14 NumPy 索引和切片

Indexing and Slicing NumPy Arrays

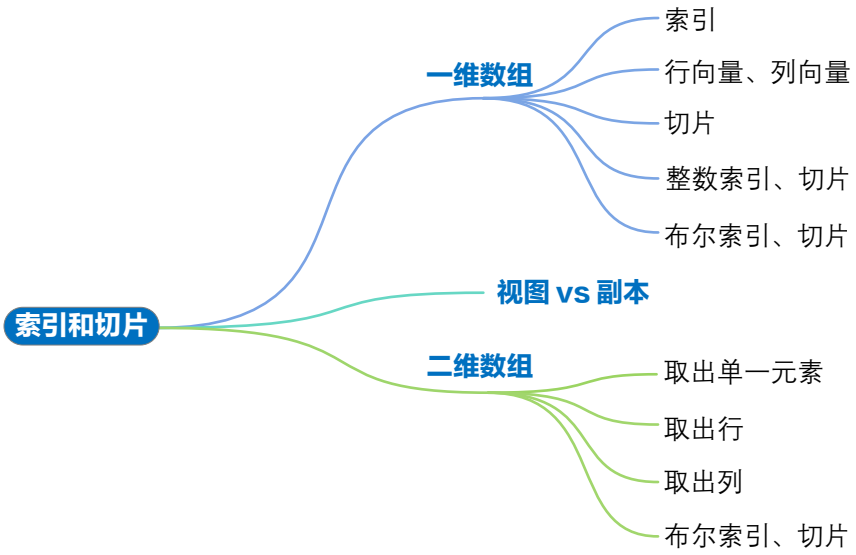
对数组切片切块、切丝切丁

做数学的艺术在于找到包含所有普遍性萌芽的特殊情况。

The art of doing mathematics consists in finding that special case which contains all the germs of generality.

—— 大卫·希尔伯特 (David Hilbert) | 德国数学家 | 1862 ~ 1943

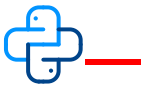
- `numpy.concatenate()` 沿指定轴将多个数组连接成一个新的数组
- `numpy.copy()` 深拷贝数组，对新生成的对象修改删除操作不会影响到原对象
- `numpy.newaxis` 在使用它的位置上为数组增加一个新的维度，可以用于在指定位置对数组进行扩展或重塑
- `numpy.r_()` 用于按行连接数组
- `numpy.reshape()` 用于重新调整数组的形状
- `numpy.squeeze()` 从数组的形状中删除大小为 1 的维度，从而返回一个形状更紧凑的数组
- `numpy.take()` 根据指定的索引从数组中获取元素，创建一个新的数组来存储这些元素
- `numpy.vstack()` 将多个数组按行堆叠



14.1 什么是索引、切片?

这个话题聊一聊 NumPy 数组的**索引** (indexing) 和**切片** (slicing)。简单来说，数组中的某个元素可以通过索引来访问。切片指的是从数组中提取“子数组”的操作。

需要反复强调的是，NumPy 数组使用基于 0 的整数索引。此外，NumPy 的切片操作返回的是原数组的**视图** (view) 而不是**副本** (copy)，因此对切片操作所得到的数组进行修改会直接影响到原数组。本书前文介绍过视图和副本这两个概念，本章后续将专门讲解 NumPy 数组视图和副本之间的区别。



本节配套的 Jupyter Notebook 文件是 Bk1_Ch14_01.ipynb。请大家一边阅读本章内容，一边在 JupyterLab 中实践。

14.2 一维数组索引、切片

索引

一维数组可以使用索引来访问和操作数组中的某个元素。

如图 1 所示，索引是一个整数值，它指定了要访问的元素在数组中的位置。

一维数组的索引从 0 开始，到数组长度 (len(a)) 减 1 结束。

如图 1 所示，想要取出数组 a 的第一个元素，可以用 a[0] 或 a[-11]。

a[-1] 或 a[10] 则取出数组 a 的最后一个元素。请大家在 Bk1_Ch14_01.ipynb 尝试取出数组不同位置元素。

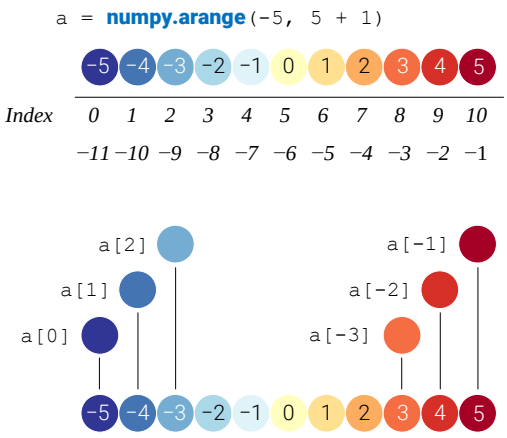


图 1. 一维数组的索引 | Bk1_Ch14_01.ipynb

行向量、列向量

上一个话题特别强调过，本书中行向量、列向量都被视作特殊的二维数组。也就是说，行向量是一行多列矩阵，而列向量是多行一列矩阵。

在 NumPy 中，numpy.newaxis 是一个特殊的索引，用于增加数组的维度。它的作用是在数组的某个位置添加一个新的轴，从而改变数组的维度。

具体来说，使用 numpy.newaxis 将会在数组的一个指定位置添加一个新的维度。如图 2 所示，对于一个一维数组 a，我们可以使用 a[:, numpy.newaxis] 将其转换为一个二维数组，其中新的维度被添加在列的方向上。这个操作将会把数组变成一个列向量。

a[numpy.newaxis, :] 则把一维数组变成行向量。本书后文还会介绍利用 numpy.reshape() 函数完成“升维”及其他变形。

Bk1_Ch14_01.ipynb 还给出其他“升维”方法，请大家自行学习。

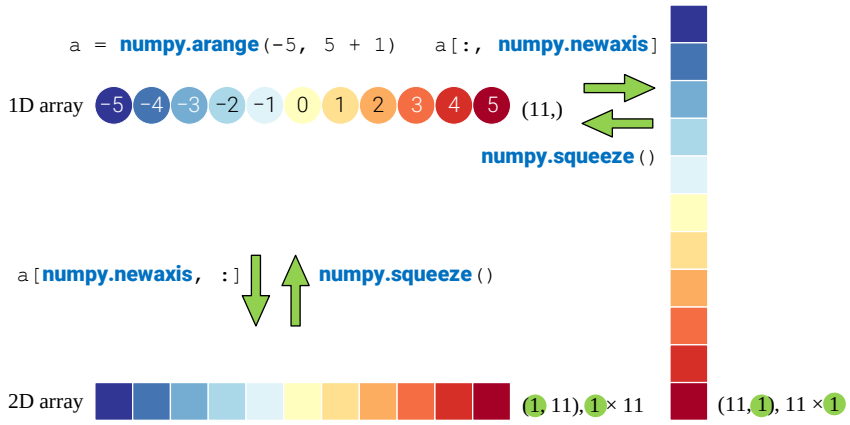


图 2. 一维数组“升维” | Bk1_Ch14_01.ipynb

相反地，在 NumPy 中，numpy.squeeze() 函数用于从数组的形状中删除长度为 1 的维度，并返回一个新的数组，其维度数目更少。

例如，对于一个形状为 (1, 3, 1, 5) 的四维数组，可以使用 numpy.squeeze(a) 函数将其转换为形状为 (3, 5) 的二维数组，其中长度为 1 的第 0 和第 2 维被删除。

如果在调用 numpy.squeeze() 时指定了参数 axis，则只有该轴上长度为 1 的维度会被删除。

总结来说，numpy.squeeze() 函数可以帮助我们简化数组的形状，使其更符合我们的需求。

切片

切片访问一维数组中的“子数组”，即多个元素。切片是一个包含开始索引和结束索引的范围，用冒号 : 分隔。

开始索引指定要获取的第一个元素的位置，结束索引指定要获取的最后一个元素的位置+1。

图 3 所示为一维数组连续切片。

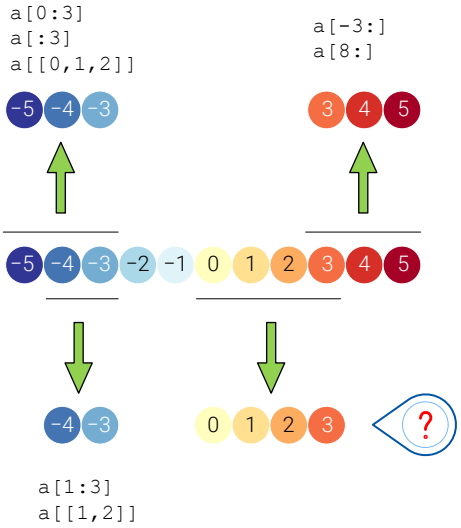


图 3. 一维数组连续切片 | Bk1_Ch14_01.ipynb

图 4 中，将步长设为 2 分别提取数组中的奇数、偶数。

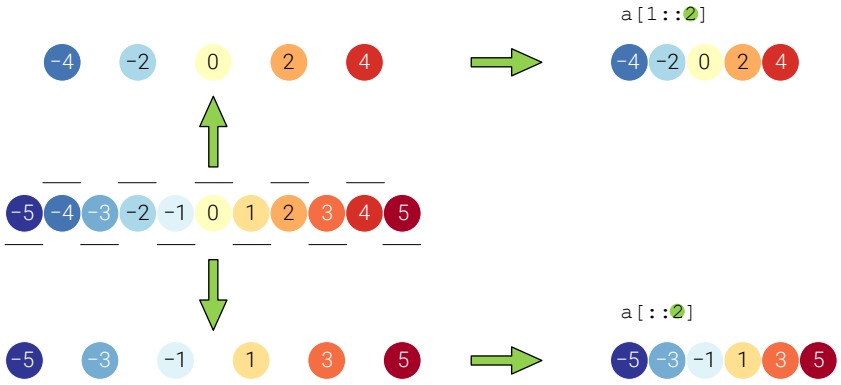


图 4. 一维数组以固定步长切片，步长为 2 | Bk1_Ch14_01.ipynb

图 5 中，将步长设为-1 将数组倒序排序。

Bk1_Ch14_01.ipynb 中还给出其他步长设置，请大家自行学习。

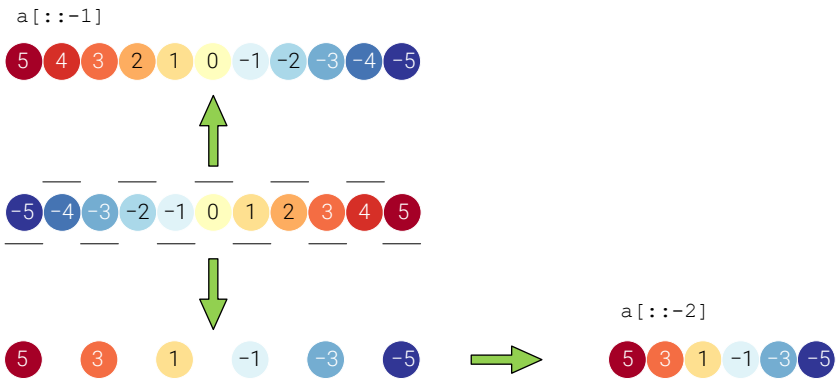


图 5. 一维数组倒序 | Bk1_Ch14_01.ipynb

整数索引、切片

在 NumPy 中，可以使用整数索引来访问和修改数组中的元素。整数索引是一种非常基本的索引方法，它允许使用一个整数或整数数组来访问数组的元素。

使用整数索引时，大家可以传递一个整数来访问数组的单个元素，或者传递一个整数数组来访问数组的多个元素。大家已经在图 1 看到这一点。

如果传递一个整数数组，则该数组的每个元素将被视为索引，从而返回一个新的数组，该数组包含原始数组中相应索引处的元素。

如图 6 所示，整数索引为数组 [0, 1, 2, -1]，我们提取一维数组的第 1、2、3 和最后一个 (-1) 元素，结果还是一维数组。

同时，我们可以用 numpy.r_[0:3, -1]构造一个数组，也能提取相同的元素组合。

numpy.r_() 是一个用于将切片对象转换为一个沿着第一个轴堆叠的 NumPy 数组的函数。它可以在数组创建和索引时使用。它的作用类似于 numpy.concatenate() 和 numpy.vstack()，但是使用切片对象作为索引来方便快捷地创建数组。

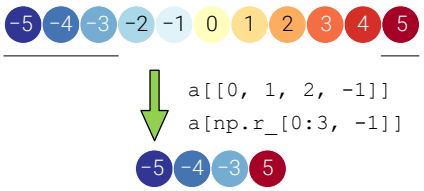


图 6. 一维数组整数索引，输入为数组 | Bk1_Ch14_01.ipynb

布尔索引、切片

布尔索引 (Boolean indexing) 是一种使用布尔值来选择数组中的元素的技术。

在使用布尔索引时，可以通过一些条件来生成一个布尔数组，该布尔数组与要索引的数组具有相同的形状，然后使用该布尔数组来选择要访问的数组元素。

图 7 所示为利用布尔值切片我们分别提取数组中大于 1、小于 0 的元素。

请大家在 JupyterLab 中查看 a > 0 返回的数组是什么。

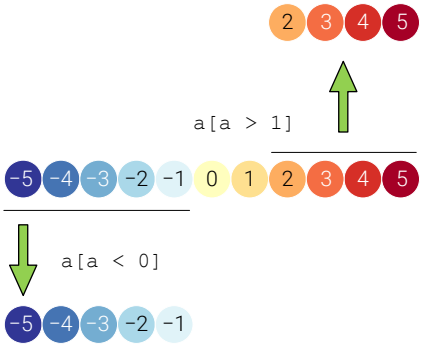


图 7. 一维数组布尔值切片 | Bk1_Ch14_01.ipynb

14.3 视图 vs 副本

在 NumPy 中，有两种不同的方式来创建新的数组对象——**视图 (view)**、**副本 (copy)**。

视图是原始数组的一个新视图，而副本是原始数组的一个新副本。它们的区别在于它们如何处理原始数据的内存和共享。

视图是原始数组的一个新视图，一种重新排列、重新解释。视图是原始数组共享相同的数据，不会创建新的内存。

换句话说，视图是原始数组的一个不同的“窗口”，它可以访问原始数组的相同数据块。当对视图进行更改时，原始数组也会发生相应的更改。

副本则是原始数组的一份完整的拷贝，修改副本不会影响原始数组。当对数组进行切片或使用 numpy.copy() 方法时，将生成一个副本。副本的创建可以使用 numpy.copy() 方法或者 numpy.array() 函数的参数 copy = True 来实现。

如图 8 所示，本节之前的各种索引、切片方法实际上创建的都只是原数组的视图，改变这些视图就会修改原数组，并“牵一发动全身”地改变所有视图。

而 numpy.copy() 则创建了全新的内存，即副本。

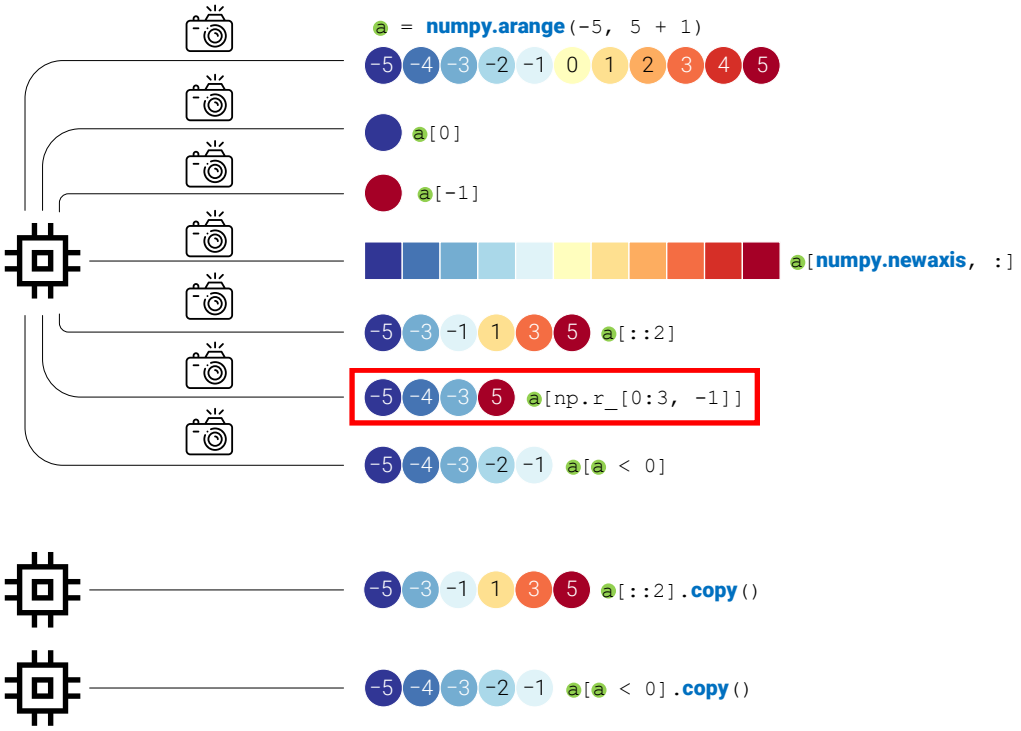


图 8. 视图，还是副本？ Bk1_Ch14_01.ipynb

在代码 1 示例中，首先 a 用 numpy.array()创建了一维数组 a；然后 b 创建了一个切片视图 s，该视图选择了数组 a 中的索引为 1、2 的元素，即第 2 个和第 3 个元素。

接下来，c 将视图中的第一个元素设置为 1000，这也会修改原始数组 a 中的元素。

然后，d 用 a.copy() 创建了一个整数数组索引副本 c，该副本选择了数组 a 中的索引为 1、3 元素，即第 2 个和第 3 个元素。

e 将副本 c 中的第一个元素设置为 888，但这不会修改原始数组 a 中的元素。

此外，我们可以使用 numpy.may_share_memory() 函数来判断两个数组是否共享内存。

在 NumPy 中，还有一些函数需要注意视图和副本的问题，比如 numpy.reshape()、numpy.transpose()、numpy.ravel()、numpy.flatten() 等等。这个话题非常重要，本书后文还会涉及。

```
# 创建一个一维数组
a a = np.array([1, 2, 3, 4, 5])

# 创建一个切片视图
b s = a[1:3]

# 修改视图中的数据
c s[0] = 1000

# 查看原始数组
print(a) # 输出：[1 0 3 4 5]

# 创建一个整数数组索引副本
d c = a[[1, 3]].copy()

# 修改副本中的数据
e c[0] = 888

# 查看原始数组
print(a) # 输出：[1 0 3 4 5]
print(c)
```

代码 1. 视图 vs 副本 | Bk1_Ch14_01.ipynb

14.4 二维数组索引、切片

取出单一元素

要取出二维 NumPy 数组中特定索引的元素，可以使用索引操作符 [] 来访问。可以将需要访问的元素的行索引和列索引作为参数传递给这个操作符。

图 9 所示为从二维数组 a 中取出单一元素，a[0, 0] 代表行索引为 0、列索引为 0 的元素，即第 1 行、第 1 列元素。

用 a[0][0]，我们也可以提取行索引为 0、列索引为 0 的元素。a[0]相当于先提取行索引为 0 的一维数组，然后再用第二个[0]从一维数组提取索引为 0 的元素。

请大家特别注意 a[[0],[0]]的结果为一维数组。

请大家自信分析图 9 其他示例。

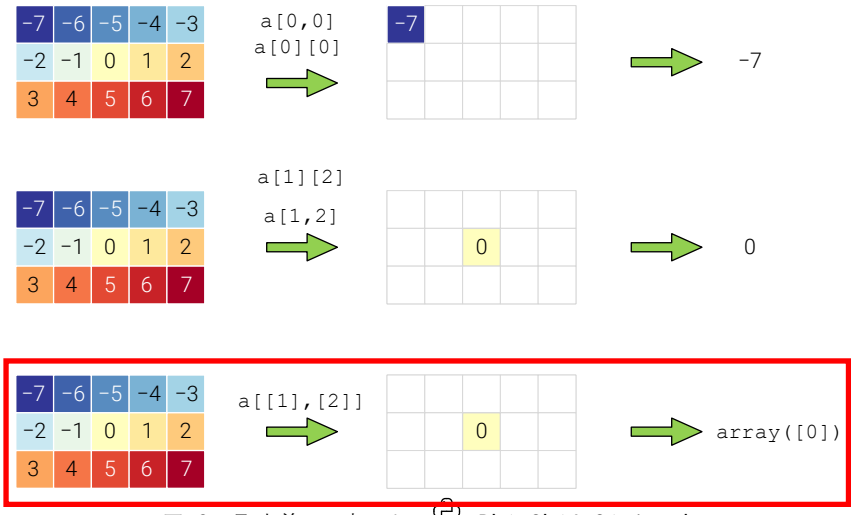


图 9. 取出单一元素 | Bk1_Ch14_01.ipynb

取出行

要取出二维 NumPy 数组中特定行的元素，也是使用索引操作符 [] 来访问。

我们可以将需要访问的行的索引作为第一个参数传递给这个操作符，用冒号 : 表示需要访问的列范围。

图 10 所示，取出第 0 行，只需 a[0]，结果为一维数组。而 a[[0], :] 取出第 0 行，结果为二维数组。

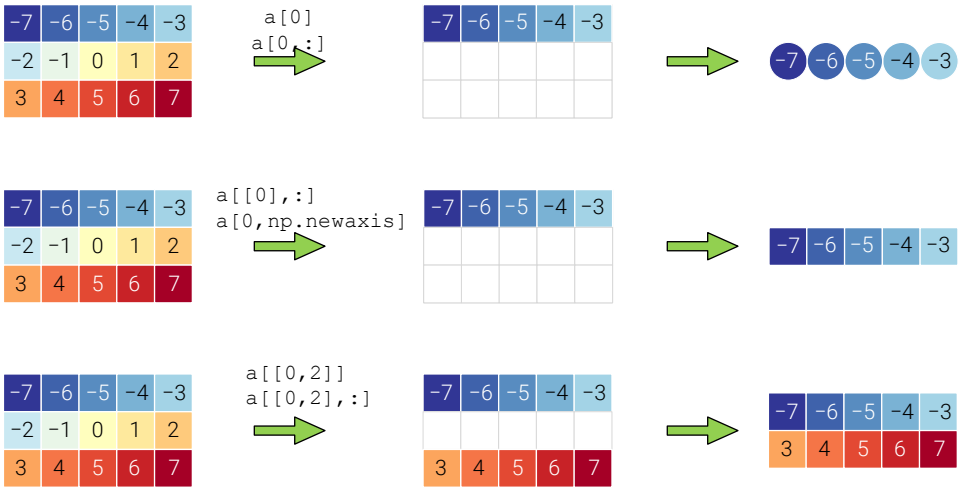


图 10. 取出行 | Bk1_Ch14_01.ipynb

取出列

如图 11 所示，我们也可以使用类似方法用取出特定列。请大家自行分析图 11 语句。

值得强调的是，本书前文提过，numpy.newaxis 是一个常用的 NumPy 函数，它用于在数组中添加一个新的维度。具体来说，numpy.newaxis 用于在现有数组的指定位置插入一个新的维度，从而改变数组的形状。

注意，在 NumPy 多维数组的索引和切片操作中，省略号 ... 可以用于代替多个连续冒号 :，从而简化操作。具体来说，省略号可以用于表示在某个维度上使用完整的切片范围。需要注意的是，省略号只能在索引或切片操作的开头、结尾或中间使用，而不能重复出现。

此外，当数组的维度比较大时，省略号可以显著提高代码的可读性和简洁性，因为它避免了写很多个冒号 : 的重复代码。

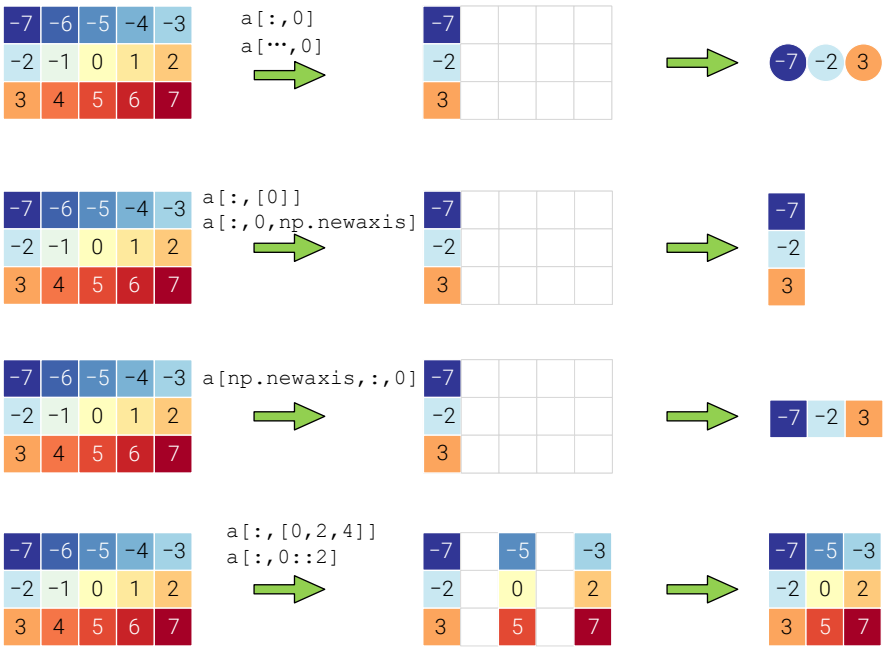


图 11. 取出列 | Bk1_Ch14_01.ipynb

图 12 所示为取出特定行列组合的方法，请大家自行学习。

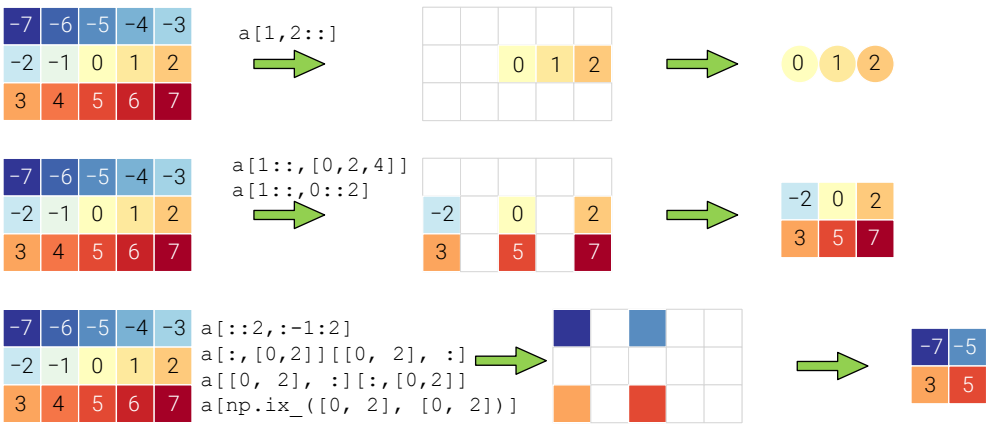


图 12. 取出特定的行列组合 | Bk1_Ch14_01.ipynb

值得一提的是，图 12 中，numpy.ix_() 是 NumPy 提供的一个函数，用于将多个一维索引数组转换为一个用于多维数组索引的元组。这个元组可以用于同时对多个维度进行索引，从而方便地选择数

组中的子集。使用 numpy.ix_() 可以让代码更加简洁和易读，避免了使用多个索引数组或切片来对多维数组进行索引的复杂性和难以理解的问题。在科学计算和数据分析中，使用 numpy.ix_() 可以方便地进行数据筛选和子集提取，提高代码效率和可读性。

布尔索引、切片

类似本章前文，二维数组也可以采用布尔索引、切片。举个例子，如图 13 所示，取出二维数组大于 0 的元素，结果为一元数组。本章配套代码还提供其他输出形式，请大家自行学习。

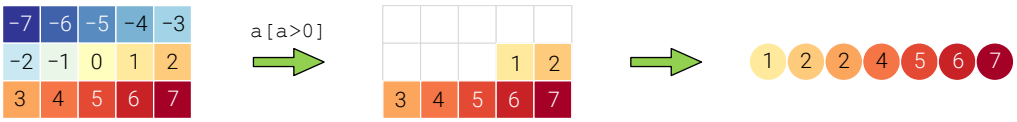


图 13. 取出大于 0 的元素 | Bk1_Ch14_01.ipynb

本章配套代码还介绍如何对三维数组进行索引、切片，也请大家自行学习。

请大家完成下面 3 道题目。

Q1. 创建一个一维数组，形状为 (10,)，用满足在 [-1, 1] 均匀分布随机数填充。切片操作提取前 5 个元素，并将结果倒序输出。

Q2. 创建一个二维数组，形状为 (3, 4)，用满足在 [-1, 1] 均匀分布随机数填充。使用切片操作选取其中的第一行和第三行。同时，使用切片操作取出第二、四列。

Q3. 创建一个三维数组，形状为 (4, 5, 6)，用满足在 [-1, 1] 均匀分布随机数填充。使用切片操作选取其中的 axis = 0、1 维度上的所有元素，以及 axis = 2 维度上的前两个元素。

* 题目不提供答案。

本章介绍的 NumPy Array 索引和切片操作和列表很相似。但是，以二维数组为例，提取 NumPy Array 的列则容易很多。

有关 NumPy Array 操作请大家务必搞清楚，我们在 Pandas 中还会用到类似的操作完成数据帧的索引和切片。当然，数据帧还有行列标签索引切片。