CS101 Homework #4

Matrix operations on April 30, 2012

Due date: Monday, May 14, 2012 (util 23:59)

Please read homework description carefully and make sure that your program meets all the requirements stated below. The homework is individual task. You can disscuss the problem with your friend but you are not allowed to program together. You will get F on entire course if your homework includes any plagiarism.

Goal

Please complete the program, 'HW4_20xxxxxx.py', to manipulate matrix operations.

Through this last homework, we hope that you are familiar with user-defined object, method, file processing, and so on.

Description

In HW4.zip file, there are five files except this one.

One is a python template file, "HW4_20xxxxxx.py", which you must complete. If your student ID is 20120123, then the file name should be "HW4_20120123.py".

Three files describe about matrix information. In the example, "a.mtx.txt", "b.mtx.txt", and "c.mtx.txt" contain the descript about matrices A, B, and C, respectively. You may construct a matrix (user-defined) object, CS101Matrix, by reading them. When your program is tested, other matrix information files may be used. Therefore, we strongly recommend that you make your own matrix information files and test your program with them. Make your own matrix files, "20xxxxxx.mtx.txt", and include them when you submit.

The last one is the sample output, "test.info.txt", which is a file generated by the statement "f.write(string(a))" in the template program.

Requirements

Object (Class)

We define a new user-defined object, 'CS101Matrix,' in the given template file.
In that object, there are at least five attributes.

_matrix_name – string, matrix name
_matrix – list of lists (2-dimensional matrix), representing matrix itself
elements of _matrix are considered as floating values
_numRows – integer, number of rows
_numCols – integer, number of columns
_isValid – bool, indicator whether it is correctly constructed or not
When a user creates an empty matrix object intentionally, it is assumed valid

By reading a matrix information file, such as 'a.mtx.txt', you must fill at least aforementioned five attributes. When you read a certain line containing a string "matrix_name", you can use string value after "=" as the attribute _matrix_name.

No white space characters must be contained on the left or right of _matrix_name.

Other lines represent rows of the matrix except blank lines. Each element on a line is separated by white spaces, such as spaces, tabs, or any combination of them. However, each line contains only one row information. Any skewed matrix is not allowed, such as:

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & & \\ 8 & 9 & 10 & 11 & 12 \\ 13 & 7 & 14 & 15 & 16 \\ 2 & 17 & 18 & 0 & \end{bmatrix}$$

When you read such a matrix, you can stop reading matrix information and set _isValid attribute as False. The followings are some examples of matrix information file.

matrix_name = A 0 2 6 1 7 9 5 3 7
a.mtx.txt
valid

matrix_name = B 1 2 3 4 5 6   7 8 9
b.mtx.txt
valid

matrix_name = Oops 1 2 3 4 5 6 7 8 9 9
c.mtx.txt
invalid

In addition, when matrix is correctly constructed, you can determine the numbers of rows and columns and assign them to _numRows and _numCols attributes, respectively.

Functions

You must complete the following 10 functions listed in the following box:

```
initialize, string,
negate, add, subract, multiply,
equal,
transpose,
scalarProduct, vectorProduct
```

1. initialize()
   - ▷ Input: string, that is, filename to read about matrix information
   - ▷ Output: CS101Matrix object
   - ▷ Initializes a CS101Matrix object – as specified in the file.
   - ▷ Extract the matrix information and set up a 'CS101Matrix' object.
   - ▷ If there is invalid matrix information, then stop reading and set _isValid attribute as False.
   - ▷ When you create CS101Matrix with default parameter None, it is considered to create a valid empty matrix.

2. string()
   - ▷ Input: CS101Matrix object
   - ▷ Output: string, description for the CS101Matrix object
   - ▷ Complete this method to nicely display matrix information on the console or file.
   - ▷ When displayed on the console, every element in the same column is right-aligned and each column is separated by enough white space (except '₩n').
   - ▷ Example (from a.mtx.txt file)

```
>>> a = initialize("a.mtx.txt")
>>> print string(a)
Matrix: A
Number of rows: 3, Number of columns: 3
|   0.00   2.00   6.00   |
|   1.00   7.00   9.00   |
|   5.00   3.00   7.00   |
```

```
>>> c = initialize("c.mtx.txt")
>>> print string(c)
Not a correct matrix!!!
```

3. negate()
   - ▷ Input: CS101Matrix object
   - ▷ Output: None, but elements in the input parameter are negated
   - ▷ Sign of every element in the matrix is inversed.
   - ▷ Examle (from a.mtx.txt file)

```
>>> a = initialize("a.mtx.txt")
>>> negate(a)
>>> print string(a)
Matrix: Result of Matrix Negation (-A)
Number of rows: 3, Number of columns: 3
|  -0.00  -2.00  -6.00  |
|  -1.00  -7.00  -9.00  |
|  -5.00  -3.00  -7.00  |
```

4. add()
   - ▷ Input: two CS101Matrix objects, operands to be added
   - ▷ Output: new CS101Matrix object or None, result of addition
   - ▷ Add two matrices, say matrix A & matrix B.
   - ▷ When two matrices are of different size, do not proceed and print error message returning None.
   - ▷ Notice that name of _matrix_name attribute must be changed correspondingly.
   - ▷ Example (from a.mtx.txt & b.mtx.txt files)

```
>>> a = initialize("a.mtx.txt")
>>> b = initialize("b.mtx.txt")
>>> print string(add(a, b))
Matrix: Result of Matrix Addition (A + B)
Number of rows: 3, Number of columns: 3
|   1.00   4.00   9.00  |
|   5.00  12.00  15.00  |
|  12.00  11.00  16.00  |
```

5. subtract()

▷ Input: two CS101Matrix objects,

If first parameter is A and the other is B, then subtract B from A

▷ Output: new CS101Matrix object or None, result of subtraction

▷ Subtract right-hand side matrix from current CS101Matrix object.

▷ When two matrices are of different size, do not proceed and print error message returning None.

▷ Notice that name of _matrix_name attribute must be changed correspondingly.

▷ Example (from a.mtx.txt & b.mtx.txt files)

```
>>> a = initialize("a.mtx.txt")
>>> b = initialize("b.mtx.txt")
>>> print string(subtract(a, b))
Matrix: Result of Matrix Subtraction (A - B)
Number of rows: 3, Number of columns: 3
|  -1.00   0.00   3.00  |
|  -3.00   2.00   3.00  |
|  -2.00  -5.00  -2.00  |
```

6. multiply()

▷ Input: two CS101Matrix object,

If first parameter is A and the other is B, then A is multiplied by B

▷ Output: new CS101Matrix object or None, result of multiplication

▷ Be aware that this operation can be performed only when the number of columns of _matrix attribute in left-hand side object and the number of rows of _matrix attribute in right-hand side object are equal. Otherwise, print error message and return None.

▷ Notice that name of _matrix_name attribute must be changed correspondingly.

▷ Example (from a.mtx.txt & b.mtx.txt files)

```
>>> a = initialize("a.mtx.txt")
>>> b = initialize("b.mtx.txt")
>>> print string(multiply(a, b))
Matrix: Result of Matrix Multiplication (A * B)
Number of rows: 3, Number of columns: 3
|  50.00  58.00  66.00  |
|  92.00 109.00 126.00  |
|  66.00  81.00  96.00  |
```

7. equal()

▷ Input: two CS101Matrix object, operands to be compared

▷ Output: bool, True if two matrices have the same elements with the same value on the same position
False, otherwise

▷ Example (from a.mtx.txt & b.mtx.txt files)

```
>>> a = initialize("a.mtx.txt")
>>> b = initialize("b.mtx.txt")
>>> print equal(a, b)
False
```

8. transpose()

▷ Input: CS101Matrix object

▷ Output: new transposed CS101Matrix object

▷ Be aware that this operation can be performed even though number of rows and columns are not equal.

▷ Examle (from a.mtx.txt file)

```
>>> a = initialize("a.mtx.txt")
>>> print string(transpose(a))
Matrix: Transposed Matrix (At)
Number of rows: 3, Number of columns: 3
|   0.00   1.00   5.00   |
|   2.00   7.00   3.00   |
|   6.00   9.00   7.00   |
```

9. scalarProduct()

▷ Input: a scale factor (float) and a CS101Matrix object

▷ Output: new CS101Matrix object, result of scalar product

$c\sum_{i=1}^{n}a_i$ for $\overrightarrow{A}=(a_1,...,a_n)$ and scale factor c

10. innerProtuct()

▷ Input: two CS101Matrix objects, that is, two vectors of CS101Matrix

▷ Output: float, a result of inner product

▷ Beware that this operation is appropriate only when two matrices are both vectors and number of dimensions of them are identical.

$|\overrightarrow{A}| \ |\overrightarrow{B}|\cos(ang(\overrightarrow{A},\overrightarrow{B}))$, where $ang(\overrightarrow{A},\overrightarrow{B})$ is angle between $\overrightarrow{A}$ and $\overrightarrow{B}$

11. getElement()

```
def getElement(matrix, x, y):
    if not matrix._isValid:
        return None
    elif matrix._numRows > x or matrix._numCols > y or x <= 0 or y <= 0:
        return None
    return matrix._matrix[x - 1][y - 1]
```

▷ Input: integers, row and column indices of matrix

x – row index, y – column index

We assume that left-top element in _matrix attribute is positioned at (0, 0). Then one row has _numCols elements, one column has _numRows elements. For example, the (1, 2)-element of matrix A (a.mtx.txt) is 9.00.

If you modify this assumption, please note that in your report and describe.

▷ Output: float, element of _matrix at position (x, y)

Submission

You need to submit your source code (changing the file name HW4_20xxxxxx.py → HW4_20120123.py), and your own matrix information files (HW4_20xxxxxx.mtx) as well as corresponding report.

If you do not follow this direction, you may get some penalty.

You must submit your zip file of source code and data through the CS101 homework submission webpage.