**Харківський національний економічний університет**

**імені Семена Кузнеця**

**ЗВІТ**

**З ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 9**

**за дисципліною: *“*Технології програмування**”

**на тему: “Робота з класами та об’єктами”**

**Варіант: 4**

**Виконав: студент факультету Інформаційних технологій**

**2 курсу, спец. Кібербезпека,**

**групи 6.04.125.010.21.2**

**Бойко Вадим Віталійович**

**Перевірив:**

**Міхєєв Іван Андрійович**

**ХНЕУ ім. С. Кузнеця**

**2022**

Завдання

1. конструктор із можливим заданням елементів матриці у вигляді двовимірного масиву (якщо елементи не надаються під час створення об’єкту, вони генеруються випадковим чином);
2. деструктор;
3. метод для виведення матриці на екран (строкове представлення матрриці);
4. Додаткова функціональність, описана у варіантах завдання.

Варіант завдання

1. Метод для визначення чи є матриця діагональною; метод для визначення чи є матриця трикутною. Перевантажити бінарний оператор додавання двох матриць, транспонування матриці.

Конструктор

*class* Matrix:  
 *def \_\_init\_\_*(self, *matrix* = *None*):  
 *if matrix* == *None*:  
 self.matrix = []  
  
 *for* i *in* range(3):  
 l = []  
 *for* j *in* range(3):  
 l.append(random.randint(-100, 100))  
 self.matrix.append(l)  
 *else*:  
 self.matrix = *matrix*

Деструктор

*def* getMatrix(self):  
 m = []  
 *for* row *in* self.matrix:  
 newRow = []  
 *for* i *in* row:  
 newRow.append(i)  
 m.append(newRow)  
 *return* m

Виведення матриці на екран

*def* printMatrix(self):  
 *for* row *in* self.matrix:  
 *for* i *in* row:  
 print('{}\t\t'.format(i), end='')  
 print()

Чи є матриця діагональною

*def* isDiagonal(self):  
 isDiagonal = len(self.matrix) == len(self.matrix[0])  
 *for* i *in* range(len(self.matrix)):  
 *if not* isDiagonal:  
 *break  
 for* j *in* range(len(self.matrix[i])):  
 *if not* isDiagonal:  
 *break  
 if* isDiagonal *and* ((j == i *and* self.matrix[i][j] == 0) *or* (j != i *and* self.matrix[i][j] != 0)):  
 isDiagonal = *False  
 return* isDiagonal

Чи є матриця трикутна

*def* isTrangle(self):  
 isTrangel = len(self.matrix) == len(self.matrix)  
 *for* i *in* range(len(self.matrix)):  
 *if not* isTrangel:  
 *break  
 for* j *in* range(len(self.matrix[i])):  
 *if not* isTrangel:  
 *break  
 if* isTrangel *and* (j < i *and* self.matrix[i][j] != 0):  
 isTrangel = *False  
 return* isTrangel

Додавання двох матриць

*def* addMatrix(self, *matrix* = *None*):  
 a1 = len(self.matrix)  
 b1 = len(self.matrix[0])  
  
 mtx = Matrix().getMatrix() *if matrix* == *None else matrix*.getMatrix()  
  
 a2 = len(mtx)  
 b2 = len(mtx[0])  
  
 a\_min = min(a1, a2)  
 a\_max = max(a1, a2)  
  
 b\_min = min(b1, b2)  
 b\_max = max(b1, b2)  
 *if* a\_min < a\_max:  
 *if* a1 < a\_max:  
 empty\_row = [0 *for* x *in* range(a\_max)]  
 self.matrix.append(empty\_row)  
 *else*:  
 empty\_row = [0 *for* x *in* range(a\_max)]  
 mtx.append(empty\_row)  
  
 newMatrix = []  
 *for* i *in* range(a\_max):  
 row = []  
 *for* j *in* range(b\_min):  
 row.append(self.matrix[i][j] + mtx[i][j])  
 *if* b\_min < b\_max:  
 *for* j *in* range(b\_min, b\_max):  
 *try*:  
 row.append(self.matrix[i][j])  
 *except*:  
 row.append(mtx[i][j])  
  
 newMatrix.append(row)  
 *return* newMatrix

Транспонування матриці

*def* transponentMatrix(self):  
 newMatrix = []  
 *for* i *in* range(len(self.matrix[0])):  
 newMatrix.append([0 *for* x *in* range(len(self.matrix))])  
  
 *for* i *in* range(len(self.matrix)):  
 *for* j *in* range(len(self.matrix[0])):  
 newMatrix[j][i] = self.matrix[i][j]  
  
 *return* newMatrix

Весь клас

*import* random  
  
# a = int(input("Введіть кількість строк матриці "))  
# b = int(input("Введіть кількість стовпців матриці "))  
  
# m = []  
#  
# for i in range(3):  
# l = []  
# for j in range(3):  
# s = "введіть значення матриці для значення {} {} ".format(i+1, j+1)  
# l.append(int(input(s)))  
# m.append(l)  
  
*class* Matrix:  
 *def \_\_init\_\_*(self, *matrix* = *None*):  
 *if matrix* == *None*:  
 self.matrix = []  
  
 *for* i *in* range(3):  
 l = []  
 *for* j *in* range(3):  
 l.append(random.randint(-100, 100))  
 self.matrix.append(l)  
 *else*:  
 self.matrix = *matrix  
  
 def* printMatrix(self):  
 *for* row *in* self.matrix:  
 *for* i *in* row:  
 print('{}\t\t'.format(i), end='')  
 print()  
  
 *def* isDiagonal(self):  
 isDiagonal = len(self.matrix) == len(self.matrix[0])  
 *for* i *in* range(len(self.matrix)):  
 *if not* isDiagonal:  
 *break  
 for* j *in* range(len(self.matrix[i])):  
 *if not* isDiagonal:  
 *break  
 if* isDiagonal *and* ((j == i *and* self.matrix[i][j] == 0) *or* (j != i *and* self.matrix[i][j] != 0)):  
 isDiagonal = *False  
 return* isDiagonal  
  
 *def* isTrangle(self):  
 isTrangel = len(self.matrix) == len(self.matrix)  
 *for* i *in* range(len(self.matrix)):  
 *if not* isTrangel:  
 *break  
 for* j *in* range(len(self.matrix[i])):  
 *if not* isTrangel:  
 *break  
 if* isTrangel *and* (j < i *and* self.matrix[i][j] != 0):  
 isTrangel = *False  
 return* isTrangel  
  
 *def* addMatrix(self, *matrix* = *None*):  
 a1 = len(self.matrix)  
 b1 = len(self.matrix[0])  
  
 mtx = Matrix().getMatrix() *if matrix* == *None else matrix*.getMatrix()  
  
 a2 = len(mtx)  
 b2 = len(mtx[0])  
  
 a\_min = min(a1, a2)  
 a\_max = max(a1, a2)  
  
 b\_min = min(b1, b2)  
 b\_max = max(b1, b2)  
 *if* a\_min < a\_max:  
 *if* a1 < a\_max:  
 empty\_row = [0 *for* x *in* range(a\_max)]  
 self.matrix.append(empty\_row)  
 *else*:  
 empty\_row = [0 *for* x *in* range(a\_max)]  
 mtx.append(empty\_row)  
  
 newMatrix = []  
 *for* i *in* range(a\_max):  
 row = []  
 *for* j *in* range(b\_min):  
 row.append(self.matrix[i][j] + mtx[i][j])  
 *if* b\_min < b\_max:  
 *for* j *in* range(b\_min, b\_max):  
 *try*:  
 row.append(self.matrix[i][j])  
 *except*:  
 row.append(mtx[i][j])  
  
 newMatrix.append(row)  
 *return* newMatrix  
  
 *def* transponentMatrix(self):  
 newMatrix = []  
 *for* i *in* range(len(self.matrix[0])):  
 newMatrix.append([0 *for* x *in* range(len(self.matrix))])  
  
 *for* i *in* range(len(self.matrix)):  
 *for* j *in* range(len(self.matrix[0])):  
 newMatrix[j][i] = self.matrix[i][j]  
  
 *return* newMatrix  
  
 *def* getMatrix(self):  
 m = []  
 *for* row *in* self.matrix:  
 newRow = []  
 *for* i *in* row:  
 newRow.append(i)  
 m.append(newRow)  
 *return* m

Приклади тестів

traingel = [  
 [1, 0, 0, ],  
 [0, 1, 0, ],  
 [0, 0, 1, ],  
 [9, 9, 9, ],  
]  
  
traingelM = Matrix(traingel)  
traingelM.printMatrix()  
print(traingelM.isDiagonal())  
print(traingelM.isTrangle())  
  
diagonal = [  
 [1, 2, 3, 4,],  
 [0, 1, 4, 5,],  
 [0, 0, 1, 6,],  
]  
  
diagonalM = Matrix(diagonal)  
diagonalM.printMatrix()  
print(diagonalM.isDiagonal())  
print(diagonalM.isTrangle())  
print(diagonalM.transponentMatrix())  
print(diagonalM.addMatrix(traingelM))  
print(diagonalM.transponentMatrix())  
print('=' \* 20)  
randomMatrix = Matrix()  
randomMatrix.printMatrix()  
print(randomMatrix.isDiagonal())  
print(randomMatrix.isTrangle())  
print(randomMatrix.addMatrix())  
print(randomMatrix.transponentMatrix())

Вивід

1 0 0

0 1 0

0 0 1

9 9 9

False

False

1 2 3 4

0 1 4 5

0 0 1 6

False

True

[[1, 0, 0], [2, 1, 0], [3, 4, 1], [4, 5, 6]]

[[2, 2, 3, 4], [0, 2, 4, 5], [0, 0, 2, 6], [9, 9, 9, 0]]

[[1, 0, 0, 0], [2, 1, 0, 0], [3, 4, 1, 0], [4, 5, 6, 0]]

====================

75 27 89

7 -84 5

-60 11 -9

False

False

[[76, -22, 131], [65, -68, -45], [-54, 6, -101]]

[[75, 7, -60], [27, -84, 11], [89, 5, -9]]