技术报告

本项目在原框架的基础上，在minimatrix.py中实现了matrix类，并在main.py中进行了测试。下展开对minimatrix.py中函数的说明。

\_\_init\_\_()初始化函数接收三个参数：矩阵数据，维数，与起始值。若用户输入矩阵数据，则判断数据中各行列表长度是否相同，即能否生成矩阵，若不能则报错。若用户输入维数，则根据初始值生成该大小的矩阵。

shape()函数返回矩阵的维数。

reshape()函数接收一个元组作为新维数，返回重新排列后的矩阵。先判断新维数对应元素数量是否与原矩阵相符，若否则报错。将原矩阵数据按序存储为列表，再根据新维数进行重新排列生成新矩阵。

dot()函数接收另一个矩阵类对象，实现两对象的乘积，返回答案矩阵。先判断二者是否可乘，若否则报错。先创建一个答案矩阵大小的全零数据列表，随后按照矩阵乘法运算法则逐位运算。最后根据答案列表生成对象并输出。

T()函数不接受参数，实现矩阵的转置，返回答案矩阵。先建立目标大小的空矩阵，随后根据原矩阵数据填充信息。

sum()函数接收axis参数作为按列或按行或按元素相加的指针，返回求和结果。

copy()函数生成一个原矩阵的复制。重新按元素创造列表以防止嵌套列表id相同产生的问题。

Kronecker\_product() pass

\_\_getitem\_\_()依据行和列的输入数据类型分类，获取元素生成矩阵输出。

\_\_setitem\_\_()依据行和列的输入数据类型分类，更改元素。

\_\_pow\_\_()接受幂指数，先判断矩阵是否为方阵，随后不断做累乘得出目标矩阵。

\_\_add\_\_(),\_\_sub\_\_(),\_\_mul\_\_()先判断二矩阵是否大小相同，若否则报错。随后遍历所有矩阵，逐位进行操作。

\_\_len\_\_()返回总元素个数。

\_\_str\_\_()

Det()

Inverse()

Rank()