2] 6.000
  ◢ [2] List
      [0] 7.000
      [1] 8.000
      [2] 9.000
```

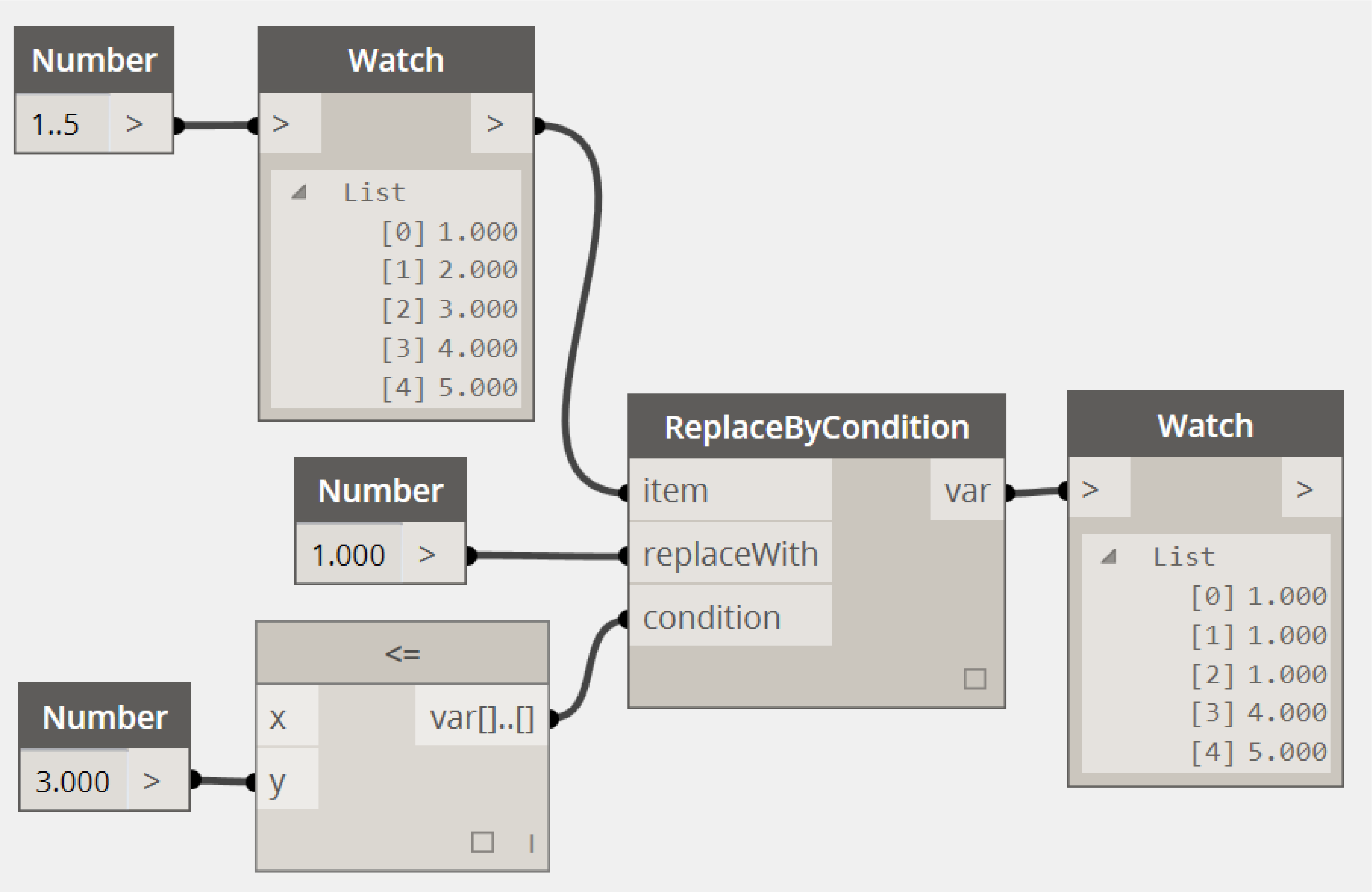
```
◢ List
  ◢ [0] List
      [0] 3.000
      [1] 2.000
      [2] 1.000
  ◢ [1] List
      [0] 6.000
      [1] 5.000
      [2] 4.000
  ◢ [2] List
      [0] 9.000
      [1] 8.000
      [2] 7.000
```

```
◢ List
  ◢ [0] List
      [0] 7.000
      [1] 8.000
      [2] 9.000
  ◢ [1] List
      [0] 4.000
      [1] 5.000
      [2] 6.000
  ◢ [2] List
      [0] 1.000
      [1] 2.000
      [2] 3.000
```

remove


Watch
>
>
List
[0] 1.000
[1] 2.000
[2] 3.000
[3] 4.000
[4] 5.000
Number
1..5
>
Number
1..2
>
Watch
>
>
List
[0] 1.000
[1] 2.000
List.RemoveItemAtIndex
list
indices
[]
Watch
>
>
List
[0] 1.000
[1] 4.000
[2] 5.000

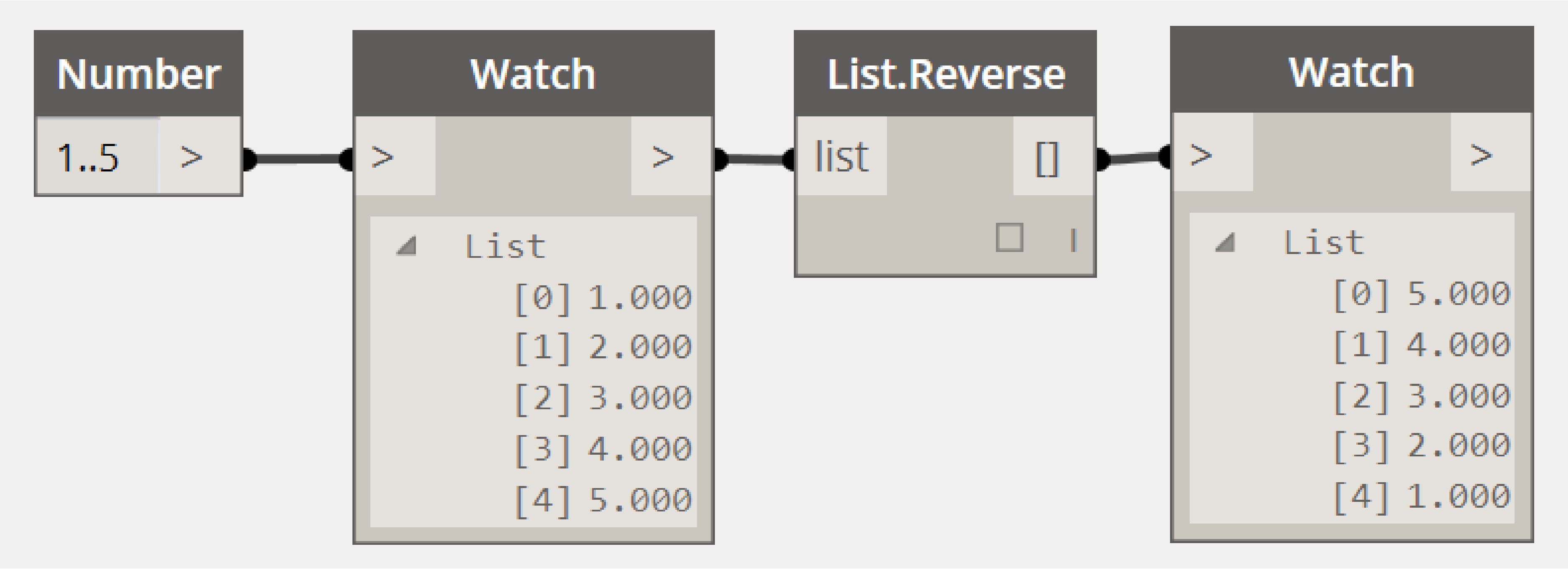
replace



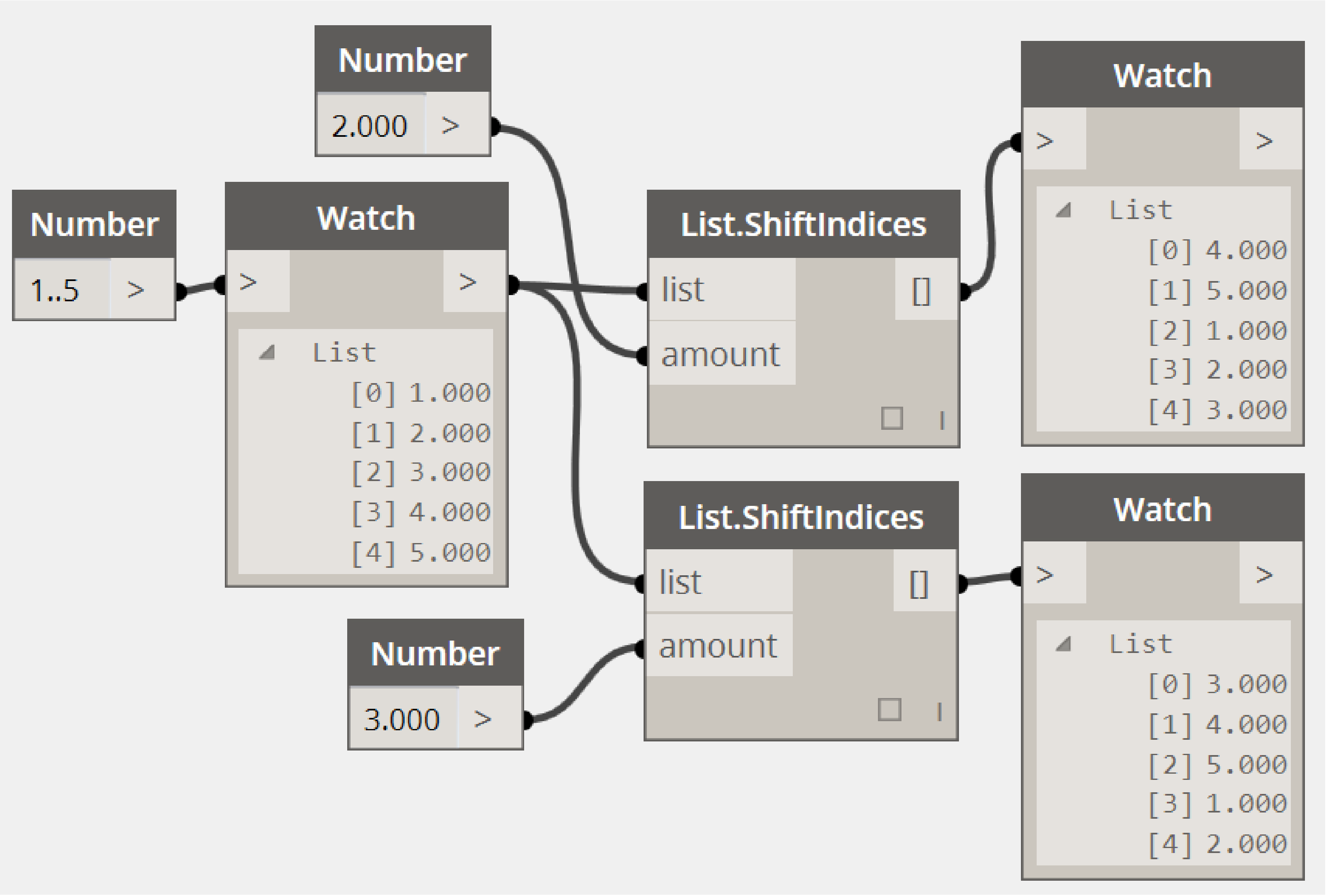
```
List
  [0] 1.000
  [1] 2.000
  [2] 3.000
  [3] 4.000
  [4] 5.000
```

```
List
  [0] 1.000
  [1] 1.000
  [2] 1.000
  [3] 4.000
  [4] 5.000
```

reverse

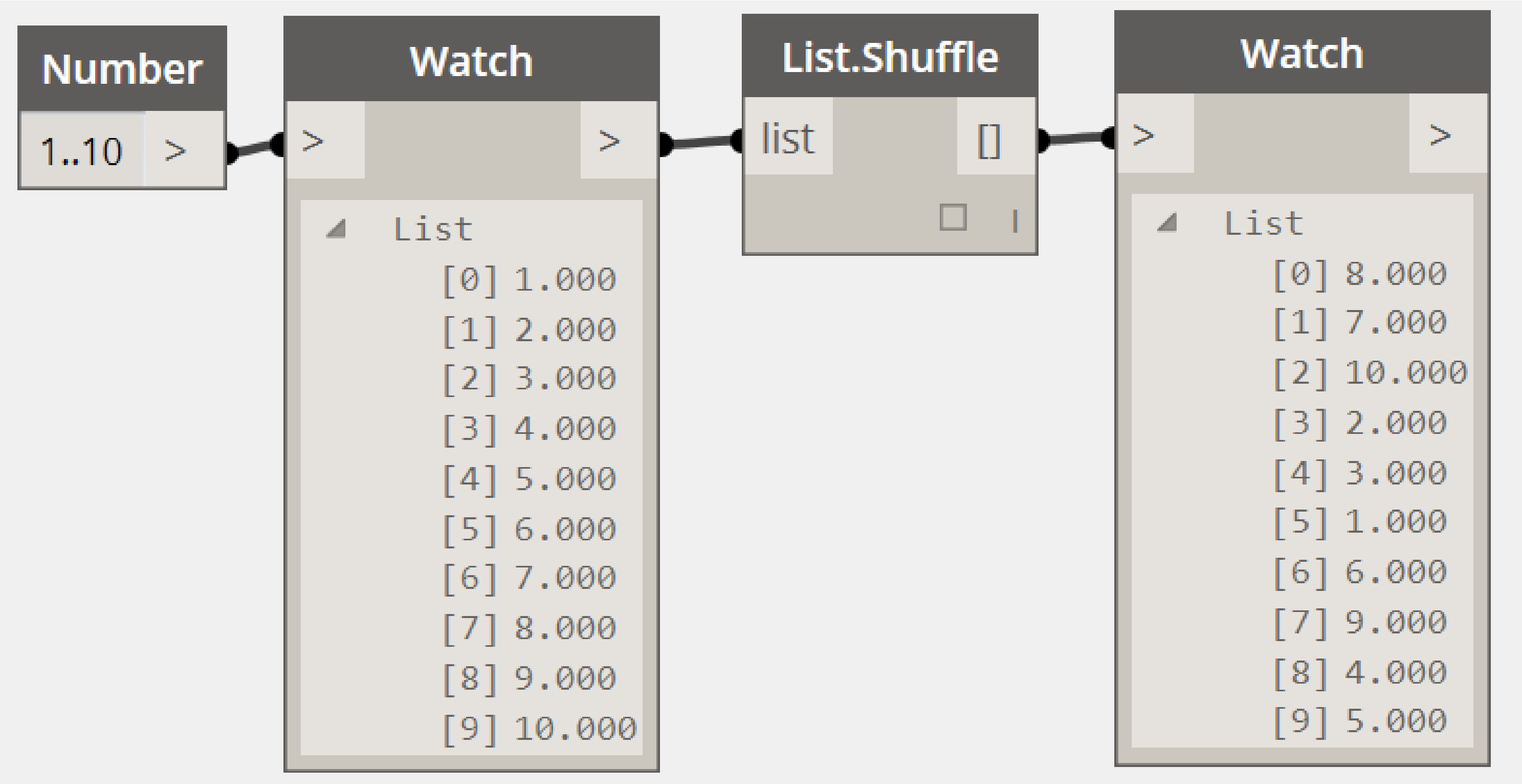

Number
1..5
>
Watch
>
>
List
[0] 1.000
[1] 2.000
[2] 3.000
[3] 4.000
[4] 5.000
List.Reverse
list
[]
Watch
>
>
List
[0] 5.000
[1] 4.000
[2] 3.000
[3] 2.000
[4] 1.000

shift



Number
2.000
>
Number
1..5
>
Watch
>
>
List
[0] 1.000
[1] 2.000
[2] 3.000
[3] 4.000
[4] 5.000
List.ShiftIndices
list
amount
[]
Watch
>
>
List
[0] 4.000
[1] 5.000
[2] 1.000
[3] 2.000
[4] 3.000
Number
3.000
>
List.ShiftIndices
list
amount
[]
Watch
>
>
List
[0] 3.000
[1] 4.000
[2] 5.000
[3] 1.000
[4] 2.000

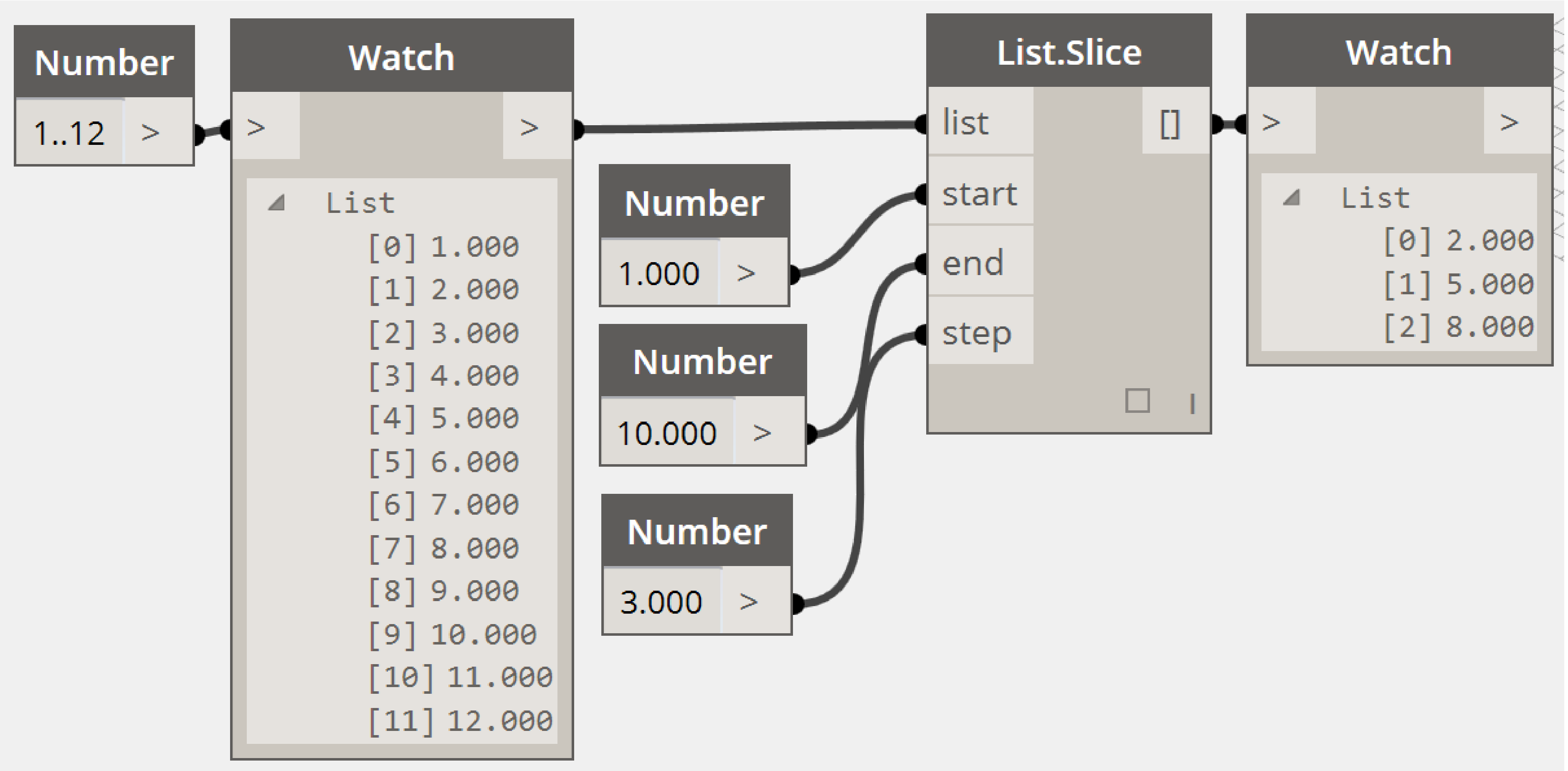
shuffle



```
List
  [0] 1.000
  [1] 2.000
  [2] 3.000
  [3] 4.000
  [4] 5.000
  [5] 6.000
  [6] 7.000
  [7] 8.000
  [8] 9.000
  [9] 10.000
```

```
List
  [0] 8.000
  [1] 7.000
  [2] 10.000
  [3] 2.000
  [4] 3.000
  [5] 1.000
  [6] 6.000
  [7] 9.000
  [8] 4.000
  [9] 5.000
```

slice



```
1..12
```

```
List
  [0] 1.000
  [1] 2.000
  [2] 3.000
  [3] 4.000
  [4] 5.000
  [5] 6.000
  [6] 7.000
  [7] 8.000
  [8] 9.000
  [9] 10.000
  [10] 11.000
  [11] 12.000
```

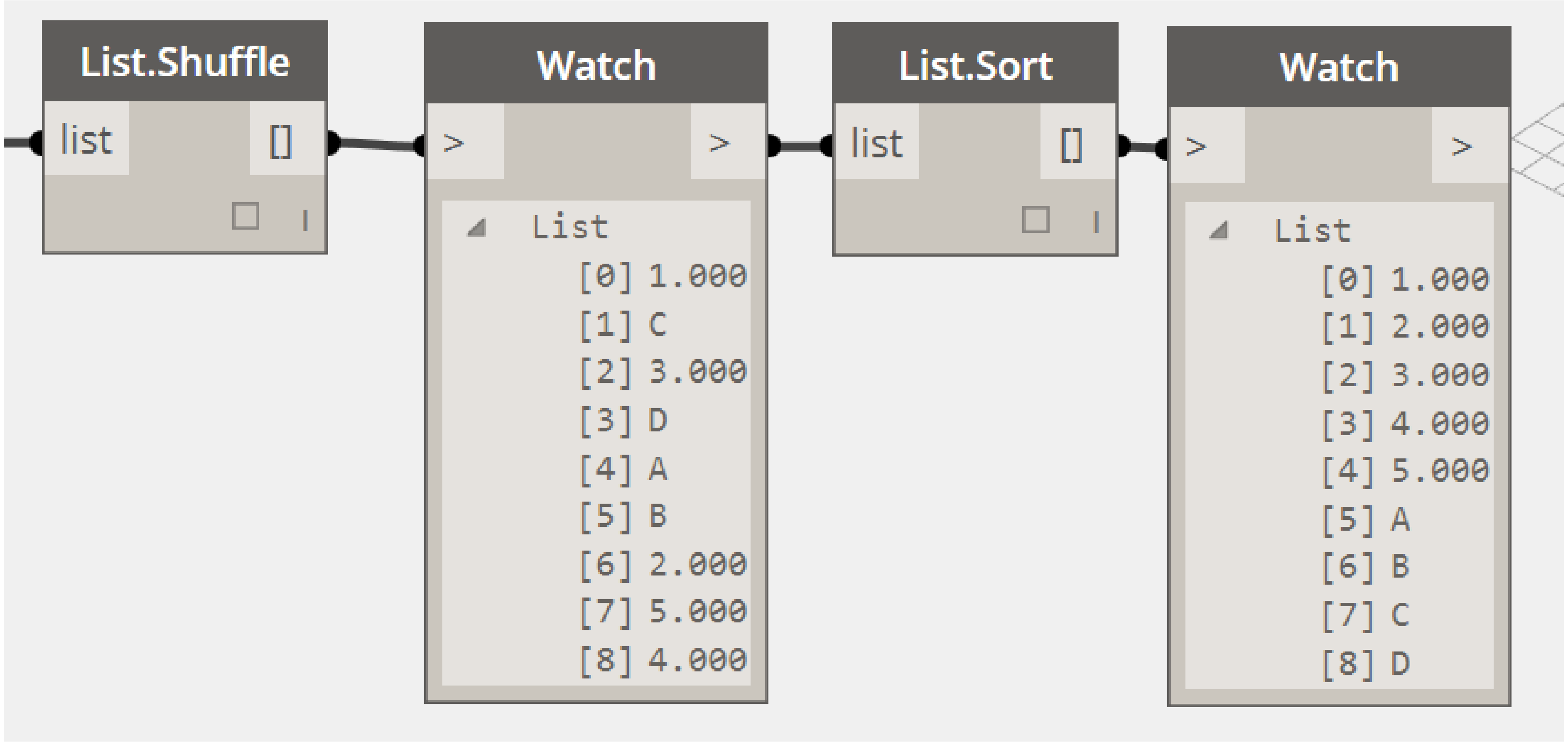
1.000

10.000

3.000

```
List
  [0] 2.000
  [1] 5.000
  [2] 8.000
```

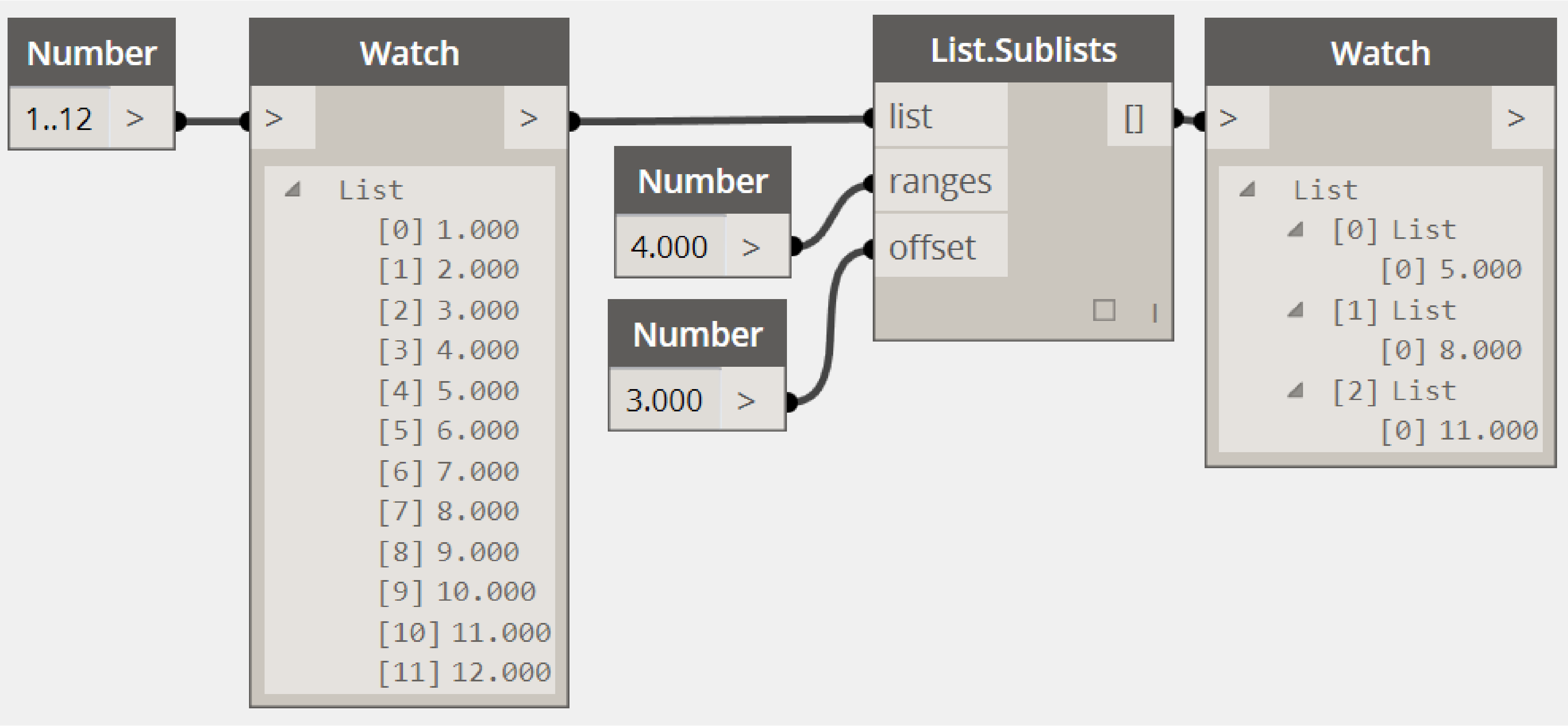
sort



```
List
  [0] 1.000
  [1] C
  [2] 3.000
  [3] D
  [4] A
  [5] B
  [6] 2.000
  [7] 5.000
  [8] 4.000
```

```
List
  [0] 1.000
  [1] 2.000
  [2] 3.000
  [3] 4.000
  [4] 5.000
  [5] A
  [6] B
  [7] C
  [8] D
```

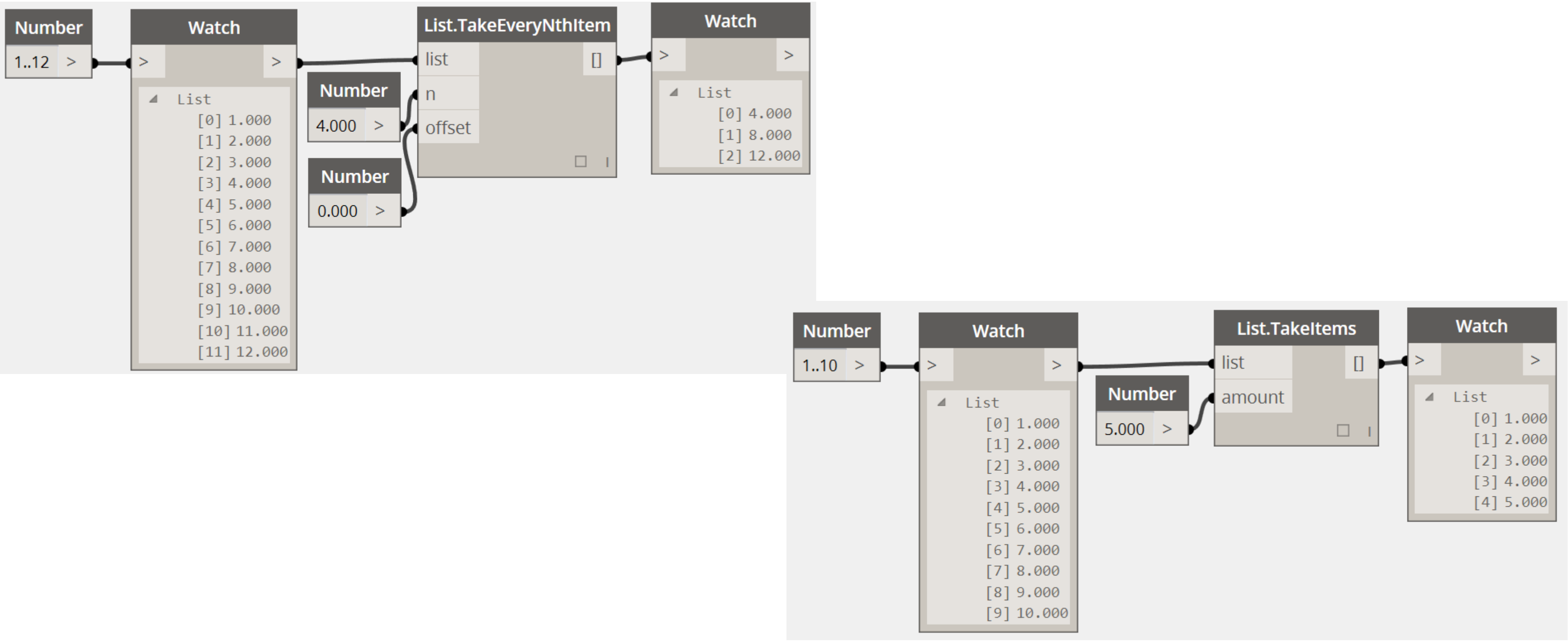
sublists



```
List
    [0] 1.000
    [1] 2.000
    [2] 3.000
    [3] 4.000
    [4] 5.000
    [5] 6.000
    [6] 7.000
    [7] 8.000
    [8] 9.000
    [9] 10.000
    [10] 11.000
    [11] 12.000
```

```
List
  [0] List
     [0] 5.000
  [1] List
     [0] 8.000
  [2] List
     [0] 11.000
```

take



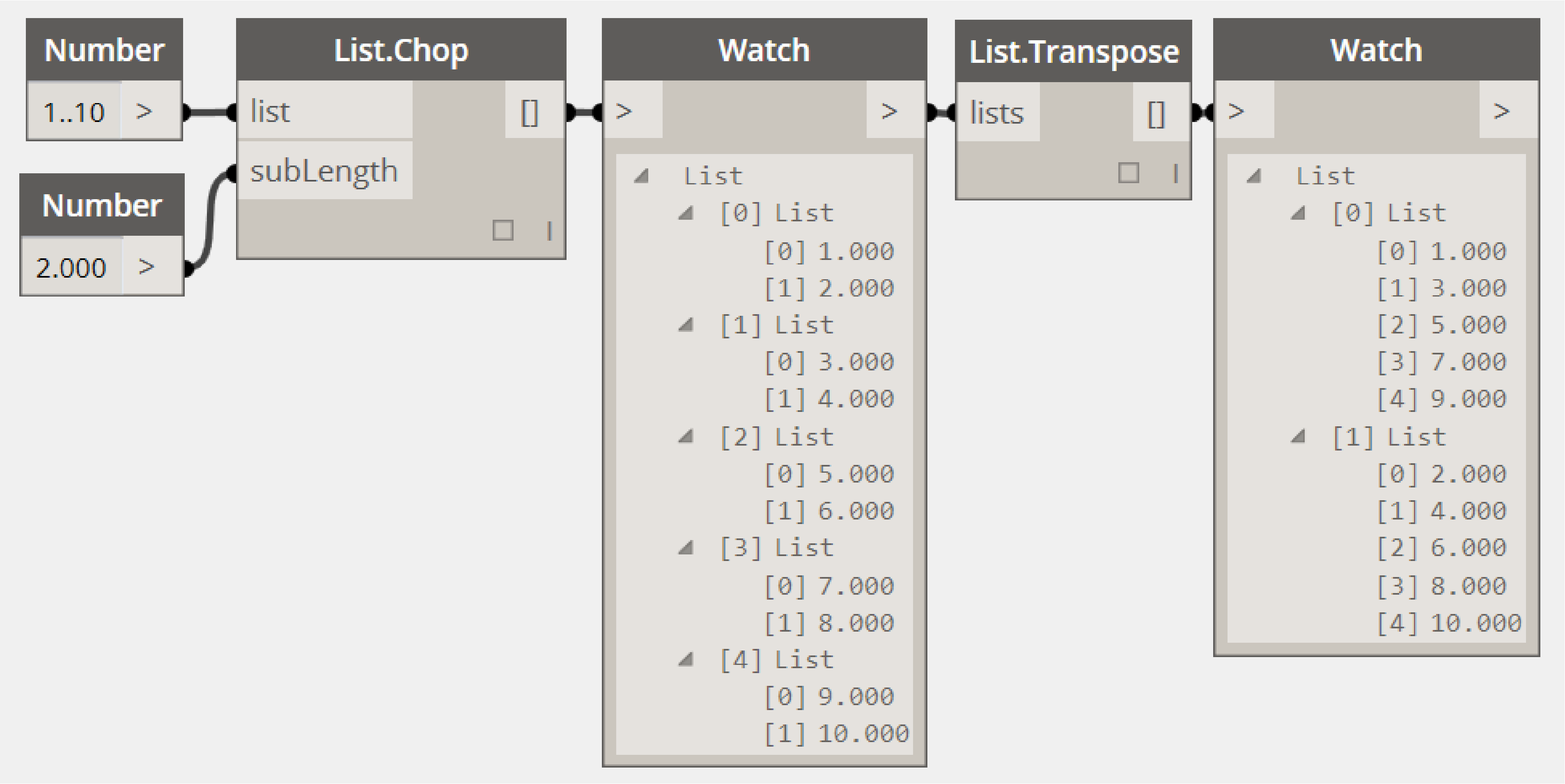
```
List
  [0] 1.000
  [1] 2.000
  [2] 3.000
  [3] 4.000
  [4] 5.000
  [5] 6.000
  [6] 7.000
  [7] 8.000
  [8] 9.000
  [9] 10.000
  [10] 11.000
  [11] 12.000
```

```
List
  [0] 4.000
  [1] 8.000
  [2] 12.000
```

```
List
  [0] 1.000
  [1] 2.000
  [2] 3.000
  [3] 4.000
  [4] 5.000
  [5] 6.000
  [6] 7.000
  [7] 8.000
  [8] 9.000
  [9] 10.000
```

```
List
  [0] 1.000
  [1] 2.000
  [2] 3.000
  [3] 4.000
  [4] 5.000
```

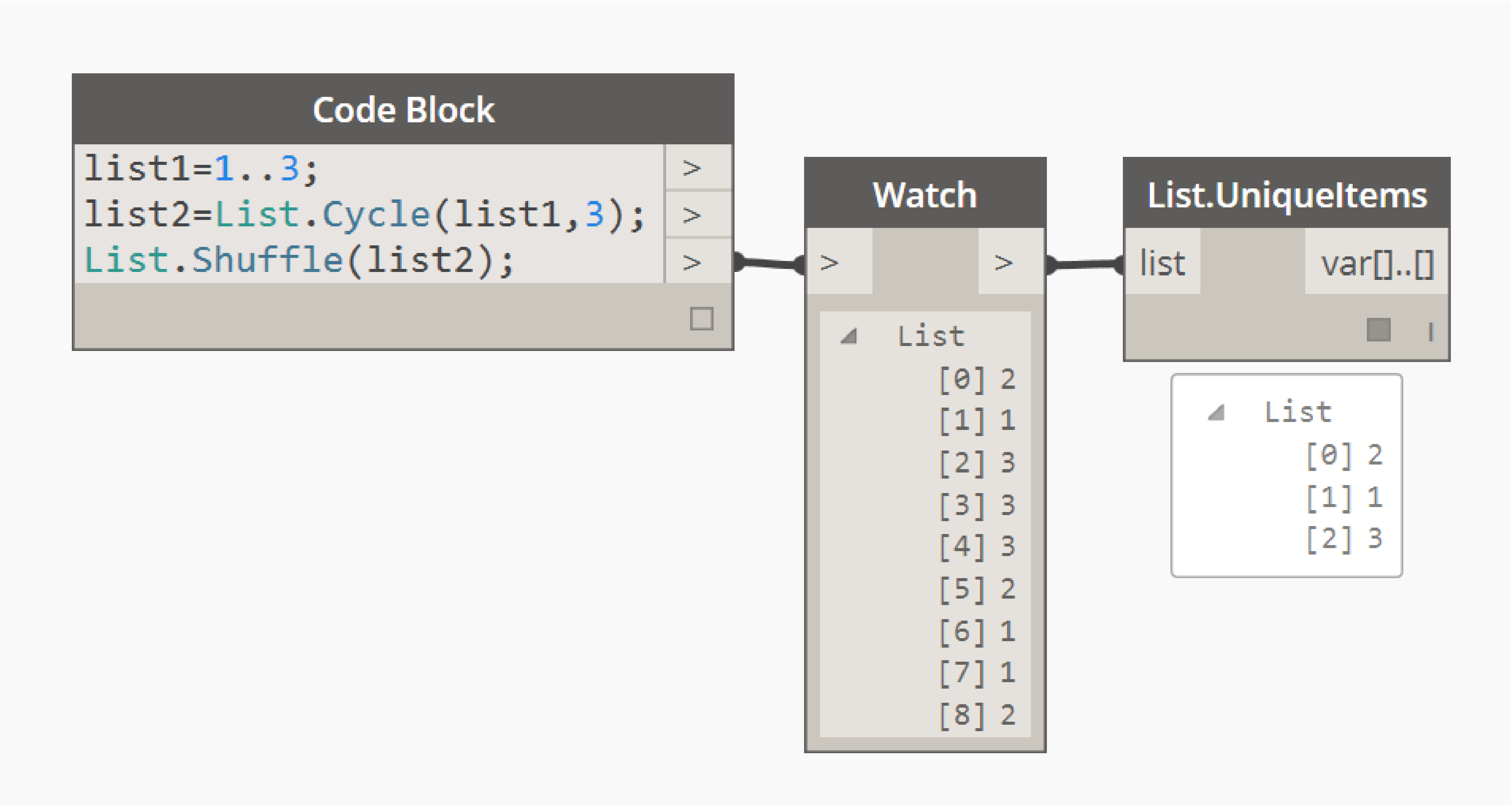
transpose



```
List
  [0] List
      [0] 1.000
      [1] 2.000
  [1] List
      [0] 3.000
      [1] 4.000
  [2] List
      [0] 5.000
      [1] 6.000
  [3] List
      [0] 7.000
      [1] 8.000
  [4] List
      [0] 9.000
      [1] 10.000
```

```
List
  [0] List
      [0] 1.000
      [1] 3.000
      [2] 5.000
      [3] 7.000
      [4] 9.000
  [1] List
      [0] 2.000
      [1] 4.000
      [2] 6.000
      [3] 8.000
      [4] 10.000
```

unique item

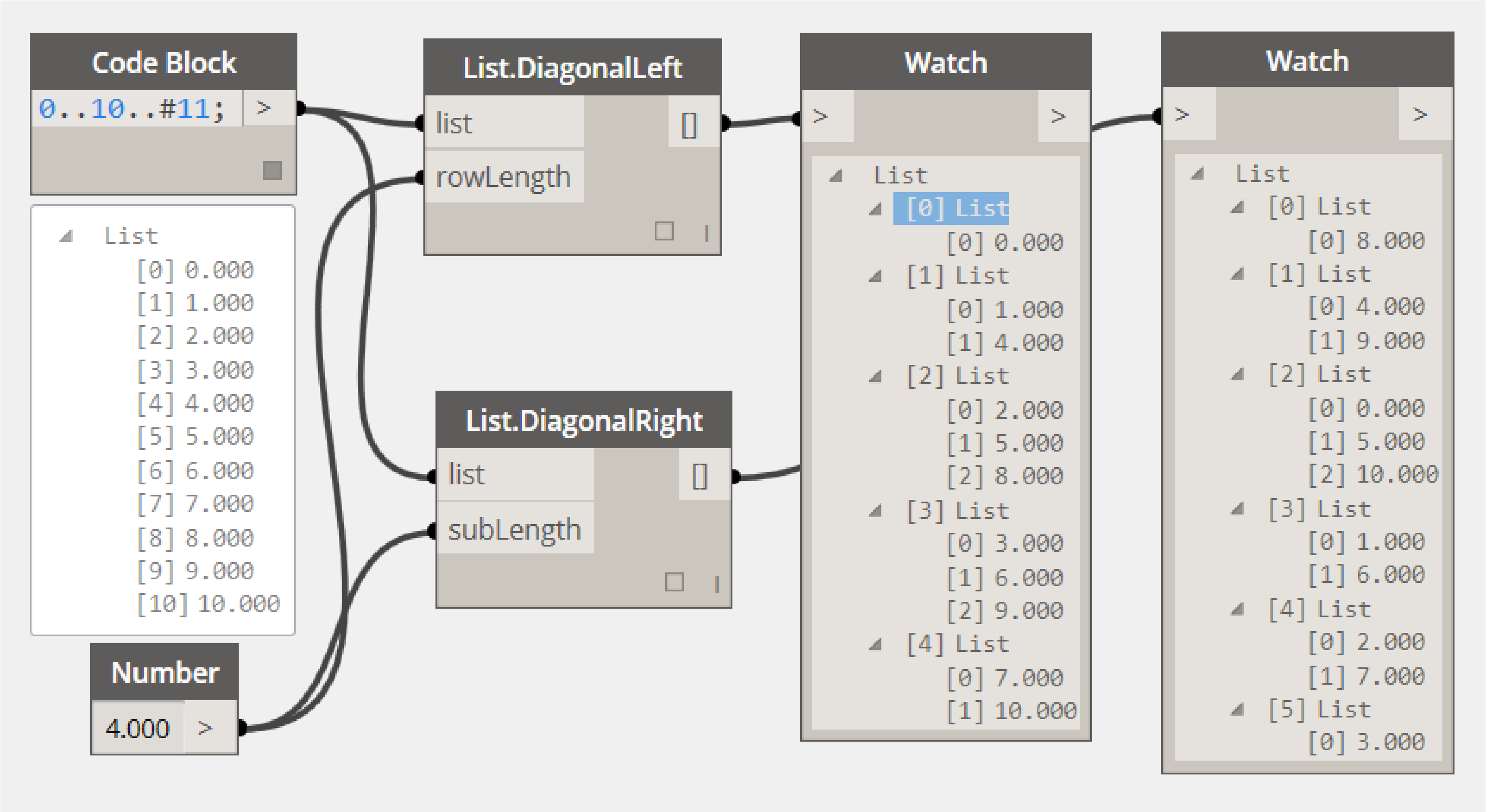


```
list1=1..3;
list2=List.Cycle(list1,3);
List.Shuffle(list2);
```

```
List
  [0] 2
  [1] 1
  [2] 3
  [3] 3
  [4] 3
  [5] 2
  [6] 1
  [7] 1
  [8] 2
```

```
List
  [0] 2
  [1] 1
  [2] 3
```

diagonal left&right



```
0..10..#11;
```

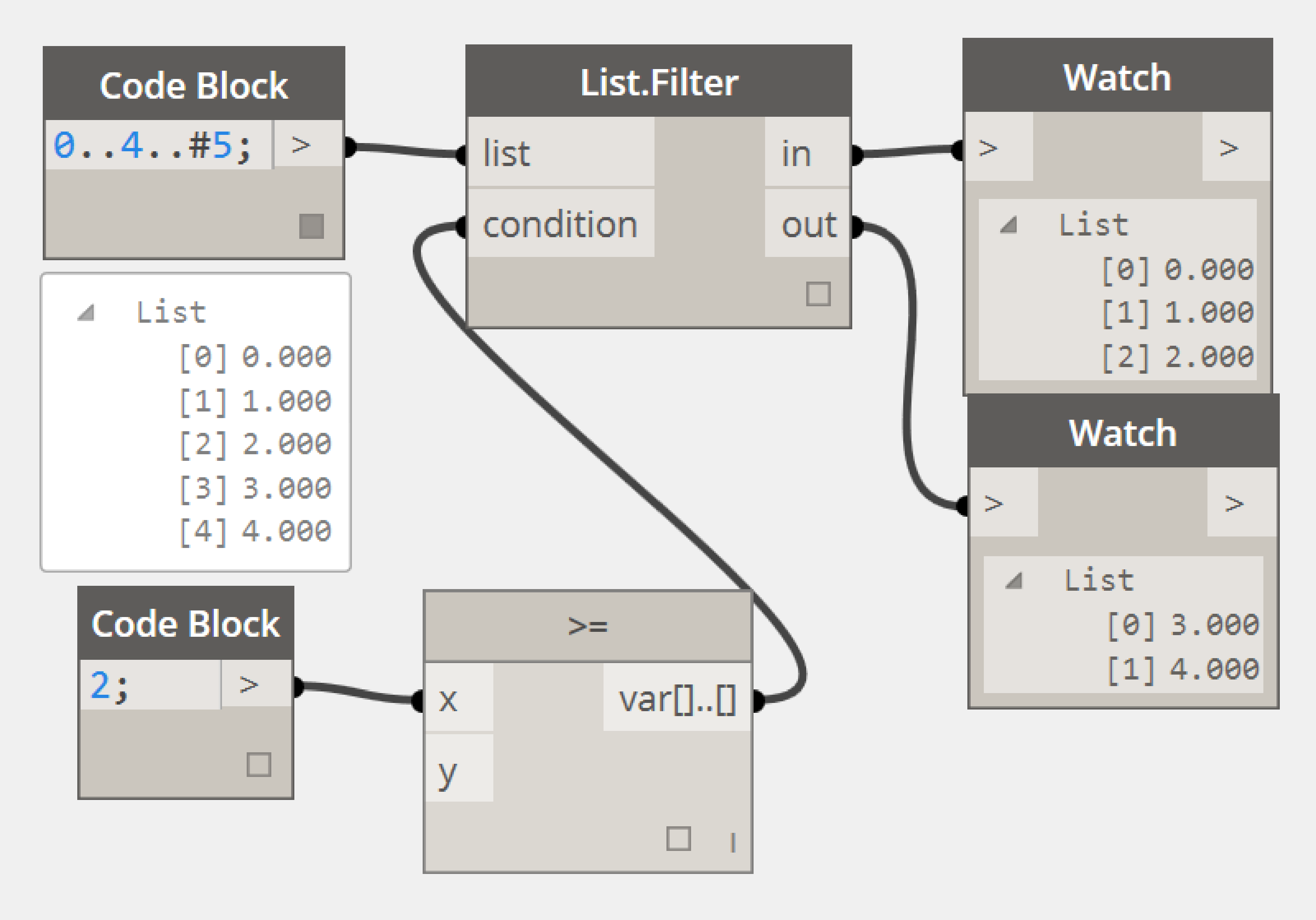
```
List
  [0] 0.000
  [1] 1.000
  [2] 2.000
  [3] 3.000
  [4] 4.000
  [5] 5.000
  [6] 6.000
  [7] 7.000
  [8] 8.000
  [9] 9.000
  [10] 10.000
```

```
4.000
```

```
List
  [0] List
    [0] 0.000
  [1] List
    [0] 1.000
    [1] 4.000
  [2] List
    [0] 2.000
    [1] 5.000
    [2] 8.000
  [3] List
    [0] 3.000
    [1] 6.000
    [2] 9.000
  [4] List
    [0] 7.000
    [1] 10.000
```

```
List
  [0] List
    [0] 8.000
  [1] List
    [0] 4.000
    [1] 9.000
  [2] List
    [0] 0.000
    [1] 5.000
    [2] 10.000
  [3] List
    [0] 1.000
    [1] 6.000
  [4] List
    [0] 2.000
    [1] 7.000
  [5] List
    [0] 3.000
```

filter



```
0..4..#5;
```

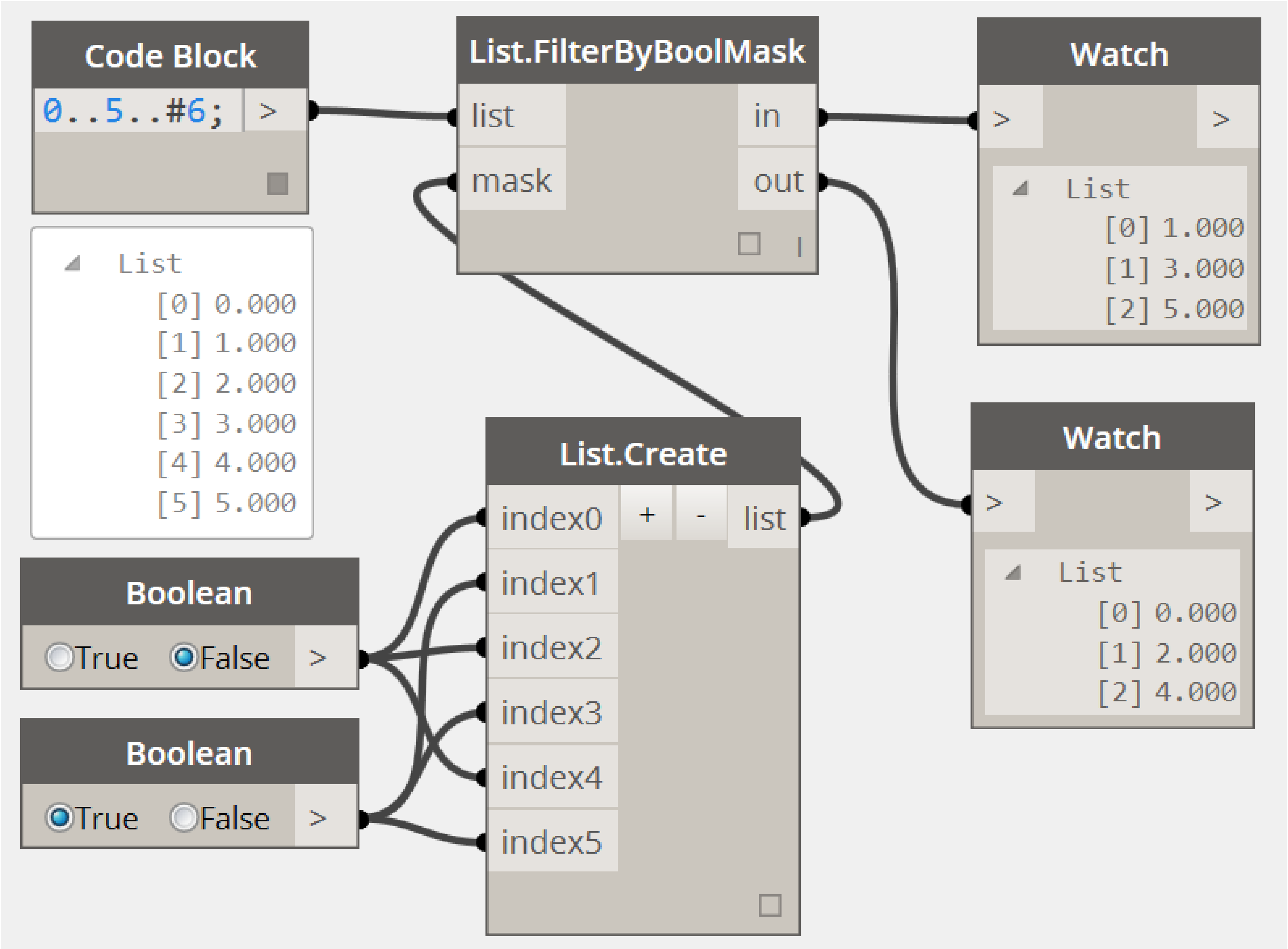
```
List
    [0] 0.000
    [1] 1.000
    [2] 2.000
    [3] 3.000
    [4] 4.000
```

```
2;
```

```
List
    [0] 0.000
    [1] 1.000
    [2] 2.000
```

```
List
    [0] 3.000
    [1] 4.000
```

filter by bool mask



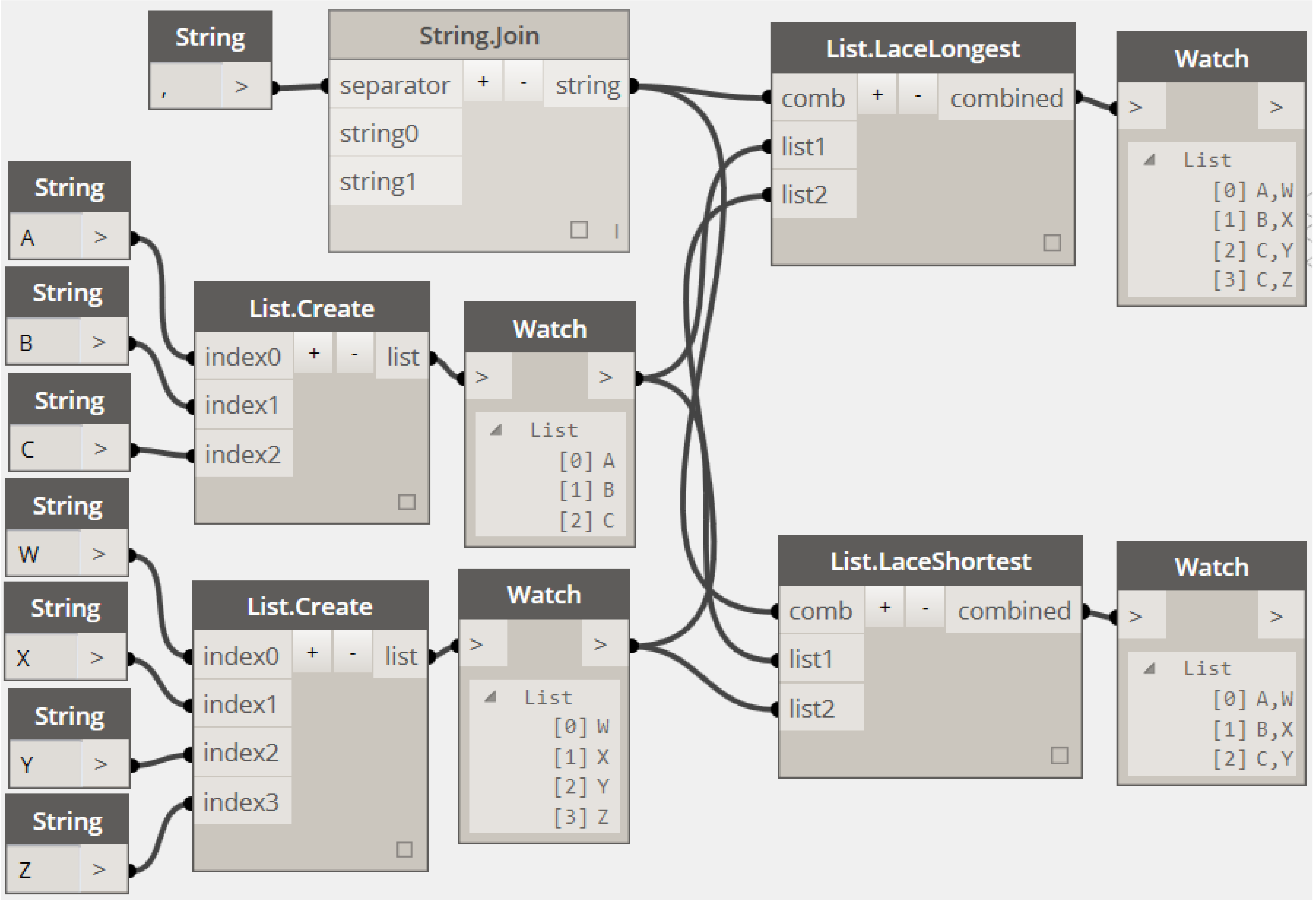
```
0..5..#6;
```

```
List
  [0] 0.000
  [1] 1.000
  [2] 2.000
  [3] 3.000
  [4] 4.000
  [5] 5.000
```

```
List
  [0] 1.000
  [1] 3.000
  [2] 5.000
```

```
List
  [0] 0.000
  [1] 2.000
  [2] 4.000
```

lace



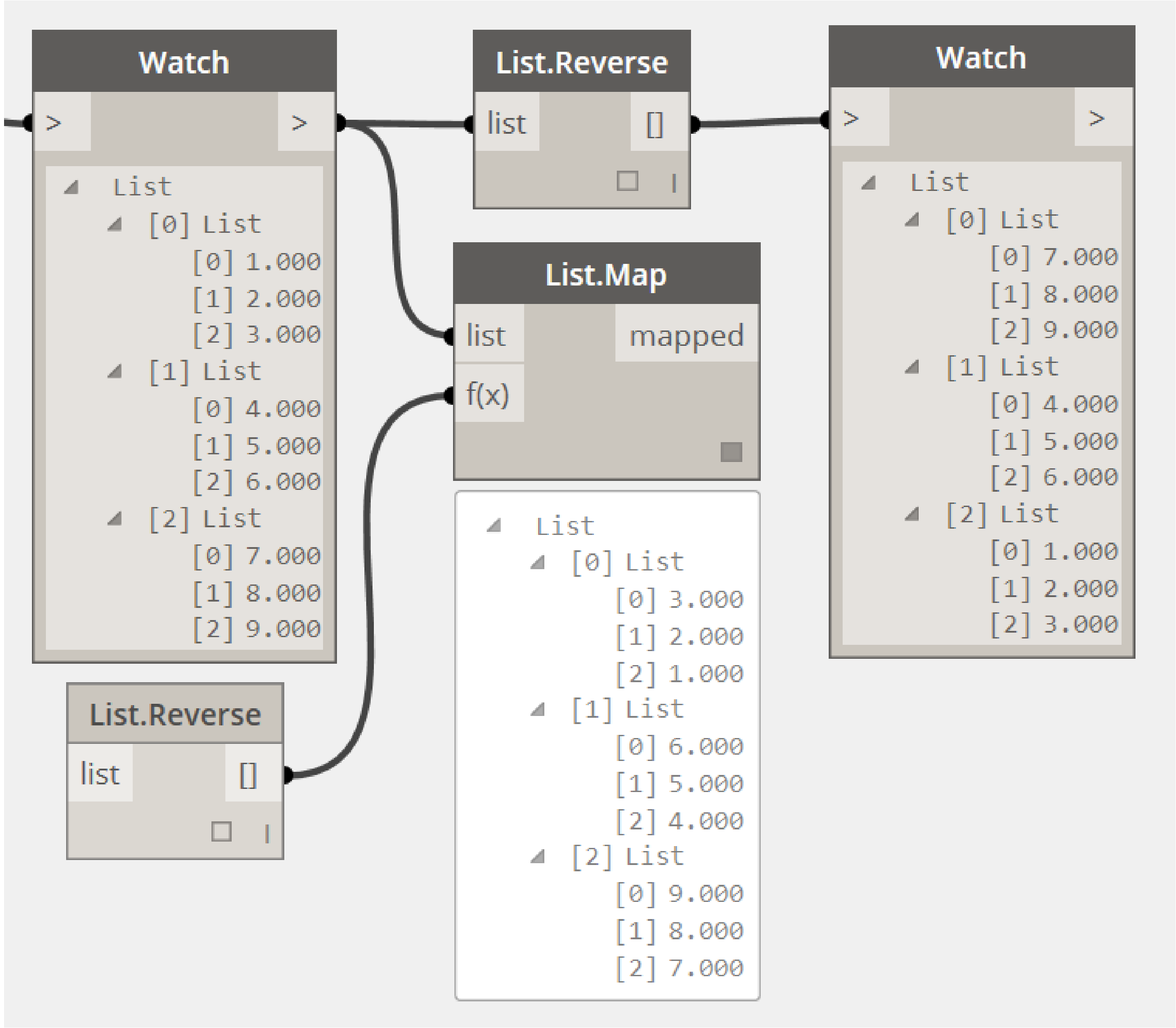
```
List
  [0] A
  [1] B
  [2] C
```

```
List
  [0] W
  [1] X
  [2] Y
  [3] Z
```

```
List
  [0] A,W
  [1] B,X
  [2] C,Y
  [3] C,Z
```

```
List
  [0] A,W
  [1] B,X
  [2] C,Y
```

map

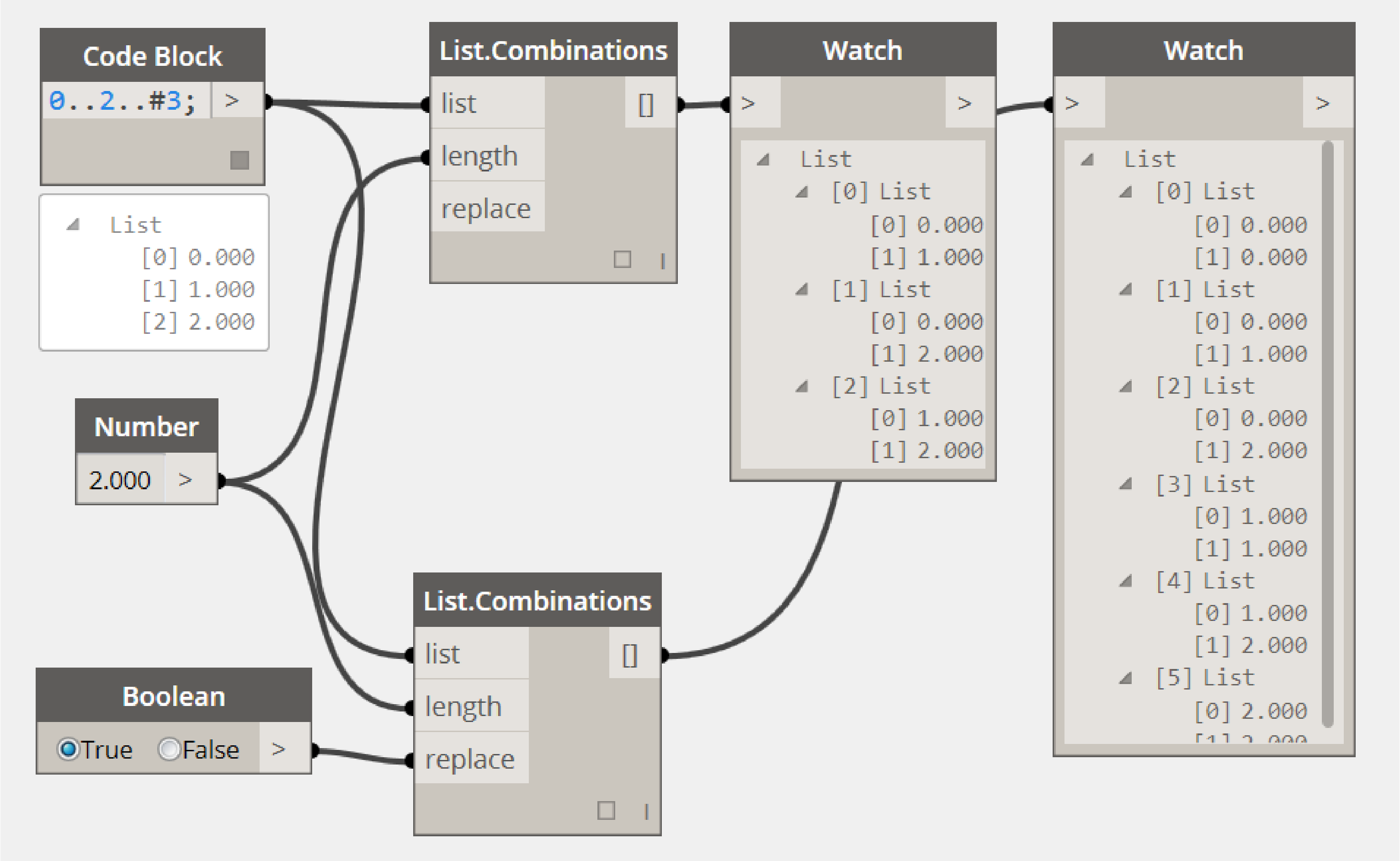


```
List
  [0] List
    [0] 1.000
    [1] 2.000
    [2] 3.000
  [1] List
    [0] 4.000
    [1] 5.000
    [2] 6.000
  [2] List
    [0] 7.000
    [1] 8.000
    [2] 9.000
```

```
List
  [0] List
    [0] 3.000
    [1] 2.000
    [2] 1.000
  [1] List
    [0] 6.000
    [1] 5.000
    [2] 4.000
  [2] List
    [0] 9.000
    [1] 8.000
    [2] 7.000
```

```
List
  [0] List
    [0] 7.000
    [1] 8.000
    [2] 9.000
  [1] List
    [0] 4.000
    [1] 5.000
    [2] 6.000
  [2] List
    [0] 1.000
    [1] 2.000
    [2] 3.000
```

combinations

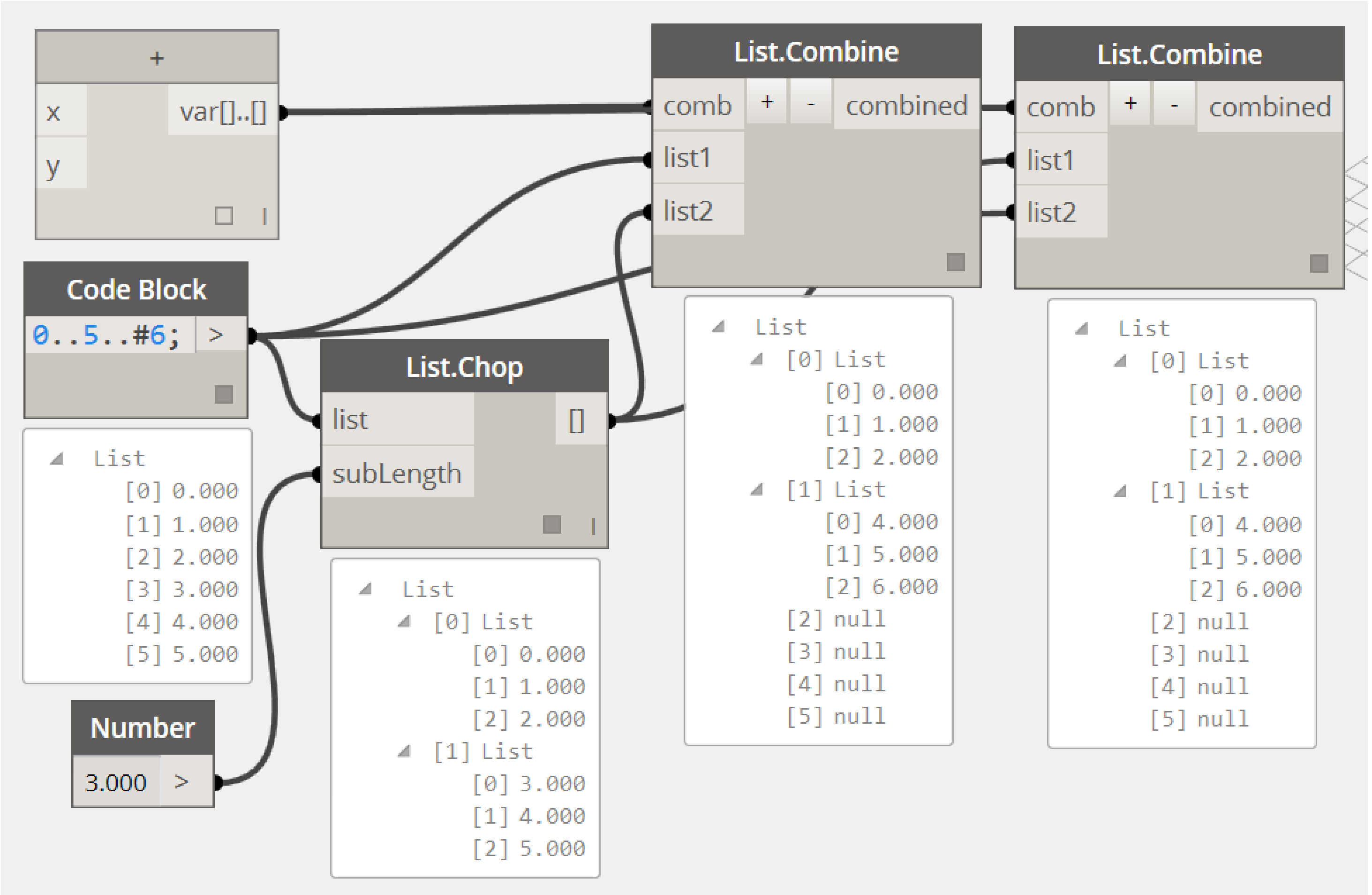


```
List
  [0] 0.000
  [1] 1.000
  [2] 2.000
```

```
List
  [0] List
    [0] 0.000
    [1] 1.000
  [1] List
    [0] 0.000
    [1] 2.000
  [2] List
    [0] 1.000
    [1] 2.000
```

```
List
  [0] List
    [0] 0.000
    [1] 0.000
  [1] List
    [0] 0.000
    [1] 1.000
  [2] List
    [0] 0.000
    [1] 2.000
  [3] List
    [0] 1.000
    [1] 1.000
  [4] List
    [0] 1.000
    [1] 2.000
  [5] List
    [0] 2.000
    [1] 2.000
```

combine



```
0..5..#6;
```

```
List
    [0] 0.000
    [1] 1.000
    [2] 2.000
    [3] 3.000
    [4] 4.000
    [5] 5.000
```

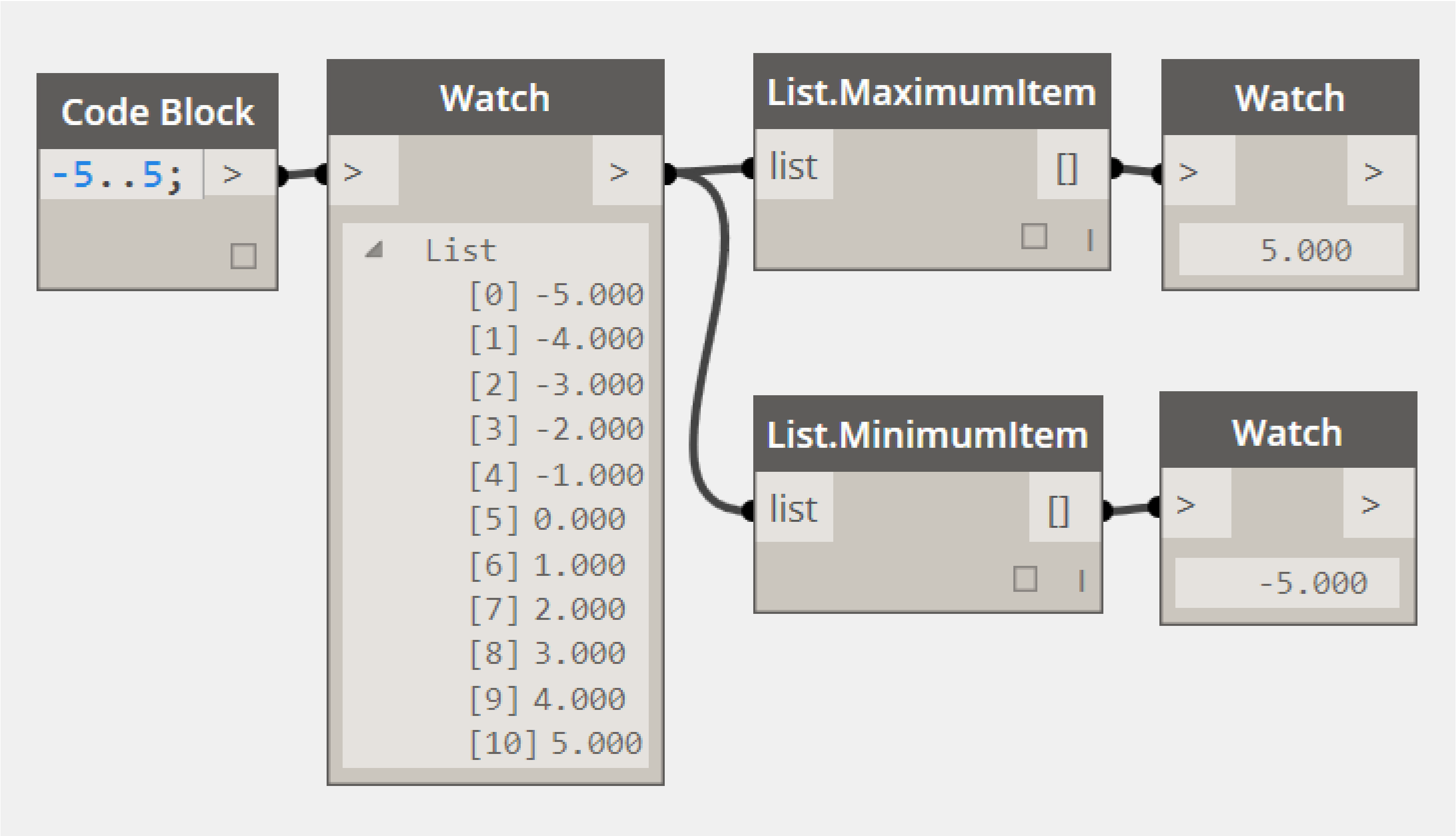
3.000

```
List
  [0] List
      [0] 0.000
      [1] 1.000
      [2] 2.000
  [1] List
      [0] 3.000
      [1] 4.000
      [2] 5.000
```

```
List
  [0] List
      [0] 0.000
      [1] 1.000
      [2] 2.000
  [1] List
      [0] 4.000
      [1] 5.000
      [2] 6.000
  [2] null
  [3] null
  [4] null
  [5] null
```

```
List
  [0] List
      [0] 0.000
      [1] 1.000
      [2] 2.000
  [1] List
      [0] 4.000
      [1] 5.000
      [2] 6.000
  [2] null
  [3] null
  [4] null
  [5] null
```

maximum/minimum item



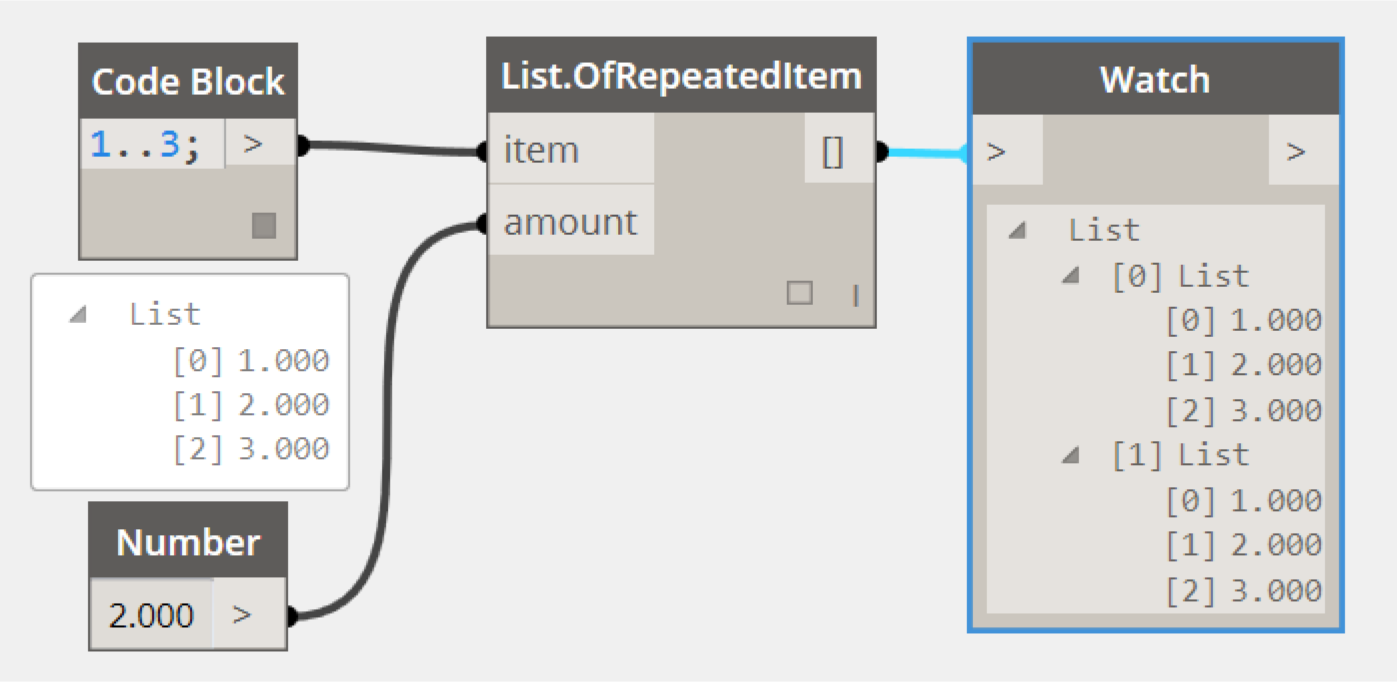
```
-5..5;
```

```
List
  [0] -5.000
  [1] -4.000
  [2] -3.000
  [3] -2.000
  [4] -1.000
  [5] 0.000
  [6] 1.000
  [7] 2.000
  [8] 3.000
  [9] 4.000
  [10] 5.000
```

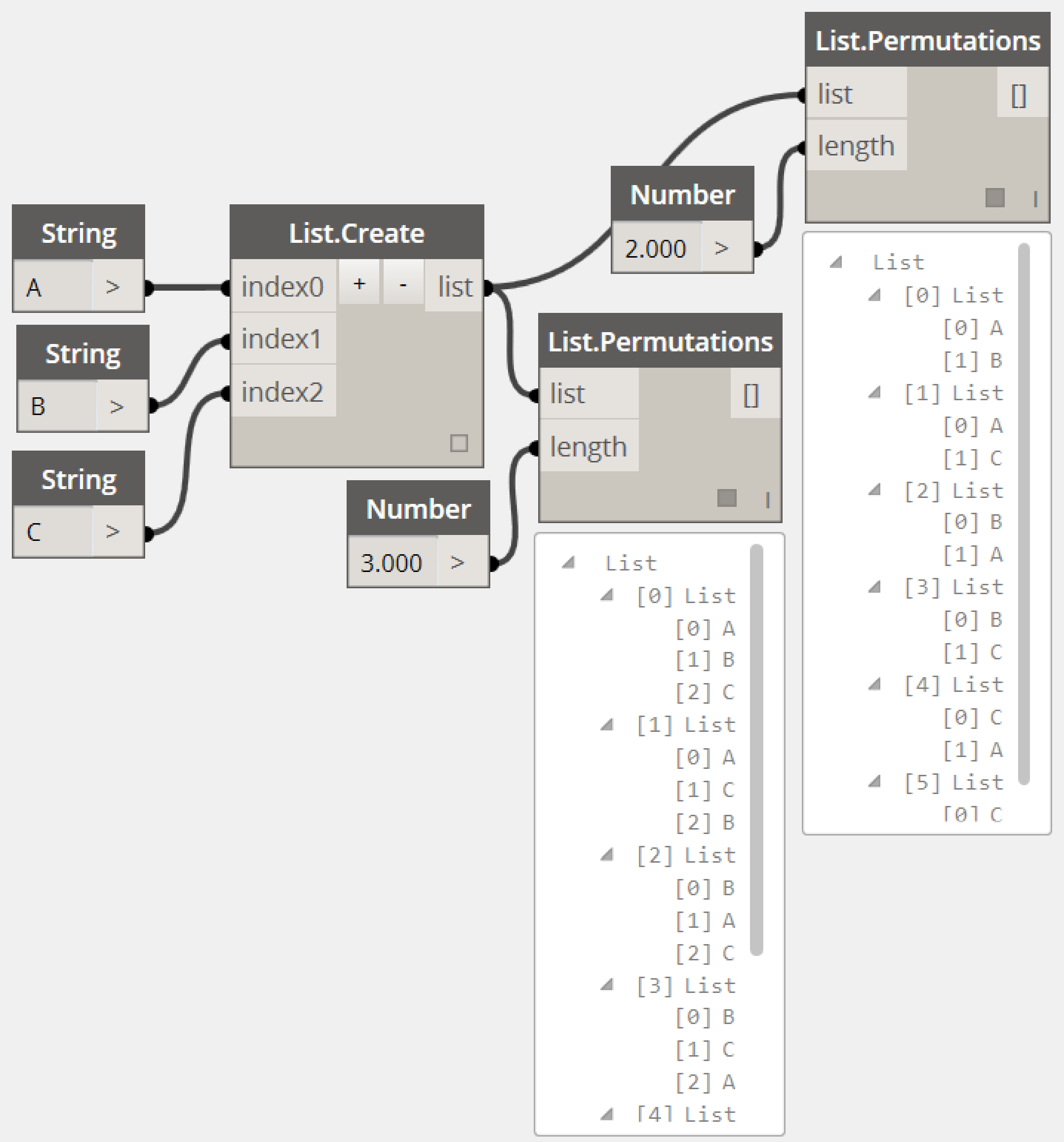
```
5.000
```

```
-5.000
```

repeated items



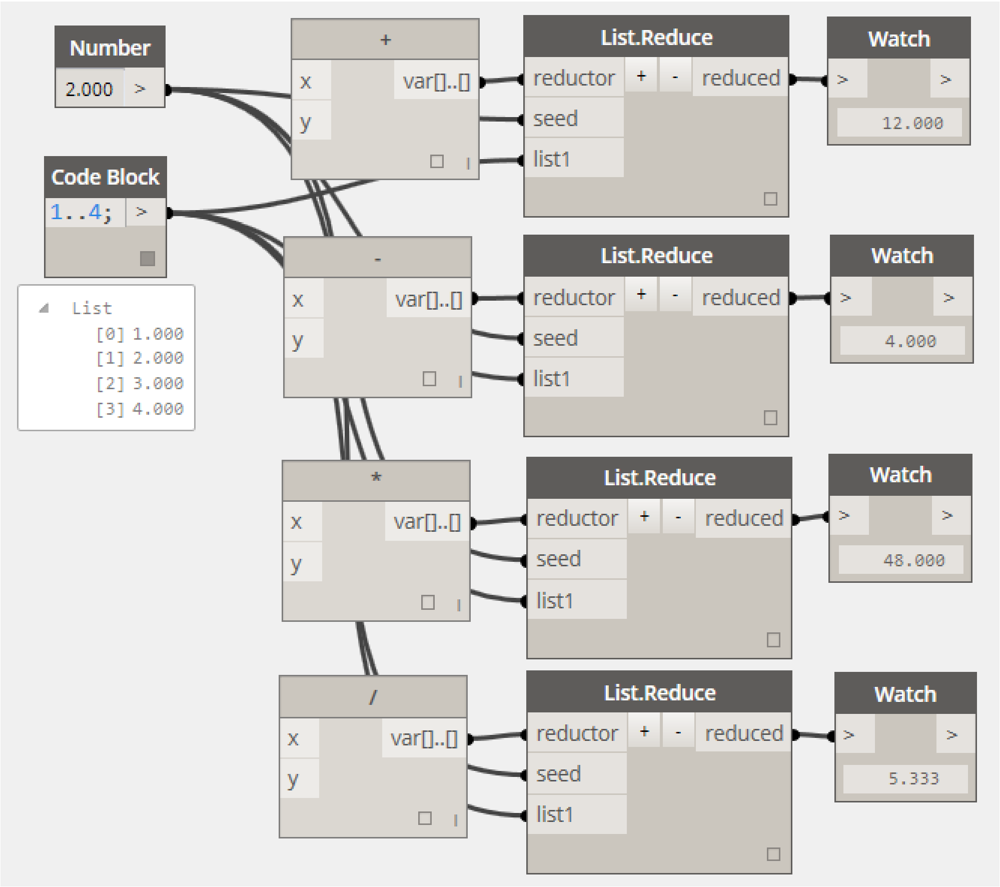
permutations



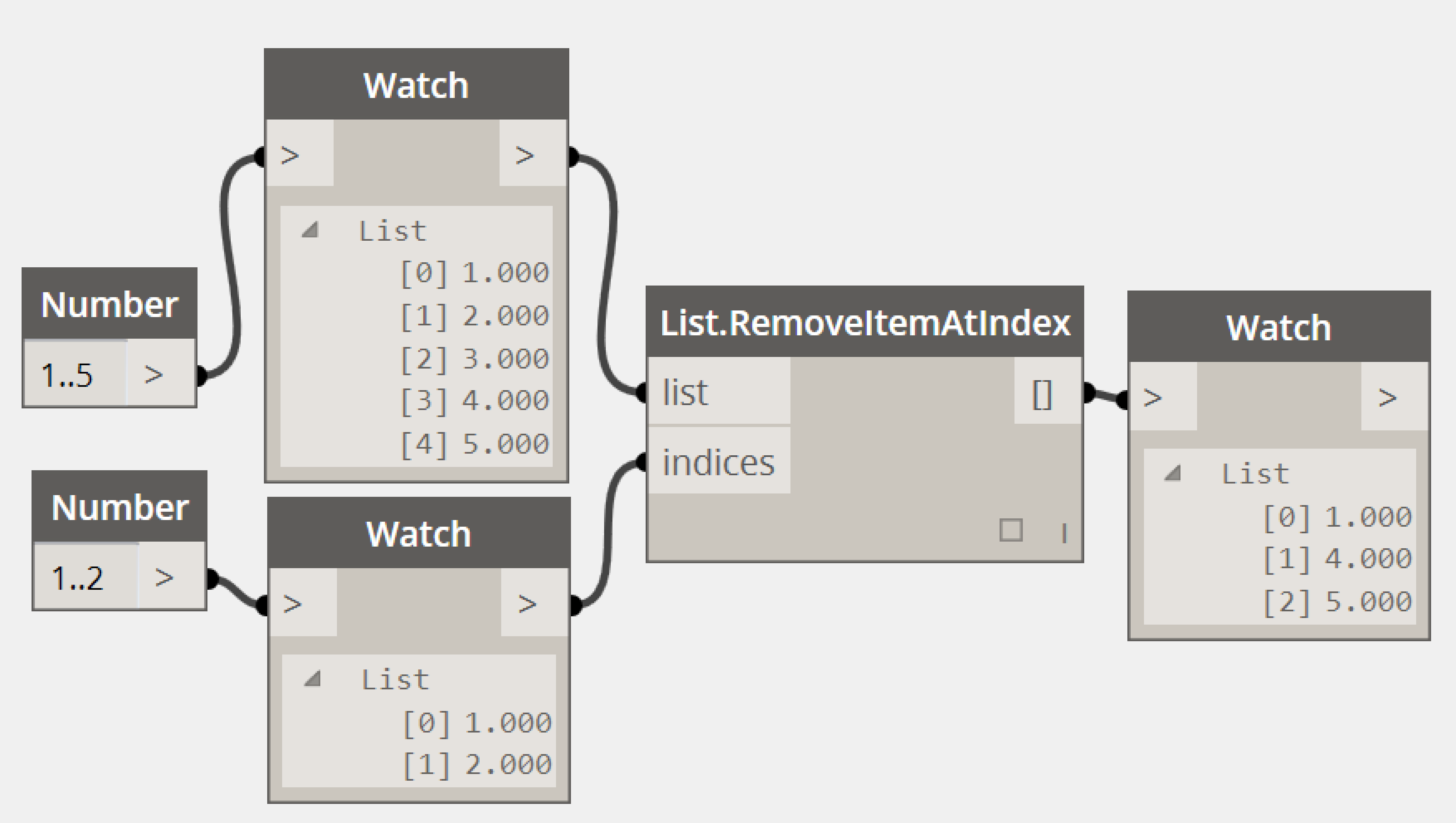
```
List
  [0] List
    [0] A
    [1] B
  [1] List
    [0] A
    [1] C
  [2] List
    [0] B
    [1] A
  [3] List
    [0] B
    [1] C
  [4] List
    [0] C
    [1] A
  [5] List
    [0] C
```

```
List
  [0] List
    [0] A
    [1] B
    [2] C
  [1] List
    [0] A
    [1] C
    [2] B
  [2] List
    [0] B
    [1] A
    [2] C
  [3] List
    [0] B
    [1] C
    [2] A
  [4] List
```

reduce


Number
2.000
>
Code Block
1..4;
>
List
[0] 1.000
[1] 2.000
[2] 3.000
[3] 4.000
+
x
y
var[]..[]
-
*
/
List.Reduce
reductor
+
-
reduced
seed
list1
Watch
>
12.000
4.000
48.000
5.333

remove item @ index

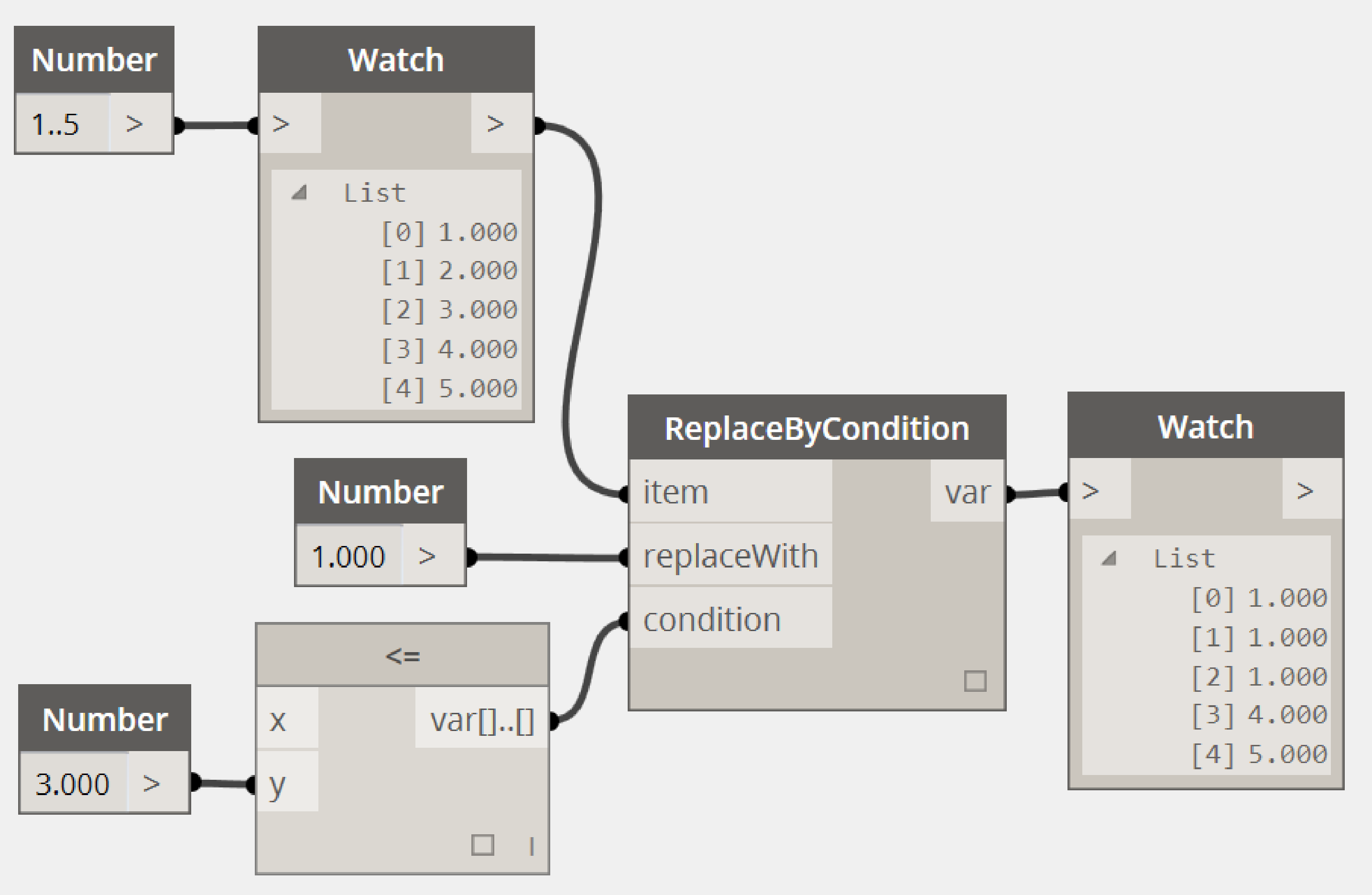


```
List
  [0] 1.000
  [1] 2.000
  [2] 3.000
  [3] 4.000
  [4] 5.000
```

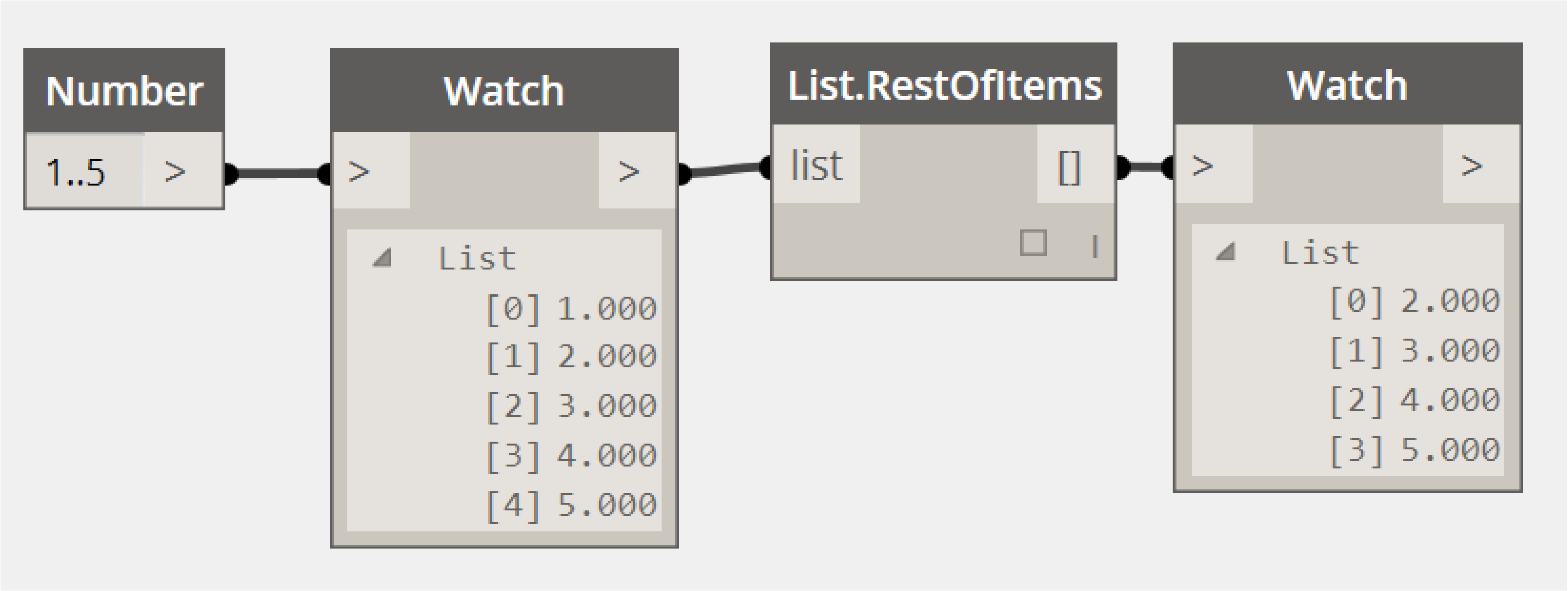
```
List
  [0] 1.000
  [1] 2.000
```

```
List
  [0] 1.000
  [1] 4.000
  [2] 5.000
```

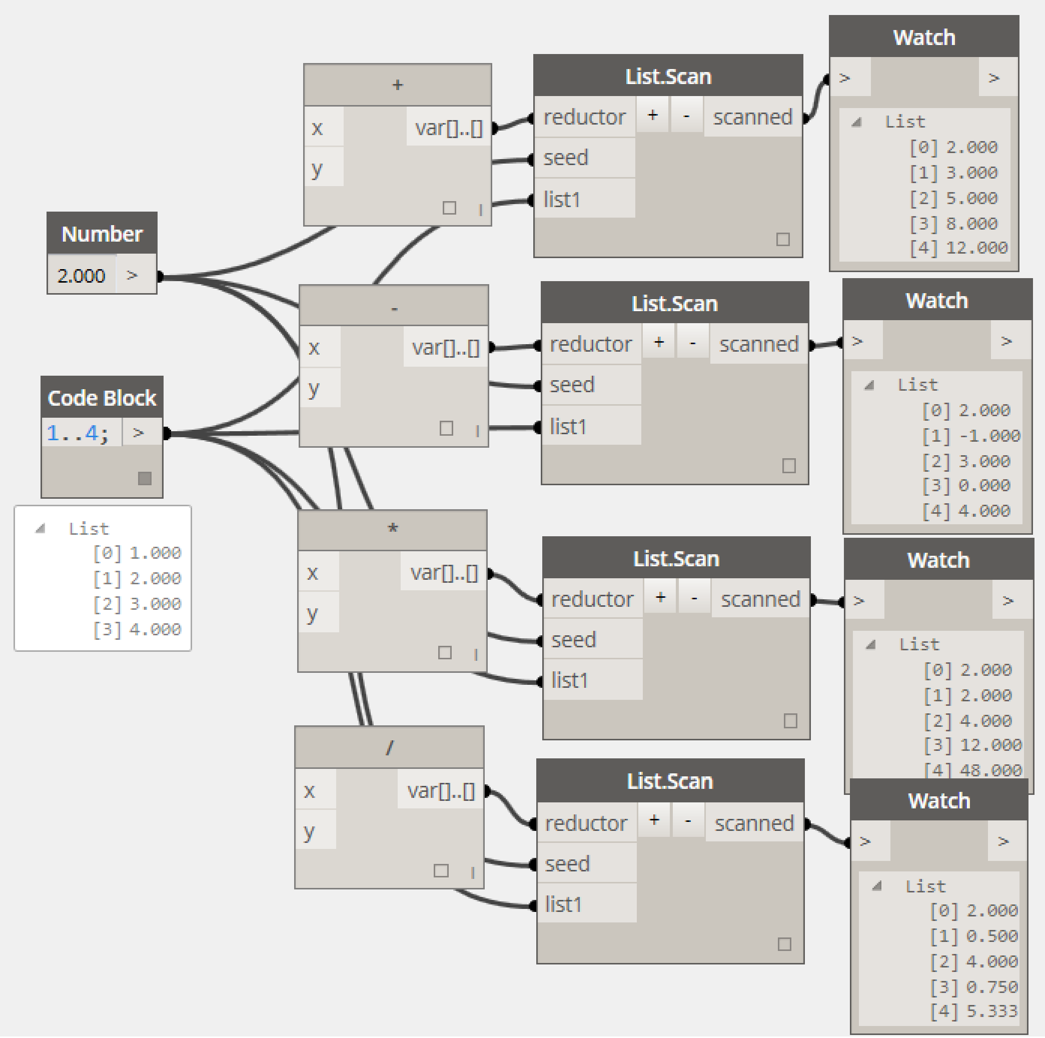
replace by condition



rest of items



scan



```
List
  [0] 1.000
  [1] 2.000
  [2] 3.000
  [3] 4.000
```

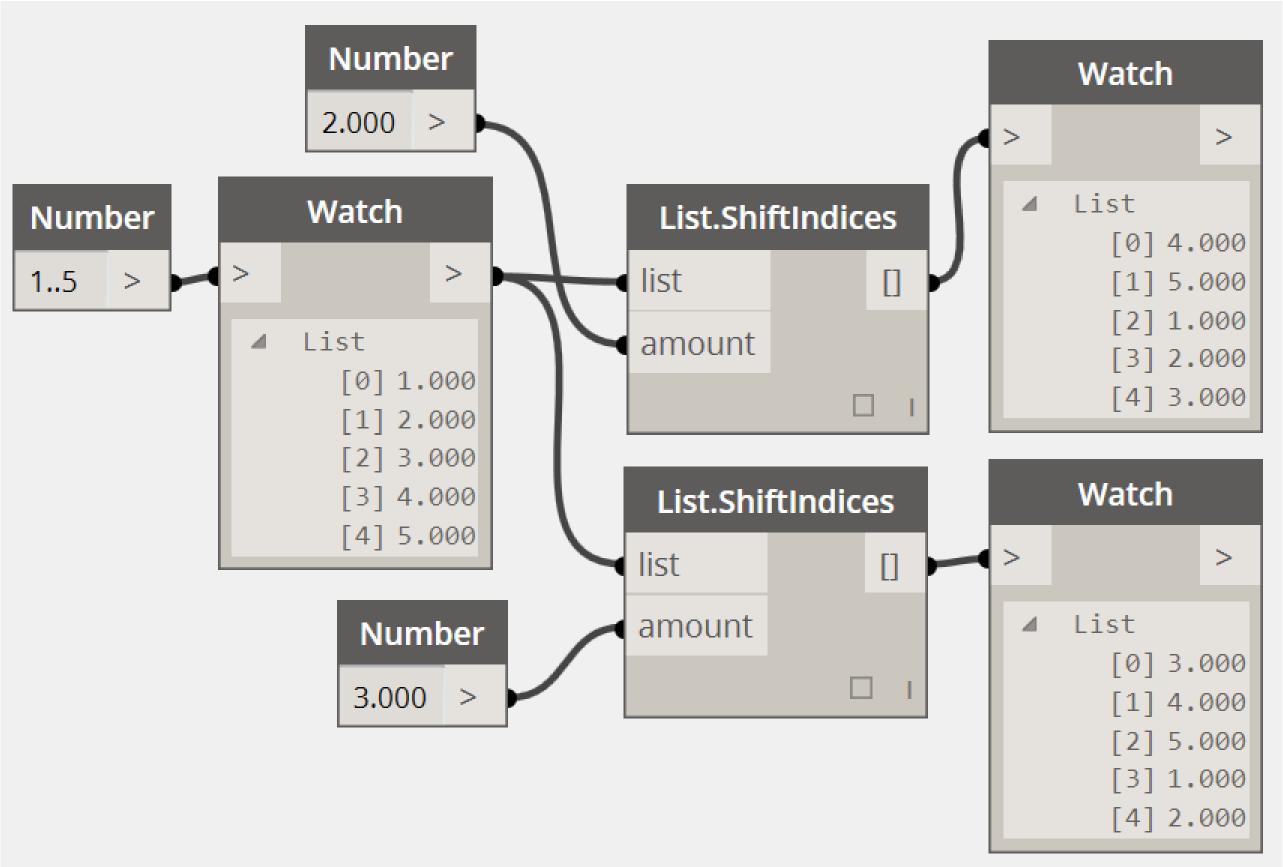
```
List
  [0] 2.000
  [1] 3.000
  [2] 5.000
  [3] 8.000
  [4] 12.000
```

```
List
  [0] 2.000
  [1] -1.000
  [2] 3.000
  [3] 0.000
  [4] 4.000
```

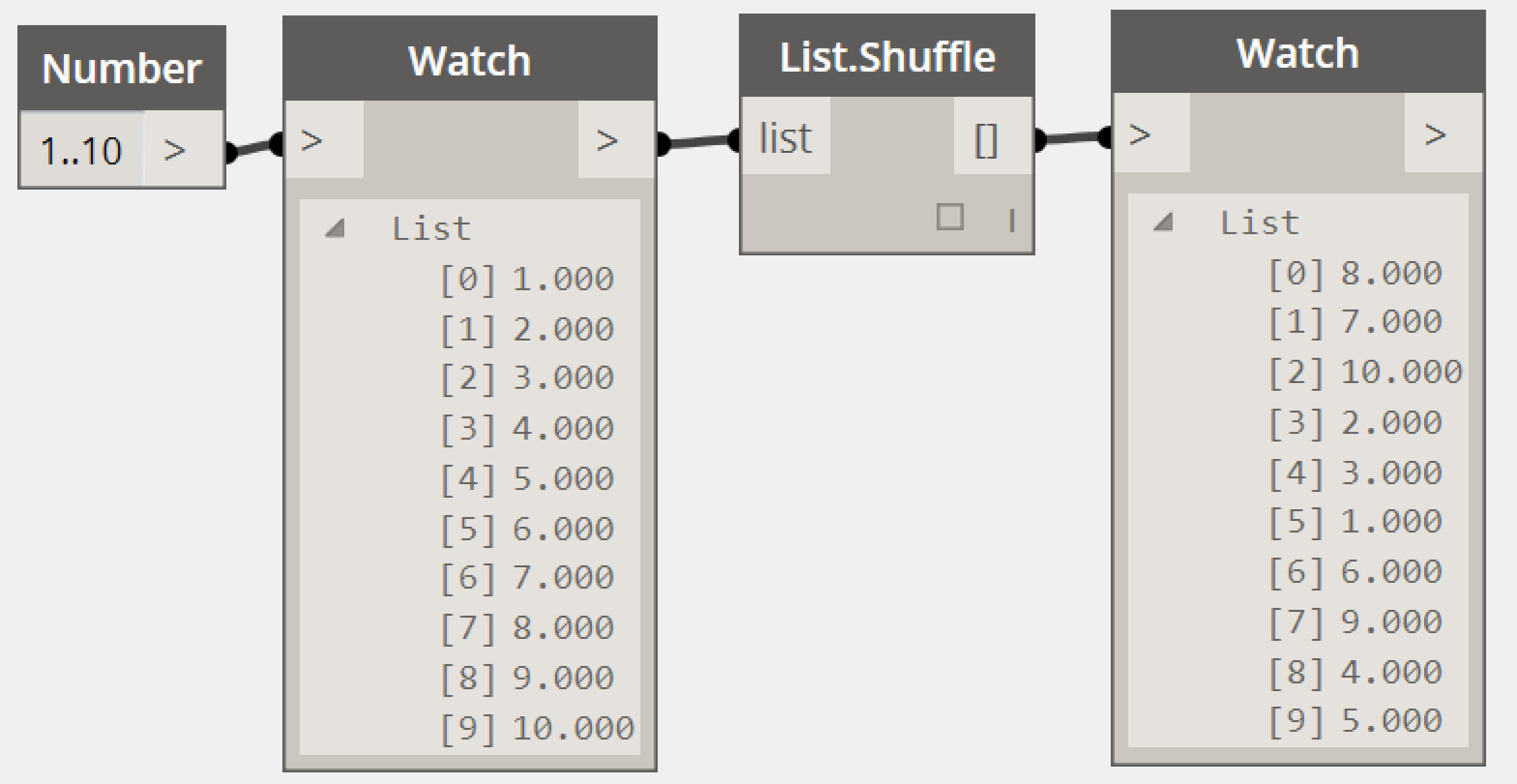
```
List
  [0] 2.000
  [1] 2.000
  [2] 4.000
  [3] 12.000
  [4] 48.000
```

```
List
  [0] 2.000
  [1] 0.500
  [2] 4.000
  [3] 0.750
  [4] 5.333
```

shift indices



shuffle



```
List
  [0] 1.000
  [1] 2.000
  [2] 3.000
  [3] 4.000
  [4] 5.000
  [5] 6.000
  [6] 7.000
  [7] 8.000
  [8] 9.000
  [9] 10.000
```

```
List
  [0] 8.000
  [1] 7.000
  [2] 10.000
  [3] 2.000
  [4] 3.000
  [5] 1.000
  [6] 6.000
  [7] 9.000
  [8] 4.000
  [9] 5.000
```

…well done!

…