**import** numpy **as** np  
*#Создайте класс Matrix размерности n на m.  
# В классе объявите массив и переменные (максимальный элемент, минимальный элемент и их позиции в массиве).  
# Создайте конструктор, в котором два входных параметра n-количество строк и m-количество столбцов.  
# В этом конструкторе происходит инициализация матрицы случайными числами. Реализуйте следующие методы:***class** Matrix:  
 **def** \_\_init\_\_(self,sizeN,sizeM):  
 self.sizeN = sizeN  
 self.sizeM = sizeM  
 self.arr = np.random.randint(1, 100, size=(self.sizeN, self.sizeM))  
 self.Max = [0,0,0] *# (n,x,y)* self.Min = [0,0,0] *# (n,x,y)* **def** draw(self):  
 print(self.arr)  
  
 **def** addition(a,b):  
 print(**'Сумма матриц равна'**)  
 print(a+b)  
 **return** a+b  
  
 **def** subtraction(a,b):  
 print(**'Разность матриц равна'**)  
 print(a-b)  
 **return** a-b  
  
 **def** multiplication(a,b):  
 print(**'Произведение матриц равно'**)  
 print(a\*b)  
 **return** a\*b  
  
 **def** transpose(a):  
 print(**'Транспонирование матрицы равно'**)  
 print(a.transpose())  
 **return** a.transpose()  
  
 **def** multiN(self,n):  
 print(**'Результат умножения на число '**,n)  
 print(self.arr \* n)  
 **return** self.arr \* n  
  
 **def** findMaxMin(self):  
 self.Max[0] = max(map(max, self.arr))  
 self.Min[0] = min(map(min, self.arr))  
 **for** i **in** range(len(self.arr)):  
 **for** j **in** range(len(self.arr[i])):  
 **if** self.arr[i][j] == self.Max[0]:  
 self.Max[1] = i  
 self.Max[2] = j  
 **if** self.arr[i][j] == self.Min[0]:  
 self.Min[1] = i  
 self.Min[2] = j  
 print(**'Нахождение максимального и минимального элемента'**)  
 print(**'Max:'**,self.Max)  
 print(**'Min:'**, self.Min)  
  
 **def** change(self):  
 self.arr[self.Max[1]][self.Max[2]] = self.Min[0]  
 self.arr[self.Min[1]][self.Min[2]] = self.Max[0]  
 print(**'Поменяли местами max и min'**)  
 print(self.arr)  
  
m1 = Matrix(5,5)  
m2 = Matrix(5,5)  
print(**'Задание 1.1 Метод, выводящий на экран матрицу. В метод передается объект'**)  
print(**'Первая матрица'**)  
m1.draw()  
print(**'Вторая матрица'**)  
m2.draw()  
print(**'---------------------------------------------------------------------------'**)  
print(**'Задание 1.2 Методы, реализующие сложение, вычитание, умножение и транспонирования матриц. '**)  
print(**'В методах передается и возвращается объект.'**)  
Matrix.addition(m1.arr,m2.arr)  
Matrix.subtraction(m1.arr,m2.arr)  
Matrix.multiplication(m1.arr,m2.arr)  
Matrix.transpose(m1.arr)  
print(**'---------------------------------------------------------------------------'**)  
print(**'Задание 1.3 Метод, реализующий умножение матрицы на число.'**)  
print(**'В метод передается число и возвращается объект.'**)  
print(**'Исходная матрица'**)  
m1.draw()  
m1.multiN(5)  
print(**'---------------------------------------------------------------------------'**)  
print(**'Задание 1.4 Методы реализующие, нахождение максимального и минимального элемента.'**)  
print(**'Методы без входных и выходных параметров.'**)  
print(**'Исходная матрица'**)  
m1.draw()  
m1.findMaxMin()  
print(**'---------------------------------------------------------------------------'**)  
print(**'Задание 1.5 Метод, меняющий местами максимальный и минимальный элементы.'**)  
print(**'Метод без входных и выходных параметров.'**)  
print(**'Исходная матрица'**)  
m1.draw()  
m1.change()  
print(**'---------------------------------------------------------------------------'**)  
*#Определить класс Vector размерности n. Определить несколько конструкторов.  
# Реализовать методы для вычисления модуля вектора, скалярного произведения,  
# сложения, вычитания, умножения на константу,  
# нахождения максимального и минимального элементов; сортировки элементов.***class** Vector:  
 **def** \_\_init\_\_(self,sizeN):  
 self.sizeN = sizeN  
 self.arr = np.random.randint(1, 100, size=self.sizeN)  
  
 **def** draw(self):  
 print(self.arr)  
  
 **def** addition(a, b):  
 print(**'Сумма векторов равна'**)  
 print(a + b)  
 **return** a + b  
  
 **def** subtraction(a, b):  
 print(**'Разность векторов равна'**)  
 print(a - b)  
 **return** a - b  
  
 **def** multiplication(a, b):  
 print(**'Произведение векторов равно'**)  
 print(a \* b)  
 **return** a \* b  
  
 **def** multiN(self, const):  
 print(**'Результат умножения на константу '**, const)  
 print(self.arr \* const)  
 **return** self.arr \* const  
  
v1 = Vector(5)  
v2 = Vector(5)  
v1.draw()  
v2.draw()  
print(**'Сложение векторов'**)  
print(**'Исходные векторы'**)  
v1.draw()  
v2.draw()  
Vector.addition(v1.arr,v2.arr)  
Vector.addition(v1.arr,v2.arr)  
*#В следующих заданиях требуется создать суперкласс (абстрактный класс, интерфейс) и определить общие методы для данного класса. Создать подклассы, в которых добавить специфические свойства и методы.  
# Часть методов переопределить. Объекты подклассов идентифицировать конструктором по имени или идентификационному номеру.  
# Использовать объекты подклассов для моделирования реальных ситуаций и объектов.*