



中山大学计算机学院

人工智能

本科生实验报告

(2024 学年春季学期)

课程名称: Artificial Intelligence

教学班级	202320346	专业(方向)	计算机科学与技术
学号	22320107	姓名	饶鉴晟

一、实验题目

矩阵加法，乘法给定两个 $n \times n$ 的整型矩阵 A 和 B ，写两个函数 `MatrixAdd` 和 `MatrixMu1`，分别得出这两个矩阵加法和乘法的结果。

两个矩阵的数据类型为嵌套列表，即 `list[list]`，且满足 `len(list)==n`，注意不要打乱原矩阵 A 和 B 中的数据。

二、实验内容

1. 算法原理

代码需要实现两个基本的线性代数操作：矩阵加法和矩阵乘法，使用 Python 语言实现。

(1) 矩阵加法：

原理：矩阵加法是指两个形状（行数和列数）相同的矩阵对应位置的元素相加。结果是一个新的矩阵，其形状与原矩阵相同，每个位置的元素是原两个矩阵对应位置元素的和。

核心公式：

如果有两个矩阵 A 和 B ，它们都是 $m \times n$ 的（即，有 m 行， n 列），那么它们的和 $C = A + B$ 也是一个 $m \times n$ 的矩阵，其中 $C_{ij} = A_{ij} + B_{ij}$ ，（ i 表示行索引， j 表示列索引）。

(2) 矩阵乘法

原理：矩阵乘法涉及到两个矩阵——一个被称为左矩阵 A ，另一个称为右矩阵 B 。矩阵乘法的结果是一个新的矩阵 C ，其中 C 的每个元素是通过取 A 的行与 B 的列对应元素的乘积和得到的。对于结果矩阵中的每个元素，其值计算如下：选择 A 的一行和 B 的一列，将这一行与这一列中对应元素的乘积相加，得到的和就是 C 中对应位置的值。



核心公式:

给定矩阵 A (大小为 $m \times n$) 和矩阵 B (大小为 $n \times p$), 它们的乘积 $C = AB$ 是一个 $m \times p$ 的矩阵, 其中每个元素由下式给出:

$$C_{ij} = \sum_{k=1}^n A_{ik} \cdot B_{kj}$$

这里, C_{ij} 是 C 的第 i 行第 j 列的元素, A_{ik} 是 A 的第 i 行第 k 列的元素, B_{kj} 是 B 的第 k 行第 j 列的元素。求和是在 k 上进行的, 意味着对于 A 的第 i 行的每个元素和 B 的第 j 列的每个元素, 它们的乘积会被加起来以得到 C_{ij} 。

(3) 实现细节:

矩阵加法: 通过嵌套列表推导式, 实现两个矩阵的逐元素加法。

矩阵乘法: 首先, 为每个目标元素初始化一个累加器。然后, 通过三层嵌套循环 (外层循环遍历结果矩阵的行, 中层循环遍历结果矩阵的列, 内层循环执行点乘并累加), 计算出结果矩阵的每个元素。

2. 伪代码

Algorithm 1: MatrixAdd: Adding two matrices

Data: Two matrices A and B

Result: The sum of matrices A and B

/* Matrix Addition */

```

1 n ← length of A;
2 Initialize matrix C as an empty list;
3 for cnt ← 0 to n-1 do
4   Append to C the sum of corresponding elements from A[cnt] and
   B[cnt];
5 end
6 return C;
```

Algorithm 2: MatrixMul: Multiplying two matrices

Data: Two matrices A and B

Result: The product of matrices A and B

/* Matrix Multiplication */

```

1 Initialize matrix C as an empty list;
2 n ← length of A;
3 for i ← 0 to n-1 do
4   Initialize tmp as an empty list;
5   for j ← 0 to n-1 do
6     value ← 0;
7     for cnt ← 0 to n-1 do
8       value ← value + (A[i][cnt] * B[cnt][j]);
9     end
10    Append value to tmp;
11  end
12  Append tmp to C;
13 end
14 return C;
```

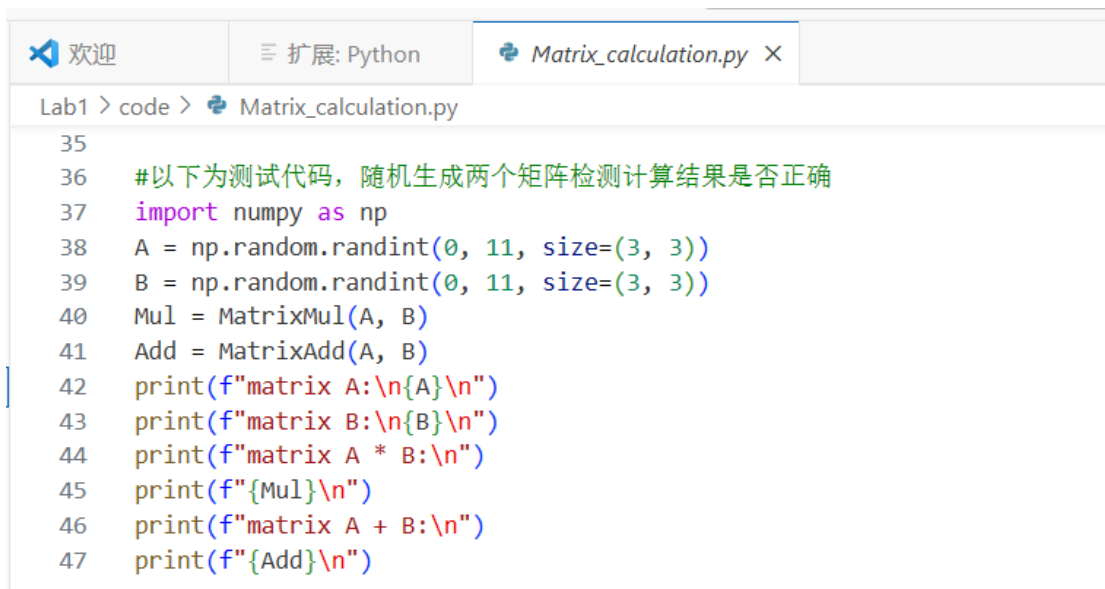


3. 关键代码展示（带注释）

```
def MatrixAdd(A, B):  
    """  
    实现两个矩阵的加法  
    :param A: list[list[int]] 第一个矩阵  
    :param B: list[list[int]] 第二个矩阵  
    :return: list[list[int]] 矩阵加法的结果  
    """  
    n = len(A)  
    C = []  
    for cnt in range(n):  
        # 对每一行的对应元素进行加法操作，并添加到结果矩阵中  
        C.append([i + j for i, j in zip(A[cnt], B[cnt])])  
    return C # 返回矩阵加法的结果  
  
def MatrixMul(A, B):  
    """  
    实现两个矩阵的乘法  
    :param A: list[list[int]] 第一个矩阵  
    :param B: list[list[int]] 第二个矩阵  
    :return: list[list[int]] 矩阵乘法的结果  
    """  
    C = []  
    tmp = []  
    value = 0  
    n = len(A)  
    for i in range(n):  
        tmp = []  
        for j in range(n): # 遍历结果矩阵的每一列  
            value = 0  
            for cnt in range(n): # 进行点乘计算  
                value += A[i][cnt] * B[cnt][j]  
            tmp.append(value) # 将点乘结果添加到临时列表中  
        C.append(tmp) # 将临时列表添加到结果矩阵中  
    return C
```

三、实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）



```
35
36 #以下为测试代码，随机生成两个矩阵检测计算结果是否正确
37 import numpy as np
38 A = np.random.randint(0, 11, size=(3, 3))
39 B = np.random.randint(0, 11, size=(3, 3))
40 Mul = MatrixMul(A, B)
41 Add = MatrixAdd(A, B)
42 print(f"matrix A:\n{A}\n")
43 print(f"matrix B:\n{B}\n")
44 print(f"matrix A * B:\n")
45 print(f"{Mul}\n")
46 print(f"matrix A + B:\n")
47 print(f"{Add}\n")
```

测试程序输入样例



```
问题 输出 调试控制台 终端 端口 1 注释
/home/codespace/.python/current/bin/python3 /workspaces/AI_2024/Lab1/code/Matrix_calculation.py
@JasonW41k3r →/workspaces/AI_2024 (main) $ /home/codespace/.python/current/bin/python3 /workspaces/AI_2024/Lab1/
matrix A:
[[ 2  1  2]
 [ 6  0  1]
 [10  5  9]]

matrix B:
[[ 5  1 10]
 [ 2  4  9]
 [ 9  4  0]]

matrix A * B:
[[30, 14, 29], [39, 10, 60], [141, 66, 145]]

matrix A + B:
[[7, 2, 12], [8, 4, 10], [19, 9, 9]]
@JasonW41k3r →/workspaces/AI_2024 (main) $
```

测试程序输出样例

如图所示，程序正确输出了两个随机生成的矩阵的加法和乘法运算结果，而且生成的结果符合题目要求的嵌套列表类型，即 `list[list]`。



四、 参考资料

[1] waq127520. (2020) numpy.random 随机数组详解.CSDN. 检索于 2024 年 3 月 10 日, <https://blog.csdn.net/waq127520/article/details/105497440>.