技术报告

Matrix类及其函数功能说明

项目简介:

本项目通过在minimatrix.py中实现Matrix类，以及在main.py中进行测试，提供了一个功能强大的矩阵操作工具。以下是对minimatrix.py中函数的详细说明。

1.\_\_init\_\_() 初始化函数:

·三个参数：矩阵数据、维数、起始值。

·若用户输入矩阵数据，则检查各行列表长度是否相同，否则报错。

·若用户输入维数，则根据初始值生成相应大小的矩阵。

2.shape() 函数:

·返回矩阵的维数。

3.reshape(new\_dim: Tuple[int, int]) -> Matrix 函数:

·接收一个元组作为新维数，返回重新排列后的矩阵。

·判断新维数元素数量是否与原矩阵相符，否则报错。

·重新排列原矩阵数据生成新矩阵。

4.dot(other: Matrix) -> Matrix 函数:

·接收另一个矩阵对象，实现两矩阵的乘积，返回结果矩阵。

·判断两矩阵是否可乘，否则报错。

·逐位进行矩阵乘法运算，生成结果矩阵。

5.T() -> Matrix 函数:

·不接受参数，实现矩阵的转置，返回结果矩阵。

·建立目标大小的空矩阵，根据原矩阵数据填充信息。

6.sum(axis: Optional[int] = None) -> Union[float, Matrix] 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。

·若 axis 为 None，则返回矩阵所有元素之和。

7.copy() -> Matrix 函数:

·生成原矩阵的复制，防止嵌套列表 id 相同的问题。

8.Kronecker\_product(other: Matrix) -> Matrix 函数:

·用于求取 Kronecker 积。

·先创建空答案列表，使用嵌套循环遍历第一个矩阵的每个元素，构建临时矩阵进行乘法操作。最后输出结果矩阵。

9.\_\_getitem\_\_(key: Union[int, Tuple[int, int]]) -> float 函数:

·根据行和列的输入数据类型分类，获取元素并生成矩阵输出。

10.\_\_setitem\_\_(key: Union[int, Tuple[int, int]], value: float) 函数:

·根据行和列的输入数据类型分类，更改元素。

11.\_\_pow\_\_(exponent: int) -> Matrix 函数:

·接受幂指数，判断矩阵是否为方阵，不断进行累乘得出目标矩阵。

12.\_\_add\_\_(), \_\_sub\_\_(), \_\_mul\_\_() 函数:

·判断两矩阵是否大小相同，否则报错。

·遍历所有矩阵，逐位进行加法、减法、乘法操作。

13.\_\_len\_\_() -> int 函数:

·返回总元素个数。

14.\_\_str\_\_() -> str 函数:

·用于美化矩阵输出，使输出结果右对齐。

·判断每个元素是否为整数，若是则转换为整数类型。

·根据元素类型和大小更新宽度值，确保输出对齐。

15.Gauss\_elimination() -> Tuple[Matrix, int] 函数:

·用于实现高斯消元法，返回高斯消元后的矩阵和交换行的次数。

·创建变量 seq 用于追踪交换行的次数，同时复制输入矩阵以避免修改原始数据。

·通过循环迭代矩阵的每一列，对角线元素用作主元素进行消元操作。

16.det() -> float 函数:

·用于求矩阵的行列式。

·判断是否为方阵，调用高斯消元法后，对主对角线上的元素进行乘积，并根据交换行的次数判断是否取相反数。

17.Inverse() -> Matrix 函数:

·通过初等行变换法求逆矩阵。

·在原矩阵右侧拼接一个单位矩阵，并通过高斯消元法进行初等行变换直到左侧变为单位矩阵，右侧即为逆矩阵。

18.rank() -> int 函数:

·通过初等行变换法求矩阵的秩。

·对原矩阵调用高斯消元法，非零行数即为秩，作为结果输出。

19.I(n: int) -> Matrix 函数:

·用于生成大小为 n x n 的单位矩阵。

20.narray(dim: Tuple[int, int], init\_value: float = 1) -> Matrix 函数:

·生成指定维度的矩阵，所有元素初始化为给定的初始值。

21.arange(start: float, end: float, step: float = 1) -> Matrix 函数:

生成一个行向量，start 是起始值，end 是终止值（不包含），step 是步长，默认为 1。

22.zeros(dim: Tuple[int, int]) -> Matrix 函数:

·生成一个零矩阵。

23.zeros\_like(matrix: Matrix) -> Matrix 函数:

·生成一个与给定矩阵相同维度的零矩阵。

24.ones(dim: Tuple[int, int]) -> Matrix 函数:

·生成一个元素全为1的矩阵。

25.ones\_like(matrix: Matrix) -> Matrix 函数:

·生成一个与给定矩阵相同维度的全1矩阵。

26.random(dim: Tuple[int, int]) -> Matrix 函数:

·生成一个指定维度的矩阵，元素为 [0, 1) 之间的随机数。

27.random\_like(matrix: Matrix) -> Matrix 函数:

·生成一个与给定矩阵相同维度的随机数矩阵。

28.concatenate(items: List[Matrix], axis: int = 0) -> Matrix 函数:

·实现矩阵的拼接。

·验证输入参数的合法性，包括拼接对象的存在、类型为矩阵、拼接维度的正确性等。

根据指定的维度，进行行或列的拼接操作。

29.vectorize(func: Callable[[float], float]) -> Callable[[Matrix], Matrix] 函数:

·接受一个函数作为参数，返回一个新的函数，该新函数可以接受矩阵作为输入，并将函数应用于输入矩阵的每个元素或标量。

总结：

Matrix 类及其函数提供了丰富的矩阵操作功能，包括基本运算、特殊矩阵生成、矩阵变换、行列式计算、逆矩阵求解、秩的计算等。通过这些功能，用户可以方便地进行矩阵运算和分析，为科学计算、线性代数等领域提供了便利的工具。