Python程序设计——第三次实验报告

2311095 宋卓伦　计算机科学与技术 2024.09.27

一、实验目的

利用函数和模块编写程序，掌握结构化程序设计方法及多脚本文件程序的编写方法，熟练运用函数和模块简化问题求解过程，了解团队协作开发程序的方法。

二、实验描述

编写矩阵运算函数库(保存在一个单独的脚本文件中，如mymatrix.py)，至少支持矩阵的乘积、哈达玛积和转置运算(注意需要能够判断传入参数矩阵是否符合计算规则要求)。编写测试程序(如main.py)，对矩阵运函数进行测试，要求矩阵中的元素可根据指定的矩阵形状随机生成、不需要人工输入。

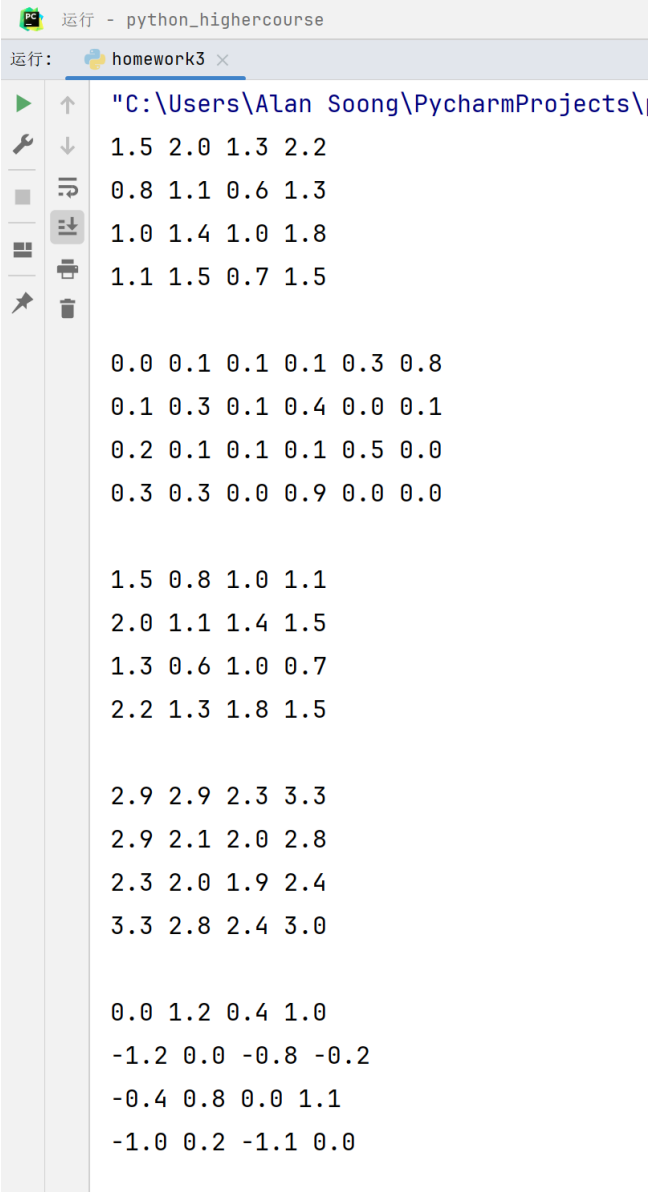
三、实验代码

```python
#martrix.py文件：

#矩阵乘法：
def multiply(A, B):

    #判断是否复符合做乘法的要求
    if len(A[0]) != len(B): return None

    #定义一个空矩阵，后面计算为其赋值
    C = [[0 for j in range(len(B[0]))] for i in range(len(A))]
    for i in range(len(A)):
        for j in range(len(B[0])):
            for k in range(len(B)):
                C[i][j] += A[i][k] * B[k][j]

    return C

#矩阵的Hadamard乘法：
def Hadamard(A,B):

    #判断是否复符合做乘法的要求
    if len(A) != len(B) | len(A[0]) != len(B[0]): return None

    #定义一个空矩阵，后面计算为其赋值
    C = [[0 for j in range(len(B[0]))] for i in range(len(A))]
    for i in range(len(A)):
        for j in range(len(B[0])):
            C[i][j] = A[i][j] * B[i][j]

    return C

```

```python
#矩阵转置：
def switch(A):

    #定义一个空矩阵为其赋值
    C = [[0 for j in range(len(A))] for i in range(len(A[0]))]
    for i in range(len(A[0])):
        for j in range(len(A)):
            C[i][j] = A[j][i]

    return C

#矩阵加法：
def add(A,B):

    #定义一个空矩阵为其赋值
    C = [[0 for j in range(len(A))] for i in range(len(A[0]))]
    for i in range(len(A[0])):
        for j in range(len(A)):
            C[i][j] = A[i][j] + B[i][j]

    return C

#矩阵减法：
def minus(A,B):

    #定义一个空矩阵为其赋值
    C = [[0 for j in range(len(A))] for i in range(len(A[0]))]
    for i in range(len(A[0])):
        for j in range(len(A)):
            C[i][j] = A[i][j] - B[i][j]

    return C

#矩阵的数乘
def num_mul(A, p):

    #定义一个空矩阵为其赋值
    C = [[0 for j in range(len(A[0]))] for i in range(len(A))]
    for i in range(len(self.A)):
        for j in range(len(self.A[0])):
            C[i][j] = A[i][j] * p

    return C

#封装一个矩阵的打印函数：
def print_matrix(C):

    for i in range(len(C)):
        for j in range(len(C[0])):
            #保留一位小数
            print('%.1f' % C[i][j], end=' ')
        print()

```

```python
#主函数的程序.py：
```

```
2
3  import numpy as np
4  import martrix as mt
5
6  #生成两个随机的给定大小的矩阵：
7  A = np.random.rand(4, 6)
8  B = np.random.rand(6, 4)
9
10 #计算：
11 C = mt.multiply(A, B)
12 E = mt.Hadamard(A, mt.switch(B))
13
14 #打印工作：
15 mt.print_matrix(C)
16 print()
17 mt.print_matrix(E)
18 print()
19 D = mt.switch(C)
20 mt.print_matrix(D)
21 print()
22 mt.print_matrix(mt.add(C,D))
23 print()
24 mt.print_matrix(mt.minus(C,D))
25
```



四、实验反思

遇到的问题1：矩阵的换行存在输出问题；

解决过程：将print函数中的end默认参数换成“”或者“ ”即可，从而正确换行；

遇到的问题2：若矩阵大小没有给定，如何判断矩阵乘法是否可以进行，不可以的话如何处理；

解决过程：无法进行矩阵乘法的将矩阵返回空，然后再打印之前判断是否为空，非空才可打印

```python
#修正的martrix.py文件，可以执行判断，错误则返回Error

def multiply(A, B):

    #判断是否符合计算条件
    if (A is None) | (B is None): return None
    if len(A[0]) != len(B): return None
    C = [[0 for j in range(len(B[0]))] for i in range(len(A))]

    for i in range(len(A)):
        for j in range(len(B[0])):
            for k in range(len(B)):
```

```
                C[i][j] += A[i][k] * B[k][j]

    return C


def Hadamard(A, B):

    #判断是否符合计算条件
    if (A is None) | (B is None): return None
    if len(A) == len(B) & len(A[0]) == len(B[0]):
        C = [[0 for j in range(len(B[0]))] for i in range(len(A))]
        for i in range(len(A)):
            for j in range(len(B[0])):
                C[i][j] = A[i][j] * B[i][j]
        return C
    else:
        return None


def switch(A):

    #判断是否符合计算条件
    if A is None: return None
    C = [[0 for j in range(len(A))] for i in range(len(A[0]))]
    for i in range(len(A[0])):
        for j in range(len(A)):
            C[i][j] = A[j][i]

    return C


def add(A, B):

    #判断是否符合计算条件
    if (A is None) | (B is None): return None
    if len(A) != len(B) | len(A[0]) != len(B[0]): return None
    C = [[0 for j in range(len(A))] for i in range(len(A[0]))]
    for i in range(len(A[0])):
        for j in range(len(A)):
            C[i][j] = A[i][j] + B[i][j]

    return C


def minus(A, B):

    #判断是否符合计算条件
    if (A is None) | (B is None): return None
    if len(A) != len(B) | len(A[0]) != len(B[0]): return None
    C = [[0 for j in range(len(A))] for i in range(len(A[0]))]
    for i in range(len(A[0])):
        for j in range(len(A)):
            C[i][j] = A[i][j] - B[i][j]

    return C

```

```
69
70  def print_matrix(C):
71
72      #判断是否符合计算条件
73      if C is not None:
74          for i in range(len(C)):
75              for j in range(len(C[0])):
76                  print('%.1f' % C[i][j], end=' ')
77              print()
78      else:
79          print('Error!')
80      print()
81
```

```
1   #主函数程序.py文件
2
3   import numpy as np
4   import martrix as mt
5   import random
6
7   #生成随机矩阵，随机大小，随机内容
8   nrA = random.randint(1, 5)
9   ncA = random.randint(1, 5)
10  A = np.random.rand(nrA,ncA)
11  nrB = random.randint(1, 5)
12  ncB = random.randint(1, 5)
13  B = np.random.rand(nrB,ncB)
14
15  #打印矩阵
16  mt.print_matrix(A)
17  mt.print_matrix(B)
18
19  C = mt.multiply(A, B)
20  mt.print_matrix(C)
21  D = mt.switch(C)
22  mt.print_matrix(D)
23
24  E = mt.Hadamard(A, mt.switch(B))
25  mt.print_matrix(E)
26
27  X = mt.add(C,D)
28  mt.print_matrix(X)
29
30  Y = mt.minus(C,D)
31  mt.print_matrix(Y)
32
```

运行结果如图：（左一左二为A、B矩阵，左三左四以及中间三个是对应的运算结果，右图是含有错误的结果）

```
0.7 0.2 0.1 0.6
0.2 0.3 0.5 0.9
0.8 0.7 0.2 0.2
0.1 0.9 0.1 0.8

0.6 0.5 0.2 0.6
1.0 0.7 0.4 0.7
0.3 0.6 0.6 0.8
0.8 0.3 0.7 0.3

1.0 0.7 0.6 0.7
1.2 0.9 1.0 0.9
1.4 1.1 0.7 1.1
1.6 1.0 1.0 1.0

1.0 1.2 1.4 1.6
0.7 0.9 1.1 1.0
0.6 1.0 0.7 1.0
0.7 0.9 1.1 1.0
```

```
0.4 0.2 0.0 0.4
0.1 0.2 0.3 0.3
0.1 0.3 0.1 0.1
0.0 0.7 0.1 0.2

2.1 1.9 2.0 2.3
1.9 1.8 2.1 1.9
2.0 2.1 1.4 2.2
2.3 1.9 2.2 2.0

0.0 -0.5 -0.7 -0.8
0.5 0.0 -0.0 -0.1
0.7 0.0 0.0 0.1
0.8 0.1 -0.1 0.0
```

```
0.8 0.5
0.9 0.3
0.6 0.9
0.3 0.7

0.3 0.7 0.9 0.2 0.3
0.3 0.0 0.1 0.7 0.2

0.4 0.6 0.7 0.5 0.4
0.4 0.6 0.8 0.4 0.4
0.5 0.5 0.6 0.7 0.4
0.3 0.2 0.3 0.5 0.2

0.4 0.4 0.5 0.3
0.6 0.6 0.5 0.2
0.7 0.8 0.6 0.3
0.5 0.4 0.7 0.5
0.4 0.4 0.4 0.2

Error!

Error!

Error!
```

将此判断操作贯穿所有函数的内部中，这种操作对于任意大小的矩阵都可以进行判断，可以运算才能够输出正确的结果，保证了代码的健壮性。