Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчёт**

***по Лабораторной работе №22***

по дисциплине ***«Языки программирования»***

Исполнитель: Колосов Артём Александрович

Группа: 10701219

Преподаватель: Лапанович Ирина Олеговна

2020

Цель работы:

Практически закрепить использования высокоуровневых типов данных языка

Python 3 на примере разработки эффективных алгоритмов для обработки ма-

тематических моделей данных, представленных в виде вектора и/или матрицы.

Основное задание:

1) Согласно своему варианту необходимо реализовать на языке Python 3 только ВТОРОЕ ЗАДАНИЕ по матрицам, которое было дано на лабораторной работе

«Массивы» по параллельной специализированной дисциплине «Основы алгоритмизации и программирования» за первый семестр первого курса обучения.

Дана целочисленная прямоугольная матрица

       1) Определить номер строки и столбца максимального элемента.

       2) Осуществить циклический сдвиг элементов прямоугольной матрицы на n элементов вправо и вниз (n вводит пользователь).

2) Напишите программу транспонирования заданной квадратной матрицы вещественных чисел.

Индивидуальное задание:

1) Найти число локальных минимумов (соседями элемента матрицы назовем

элементы, имеющие с ним общую сторону или угол; элемент матрицы назы-

вается локальным минимумом, если он строго меньше всех своих соседей).

2) Перестроить заданную матрицу, переставляя в ней столбцы так, чтобы зна-

чения их характеристик убывали (характеристикой столбца прямоугольной

матрицы называется сумма модулей его элементов).

3) Путем перестановки элементов квадратной вещественной матрицы до-

биться того, чтобы ее максимальный элемент находился в левом верхнем

углу, следующий по величине – в позиции (2,2), следующий по величине – впозиции (3,3) и т. д., заполнив таким образом всю главную диагональ.

4) Найти наибольший среди локальных максимумов (элемент матрицы называ-

ется локальным максимумом, если он строго больше всех своих соседей).

5) Поменять местами строку, содержащую элемент с наибольшим значением в

матрице со строкой, содержащей элемент с наименьшим значением. Выве-

сти на экран полученную матрицу. Для каждой строки с нулевым элементом

на главной диагонали вывести ее номер и значение наибольшего из элемен-

тов этой строки.

6) Определить, является ли матрица симметричной (относительно главной диа-

гонали). Вывести на экран соответствующее сообщение. Найти максималь-

ный элемент среди стоящих на главной и побочной диагонали и поменять

местами с элементом, стоящим на пересечении этих диагоналей.

7) Выведите номера столбцов, все элементы, которых четны. Для каждого

столбца с отрицательным элементом на главной диагонали вывести его но-

мер и сумму всех элементов этого столбца.

8) Среди столбцов заданной матрицы, содержащих только такие элементы, ко-

торые по модулю не больше n, найти столбец с минимальным произведе-

нием элементов.

9) Определить, становится ли симметричной (относительно главной диагонали)

заданная матрица после замены на число 0 каждого локального минимума.

Элемент матрицы называется локальным минимумом, если он строго

меньше всех имеющихся у него соседей. Соседями элемента ajj в матрице

назовем элементы aki с i-1ki+1, j-1lj+1,(k,l)(i,j).

10) Вычислить определитель матрицы. Вычислить норму матрицы.

11) Определить, становится ли симметричной (относительно главной диагонали)

заданная матрица после замены на число 0 каждого локального максимума.

Элемент матрицы называется локальным максимумом, если он строго

больше всех имеющихся у него соседей. Соседями элемента ajj в матрице

назовем элементы aki с i-1ki+1, j-1lj+1,(k,l)(i,j).

12) Выведите номера столбцов, элементы каждой из которых образуют моно-

тонную последовательность (монотонно убывающую или монотонно воз-

растающую).

13) Найти максимальное из чисел, встречающихся в заданной матрице ровнодва раза.

14) Найти строку заданной матрицы, в которой длина максимальной серии (по-

следовательности одинаковых элементов) минимальна.

15) Среди строк заданной матрицы, содержащих только нечетные элементы,

найти строку с максимальной суммой модулей элементов.

Требования к выполнению:

1) Необходимо спроектировать блок-схемы алгоритмов решений соответствую-

щих заданий согласно своему варианту (варианты назначаются преподавате-

лем или выбираются самостоятельно студентом, но ни один вариант не дол-

жен повторяться у студентов из одной группы) и на базе данных алгоритмов

разработать интерактивные консольные приложения с использованием архи-

тектурного шаблона проектирования ***MVC***.

2) Для каждого алгоритма решения задания оценить (вычислить) алгоритмиче-

скую сложность по необходимому времени и затрачиваемой памяти исполь-

зуя определение ***Big O***.

3) Для масштабируемости разрабатываемого программного решения размер

последовательности (списка) и его элементы должны задаваться пользовате-

лем во время выполнения программы или с помощью генератора псевдослу-

чайных чисел.

4) Для автоматизации заполнения различными значениями искомого контей-

нера рекомендуется использовать соответствующие функции генерирования

псевдослучайных чисел модуля ***random***.

5) Рекомендуется избегать использования глобальных переменных при написа-

нии основной логики приложения.

6) Если логически не подразумевается или в заданиях иного не указано, то вход-

ными и выходными данными являются вещественные числа (числа с плаваю-

щей запятой).

7) Все программы должны быть разбиты на отдельные функции. Среди данных

функций рекомендуется добавлять стартовую функцию ***main***, с которой лучше

производить запуск программы.

8) При выполнении заданий необходимо по максимуму пытаться разрабатывать

универсальный, масштабируемый, легко поддерживаемый и читаемый код.

9) Также рекомендуется придерживаться ***Single Responsibility Principle****,* ***SRP***

(принципа единственной ответственности) – постарайтесь вынести основную

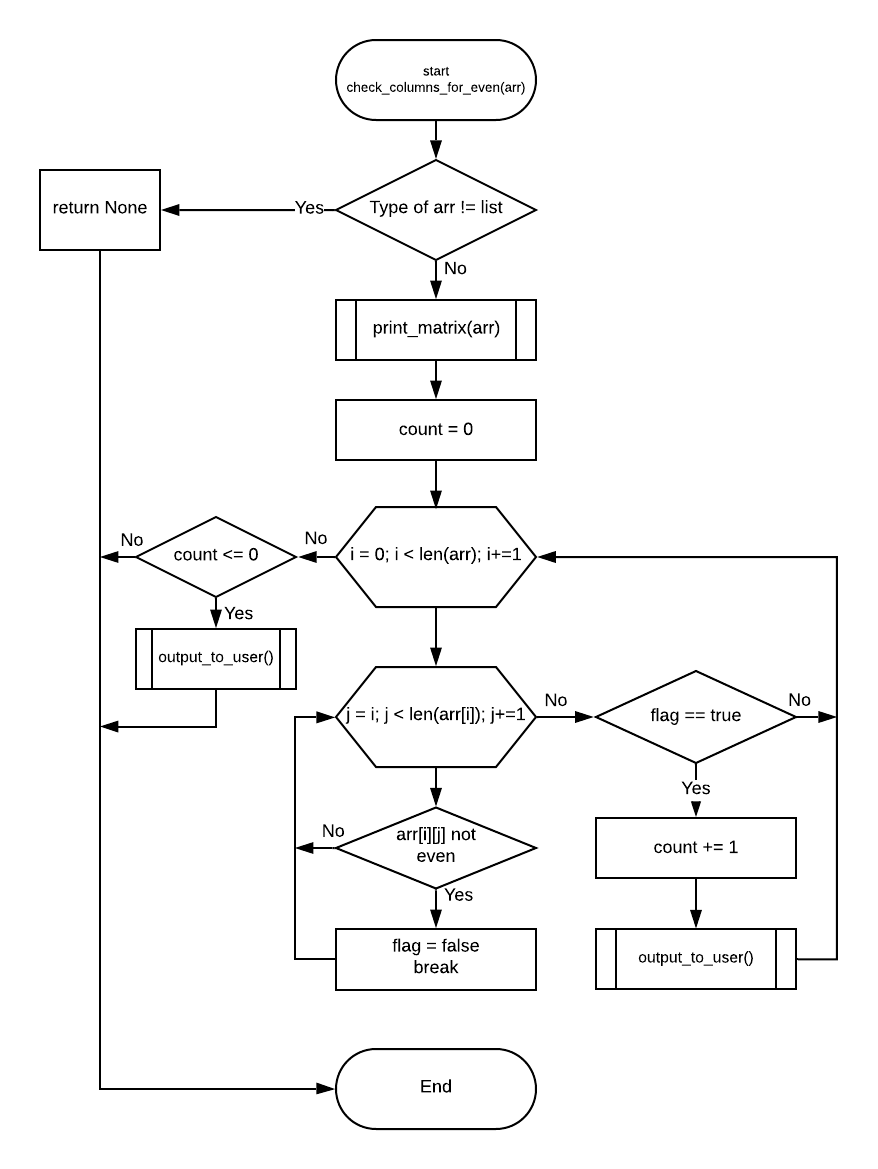
бизнес-логику задания в отдельную функцию (т.е. архитектура приложения

должна минимум состоять из нескольких функций).

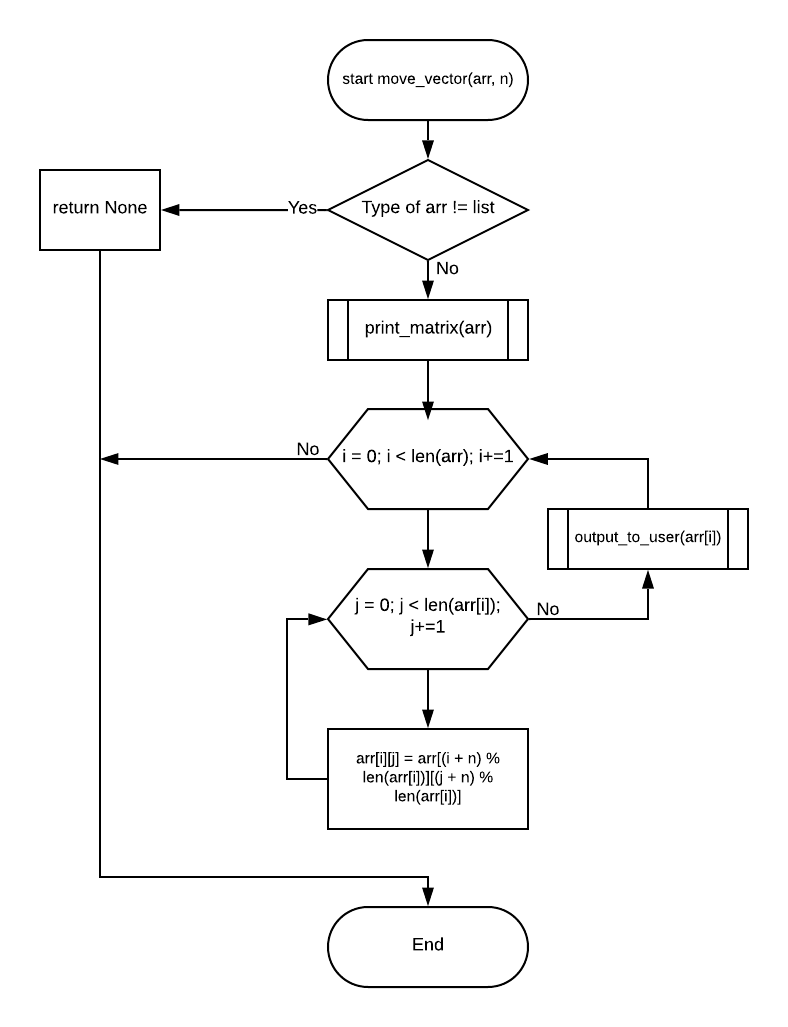
10) В соответствующих компонентах (функциях) бизнес-логики необходимо

предусмотреть «защиту от дурака».

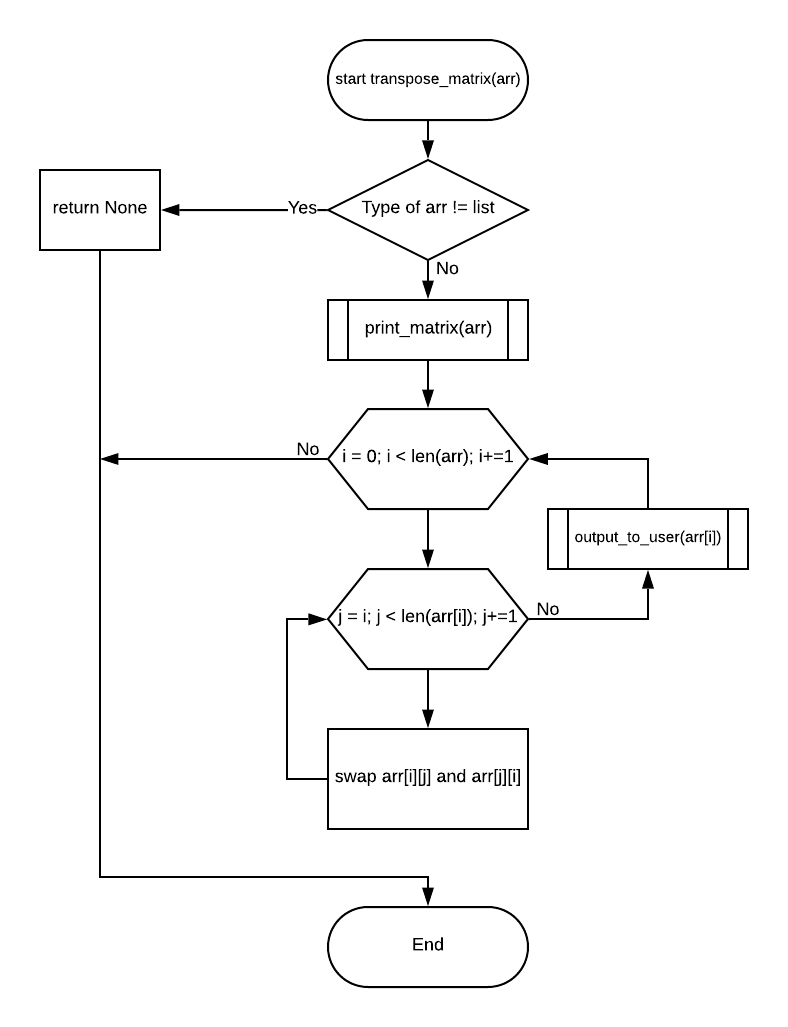
**Блок-схемы основных алгоритмов задания:**



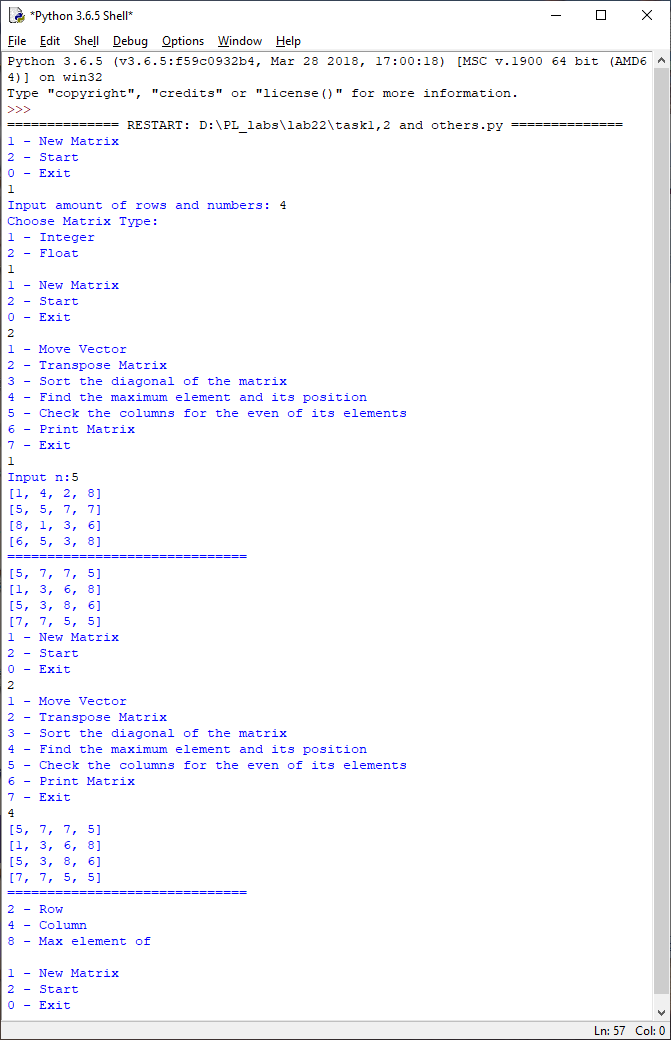
**Рис 22.1 Блок-схема функции check\_columns\_for\_even**



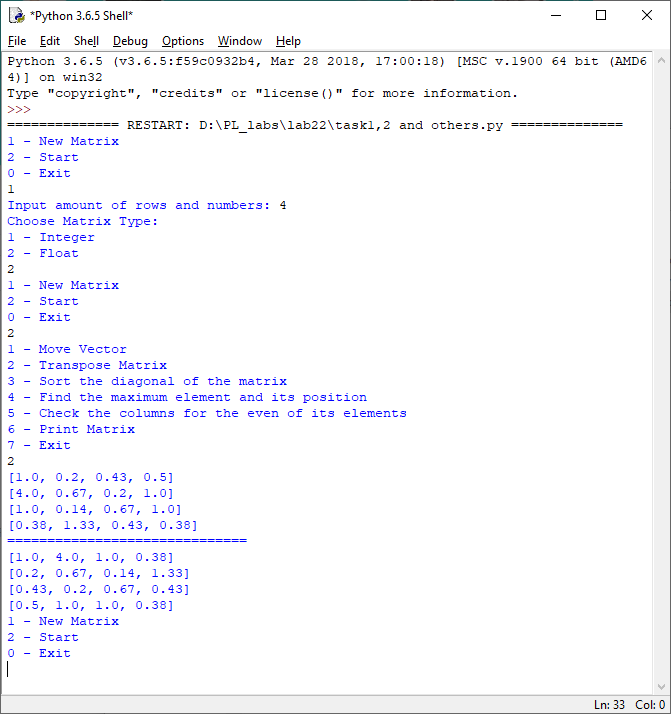
**Рис 22.2 Блок-схема функции move\_vector**



**Рис 22.3 Блок-схема функции move\_vector**

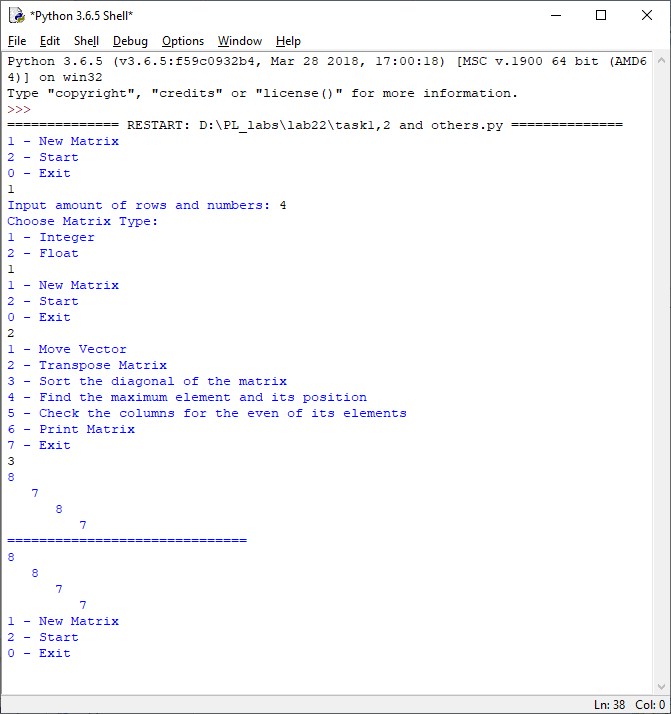


**Рис 22.4 Результаты выполнения первого основного задания**

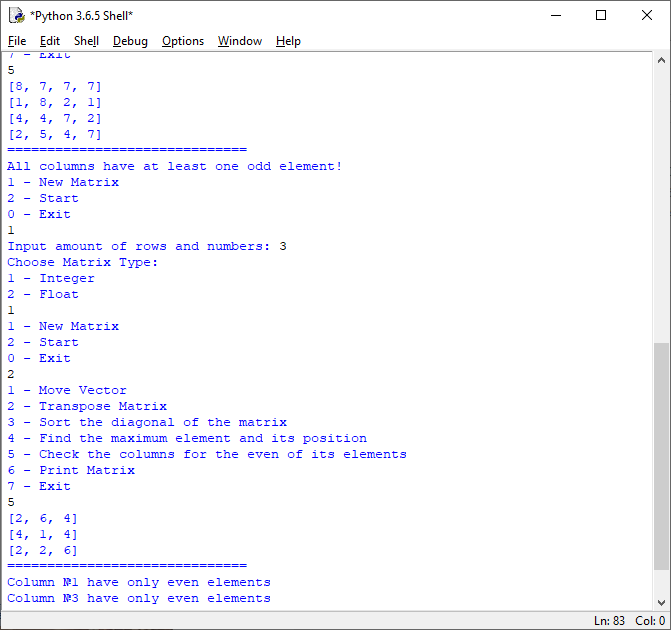
****

**Рис 22.5 Результаты выполнения второго основного задания**

Результаты выполнения индивидуального задания:



**Рис 22.6 Результаты выполнения индивидуального задания**



**Рис 22.7 Результаты выполнения индивидуального задания**

Оценка алгоритмов по BigO:

Алгоритм check\_columns\_for\_even:

Худшее время O(n^2)

Лучшее время O(n^2)

Среднее время O(n^2)

Затраты памяти O(1)

Алгоритм transpose\_matrix:

Худшее время О(n^2)

Лучшее время О(n^2)

Среднее время О(n^2)

Затраты памяти О(1)

Сортировка вставками:

Худшее время О(n^2) сравнений, обменов

Лучшее время O(n^2) сравнений, O(n^2) обмен

Среднее время О(n^2) сравнений, обменов

Затраты памяти O(1)

Что я освоил в процессе выполнения лабораторной работы (выводы):

Научился разрабатывать эффективные алгоритмы с использованием матриц. Закрепил умения работать со сложными типами данных в языке программирования Python.

*ПРИЛОЖЕНИЕ A*

Листинг исходных кодов программ

from random import randint

# Выполнение различных действий с матрицами

# Version: 1.0

# Group: 10701219

# Author: Колосов Артём Александрович

# Date: 2.28.2020

def output\_to\_user(msg, s='', e='\n'):

print(msg, sep=s, end=e)

def get\_input(msg):

return input(str(msg))

def random\_float\_matrix(i=10, j=10):

return [[round(randint(1, j) / randint(1, j \* 2), 2) for \_ in range(j)] for \_ in range(i)]

def random\_int\_matrix(i=10, j=10):

return [[randint(1, j \* 2) for \_ in range(j)] for \_ in range(i)]

def found\_max\_element(arr):

if not isinstance(arr, list):

return None

m\_i = 0

m\_j = 0

maximum = arr[0][0]

for i in range(len(arr)):

for j in range(len(arr[i])):

if arr[i][j] > maximum:

maximum = arr[i][j]

m\_i = i

m\_j = j

return m\_i + 1, m\_j + 1, maximum

def var\_to\_int(var):

if isinstance(var, str) and var.isdigit():

var = int(var)

else:

var = None

return var

def print\_matrix\_diagonal(arr, splitter=True):

if not isinstance(arr, list):

return None

for i in range(len(arr)):

output\_to\_user("{0}{1}".format(' ' \* i, arr[i][i]))

if splitter:

output\_to\_user('=' \* 30)

def print\_matrix(arr, splitter=True):

if not isinstance(arr, list):

return None

for i in range(len(arr)):

output\_to\_user(arr[i])

if splitter:

output\_to\_user("=" \* 30)

def move\_vector(arr, n):

if not isinstance(arr, list):

return None

print\_matrix(arr)

for i in range(len(arr)):

for j in range(len(arr[i])):

arr[i][j] = arr[(i + n) % len(arr[i])][(j + n) % len(arr[i])]

output\_to\_user(arr[i])

def transpose\_matrix(arr):

if not isinstance(arr, list):

return None

print\_matrix(arr)

for i in range(len(arr)):

for j in range(i, len(arr[i])):

arr[i][j], arr[j][i] = arr[j][i], arr[i][j]

output\_to\_user(arr[i])

def check\_columns\_for\_even(arr):

if not isinstance(arr, list):

return None

print\_matrix(arr)

count = 0

for i in range(len(arr)):

flag = True

for j in range(len(arr[i])):

if arr[j][i] % 2 != 0:

flag = False

break

if flag:

count += 1

output\_to\_user("Column №{0} have only even elements".format(i + 1))

if count <= 0:

output\_to\_user("All columns have at least one odd element!")

def sort\_matrix\_diagonal(arr):

if not isinstance(arr, list):

return None

print\_matrix\_diagonal(arr)

for i in range(len(arr) - 1):

for j in range(len(arr) - 1):

if arr[j][j] < arr[j + 1][j + 1]:

arr[j][j], arr[j + 1][j + 1] = arr[j + 1][j + 1], arr[j][j]

print\_matrix\_diagonal(arr, False)

def main():

arr = []

while True:

command = input("1 - New Matrix\n"

"2 - Start\n"

"0 - Exit\n")

if command == '1':

n = var\_to\_int(get\_input("Input amount of rows and numbers: "))

if n:

command = get\_input("Choose Matrix Type:\n"

"1 - Integer\n"

"2 - Float\n")

if command == '1':

arr = random\_int\_matrix(n, n)

elif command == '2':

arr = random\_float\_matrix(n, n)

else:

arr = random\_int\_matrix(n, n)

else:

output\_to\_user("Error: invalid input")

elif command == '2':

if isinstance(arr, list) and arr is not []:

command = get\_input("1 - Move Vector\n"

"2 - Transpose Matrix\n"

"3 - Sort the diagonal of the matrix\n"

"4 - Find the maximum element and its position\n"

"5 - Check the columns for the even of its elements\n"

"6 - Print Matrix\n"

"7 - Exit\n")

if command == '1':

n = var\_to\_int(get\_input("Input n:"))

move\_vector(arr, 1)

elif command == '2':

transpose\_matrix(arr)

elif command == '3':

sort\_matrix\_diagonal(arr)

elif command == '4':

print\_matrix(arr)

new\_arr = found\_max\_element(arr)

output\_to\_user("{0} - Row\n"

"{1} - Column\n"

"{2} - Max element of\n".format(\*new\_arr))

elif command == '5':

check\_columns\_for\_even(arr)

elif command == '6':

print\_matrix(arr, splitter='=')

elif command == '7':

break

elif command == '0':

break

if \_\_name\_\_ == "\_\_main\_\_":

main()