第 5 节：GEE 的数据类型(Dictionary，List，Array)

文本和数字的功能是进行描述和数据储存，而 Dictionary，List 和 Array 可以看作是文本和数字间通过不同结合形式而形成的新的数据类型。简单的说，Dictionary，List 和 Array 是拥有了一定“格式”的文本或数字。通过本节的学习，我们将初步体会这三种数据形式在 GEE 中的语法和功能，以及从概念上了解为什么它们必须遵守一定格式要求。

5.1 Dictionary

Dictionary 的中文含义是“字典”。直觉上，字典给人的印象是通过 A-Z 的排序规则将一系列词汇进行整理的文本集合，而且每个词汇都有对应的含义解释。字典的这种特征也反映在了 GEE 的 Dictionary 上，但 GEE 的 Dictionary 并不将内容限定为词汇(文本)，而是包含了数字、词汇和符号。我们通过下边的学习来理解 Dictionary 的格式和用途。

下边介绍 Dictionary 变量的创建，代码及执行效果如下：

```
var Dictionary_Profile = ee.Dictionary( {
                        Name: 'Jinzhu Wang',
                        Gender: 'Male',
                        Age: '> 20',
                        Location: 'Chongqing'
} );

print( Dictionary_Profile );
```

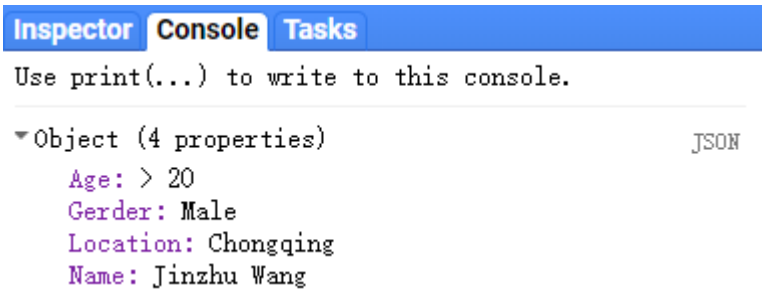


图 5.1 字典的创建

本例中，字典的创建格式与前述是相同的，在具体命令上，通过 ee.Dictionary({})命令来告诉 GEE 这个变量是字典形式的。这里需要指出两点，第一，创建命令里出为什么会出现大括号？第二，字典的“对应关系”是怎么体现的？针对第一点，答案是 GEE 中还存在 List 和 Array 格式的数据形式，如果没有大括号{}，GEE 在执行命令的时候就会犯晕，因为它分不清小括号里边到底是哪种数据格式，然后就会报错。所以当声明字典格式的数据时，普遍采用的方法就是在字典两边加上大括号{}。针对第二点，对应关系是通过冒号“：”来实现的，冒号左边是关键词(Key)，冒号右边是关键词对应的内容(content)。应该指出，关键词相当于变量名，因此即使是文本也不需要加引号，而内容作为数据，应该遵守数据格式的规范，即文本要加引号，数字不用加引号等。

下边介绍字典的合并命令，具体代码和执行效果如下所示：

```
var Dict_1 = ee.Dictionary( { Weight:'50kg', Hight:'160cm'    } );
var Dict_2 = ee.Dictionary( { Weight:'70g', Age: 26 } );
var Dict_Combine = Dict_1.combine( Dict_2, true );

print( Dict_Combine );
```

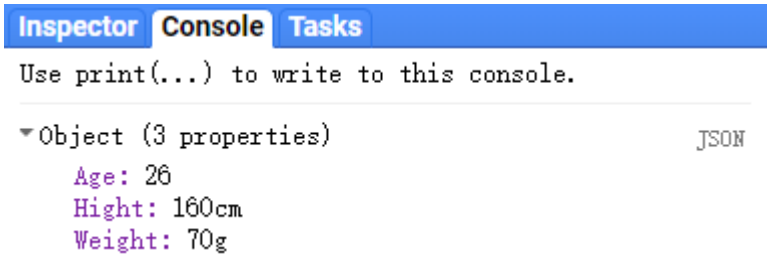


图 5.2 字典的合并 1

在本例中，两个例子在合并时遇到了 Key 相同的情况，此时如果在.combine(,)命令中指定第二个参数为 true，那么这时在合并的字典中就会将重复的内容保留为第二个变量的值。相反的，如果将第二个参数指定为 false，那么合并字典中的重复内容就是第一个字典的值。这种情况的具体代码和执行效果如下所示：

```
var Dict_1 = ee.Dictionary( { Weight:'50kg', Hight:'160cm'    } );
var Dict_2 = ee.Dictionary( { Weight:'70g', Age:26 } );
var Dict_Combine = Dict_1.combine( Dict_2, false);

print( Dict_Combine );
```

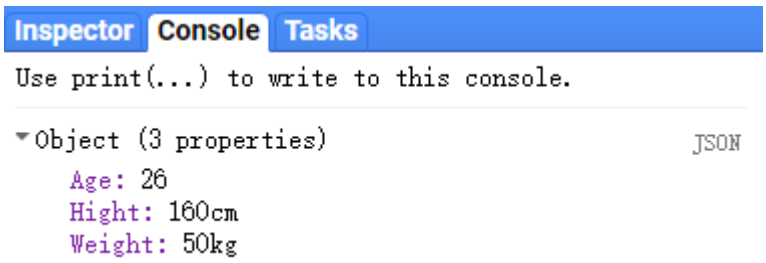


图 5.3 字典的合并 2

下边介绍字典内容的更改命令，具体代码和执行效果如下所示：

```
var Dict_1 = ee.Dictionary( {
                                Name: 'Jinzhu Wang',
                                Gender: 'Male',
                                Age: '> 20',
                                Location: 'Chongqing'
} );
var Dict_Change = Dict_1.set('Age', '<30' );

print( Dict_Change);
```

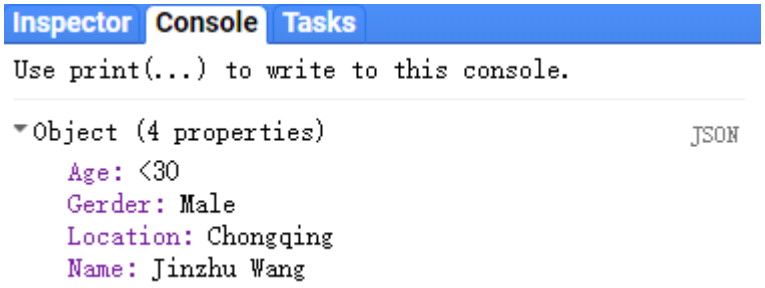


图 5.4 字典的更改

本例中，.set(,)有两个参数，第一个参数是要更改内容的 Key，第二个参数是将要新写入的内容。

下边介绍字典关键词(Key)陈列命令，具体代码和操作效果如下所示：

```
var Dict_1 = ee.Dictionary( {
                                Name: 'Jinzhu Wang',
                                Gender: 'Male',
                                Age: '> 20',
                                Location: 'Chongqing'
} );

var List_Keys= Dict_1.keys( );

print( List_Keys);
```

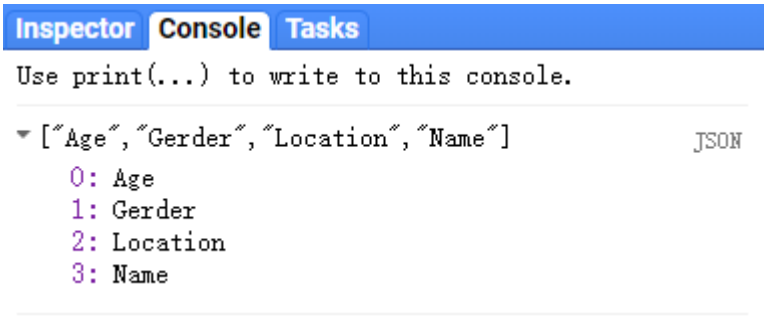


图 5.5 字典的关键词陈列

下边介绍字典的内容读取命令，代码及执行效果如下：

```
var Dict_1 = ee.Dictionary( {
                                Name: 'Jinzhu Wang',
                                Gender: 'Male',
                                Age: '> 20',
                                Location: 'Chongqing'
} );

var The_Name = Dict_1.get('Name' );

print( The_Name);
```

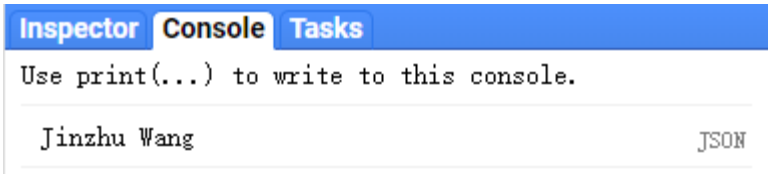


图 5.6 字典的内容读取

本例中，通过在.get()中输入 Key，就可以获得相应 Key 中的内容。

下边介绍字典的多内容查询，代码及执行效果如下：

```
var Dict_1 = ee.Dictionary( {
                                Name: 'Jinzhu Wang',
                                Gender: 'Male',
                                Age: '> 20',
                                Location: 'Chongqing'
} );

var The_Values = Dict_1.values(['Name','Age','Location' ]);

print( The_Values);
```

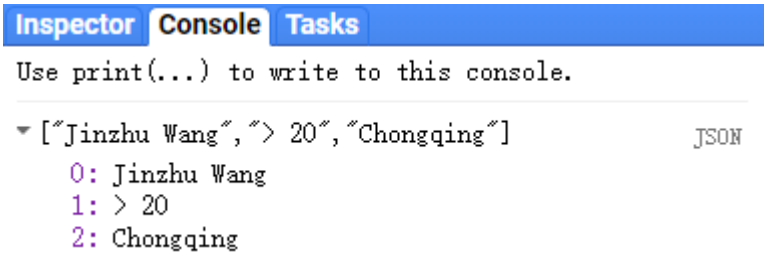


图 5.7 字典的多内容读取

本例中，利用一个方括号，我们可以输入多个 Key，进而获得多个内容。方括号[]所表明的数据格式是 List，具体细节将在下一小节讲解，这里可以将它理解成为“放数据的篮子”。

下边介绍字典的关键词存否命令，代码及执行效果如下：

```
var Dict_1 = ee.Dictionary( {
                                Name: 'Jinzhu Wang',
                                Gender: 'Male',
                                Age: '> 20',
                                Location: 'Chongqing'
} );

var The_Contain = Dict_1.contains('Hight');

print( The_Contain);
```

本例中，.contains()的功能是查询括号内的关键词是否存在于字典中。需要注意的是，因为字典中的关键词是变量，所以不用在两边加引号，但当查询这些关键词是否存在时，由于查询的是“词”是否存在，因此需要用文本格式，即在查询内容两边加上引号。

下边介绍字典的尺寸查询命令，代码及执行效果如下：

```
var Dict_1 = ee.Dictionary( {
                                    Name: 'Jinzhu Wang',
                                    Gender: 'Male',
                                    Age: '> 20',
                                    Location: 'Chongqing'
} );

var The_Size = Dict_1.size();

print( The_Size);
```

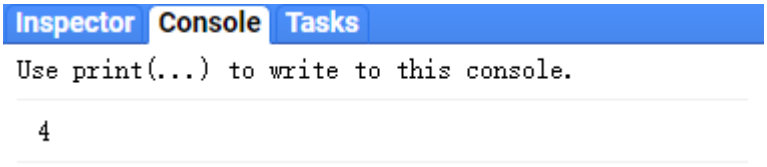


图 5.8 字典的尺寸查询

关于字典的常用命令已经介绍完毕，下边给出本节所用到的字典命令，尝试回忆其格式及用法。

ee.Dicionary()	dictionary.combine()	dinctionary.set()	dictionary.keys()
dictionary.get()	dictionary.values()	dictionary.contains()	dictionary.size()

5.2 List

List 的中文含义是“单子，字条”，在 GEE 的语境下，List 主要用来存储一系列数据，这些数据可以由不同的格式(比如数字，文本，字典等)组成。我们可以把 List 理解为“文件夹”，用来在 GEE 的代码中存储各种数据。

下边介绍 List 的创建命令，代码如下：

```
var List_Example = ee.List( [1,2,3,'A','B','C',['Hello','Good','Bye']] );

print( List_Example );
```

在本例中，需要注意两点，第一点是 List 区别于其他数据格式的标志是两端的方括号[]。采用方括号[]的原因与之前文本的引号和 Dictionary 的大括号是一致的，即都是为了区分这种独特的数据格式。第二点是 List 内部通过逗号“,”来分割空间，同时其中的每一个空间都可以用来存储任意数据格式(包括遥感图像和矢量文件)的数据。

本例中，List 的第 7 个位置(代码语境中编号从 0 开始，因此编号为 6 的位置是第七位)是另一个 List，因此这个 List 还可以进一步的展开。代码的执行效果如下：

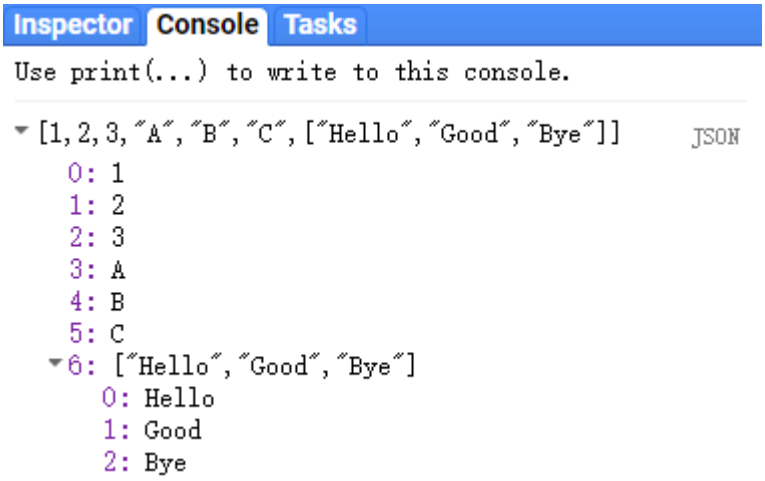


图 5.9 List 的创建

下边介绍 List 的重复创建命令，代码及执行效果如下：

```
var List_Repeat = ee.List.repeat( 'Yeah!', 5 );

print( List_Repeat );
```

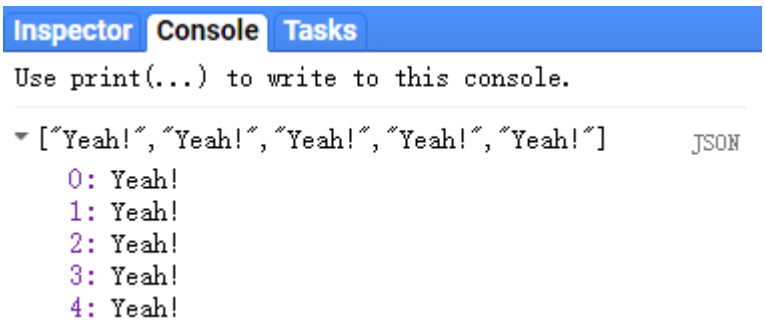


图 5.10 List 的重复创建

下边介绍 List 的等差创建命令，代码及执行效果如下：

```
var List_Sequence= ee.List.sequence( 0, 9, 1 , null );

print(List_Sequence);
```

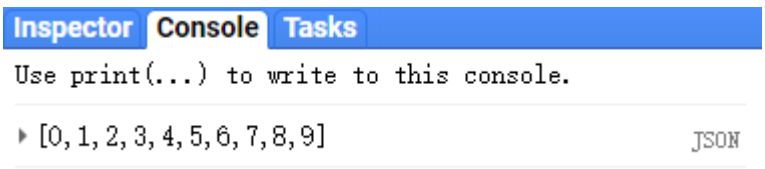


图 5.11 List 的等差创建 1

本例是一个多参数命令。虽然之前介绍的命令多数是单参数的，但事实上在 GEE 中多数命令是多参数的。对于多参数命令，我们必须熟悉每一个参数代表的含义及控制的功能，以上述代码为例，ee.List.sequence(， ， ，)一共有四个参数，他们分别代表等差数列的“首部数字”，“尾部数字”，“公差”和“项数”，其中“公差”和“项数”只能存在一个。那么该如何学习多参数命令呢？答案是在“代码及个人文件”的子栏目“Docs 中”输入要查询的命令，就可以查看这个命令的参数个数及功能。

将 ee.List.sequence 输入 Docs 的查询栏中，得到以下解释：

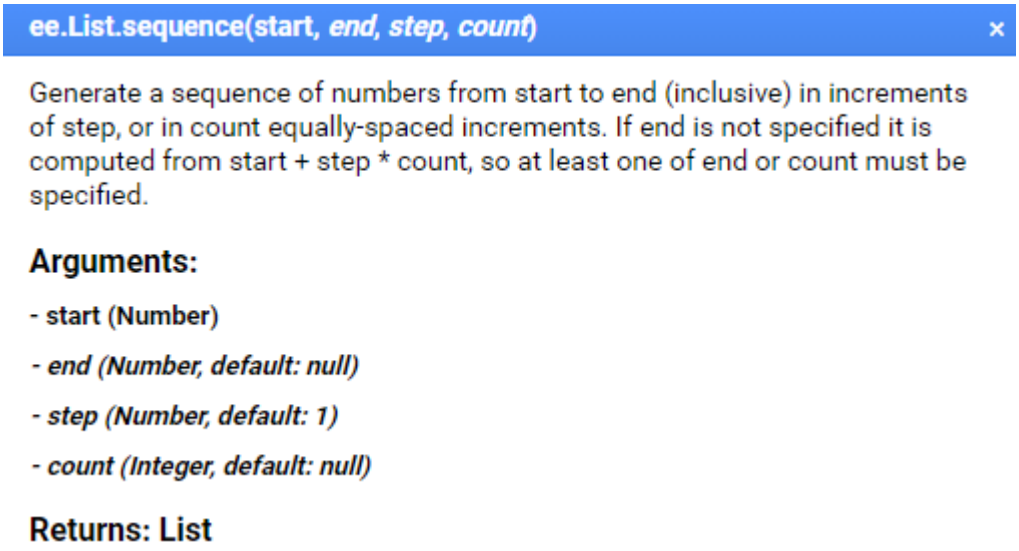


图 5.12 ee.List.sequence 的命令解释

需要强调的是，在 ee.List.sequence (0, 9, 1 , null)中，start = 0，end = 9，step = 1，count = null (空)，即参数位置决定参数类型，因此必须十分注意参数位置。同样的，如果我们想创建一个 0 到 9 之间由四个数字构成的等差数列，命令必须是 ee.List.sequence (0, 9, null ,4)，即使其中第三个位置不存在参数，也必须以 null 进行填位，进而保证第四个参数的准确。

另外，可不可以不用位置确定参数类型呢？答案是可以的，以 ee.List.sequence (0, 9, null ,4)为例，具体代码和执行效果如下，其中因为位置信息的缺失，所以原来的参数变成了一个 Dictionary，通过字典的数据结构将参数名和参数联系到了一起。

```
var List_Sequence= ee.List.sequence(
      { start  :  0,  end  :  9,  count :  4 }          );

print(List_Sequence);
```

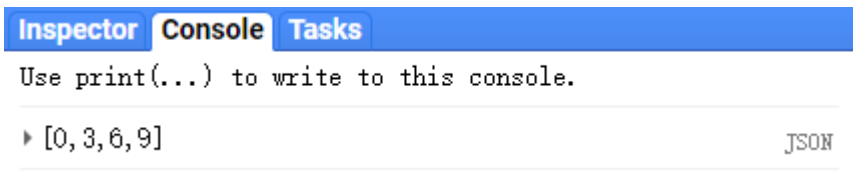


图 5.13 List 的等差创建 2

下边介绍 List 的改写命令，代码及执行效果如下：

```
var List_1 = ee.List( [1992,01,20,'No.1'] );
var List_2 = List_1.set(0,1993);
var List_3 = List_1.set(-1,'No.2');

print( List_1, List_2, List_3 );
```

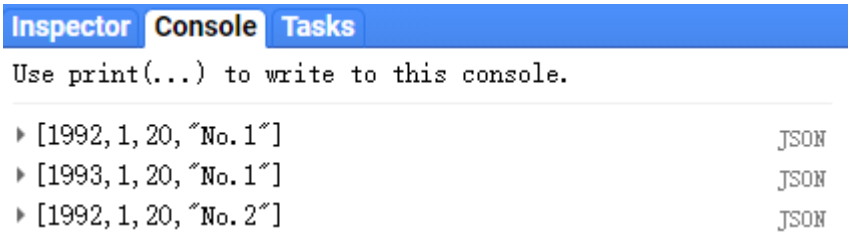


图 5.14 List 的改写

通过本例可以看出，List 的改写命令的两个参数分别表示“改写位置”和“改写内容”。当改写位置的值为负数时，代表从右往左方向的位置。

下边介绍 List 的替换命令，代码及执行效果如下：

```
var List_1 = ee.List( [1992,01,20,'No.1', 1993,01,15,'No.2'] );

var List_2 = List_1.replace(01,05);
var List_3 = List_1.replaceAll(01,05);

print( List_1, List_2, List_3 );
```

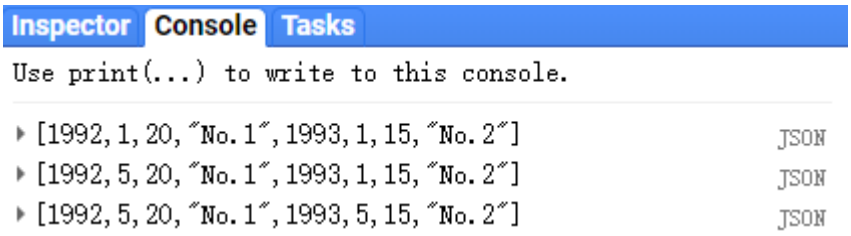
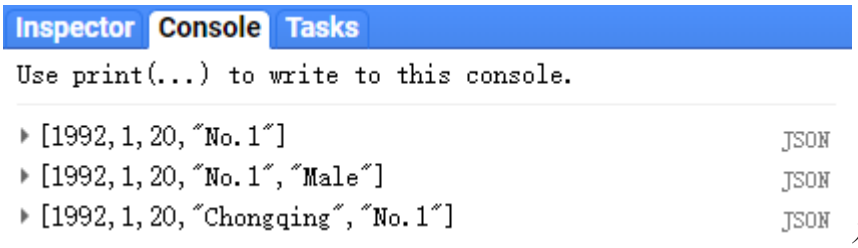


图 5.15 List 的替换

下边介绍 List 的添加和插入命令，代码及执行效果如下：

```
var List_1 = ee.List( [1992,01,20,'No.1']);
var List_2 = List_1.add('Male');
var List_3 = List_1.insert(3,'Chongqing');


print( List_1, List_2, List_3 );
```



入

图 5.16 List 的添加和插入

下边介绍 List 的打包命令，代码及执行效果如下：

```
var List_1 = ee.List( [1992,01,20,'No.1']);
var List_2 = List_1.zip(['Year','Month','Day']);

print( List_1, List_2);
```

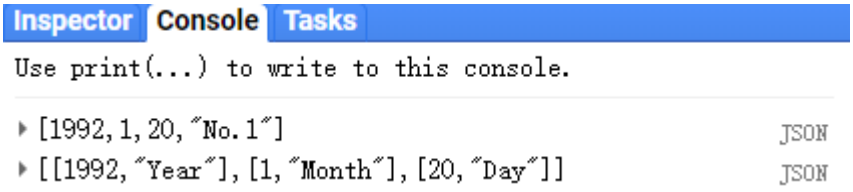


图 5.17 List 的打包

下边介绍 List 的倒置和转置功能，代码及执行效果如下：

```
var List_1 = ee.List( [ 0,1,2,3,4,5,6,7,8,9 ] );
var List_Reverse = List_1.reverse( );
var List_Rotate = List_1.rotate(5);

print( List_Reverse, List_Rotate );
```

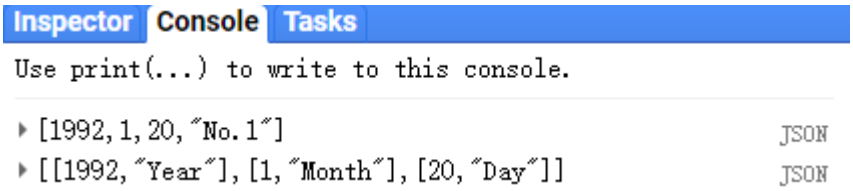


图 5.18 List 倒置和转置

下边介绍 List 的排序命令，代码及执行效果如下：

```
var List_1 = ee.List( [ 'Zhao', 'Wang', 'Ma', 'Liu', 'Bai' ] );
var List_Sort = List_1.sort( );

print( List_1, List_Sort );
```

图 5.19 List 排序

下边介绍 List 的位置互换命令，代码及执行效果如下：

```
var List_1 = ee.List( [0,1,2,3,4,5,6,7,8,9,0] );
var List_2 = List_1.swap( 5,8 );

print( List_1, List_2 )
```

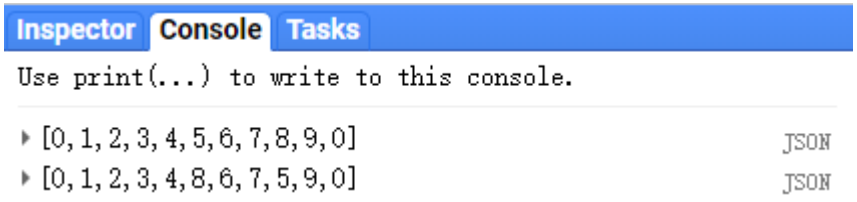


图 5.20 List 位置互换

下边介绍 List 的一维化命令，代码及执行效果如下：

```
var List_1 = ee.List( [     [[123,456], [798,101]]   ,
                            [987,654], [321,909]]
                                                    ]);
var List_2 = List_1.flatten( );

print( List_1, List_2 );
```

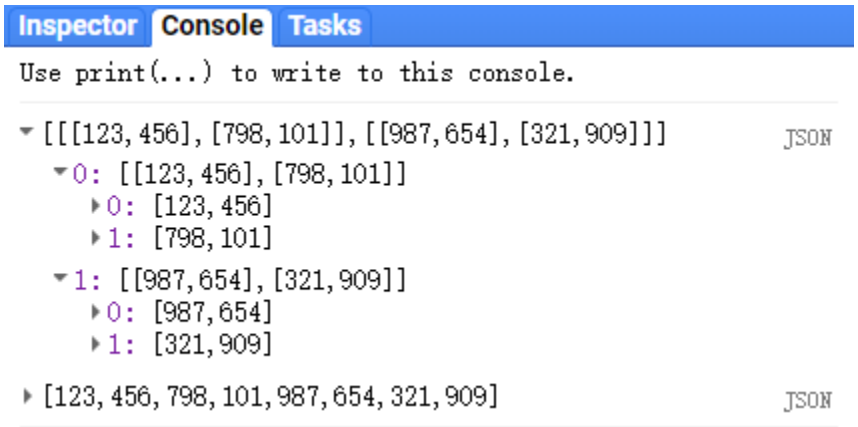


图 5.21List 的一维化

下边介绍 List 的查询和删除，代码及执行效果如下：

```
var List_1 = ee.List([1,1,2,2,3,3,4,4,5,5])
var List_2 = List_1.get(3);
var List_3 = List_1.remove(3);
var List_4 = List_1.removeAll([3,4]);

print( List_1,List_2,List_3,List_4 );
```

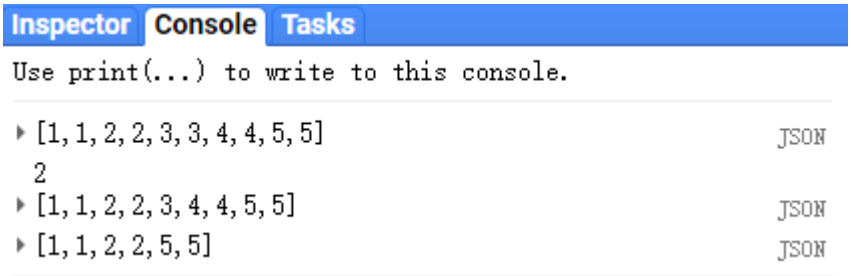


图 5.22 List 的查询和删除

下边介绍 List 的是否等于以及是否包含命令，代码及执行效果如下：

```
var List_1 = ee.List( ['zhao','qian','sun','li'] );
var List_2 = ee.List( ['zhao','qian','li','sun'] );
var List_3 = ee.List( ['zhao','qian','li'] );

var True_False_1 = List_1.equals( List_2 );
var True_False_2 = List_1.contains('li');
var True_False_3 = List_1.containsAll(List_3);

print( True_False_1,True_False_2,True_False_3);
```

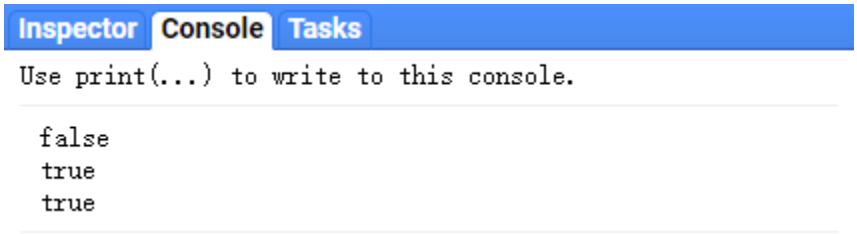


图 5.23 List 的是否包等于与是否包含

下边介绍 List 的内容位置和频率查询命令，代码及执行效果如下：

```
var List_Number = ee.List( [1,2,3,4,5,6,4,5,6,5,5,5] );

var Index_Number = List_Number.indexOf( 5 );
var Index_Sub = List_Number.indexOfSublist([4,5,6]);
```

```
var Index_Last_Sub = List_Number.lastIndexOfSubList([4,5,6])
var Frequency_Number = List_Number.frequency(5)

print( List_Number );
print( Index_Number );
print( Index_Sub );
print( Index_Last_Sub );
print(Frequency_Number)
```

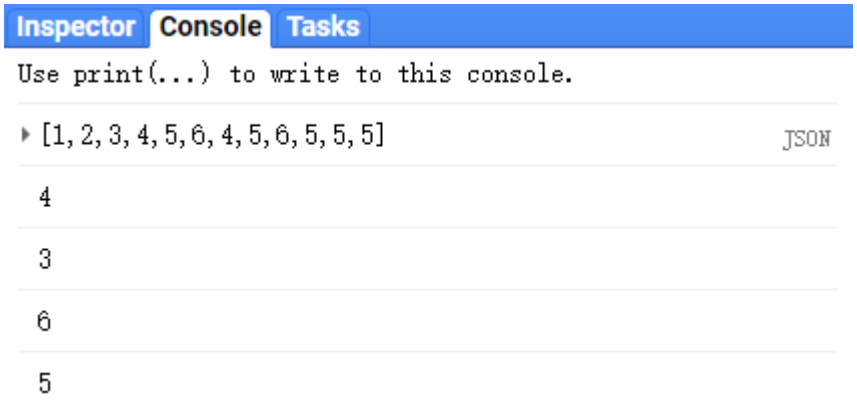


图 5.24 List 的内容位置和频率查询

下边介绍 List 的 map 命令，代码及执行效果如下：

```
var List_1 = ee.List( ['Zhao','Qian','Sun','Li'] );

function Do (Name) {
        return ee.List.repeat(Name,3);
    }
var List_2 = List_1.map( Do );
print( List_1, List_2 );
```

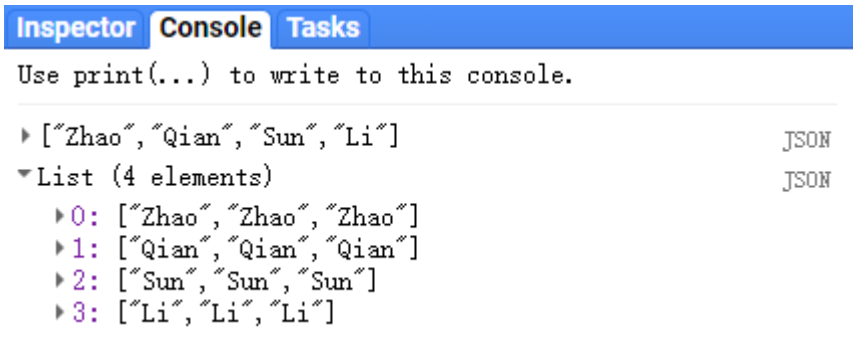


图 5.25 List 的 map 命令

本例中，我们只需要理解.map 命令的作用是对 List 中的每一个对象(Object)都执行了某种相同的操作，这种相同的操作由 function 命令进行编写。总体上，.map 命令的价值在于减少重复工作，提高工作效率。

下边介绍 List 的循环命令，代码及执行效果如下：

```
var List_1 = ee.List.sequence(1,100,1);

function Do ( Number_1, Number_2 )
        { return ee.Number(Number_1).add(Number_2);
        }

var List_2 = List_1.iterate( Do, 0 );

print( List_1, List_2 );
```

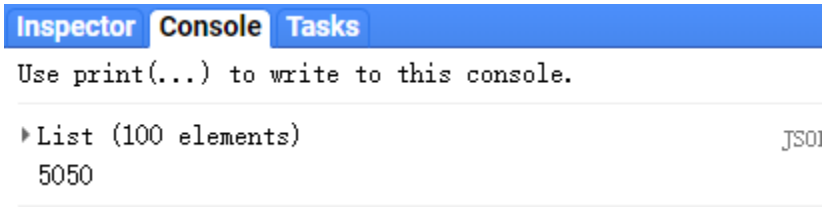


图 5.26 List 的循环命令

本例中的目的是求取 1-100 这 100 个自然数的和。现阶段，我们只需要理解.iterate()命令有两个参数，第一个参数是执行循环要执行的参数方程，第二个是循环的初始量就行。

下边介绍 List 的尺寸计算命令，代码及执行效果如下：

```
var List_1 = ee.List([          ['Zhao','Qian'],
                                 ['Sun','Li'],
                                 ['Zhou','Wu'],
                                 ['Zheng','Wang']
                                                    ]);

var Length_Number_1 = List_1.size( );
var Length_Number_2 = List_1.length( );

print( List_1 );
print( Length_Number_1 );
print( Length_Number_2 );
```

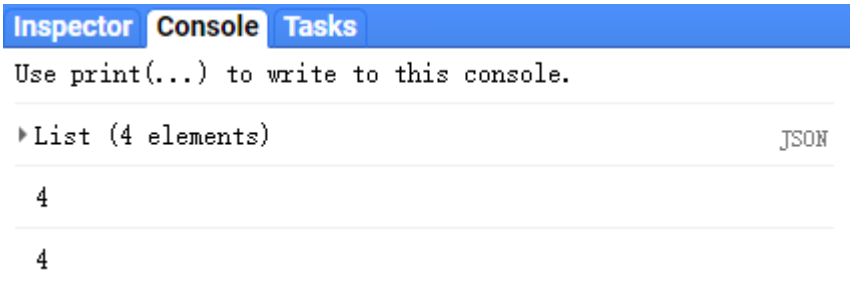


图 5.27 List 的大小计算

可以看出，.size()和.length()指令具有相同的效果。

下边整理了本节介绍的常用 List 命令：

ee.List()	list.repeat()	list.sequence()	list.set()	list.replaceAll()
ist.add()	list.insert()	list.zip()		
list.reverse()	list.rotate()	list.sort()	list.swap()	list.flatten()
list.get()	list.remove()	list.removeAll()		
list.equals()	list.contains()	list.containsALL()	list.indexOfSubList()	
list.indexOf()	list.frequency()	list.lastIndexOfSubList()		
list.map ()	list.iterate()	list.length()	list.size()	

5.3 Array

在上述数据格式中，Dictionary 可以说是变量与内容间“对应关系”，List 可以说是存储变量的“容器”，那么 Array 应该怎么理解呢？Array 的中文含义是“数组，阵列，矩阵”，其本质上仍属于 List 的范畴。作为高阶遥感分析中的核心数据格式，我们可以这样理解 Array：“带有方向的 List”，同时应该注意这种 List 只能由数字构成。

下边介绍矩阵的构建，代码及执行效果如下：

```
var Array_1 = ee.Array( [[1], [2], [3]]);          //3*1
var Array_2 = ee.Array( [[1, 2, 3]]);              //1*3

print( Array_1,Array_2 );
```

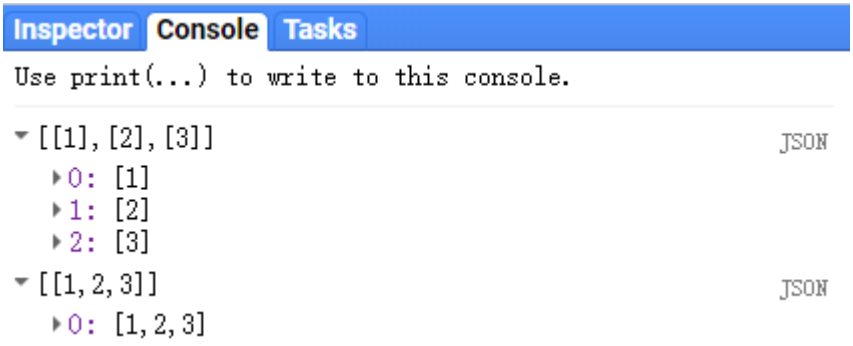


图 5.28 矩阵的创建

本例中分别创建了一个 3×1 和 3×1 的 Array。那么如何理解 Array 中方向呢？以 ee.Array([[1], [2], [3]]) 为例，可以把括号内的逗号想象成进行 Word 编辑时的回车键，因为其中的 List 间有两个逗号，因此相当于换了两次行，而每一个的内容只有一个数字，因此这是一个 3 行 1 列的矩阵。对于 ee.Array([[1, 2, 3]])，因为 List 间没有逗号，所以整个“文档”里只有一行，而对于这一行来说，存在 3 个数字，因此这是一个 1 行 3 列的矩阵。

下边介绍单位矩阵的构建，代码及执行效果如下：

```
var Array_1 = ee.Array.identity( 5 );

print( Array_1 );
```

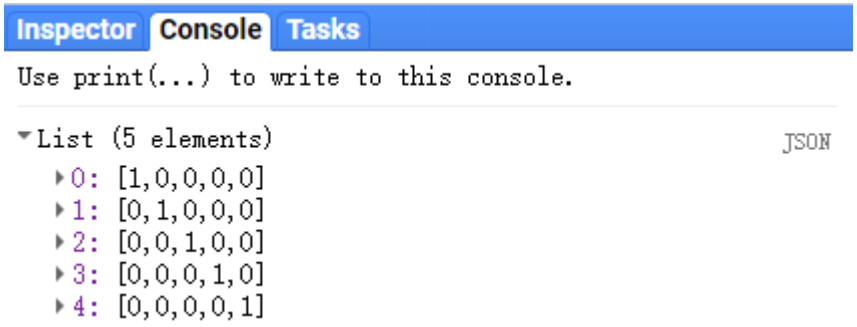


图 5.29 单位矩阵的创建

下边介绍矩阵的重复命令，代码及执行效果如下：

```
var Array_1 = ee.Array([        [ 1,2,3 ],
                                [ 4,5,6 ]       ]);

var Array_2 = Array_1.repeat( 1,2 );
print( Array_1, Array_2 );
```

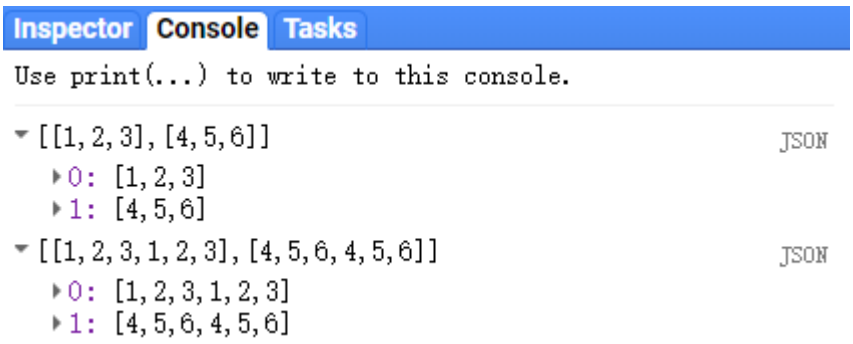


图 5.30 矩阵的重复

在本例中，.repeat(,)的作用是将原矩阵沿着某个方向重复若干次，其中第一个参数是重复方向，第二个参数是拷贝个数。这个操作的示意图如下：

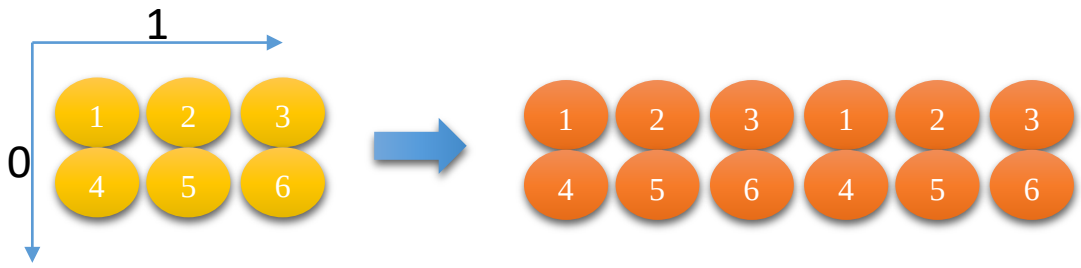


图 5.31 矩阵的重复命令示意图

下边介绍矩阵的掩膜命令，代码及执行效果如下：

```
var Array_1 = ee.Array( [ [1,1],[2,2],[3,3],[4,4]] );
var Array_2 = ee.Array( [ [    0],[    0],[    1],[    0]] );
var Array_3 = Array_1.mask( Array_2 );

print( Array_1, Array_2, Array_3 );
```

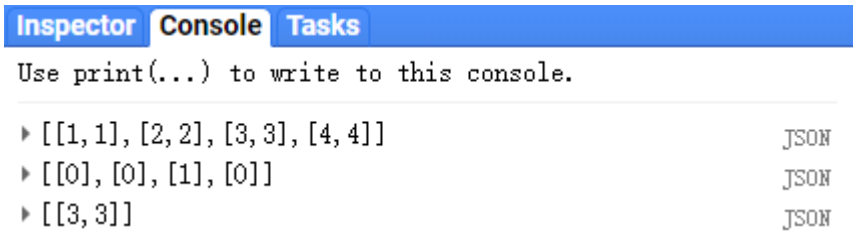


图 5.32 矩阵的掩膜

下边介绍矩阵的倒置命令，代码及执行效果如下：

```
var Array_1 = ee.Array( [       [111,111,111],
                                [222,222,222]       ]);
var Array_2 = Array_1.transpose();
var Array_3 = Array_2.transpose()

print( Array_1, Array_2,Array_3);
```

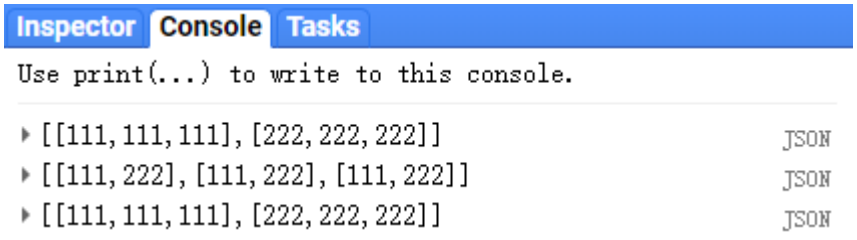


图 5.33 矩阵的倒置

下边介绍矩阵的数据转换命令，代码及执行效果如下：

```
var Array_1 = ee.Array( [ [1.1, 2.2, 3.3], [4.4, 5.5, 6.6] ] );
var Array_2 = Array_1.uint8();

print( Array_1, Array_2 );
```

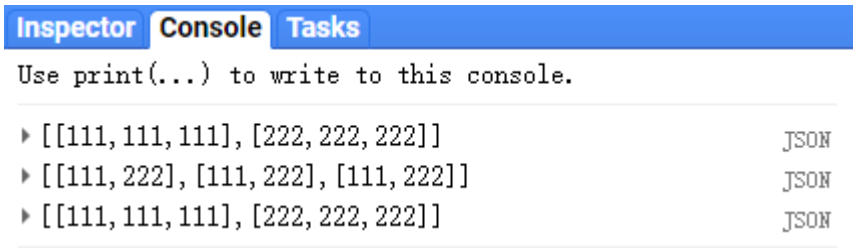


图 5.34 矩阵的数据转换

下边介绍矩阵的比较命令，代码及执行效果如下，类似的命令参照 Number 部分：

```
var Array_1 = ee.Array( [ [1,1],[2,2],[3,3],[4,4]] );
var Array_2 = ee.Array( [ [1,2],[3,4],[4,3],[2,1]] );
var Array_3 = Array_1.eq( Array_2 );

print( Array_1, Array_2, Array_3 );
```

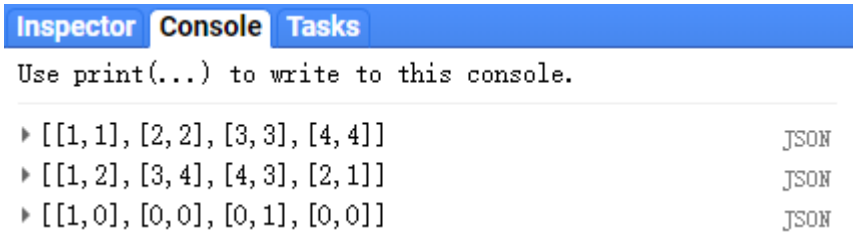


图 5.35 矩阵的比较

下边介绍矩阵的交并命令，代码及执行效果如下，类似的命令参照 Number 部分：

```
var Array_1 = ee.Array( [ [1,0],[0,0],[3.14,1.41]] );
var Array_2 = ee.Array( [ [1,2],[0,0],[3,4]] );
var Array_and  = Array_1.and( Array_2 );
var Array_or   = Array_1.or( Array_2 );

print( Array_1, Array_2, Array_and, Array_or );
```

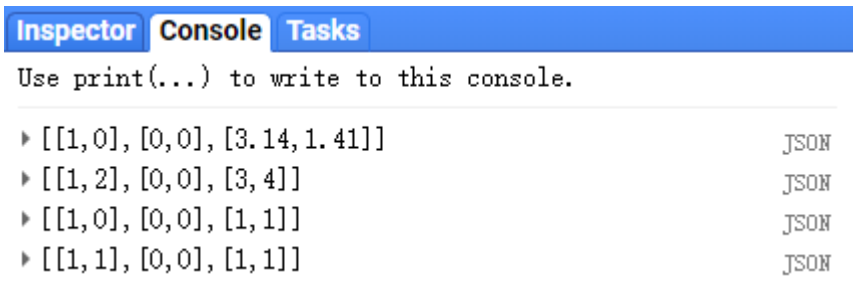


图 5.36 矩阵的交并

下边介绍矩阵的函数命令，代码及执行效果如下，类似的命令参照 Number 部分：

```
var Array_1 = ee.Array( [-3.14, 1.414] );
var Array_2 = Array_1.ceil().abs();

print ( Array_2 );
```

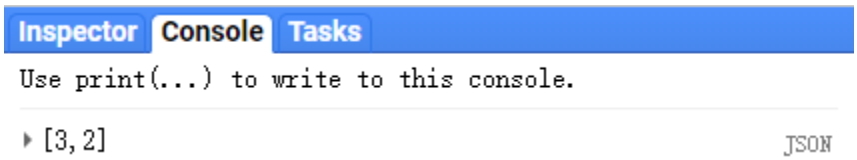


图 5.37 矩阵的函数运算

下边介绍矩阵的数学命令，代码及执行效果如下，类似的命令参照 Number 部分：

```
var Array_1 = ee.Array( [ [1,2,3,4] ] );
var Array_2 = ee.Array( [ [5,6,7,8] ] );
var Result_Array = Array_1.add(Array_2);

print ( Array_1, Array_2, Result_Array );
```

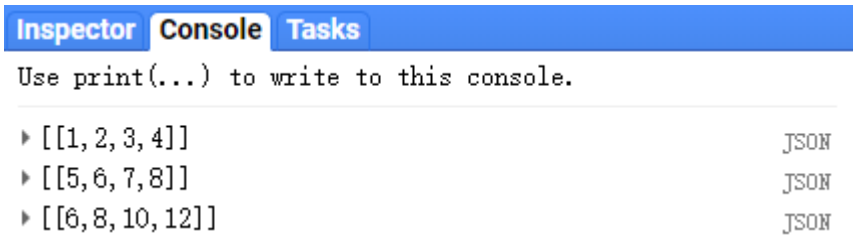


图 5.38 矩阵的数学运算

下边介绍矩阵的位运算命令，代码及执行效果如下，类似的命令参照 Number 部分：

```
var Array_1 = ee.Array( [3,2] );
var Array_2 = ee.Array( [3,3] );
var Array_Bit_And = Array_1.bitwiseAnd( Array_2 );
var Array_Bit_or  = Array_1.bitwise_or( Array_2 );

var Array_3    = ee.Array( [1,2,3] );
var Array_Left  = Array_3.leftShift(  [1,1,1] );
var Array_Right = Array_3.rightShift( [1,1,1] );

print ( 'Bitwise [11,10] and [11,11] =', Array_Bit_And    );
print ( 'Bitwise [11,10]  or [11,11] =', Array_Bit_or );
print (  Array_Left   );
print (  Array_Right );
```

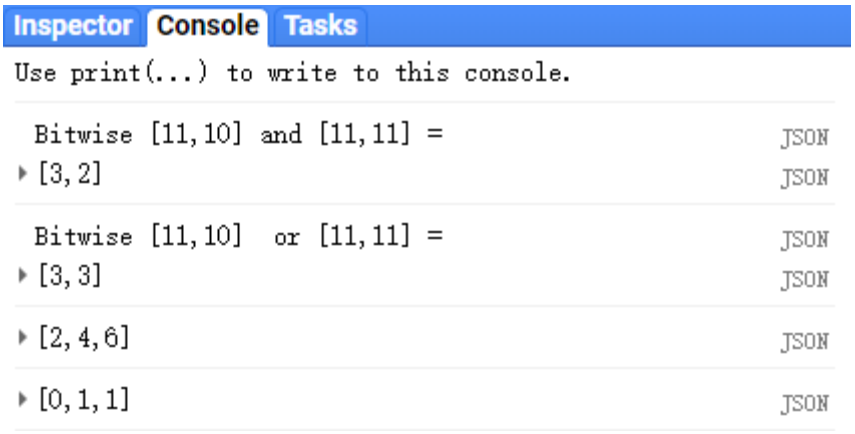


图 5.39 矩阵的位运算

本节有关矩阵的常用命令整理如下：

ee.Aray()	ee.Aray.identity()	ee.Array.repeat()	
array.mask()	array.transpose()	array.uint8()	
array.eq()	array.and()	array.or()	array.round()
rray.bitwiseAnd()	array.leftShift()		

本材料由王金柱（西南大学&迪肯大学）创作。如有需要请与我联系。
This doc contributed by Jinzhu Wang of Southwest University & Deakin University.
Email: wangjinzhulala@gmail.com