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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Информатика»

Тема: Основные управляющие конструкции языка Python

Студентка гр. 2381 _____ Газукина Д.Д.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2022

Цель работы.

Изучить основные управляющие конструкции языка Python, 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 (np.linalg.matrix_rank) для нахождения ранга матрицы и метод numpy.hstack (np.hstack) для объединения двух матриц (добавления столбцов).

Задача 3: используется метод numpy.linalg.norm (np.linalg.norm) для нахождения длины вектора.

Выполнение работы.

Подключается модуль numpy.

Функция check_crossroad(robot, point1, point2, point3, point4).

Принимает на вход 5 кортежей — координаты дакибота и углов перекрестка. Проверяем нахождение робота в перекрестке с помощью сравнения первого элемента первого кортежа (x-координаты робота): он должен быть больше либо равен x-координате первой точки и меньше либо равен x-координате третьей точки, второго элемента первого кортежа (y-координаты робота): он должен быть больше либо равен y-координате второй точки и меньше либо равен y-координате четвертой точки. Если условия выполняются, функция возвращает True, иначе False.

Функция check_collision(coefficients).

Функция принимает на вход матрицу ndarray Nx3 (N — количество ботов) коэффициентов уравнения траекторий coefficients. Циклами (используется вложенный цикл) перебираются все пары уравнений для двух произвольных роботов. Создаются матрицы — AB (с коэффициентами a и b), и ABC — расширенная матрица коэффициентов, которая создается с помощью метода numpy.hstack((AB, C)). C — матрица с последними элементами каждой строки входной матрицы. Для поиска столкновений дакиботов находим ранги матриц с помощью метода np.linalg.matrix_rank(). Если они равны, значит есть единственное решение системы, следовательно, дакиботы столкнулись. Записываем их номера в список result[], который после прохождения циклов и сортировки этого списка возвращает функция.

Функция check_path(points_list).

На вход функции подается список двумерных точек points_list. С помощью цикла for перебираются последовательные пары координат. С помощью метода numpy.linalg.norm находится длина вектора, то есть длина пути от одной точки (A) до соседней (B), она прибавляется к переменной

result (длина всего пути), которая затем округляется до двух знаков после запятой и возвращается функцией.

Разработанный программный код см. в приложении А.

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	Дакибот за границами перекрестка.
2.	(5, 8) (0, 3) (12, 3) (12, 16) (0, 16)	True	Дакибот в пределах перекрестка.
3.	[[-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)]	Выводится массив пар столкнувшихся дакиботов.
4.	[(1.0, 2.0), (2.0, 3.0)]	1.41	Выводится длина пути, который прошел дакибот.
5.	[(2.0, 3.0), (4.0, 5.0)]	2.83	Выводится длина пути, который прошел дакибот.

Выводы.

Были изучены основные управляющие конструкции языка Python и модуль numpy. С помощью полученных знаний были составлены функции для решения практических задач: функция для проверки, находится ли дакибот на перекрестке, функция нахождения столкнувшихся дакиботов и функция нахождения длины пути, пройденного дакиботом.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#import numpy as np


def check_crossroad(robot, point1, point2, point3, point4):
    if robot[0] >= point1[0] and robot[0] <= point3[0]\
            and robot[1] >= point2[1] and robot[1] <= point4[1]:
        return True
    else:
        return False


def check_collision(coefficients):
    N = coefficients.shape[0]
    result = []
    for i in range(N):
        for j in range(i+1, N):
            AB = np.array([coefficients[i][:2], coefficients[j]
[:2]])
            C = np.array([[coefficients[i][2]],[coefficients[j]
[2]]])
            ABC = np.hstack((AB, C))
            rank_AB = np.linalg.matrix_rank(AB)
            rank_ABC = np.linalg.matrix_rank(ABC)
            if rank_AB == rank_ABC:
                result.append((i, j))
                result.append((j, i))
    result = sorted(result)
    return (result)


def check_path(points_list):
    result = 0.0
    for i in range(len(points_list) - 1):
        A = np.array(points_list[i])
        B = np.array(points_list[i+1])
        result += np.linalg.norm(B - A, ord=2)
    return (np.round(result, 2))
```