作者	**生姜 DrGinger**
脚本	**生姜 DrGinger**
视频	**崔崔 CuiCui**
开源学习资源	https://github.com/Visualize-ML
平台	https://www.youtube.com/@DrGinger_Jiang https://space.bilibili.com/3546865719052873 https://space.bilibili.com/513194466

1.1 什么是向量?

向量既有大小、又有方向，在物理、计算机科学等领域广泛应用。本节首先介绍了向量的基本概念，并通过速度、位移等实例，说明向量、标量的区别。然后，引入了向量场的概念，展示了风速、水流等现象如何用向量场描述。接着，讲解了向量的大小、方向，并通过箭头示意图强调其特性。随后，介绍了 RGB 颜色模型，说明颜色如何用三维向量表示，并阐述了向量的维数概念。接下来，讨论了行向量、列向量的区别，并通过矩阵视角分析它们的排列方式，及通过转置完成相互转换。此外，本节还介绍了 NumPy 如何创建一维、二维数组，演示了索引、切片操作，并通过 Python 代码展示行向量、列向量相互转置。

向量无处不在

向量 (vector) 是一种既有大小、又有方向的量。

几何上来看，向量是个**有向线段** (directed line segment)。

相比之下，**标量** (scalar) 就像一个单纯的数值，只有大小，没有方向。

汽车仪表盘上显示的车速，比如 60 千米/小时，是一个**标量**；因为它只表示速度的大小。

然而，如果加上行驶的方向，比如“以 60 千米/小时向正东行驶”，这个速度就变成了一个**向量**；因为它既包含了大小 (60 千米/小时)，也明确了方向 (正东)。

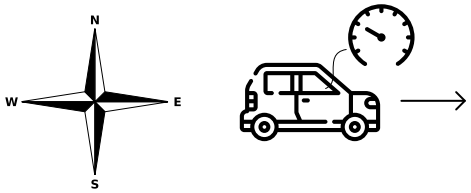


图 1．速度，标量和矢量

再举个例子，在徒步越野时，假设你距离目的地还有 5 千米。这个距离是一个**标量**，因为它只表示大小，并不涉及方向。

但如果我们再加上方向，例如“西南方向 5 千米”，那么这个量就变成了位移 (向量)。位移不仅告诉我们要走多远，还确定了了行进的方向。

生活中向量无处不在，风、水流、重力场、电磁场等现象，都可以用**向量场** (vector field) 来描述。向量场是一个数学描述，其中空间中每个点都关联一个向量，用来表示该点的某种物理量的大小和方向。

比如，某个具体位置来看，风不仅有速度的大小 (如每秒 10 米)，还有明确的方向 (如从北向南吹)。

河流中某一点处的水流也是如此，既有流速的大小，也有流动的方向。

这些特定位置的物理现象可以用单个向量描述。但当我们观察整个区域时，就会发现风速、流速在不同位置各不相同，从而形成一个向量场。

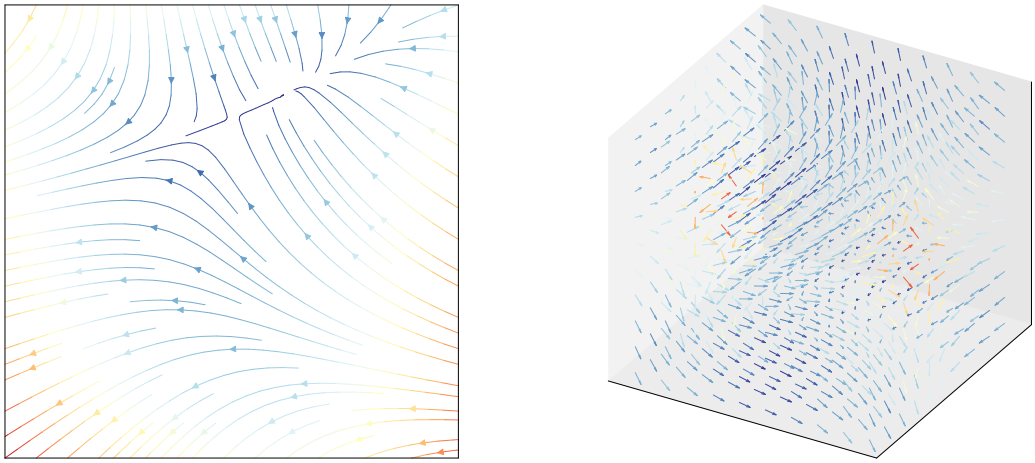


图 2．平面、空间向量场

向量：既有大小、又有方向

我们可以把向量想象成一支箭，它有两个重要的特性——大小，箭的长度；方向，箭的指向。

这也就是为什么物理学中向量常被称为矢量的原因。

如图 3 所示，箭的长度代表**向量大小** (magnitude of a vector)，也叫**向量长度** (length of a vector)。

简单来说，向量长度定义为**起点** (initial point, start point, tail, origin)、**终点** (terminal point, end point, head) 之间的直线距离。

箭头指向的方向代表**向量方向** (direction of a vector)。

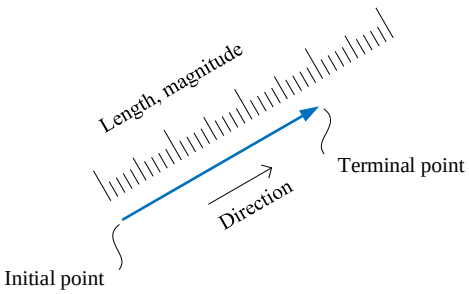


图 3．向量既有大小又有方向

向量大小是一个标量 (只有大小，没有方向)。

如图 4 所示，即使两个**向量大小**完全相同，只要它们的方向不同，它们就代表了不同的物理量。例如，一个向量可能指向西方，而另一个则指向东方；虽然它们的长度一样，但在描述力、速度等现象时，它们所起的作用却截然不同。

图 4 中，有的向量彼此**垂直** (perpendicular)，这意味着它们在各自的方向上是独立的，不会相互影响；而有的向量方向完全**相反** (opposite)，它们之间的相互作用可能会导致相互抵消。

另外，我们可以发现把这些平面上长度相同的向量起点放到一起，这些 (平面上的) 向量的终点在同一个**正圆** (circle) 上。

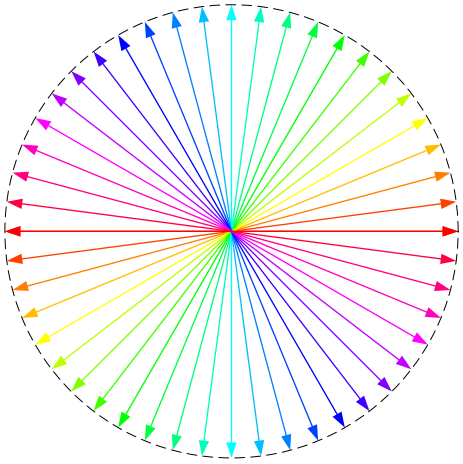


图 4．大小相同、但是方向不同的向量

RGB 颜色模型

在计算机图形学中，颜色也可以用向量表示。

例如，**RGB 颜色模型** (Red Green Blue color model) 中的颜色可以用 **3 维向量** (three-dimensional vector) 描述。**R** (red)、**G** (green)、**B** (blue) 每个**分量** (component) 代表一种颜色的强度。

向量维数是指描述该向量所需的独立**分量**的数量。比如，**2 维向量**有两个独立分量，**3 维向量**有三个独立分量。

在本书中，R、G、B 颜色的每个分量取值范围为 [0, 1]。

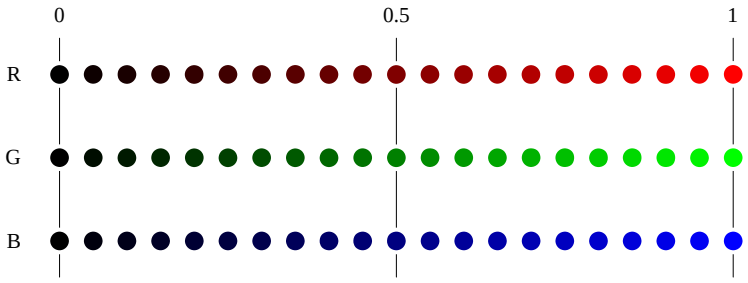


图 5. RGB 三原色分量取值

当 R = 1，而 G 和 B 均为 0 时，对应的色号 (1, 0, 0) 表示纯红色；类似地，纯绿色的色号为 (0, 1, 0)，纯蓝色的色号为 (0, 0, 1)。本书中，我们会用半角圆括号表示**元组** (tuple)。简单来说，元组是有序的元素序列。例如，坐标经常用**元组**来表示，在平面直角坐标系中可写作 (x_1, x_2)。我们会用方括号 [] 表示向量。

在某些实际应用中，R、G、B 颜色分量可能以 [0, 255] 区间内的整数表示，这是另一种常见的颜色编码方式。

⚠ 值得一提的是，RGB 是基于光的混合，而不是颜料的混合。

如图 6 所示，红光、绿光混合会产生黄光，这是因为光的叠加是加色混合，当红色和绿色的光同时作用于人眼时，我们感知到的就是黄色。

再如，蓝光、绿光混合会产生青色，而红光和蓝光混合则会产生品红色。

当红、绿、蓝三种光以最大强度混合时，就会产生白光，这是因为所有可见光谱的颜色都被叠加在一起。这种加色混合的原理广泛应用于显示器、电视和投影仪等设备中，通过调节红、绿、蓝三个通道的强度，可以呈现出丰富多彩的颜色。

本书后续会用 RGB 讲解各种线性代数概念。

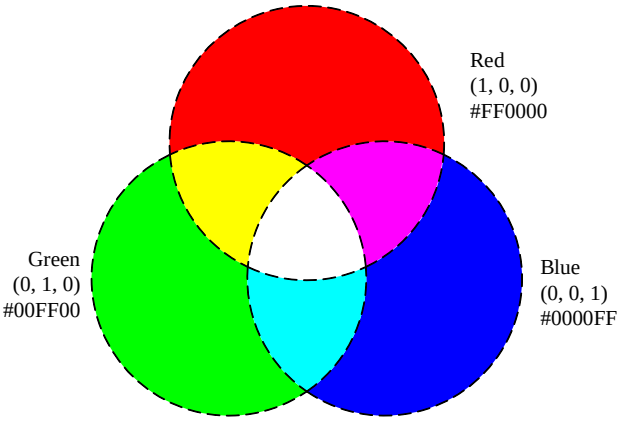


图 6. RGB 三原色混合

行向量

行向量 (row vector) 中的**分量** (component) 按一行排布，比如

$$\begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} \tag{1}$$

向量中**索引** (index) 为 i 的**分量**为 a_i。

索引 i 的取值范围是从 1 到 n 的正整数。

再次强调，**向量维数**是指该向量中分量数量；显然，上述向量为 n 维向量。

例如，一个 **2 维行向量**拥有 2 个分量，一个 **3 维行向量**则有 3 个分量。

$$\begin{bmatrix}1 & 2\end{bmatrix},\ \begin{bmatrix}1 & 2 & 3\end{bmatrix} \tag{2}$$

在机器学习中，一个样本的数据可以表示为一个行向量。

NumPy 创建一维数组

本书将在讲解线性代数概念的同时，介绍如何使用 Python 进行相关计算。本书特别依赖 NumPy；NumPy 是一个强大的数值计算库，专门用于处理多维数组和矩阵运算，并提供了丰富的线性代数函数。

NumPy **数组** (array) 是其核心数据结构，它比 Python 的**列表** (list) 更高效，支持向量化计算，使得数学运算更加简洁、快速且易读。

代码 1 这段代码首先导入了 NumPy 模块，并用一个包含 1 到 10 的列表构造了一维数组。接着，通过数组的属性获取了其维数和形状，然后展示了如何使用索引、切片操作来访问和提取数组中的元素。

如图 7 所示，NumPy 数组使用零索引，从 0 开始对数组的元素进行索引，这意味着数组的第 1 个元素在索引 0 位置上。然而，在数学中，通常我们从 1 开始索引。因此，在数学的视角下，向量的第 1 个元素的索引为 1。

值得注意的是，NumPy 中，一维数组最后一个分量的索引也可以用-1 表示；倒数第二个分量的索引可以是-2，以此类推。

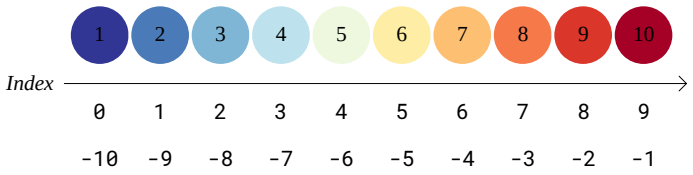


图 7. 一维 NumPy 数组的索引

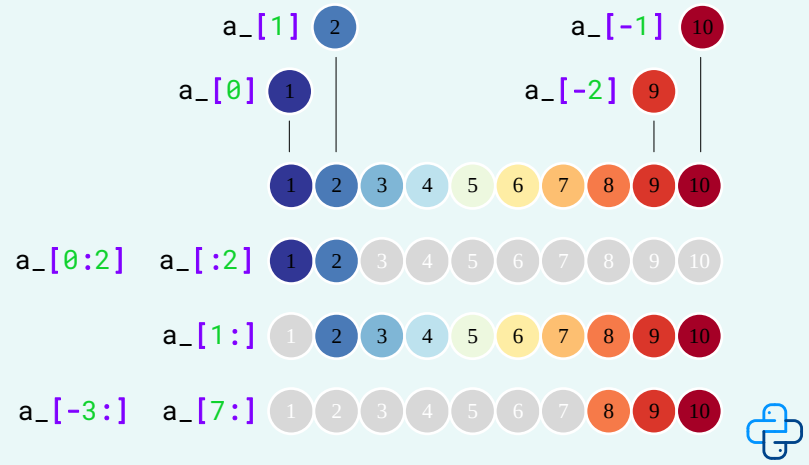
代码 1. 一维数组 | LA_01_01_01.ipynb

```
## 初始化
a import numpy as np

## 创建数组
b a_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
c a_ = np.array(a_list)
d # a_ = np.arange(1,11)
e dim = a_.ndim
f shape = a_.shape

## 索引
g a_[0]
h a_[1]
i a_[-1]
j a_[-2]

## 切片
k a_[0:2]
l a_[:2]
m a_[1:]
n a_[-3:]
o a_[7:]
```



下面让我们逐行解释代码 1。

a 的作用是导入 NumPy 模块，并将它命名为 np。这样，在后续代码中，就可以使用 np 来调用 NumPy 提供的函数和功能。

b 创建了一个 Python 列表，名字叫做 a_list，里面包含了从 1 到 10 的整数。列表是 Python 中的一种数据结构，用于存储一系列数据。

c 使用 NumPy 的 numpy.array() 函数将前面创建的列表 a_list 转换为一个 NumPy 数组，并赋值给变量 a_。NumPy 数组与 Python 列表类似，但提供了更多数学运算和操作功能。

d 这行代码前面的 # 表示这是注释，不会被执行。注释中的代码提供了另一种创建相同数组的方法：np.arange(1,11) 会生成从 1 到 10 (不包含 11) 的连续整数序列。

e 获取数组 a_的数组的“轴”数，并将结果存储在变量 dim 中。对于一维数组来说，ndim 的值为 1。

f 获取数组 a_的形状，即数组在每个“轴”上的大小。形状是一个元组 (tuple)，对于这个一维数组，shape 会返回 (10,)，表示数组中有 10 个元素。

g 取数组中索引为 0 的元素。由于索引从 0 开始，这里的元素是第一个数字 1。

h 取数组中索引为 1 的元素，即第二个数字 2。

i 取数组中最后一个元素。负数索引-1 表示倒数第一个元素，即数字 10。

j 取数组中倒数第二个元素 (索引为-2)。

k 取数组中从索引 0 到索引 2 (不包含索引 2) 的元素。结果是一个包含第一个和第二个元素的子数组，即 array([1, 2])。

l 中切片的开始索引省略时，默认从头开始，所以这行代码也返回 array([1, 2])。

m 中当切片的结束索引省略时，默认取到数组末尾，因此返回从索引 1 开始到最后的所有元素。

n 使用负数索引表示从数组末尾开始切片，-3: 表示取倒数第三个元素到最后一个元素，所以返回 array([8, 9, 10])。

从索引 7 开始取到最后的元素，即返回 array([8, 9, 10])。

注意，本书不会专门讲解 Python 编程基础，相关内容请大家参考《编程不难》；此外，本书也不会讲解数学、数据等复杂的可视化方案，相关内容请大家参考《可视之美》；两本书的草稿、代码等开源资源在 https://github.com/Visualize-ML。

列向量

列向量 (column vector) 中的分量按一列排列。

n 维列向量 $\boldsymbol{a}$，可以记作

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \tag{3}$$

不特殊说明，本书中向量都默认为列向量。本书中，列向量用小写、粗体、斜体字母表示，比如 $\boldsymbol{a}$、$\boldsymbol{b}$、$\boldsymbol{c}$、$\boldsymbol{u}$、$\boldsymbol{v}$、$\boldsymbol{x}$、$\boldsymbol{y}$。

总结来说，行向量、列向量是向量的两种基本表示形式，它们在数学、物理和计算机科学中都有广泛应用。它们的本质区别在于排列方式，而行向量和列向量的选择通常取决于具体的应用场景和计算需求。

RGB 颜色列向量

如图 8 所示，RGB 颜色中的红色、绿色、蓝色、**青色** (cyan)、**品红** (magenta)、**黄色** (yellow)、白色可以写成列向量。

白色对应的列向量为 $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$，我们管这个每个分量都是 1 的列向量叫做**全 1 列向量** (all-ones column vector)，本书记作 $\boldsymbol{1}$ (数字 1 的粗体、斜体)。

如果什么光都没有，我们就会看到黑色，对应列向量 $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$。我们管这个每个分量都是 0 的列向量叫做**零向量** (zero vector)，记作 $\boldsymbol{0}$ (数字 0 的粗体、斜体)。

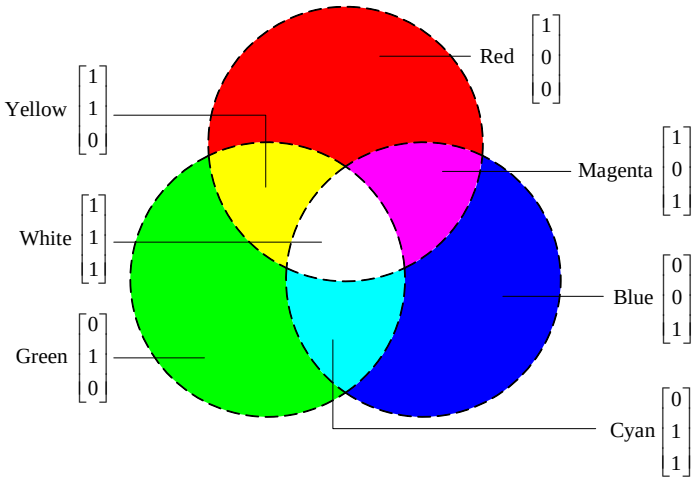


图 8. RGB 颜色的列向量

大家可能已经发现，红色 + 蓝色 = 品红，蓝色 + 绿色 = 青色，红色 + 绿色 = 黄色，红色 + 绿色 + 蓝色 = 白色；而这些运算从线性代数角度来看相当于向量加法。本章后续将介绍向量基本运算。

矩阵视角

为了区分行向量、列向量，本书中行向量会用上角标，比如 $\boldsymbol{a}^{(1)}$；列向量则用下角标，比如 $\boldsymbol{a}_1$。

比如，n 维行向量 $\boldsymbol{a}^{(1)}$

$$\boldsymbol{a}^{(1)} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \end{bmatrix} \tag{4}$$

实际上，如图 9 (a) 所示，$\boldsymbol{a}^{(1)}$为一行、多列的**矩阵** (matrix)。

简单来说，**矩阵**是由若干行、若干列排列成的矩形数组，其中每个分量都可以通过行索引、列索引唯一确定。

⚠ 注意，行索引在先，列索引在后。

比如，$\boldsymbol{a}^{(1)}$ 中分量的行索引都是 1，它们的列索引为从 1 到 n 的正整数。图 9 (a) 所示行向量对应的矩阵形状记作 $1 \times n$。

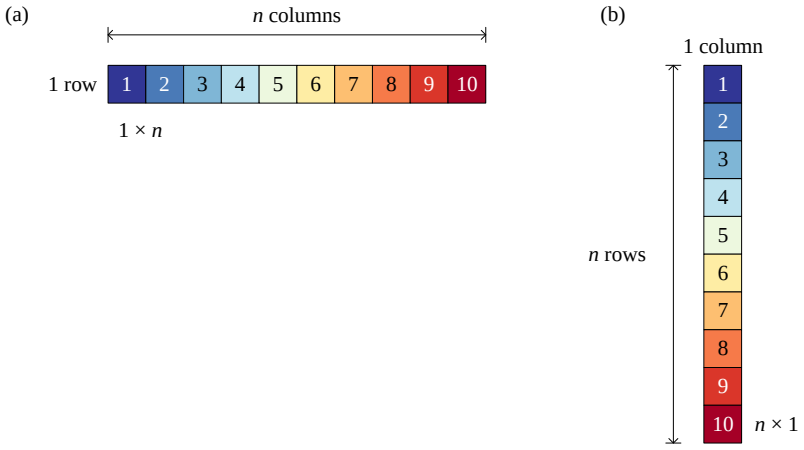


图 9. 从矩阵视角看行向量、列向量

如图 9 (b) 所示，n 维列向量 $\boldsymbol{a}_1$ 为

$$\boldsymbol{a}_1 = \begin{bmatrix} a_{1,1} \\ a_{2,1} \\ \vdots \\ a_{n,1} \end{bmatrix} \tag{5}$$

类似地，$\boldsymbol{a}_1$ 为多行、一列的矩阵。$\boldsymbol{a}_1$ 中分量的行索引为从 1 到 n 的正整数，它们的列索引都是 1。

图 9 (a) 所示行向量对应的矩阵形状记作 $n \times 1$。

转置 (transpose) 将矩阵的行、列进行互换。

如图 10 所示，行向量转置后变成列向量，列向量转置后变成行向量。

本书中，矩阵转置的记号用上角标 $^{\mathrm{T}}$ 表示，比如下例

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \quad \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \tag{6}$$

转置不改变行向量、列向量分量数量，也不改变分量之间的相对位置关系。

很多时候，为了减小行间距，本书会用转置来表示列向量，比如 $\boldsymbol{a} = [3, 4]^{\mathrm{T}}$。这个列向量偶尔也会记作 $\boldsymbol{a} = [3; 4]$，分号 ; 表示分行。

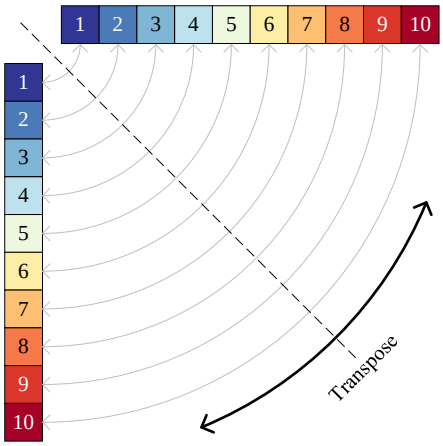


图 10. 行向量、列向量之间的转置变换

NumPy 创建二维数组

代码 2 利用 NumPy 创建二维数组，构造 1 行、10 列的行向量，并通过转置得到 10 行、1 列的列向量。示例展示了数组的索引和切片操作，如访问特定元素和提取子数组。

代码 2. 一维数组 | LA_01_01_02.ipynb

```
## 初始化
import numpy as np

## 创建行向量（二维）
row_vector = np.array([[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]])
a row_vector.ndim
  row_vector.shape

### 索引
  row_vector[0, 0]
b row_vector[0, 1]
  row_vector[0, -1]

### 切片
  row_vector[:, :2]
c row_vector[:, 1:]
  row_vector[:, -3:]

## 创建列向量（二维）

### 转置
d col_vector = row_vector.T

### 索引
  col_vector[0, 0]
e col_vector[1, 0]
  col_vector[-1, 0]

### 切片
  col_vector[:2, :]
f col_vector[1:, :]
  col_vector[-3:, :]
```

下面讲解这段代码。

a 使用 numpy.array() 函数将一个嵌套的列表转换成二维数组，其中内层列表包含 1 到 10 的数字，外层列表表示这个数组只有一行，因此它构成了一个 1×10 的行向量。row_vector.ndim 调用 row_vector 的 ndim 属性，用来获取数组的维数；由于数组是以二维形式存储，所以返回值为 2。row_vector.shape 调用 row_vector 的 shape 属性，返回一个元组 (1, 10)，表示该数组有 1 行、10 列，即数组的行数、列数。

b 中 row_vector[0, 0] 通过索引访问数组中第 1 行、第 1 列的元素 (注意 Python 中索引从 0 开始)，返回的值为 1，即行向量的第 1 个分量。

row_vector[0, 1] 访问 row_vector 中第 1 行、第 2 列的元素，返回值为 2，对应于行向量中的第 2 个分量。

row_vector[0, -1] 使用负索引来访问 row_vector 的元素，其中 -1 表示最后一列，因此返回值为 10，即行向量中的最后一个元素。

ⓒ 中 row_vector[:, :2] 使用切片操作，冒号 : 表示选取所有行，而 :2 表示选取每行中从索引 0 到 1 的元素 (不包含索引 2)，结果为一个子数组 array([[1, 2]])。注意，这个结果还是二维数组。

row_vector[:, 1:] 操作中，1: 表示从索引 1 开始一直到最后，冒号表示选取所有行。

row_vector[:, -3:] 使用切片选取每行中从倒数第三个元素到最后一个元素，-3: 表示从倒数第三列开始。

ⓓ 利用 .T 属性将行向量 row_vector 进行转置操作，得到一个列向量，即将原来一行多列的数组变换为多行一列的数组。

ⓔ 中 col_vector[0, 0] 通过索引访问转置后列向量的第 1 行、第 1 列的元素，返回值为 1，即列向量中的第一个分量。

col_vector[1, 0] 访问 col_vector 的第 2 行第 1 列的元素，返回值为 2，对应列向量中的第二个分量。

col_vector[-1, 0] 用负索引 -1 表示最后一行，访问转置后列向量的最后一个元素，返回值为 10，即列向量中的最后一个分量。

ⓕ 中 col_vector[:2, :] 使用切片操作，:2 表示选取前两行，而冒号 : 表示选取所有列，因此返回一个包含列向量前两个元素的子数组。这个子数组也是二维数组。

col_vector[1:, :] 使用切片操作，1: 表示从第 2 行开始一直到最后一行，选取所有列，因此返回从第二个元素到最后一个元素构成的子数组。

col_vector[-3:, :] 这一行代码的切片操作中，-3: 表示从倒数第三行开始选取到最后一行，冒号表示所有列，因此返回包含列向量最后三个元素的子数组。

请大家用 DeepSeek/ChatGPT 等工具完成本节如下习题。

Q1. 请安装 Anaconda，并学习使用 JupyterLab。

Q2. 请学习 Python 中列表 list、元组 tuple、变量 variable 等概念。

Q3. 定义一维 NumPy 数组 numpy.array([1, 2, 3, 4, 5, 6])，分别取出其中

► 奇数；

► 偶数。

Q4. 定义二维 NumPy 数组 numpy.array([[−3, −2, −1, 0, 1, 2, 3]])，分别取出其中

► 大于 0 的分量；

► 大于等于 0 的所有分量；

► 小于 2 的所有分量；

► 小于等于 2 的所有分量。

Q5. 请对一维数组 numpy.array([1, 2, 3, 4, 5, 6]) 进行转置运算，并解释结果。

Q6. 请对二维数组 numpy.array([[−3, −2, −1, 0, 1, 2, 3]]) 进行转置运算，并解释结果。

Q7. 请逐个学习、使用如下这些 NumPy 函数。

► numpy.linspace()

► numpy.zeros()

► numpy.ones()

► numpy.random.rand()

Q8. 请学习如何在 Matplotlib 中定义颜色。

https://matplotlib.org/stable/users/explain/colors/colors.html

Q9. 请 DeepSeek/ChatGPT 逐行注释下例，并学习绘制水流图。

https://matplotlib.org/stable/gallery/images_contours_and_fields/plot_streamplot.html

Q10. 请 DeepSeek/ChatGPT 逐行注释下例，并学习绘制三维箭头图。

https://matplotlib.org/stable/gallery/mplot3d/quiver3d.html