

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)，对矩阵运算函数进行测试，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

```
1  #martrix.py文件:
2
3  #矩阵乘法:
4  def multiply(A, B):
5
6      #判断是否复符合做乘法的要求
7      if len(A[0]) != len(B): return None
8
9      #定义一个空矩阵，后面计算为其赋值
10     C = [[0 for j in range(len(B[0]))] for i in range(len(A))]
11     for i in range(len(A)):
12         for j in range(len(B[0])):
13             for k in range(len(B)):
14                 C[i][j] += A[i][k] * B[k][j]
15
16     return C
17
18 #矩阵的Hadamard乘法:
19 def Hadamard(A,B):
20
21     #判断是否复符合做乘法的要求
22     if len(A) != len(B) | len(A[0]) != len(B[0]): return None
23
24     #定义一个空矩阵，后面计算为其赋值
25     C = [[0 for j in range(len(B[0]))] for i in range(len(A))]
26     for i in range(len(A)):
27         for j in range(len(B[0])):
28             C[i][j] = A[i][j] * B[i][j]
29
30     return C
31
```

```

32 #矩阵转置:
33 def switch(A):
34
35     #定义一个空矩阵为其赋值
36     C = [[0 for j in range(len(A))] for i in range(len(A[0]))]
37     for i in range(len(A[0])):
38         for j in range(len(A)):
39             C[i][j] = A[j][i]
40
41     return C
42
43 #矩阵加法:
44 def add(A,B):
45
46     #定义一个空矩阵为其赋值
47     C = [[0 for j in range(len(A))] for i in range(len(A[0]))]
48     for i in range(len(A[0])):
49         for j in range(len(A)):
50             C[i][j] = A[i][j] + B[i][j]
51
52     return C
53
54 #矩阵减法:
55 def minus(A,B):
56
57     #定义一个空矩阵为其赋值
58     C = [[0 for j in range(len(A))] for i in range(len(A[0]))]
59     for i in range(len(A[0])):
60         for j in range(len(A)):
61             C[i][j] = A[i][j] - B[i][j]
62
63     return C
64
65 #矩阵的数乘
66 def num_mul(A, p):
67
68     #定义一个空矩阵为其赋值
69     C = [[0 for j in range(len(A[0]))] for i in range(len(A))]
70     for i in range(len(self.A)):
71         for j in range(len(self.A[0])):
72             C[i][j] = A[i][j] * p
73
74     return C
75
76 #封装一个矩阵的打印函数:
77 def print_matrix(C):
78
79     for i in range(len(C)):
80         for j in range(len(C[0])):
81             #保留一位小数
82             print('%.1f' % C[i][j], end=' ')
83         print()
84

```

```
2
3 import numpy as np
4 import matrix 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
```

```
运行 - python_highercourse
运行: homework3 x
"C:\Users\Alan Soong\PycharmProjects\
1.5 2.0 1.3 2.2
0.8 1.1 0.6 1.3
1.0 1.4 1.0 1.8
1.1 1.5 0.7 1.5

0.0 0.1 0.1 0.1 0.3 0.8
0.1 0.3 0.1 0.4 0.0 0.1
0.2 0.1 0.1 0.1 0.5 0.0
0.3 0.3 0.0 0.9 0.0 0.0

1.5 0.8 1.0 1.1
2.0 1.1 1.4 1.5
1.3 0.6 1.0 0.7
2.2 1.3 1.8 1.5

2.9 2.9 2.3 3.3
2.9 2.1 2.0 2.8
2.3 2.0 1.9 2.4
3.3 2.8 2.4 3.0

0.0 1.2 0.4 1.0
-1.2 0.0 -0.8 -0.2
-0.4 0.8 0.0 1.1
-1.0 0.2 -1.1 0.0
```

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

```
1  #修正的matrix.py文件，可以执行判断，错误则返回Error
2
3  def multiply(A, B):
4
5      #判断是否符合计算条件
6      if (A is None) | (B is None): return None
7      if len(A[0]) != len(B): return None
8      C = [[0 for j in range(len(B[0]))] for i in range(len(A))]
9
10     for i in range(len(A)):
11         for j in range(len(B[0])):
12             for k in range(len(B)):
```

```
13         C[i][j] += A[i][k] * B[k][j]
14
15     return C
16
17
18 def Hadamard(A, B):
19
20     #判断是否符合计算条件
21     if (A is None) | (B is None): return None
22     if len(A) == len(B) & len(A[0]) == len(B[0]):
23         C = [[0 for j in range(len(B[0]))] for i in range(len(A))]
24         for i in range(len(A)):
25             for j in range(len(B[0])):
26                 C[i][j] = A[i][j] * B[i][j]
27         return C
28     else:
29         return None
30
31
32 def switch(A):
33
34     #判断是否符合计算条件
35     if A is None: return None
36     C = [[0 for j in range(len(A))] for i in range(len(A[0]))]
37     for i in range(len(A[0])):
38         for j in range(len(A)):
39             C[i][j] = A[j][i]
40
41     return C
42
43
44 def add(A, B):
45
46     #判断是否符合计算条件
47     if (A is None) | (B is None): return None
48     if len(A) != len(B) | len(A[0]) != len(B[0]): return None
49     C = [[0 for j in range(len(A))] for i in range(len(A[0]))]
50     for i in range(len(A[0])):
51         for j in range(len(A)):
52             C[i][j] = A[i][j] + B[i][j]
53
54     return C
55
56
57 def minus(A, B):
58
59     #判断是否符合计算条件
60     if (A is None) | (B is None): return None
61     if len(A) != len(B) | len(A[0]) != len(B[0]): return None
62     C = [[0 for j in range(len(A))] for i in range(len(A[0]))]
63     for i in range(len(A[0])):
64         for j in range(len(A)):
65             C[i][j] = A[i][j] - B[i][j]
66
67     return C
68
```

```

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 matrix 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	0.4 0.2 0.0 0.4	0.8 0.5
0.2 0.3 0.5 0.9		0.9 0.3
0.8 0.7 0.2 0.2	0.1 0.2 0.3 0.3	0.6 0.9
0.1 0.9 0.1 0.8	0.1 0.3 0.1 0.1	0.3 0.7
	0.0 0.7 0.1 0.2	
0.6 0.5 0.2 0.6		0.3 0.7 0.9 0.2 0.3
1.0 0.7 0.4 0.7		0.3 0.0 0.1 0.7 0.2
0.3 0.6 0.6 0.8	2.1 1.9 2.0 2.3	
0.8 0.3 0.7 0.3	1.9 1.8 2.1 1.9	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
1.0 0.7 0.6 0.7	2.0 2.1 1.4 2.2	
1.2 0.9 1.0 0.9	2.3 1.9 2.2 2.0	0.4 0.4 0.5 0.3
1.4 1.1 0.7 1.1		0.6 0.6 0.5 0.2
1.6 1.0 1.0 1.0		0.7 0.8 0.6 0.3
	0.0 -0.5 -0.7 -0.8	0.5 0.4 0.7 0.5
		0.4 0.4 0.4 0.2
1.0 1.2 1.4 1.6	0.5 0.0 -0.0 -0.1	Error!
0.7 0.9 1.1 1.0	0.7 0.0 0.0 0.1	Error!
0.6 1.0 0.7 1.0		
0.7 0.9 1.1 1.0	0.8 0.1 -0.1 0.0	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。