**### NumPy 快速入门：数组运算基础**

**欢迎来到 NumPy 的世界！NumPy (Numerical Python) 是 Python 中用于科学计算的核心库。它提供了一个高性能的多维数组对象 `ndarray`，以及用于处理这些数组的工具。**

**把它想象成 Python 列表的“超级赛亚人”版本，尤其是在处理大量数字时，它的速度和便利性是无与伦比的。**

**这篇文档将带你了解 NumPy 数组最基础、也最重要的运算。**

**\*\*首先，我们需要导入 numpy 库。按照惯例，我们通常给它一个别名 'np'，这样写起来更方便。\*\***

**[CODE\_START]**

**import numpy as np**

**# 我们可以打印出安装的版本号来确认导入成功**

**print(f"NumPy version installed: {np.\_\_version\_\_}")**

**[CODE\_END]**

**#### 1. 创建 NumPy 数组**

**创建数组最常见的方式就是从一个 Python 列表转换。**

**\*\*下面我们将分别从一个简单的一维列表和一个嵌套的二维列表来创建数组。\*\***

**[CODE\_START]**

**# 从一个简单的一维 Python 列表创建数组**

**python\_list = [10, 20, 30, 40, 50]**

**my\_array = np.array(python\_list)**

**print("这是一个一维 NumPy 数组:")**

**print(my\_array)**

**# 从一个嵌套的列表（二维）创建矩阵**

**python\_matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]**

**my\_matrix = np.array(python\_matrix)**

**print("\n这是一个二维 NumPy 数组（矩阵）:")**

**print(my\_matrix)**

**[CODE\_END]**

**NumPy 也提供了很多便捷的函数来快速创建特定类型的数组。**

**\*\*例如 `arange`, `zeros`, 和 `ones`。\*\***

**[CODE\_START]**

**# 使用 np.arange() 创建一个类似 range 的序列**

**# 参数是：起始（包含），结束（不包含），步长**

**range\_array = np.arange(0, 10, 2)**

**print("使用 arange 创建的数组:", range\_array)**

**# 使用 np.zeros() 创建一个全是0的数组**

**# 需要提供一个元组(tuple)来指定形状，比如 (2, 3) 代表2行3列**

**zeros\_array = np.zeros((2, 3))**

**print("\n一个 2x3 的全零矩阵:\n", zeros\_array)**

**# 使用 np.ones() 创建一个全是1的数组**

**ones\_array = np.ones((3, 2))**

**print("\n一个 3x2 的全一矩阵:\n", ones\_array)**

**[CODE\_END]**

**#### 2. NumPy 的核心优势：向量化运算**

**这就是 NumPy 如此强大的原因！对于数组的数学运算，NumPy 会自动地应用到数组的\*\*每一个元素\*\*上，而不需要我们手写 `for` 循环。这个特性叫做“向量化”(Vectorization)。**

**\*\*来看一个简单的例子，我们想给数组中的每个数字都加上5。\*\***

**[CODE\_START]**

**# 准备一个数组**

**data = np.arange(1, 6) # 创建数组 [1 2 3 4 5]**

**print("原始数组:", data)**

**# 直接对整个数组进行数学运算**

**print("数组 + 5 =", data + 5)**

**print("数组 - 1 =", data - 1)**

**print("数组 \* 10 =", data \* 10)**

**print("数组 / 2 =", data / 2)**

**print("数组 \*\* 2 =", data \*\* 2) # 每个元素求平方**

**[CODE\_END]**

**这种向量化运算也适用于两个形状相同的数组之间。**

**[CODE\_START]**

**array\_a = np.array([10, 20, 30, 40])**

**array\_b = np.array([1, 2, 3, 4])**

**# 两个数组相加，对应位置的元素会相加**

**print("A + B =", array\_a + array\_b)**

**# 两个数组相乘**

**print("A \* B =", array\_a \* array\_b)**

**[CODE\_END]**

**#### 3. 数组索引与切片**

**获取数组中的数据和 Python 列表非常相似。**

**[CODE\_START]**

**# 创建一个测试数组**

**test\_array = np.arange(100, 110) # [100, 101, ..., 109]**

**print("测试数组:", test\_array)**

**# 获取索引为 3 的元素**

**print("索引为 3 的元素:", test\_array[3])**

**# 获取索引从 2 到 5 (不包含5) 的片段**

**print("从 2 到 5 的切片:", test\_array[2:5])**

**# 对于多维数组，用逗号分隔不同维度的索引**

**matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])**

**print("\n测试矩阵:\n", matrix)**

**# 获取第1行，第2列的元素（行和列都从0开始计数）**

**# 所以是数字 6**

**print("第 1 行, 第 2 列的元素:", matrix[1, 2])**

**# 获取整个第0行**

**print("第 0 行:", matrix[0, :])**

**# 获取整个第1列**

**print("第 1 列:", matrix[:, 1])**

**[CODE\_END]**

**#### 4. 常用统计函数**

**NumPy 提供了大量快速的统计函数。**

**[CODE\_START]**

**# 创建一个包含随机数的数组用于统计**

**# np.random.randint(low, high, size)**

**stats\_array = np.random.randint(0, 100, 10) # 10个0到99之间的随机整数**

**print("用于统计的随机数组:", stats\_array)**

**# 计算总和**

**print("总和:", stats\_array.sum()) # 或者 np.sum(stats\_array)**

**# 计算平均值**

**print("平均值:", stats\_array.mean())**

**# 找出最大值**

**print("最大值:", stats\_array.max())**

**# 找出最小值**

**print("最小值:", stats\_array.min())**

**[CODE\_END]**

**恭喜你！你已经掌握了 NumPy 最核心的基础知识。利用这些简单的操作，已经可以解决许多数据分析和科学计算中的问题了。继续探索，你会发现 NumPy 更多的强大功能！**