МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Информатика»

Тема: Основные управляющие конструкции языка Python

Студентка гр. 2381 _____ Потапова Д.М.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2022

Цель работы.

Изучить основные управляющие конструкции языка Python, включая модуль numpy.

Задание.

Вариант 2. Вариант лабораторной работы состоит из 3 задач, оформите каждую задачу в виде отдельной функции согласно условиям задач. Приветствуется использование модуля numpy, в частности пакета numpy.linalg. Вы можете реализовывать вспомогательные функции, главное — использовать те же названия основных функций, что требуются в задании. Сами функции вызывать не надо, это делает за вас проверяющая система.

Задача 1. Содержательная постановка задачи

Дакибот приближается к перекрестку. Он знает 4 координаты, соответствующие координатам углов перекрестка (координаты образуют прямоугольник), и свои координаты. По правилам движения дакибот должен остановиться сразу, как только оказывается на перекрестке. Ваша задача — помочь дакиботу понять, находится ли он на перекрестке (внутри прямоугольника).

Формальная постановка задачи

Оформите задачу как отдельную функцию: *def check_rectangle(robot, point1, point2, point3, point4).* На вход функции подаются: координаты дакибота *robot* и координаты точек, описывающих перекресток: *point1, point2, point3, point4*. Точка — это кортеж из двух целых чисел (x, y). Функция должна возвращать **True**, если дакибот на перекрестке, и **False**, если дакибот вне перекрестка.

Задача 2. Содержательная часть задачи

Несколько дакиботов прибыли на базу, но их корпуса оказались поврежденными. В логах ботов программисты нашли сведения про их траектории движения, которые задаются линейными уравнениями вида: ax+by+c=0. В логах хранятся коэффициенты этих уравнений a, b, c.

Ваша задача — вывести список номеров ботов (кортежи), которые столкнулись с друг другом (боты нумеруются с нуля, порядок следования коэффициентов уравнений соответствует порядку ботов).

Формальная постановка задачи

Оформите решение в виде отдельной функции *check_collision()*. На вход функции подается матрица **ndarray Nx3** (N — количество ботов, может быть разным в разных тестах) коэффициентов уравнений траекторий *coefficients*. Функция возвращает список пар — номера столкнувшихся ботов (если никто из ботов не столкнулся, возвращается пустой список).

Задача 3. Содержательная часть задачи

При перемещении по дакитауну дакибот должен регулярно отправлять на базу сведения, среди которых есть длина пройденного пути. Дакиботу известна последовательность своих координат (x, y), по которым он проехал. Ваша задача — помочь дакиботу посчитать длину пути.

Формальная постановка задачи

Оформите задачу как отдельную функцию *check_path()*, на вход которой передается последовательность (список) двумерных точек (пар) *points_list*. Функция должна возвращать число — длину пройденного дакиботом пути (выполните округление до 2 знака с помощью *round(value, 2))*.

Основные теоретические положения.

Задача 2: используется метод *numpy.linalg.matrix_rank* для нахождения ранга матрицы.

Задача 3: используется метод *numpy.linalg.norm* для нахождения длины вектора.

Выполнение работы.

Подключается модуль numpy.

Функция *check_rectangle(robot, point1, point2, point3, point4)*.

Функция принимает на вход 5 кортежей с координатами дакибота и точек, которые ограничивают перекресток. Проверяя условие нахождения дакибота в прямоугольнике, образующимся точками — координата дакибота по оси X больше x-координаты точки 1, но меньше x-координаты точки 4; координата дакибота по оси Y больше y-координаты точки 1, но меньше y-координаты точки 2 — функция возвращает **True** (дакибот на перекрестке) или **False** (дакибот не на перекрестке).

Функция *check_collision()*.

Функция принимает на вход матрицу *ndarray Nx3* (N — количество ботов) коэффициентов уравнения траекторий *coefficients*. Циклами for перебираются все возможные пары коэффициентов уравнений траекторий. В каждом прохождении циклов создаются 2 матрицы. Первая (*mat1*) — матрица коэффициентов a и b, вторая (*mat2*) — расширенная матрица коэффициентов (a, b, c). Для поиска столкновений дакиботов, т.е. решений систем полученных уравнений, находим ранги матриц методом *np.linalg.matrix_rank()*. Если ранг основной матрицы системы равен рангу расширенной матрицы, то система имеет единственной решение, а значит дакиботы врезаются и их номера добавляются в результирующий список *rez[]*. Функция возвращает отсортированный методом *sorted()* список пар (кортежей номеров столкнувшихся дакиботов). Если никто не столкнулся, функция возвращает пустой список.

Функция *check_path()*.

Функция принимает на вход последовательность (список) двумерных точек (пар) *points_list*. Циклом for перебираются все пары координат и формируются соответствующие им вектора, по которым движется дакибот. С помощью метода *numpy.linalg.norm(ord=2)* находим длину вектора и суммируем ее с переменной *rez*. Функция возвращает округленное с помощью метода *round(value, 2)* число *rez* — длину пройденного дакиботом пути.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	(9, 3) (14, 13) (26, 13) (26, 23) (14, 23)	False	Результат работы функции — False, значит дакибот находится за границами перекрестка.
2.	[[-1 -4 0] [-7 -5 5] [1 4 2] [-5 2 2]]	[(0, 1), (0, 3), (1, 0), (1, 2), (1, 3), (2, 1), (2, 3), (3, 0), (3, 1), (3, 2)]	Результат работы функции — массив пар столкнувшихся дакиботов.
3.	[(1.0, 2.0), (2.0, 3.0)]	1.41	Результат работы функции — длина пути, пройденного дакиботом.

Выводы.

Были изучены основные управляющие конструкции языка Python и методы библиотеки numpy. С помощью полученных знаний были составлены функции для решения практических задач, с которыми можно встретиться в Лаборатории Автономных Транспортных Систем, а именно: функция проверки нахождения дакибота на перекрестке, функция нахождения номеров столкнувшихся роботов и функция нахождения пути, пройденного дакиботом.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_lb1.py

```
    def check_crossroad(robot, point1, point2, point3, point4):
        x, y = robot[0], robot[1]
        x1, y1 = point1[0], point1[1]
        x2, y2 = point2[0], point2[1]
        x3, y3 = point3[0], point3[1]
        x4, y4 = point4[0], point4[1]
        if (x1 <= x <= x2) and (y1 <= y <= y4):
            return True
        else:
            return False


    def check_collision(coefficients):
        rez = []
        for i in range(0, len(coefficients)):
            for j in range(i + 1, len(coefficients)):
                mat1 = np.array([coefficients[i][:2], coefficients[j]
[:2]])
                coefficients[i][2] = coefficients[i][2] * (-1)
                coefficients[j][2] = coefficients[j][2] * (-1)
                mat2 = np.array([coefficients[i], coefficients[j]])
                                    if  np.linalg.matrix_rank(mat1)  ==
np.linalg.matrix_rank(mat2):
                    rez.append((i, j))
                    rez.append((j, i))
        return sorted(rez)


    def check_path(points_list):
        rez = 0
        for i in range(0, len(points_list) - 1):
            s1 = np.array(points_list[i])
            s2 = np.array(points_list[i + 1])
            rez += np.linalg.norm(s2 - s1, ord=2)
        return round(rez, 2)
```