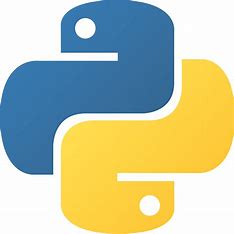
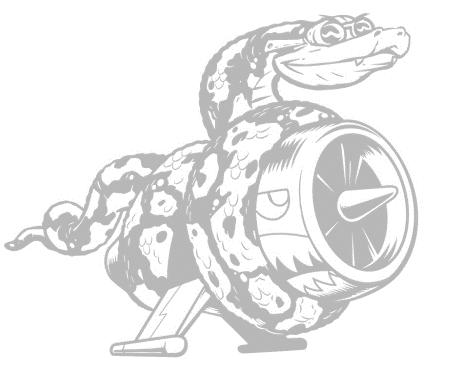
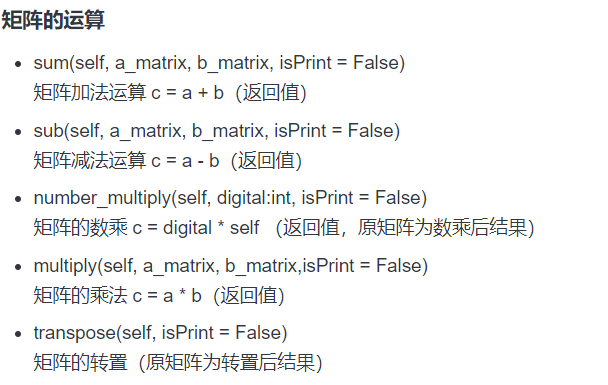
Python **大作业**

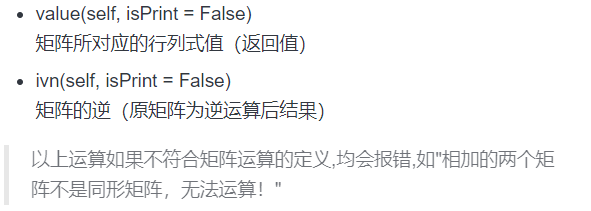












代码 Matrix.py

class Matrix():

'''

这里是矩阵类，看我从0开始硬搓矩阵

虽然我不喜欢写注释，文档但是这次我不得不写

'''

def \_\_init\_\_(self, line:int, row:int, data:list) :

'''这里是矩阵的基本设置'''

self.data = data # 这里储存矩阵的数据，列表

self.line = line # 矩阵的行，int

self.row = row # 矩阵的列，int

'''以下是有关矩阵的输出'''

def get\_index(self, line\_get:int, row\_get:int, isPrint = False):

'''

根据矩阵下标获得对应数字

:param line\_get: 行

:param row\_get: 列

:param isPrint: True 为打印该数字 默认为False

:return: 返回下标对应数字

'''

if line\_get > self.line or row\_get > self.row:

print('下标超出范围！')

return False

data\_get = self.data[(line\_get - 1) \* self.row + row\_get - 1]

if isPrint:

print(data\_get)

return data\_get

def get\_number(self, number:int, isPrint = False):

'''

根据编号获得数字

:param number: 编号

:param isPrint: True 为打印该数字 默认为False

:return: 返回下标对应数字

'''

if number > self.line \* self.row:

print('编号超出范围！')

return False

data\_get = self.data[number - 1]

if isPrint:

print(data\_get)

return data\_get

def print\_matrix(self):

'''

按照矩阵格式打印矩阵

'''

for i in range(self.line):

for j in range(self.row):

print(self.data[i \* self.row + j],end = ' ')

print('\n')

'''以下是矩阵有关运算'''

def sum(self, a\_matrix, b\_matrix, isPrint = False):

'''

矩阵加法运算 c = a + b

:param a\_matrix: 矩阵a

:param b\_matrix: 矩阵b

:param isPrint: True 为打印该矩阵 默认为False

:return: 矩阵c

'''

if a\_matrix.line != b\_matrix.line or a\_matrix.row != b\_matrix.row:

print('相加的两个矩阵不是同形矩阵，无法运算！')

return False

c\_matrix\_line, c\_matrix\_row = a\_matrix.line, a\_matrix.row

all\_number = c\_matrix\_line \* c\_matrix\_row

# 列表推导式，求矩阵加法

c\_matrix\_data = [a\_matrix.data[number] + b\_matrix.data[number] for number in range(all\_number)]

c\_matrix = Matrix(c\_matrix\_line, c\_matrix\_row, c\_matrix\_data) # 生成矩阵c

if isPrint:

c\_matrix.print\_matrix()

return c\_matrix

def sub(self, a\_matrix, b\_matrix, isPrint = False):

'''

矩阵减法运算 c = a - b

:param a\_matrix: 矩阵a

:param b\_matrix: 矩阵b

:param isPrint: True 为打印该矩阵 默认为False

:return: 矩阵c

'''

if a\_matrix.line != b\_matrix.line or a\_matrix.row != b\_matrix.row:

print('相减的两个矩阵不是同形矩阵，无法运算！')

return False

c\_matrix\_line, c\_matrix\_row = a\_matrix.line, a\_matrix.row

all\_number = c\_matrix\_line \* c\_matrix\_row

# 列表推导式，求矩阵减法

c\_matrix\_data = [a\_matrix.data[number] - b\_matrix.data[number] for number in range(all\_number)]

c\_matrix = Matrix(c\_matrix\_line, c\_matrix\_row, c\_matrix\_data) # 生成矩阵c

if isPrint:

c\_matrix.print\_matrix()

return c\_matrix

def number\_multiply(self, digital:int, isPrint = False):

'''

矩阵的数乘 c = digital \* self

:param digital: 所乘的数字

:param a\_matrix: 矩阵a

:param isPrint: True 为打印该矩阵 默认为False

:return: 返回矩阵c

'''

c\_matrix\_line, c\_matrix\_row = self.line, self.row

all\_number = c\_matrix\_line \* c\_matrix\_row

# 列表推导式，求矩阵数乘

c\_matrix\_data = [self.data[number] \* digital for number in range(all\_number)]

self.data = c\_matrix\_data

c\_matrix = Matrix(c\_matrix\_line, c\_matrix\_row, c\_matrix\_data) # 生成矩阵c

if isPrint:

c\_matrix.print\_matrix()

return c\_matrix

def multiply(self, a\_matrix, b\_matrix,isPrint = False):

'''

矩阵的乘法

:param a\_matrix: 矩阵a

:param b\_matrix: 矩阵b

:param isPrint: True 为打印该矩阵 默认为False

:return: 矩阵c

'''

if a\_matrix.row != b\_matrix.line:

print('相乘的两个矩阵行列不匹配，无法运算！')

return False

c\_matrix\_line, c\_matrix\_row = a\_matrix.line, b\_matrix.row

all\_number = c\_matrix\_line \* c\_matrix\_row

c\_matrix\_data = [0 for i in range(all\_number)] # 初始化c

ans = 0 # 初始化和

for i in range(a\_matrix.line):# 行

for j in range(b\_matrix.row):# 列

for k in range(a\_matrix.row):

ans = ans + a\_matrix.data[i \* a\_matrix.row + k] \* b\_matrix.data[k \* b\_matrix.row + j]

c\_matrix\_data[i \* c\_matrix\_row + j] = ans

ans = 0 # 重置为0

c\_matrix = Matrix(c\_matrix\_line, c\_matrix\_row, c\_matrix\_data) # 生成矩阵c

if isPrint:

c\_matrix.print\_matrix()

return c\_matrix

def transpose(self, isPrint = False):

'''

矩阵的转置

:param isPrint: True 为打印该矩阵 默认为False

'''

data = tuple(self.data) # 注意 这里要是不用元组，则是浅拷贝，下面程序执行时会导致data数值发生变化

# 转置

for i in range(self.line):

for j in range(self.row):

self.data[j \* self.line + i] = data[i \* self.row + j]

self.line, self.row = self.row, self.line

if isPrint:

self.print\_matrix()

def value(self, isPrint = False):

'''

行列式的值

:param isPrint: True 为打印结果 默认为False

:return: 行列式的值

'''

if self.row != self.line:

print("不是方阵无法运算！")

return 0

# 当我写完之后我才想起来，原来对角线法不能用于2\*2行列式

# 2\*2

if self.row == 2:

ans = self.data[0] \* self.data[3] - self.data[1] \* self.data[2]

if isPrint: # 我知道这里写的很蠢，但是下面我实在不想动了，就当是屎山得了

print(ans)

return ans # 能跑就行，我加2\*2后第一遍写完发现报错了，吓死了，还好只是缩进错了（就是这里）

# !2\*2(这个感叹号是不是很妙啊，哈哈哈哈）

# 这里其实用的是对角线法,现在是加数值

ans = 0

for k in range(0, self.row):

ans\_middle = 1

for i, j in zip(range(0, self.row), range(k, self.line + k)):

ans\_middle \*= self.data[i \* self.row + j % self.line]

ans += ans\_middle

# 现在是减数值

for k in range(0, self.row):

ans\_middle = 1

for i, j in zip(range(0, self.row), range(k, self.line + k)):

ans\_middle \*= self.data[i \* self.row + (3 - j) % self.line]

ans -= ans\_middle

# 上面要是看不懂，自己画个3\*3行列式找个规律就行了

if isPrint:

print(ans)

return ans # 矩阵的逆第一步我算是写完了

def ivn(self, isPrint = False):

'''

矩阵的逆

:param isPrint: True 为打印结果 默认为False

'''

if self.row != self.line:

print("不是方阵无法运算！")

return 0

# 这里先进行一些代数余子式的运算

# 每个代数余子式可以看成是n - 1 \* n - 1的行列式计算，关键在于怎么把行和列剔除呢

matrix\_middle = Matrix(self.row - 1, self.line - 1, [0])

matrix\_middle\_data = []

data = []

value = self.value()

for k in range(self.line):

for p in range(self.row):

# 以上是控制 n \* n 矩阵的

# 以下是控制 n - 1 \* n - 1 矩阵的（代数余子式）

matrix\_middle\_data.clear()

for i in range(self.line):

for j in range(self.row):

if i != k and j != p:

matrix\_middle\_data.append(self.data[i \* self.row + j])

matrix\_middle.data = matrix\_middle\_data

# 现在我余子式那个行列式构建完成，下面我开始计算,并把他放到新的列表里面

data.append(matrix\_middle.value() \* -1 \*\* (k + p))

# 此时，我有关代数余子式计算已经完成力(喜)

self.data = data

self.transpose()

data\_middle = []

for i in range(self.line):

for j in range(self.row):

data\_middle.append(self.data[i \* self.row + j] / value)

self.data = data\_middle

if isPrint:

self.print\_matrix()

测试集main.py

from matrix import Matrix

# 这里是简单的测试集

a = Matrix(3, 2, [9, 2, 3, 21, 5, 6])

b = Matrix(2, 4, [1, 2, 10, 4, 23, 6, 15, 8])

c = Matrix(3, 2, [18, 19, 23, 1, 3, 51])

print('a = ')

a.print\_matrix()

print('b = ')

b.print\_matrix()

print('c = ')

c.print\_matrix()

print('a + c =')

a.sum(a, c, True)

print('a - c =')

a.sub(a, c, True)

print('c的转置')

c.transpose(True)

print('a \* b =')

a.multiply(a, b, True)

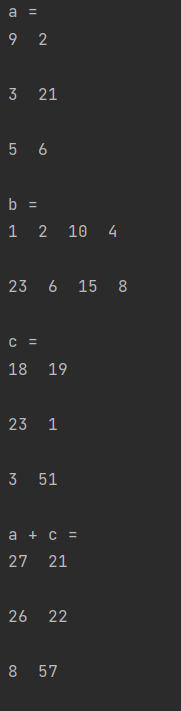
print('2 \* a = ')

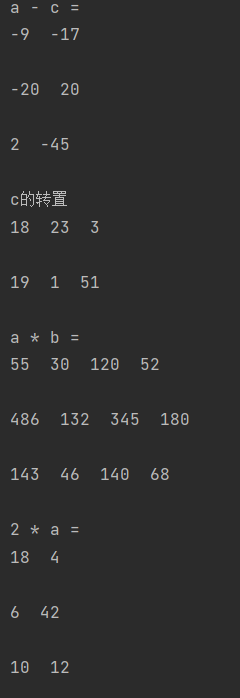
a.number\_multiply(2, True)

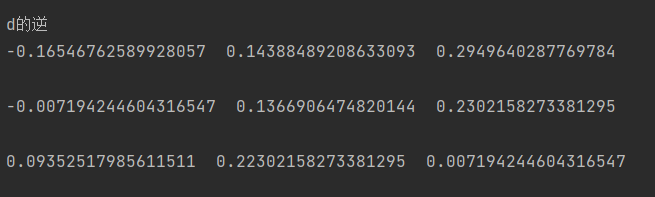
print('d的逆')

d.ivn(True)

测试截图







课

课程总结

经过本学期python课程的学习，让我对python有了更深入了解，通过此次大作业，更加深刻体会到了面向对象的编程的含义。让我更加熟练掌握了类的定义使用，并有机的与前面所学内容相结合起来。