Jaewoo An

CS101 Basic Programming

14 May 2012

Homework #4 Matrix

|  |  |
| --- | --- |
| Name of the Function | Implementation |
| initialize(matrixFile = None) | Case *a)* if matrixFile = None:  Then returns a valid CS101Matrix with name(None), empty 0x0 matrix.  Case *b)* otherwise; open and read the text file of the filename = matrixFile, and takes all the necessary information from the file. Declare a CS101Matrix() as result\_matrix, and assign the obtained information to the corresponding attributes.  Using  for line in f:  row = line.strip().replace('\"', "").split()  I could retrieve the row vectors of the matrix, and save it in a previously declared list called row. Using a for loop inside the for loop, I could then make the program to keep taking next row vectors and save it as row, which is then saved as an element of the list result\_matrix.\_matrix. |
| string(matrix) | Using %s, %d, and %f, I could easily combine a certain set of strings with attributes from the CS101Matrix object called matrix.  With “\n” I could change the line.  Using a for loop, I could retrieve row data from the list of list–matrix.\_matrix.  Lastly, this function returns the resulting string. |
| getElement(matrix, x, y) | Returns the element at the position (x, y) of the parameter matrix. If either the matrix is invalid or the position (x, y) is out of the matrix’s size, then this function returns None.  If not, this function returns matrix.\_matrix[x-1][y-1] in float. |
| negate(matrix) | This function first retrieves each element of the parameter matrix through double for loop, and multiplies the retrieved elements by (-1) and saves it on the original matrix, thereby negating the entire matrix. |
| add(leftMatrix, rightMatrix) | Using %s, I could combine the determined set of strings with leftMatrix.\_matrix\_name and rightMatrix.\_matrix\_name.  With the CS101Matrix object called newMatrix, this function retrieves all the data from both leftMatrix and rightMatrix by accessing their attributes and adds each corresponding elements together in the double for loop. Then this function saves those values in newMatrix as its attributes and returns the completed newMatrix.  However, if the two matrices have different sizes, then this function simply prints an error message and returns None. |
| subtract(leftMatrix, rightMatrix) | This function acts exactly like add(leftMatrix, rightMatrix) except that this function subtracts the element of rightMatrix from the corresponding element of leftMatrix rather than adding them together. |
| multiply(leftMatrix, rightMatrix) | This function acts exactly like the previous add() and subtract() functions, except the double for loop part.  Rather than a double for loop, this function has a triple for loop. When two matrices are multiplied, the row vector of the leftMatrix and the column vector of the rightMatrix are multiplied, and the result is saved as a element at position (the row index of leftMatrix, the column index of the rightMatrix).  Since this process requires three variables (the row index of leftMatrix, the column index of the rightMatrix, and the index of the element being multiplied within the row or column), I had to use a triple for loop structure.  Lastly, this function prints an error message and returns None when the numCols of the leftMatrix and the numRows of the rightMatrix have different values. |
| equal(leftMatrix, rightMatrix) | This function acts exactly like add() and subtract(), except that this function returns a Boolean value (T of F) depending on the equality of the two parameter matrices.  Using a double for loop, this function retrieves and compares the elements of leftMatrix and rightMatrix at the same position. If any of the element pairs is not equal, then this function returns False. Otherwise, True. |
| transpose(matrix) | This function retrieves elements from the parameter matrix at the position (x, y) and saves it as an element of the new CS101Matrix called newMatrix at the position (y, x), where x and y are for-loop variables working as indexes. |
| scalarProduct(scale, matrix): | This function retrieves elements from the parameter matrix and multiplies them by the scalar parameter “scale”. The resulting elements are then saved at the newly created CS101Matrix called newMatrix. |
| innerProduct(leftMatrix, rightMatrix) | Only if two matrices are both vectors (that is, when they are both 1 x n or n x 1 matrices) and number of dimensions of them are identical, this function works. If both matrices are row vectors, then this function transposes the rightMatrix and multiplies each element (element at position (a, b) at leftMatrix and element at position (b, a) at rightMatrix) and adds the result together. (which is basically multiply(leftMatrix, leftMatrix)). The added result is returned.  A \* B = |A||B|cos(ang(A, B)).  Ang(A, B) = 0 since both vectors have the same dimension (either 1 x n or n x 1).  cos(0) = 1.  Thus, A \* B is simply multiplication of the magnitude. |