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

Студент гр. 3342 _____ Галеев А.Д.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Изучить основные управляющие конструкции языка Python, а так же применение библиотеки NumPy

Задание

Вариант №2

Задача 1

Требуется помочь дакиботу понять, находится ли он на перекрестке (внутри прямоугольника), на вход функции подаются: координаты дакибота *robot* и координаты точек, описывающих перекресток: *point1, point2, point3, point4*. Функция должна возвращать True, если дакибот на перекрестке, и False, если дакибот вне перекрестка.

Задача 2

В логах ботов программисты нашли сведения про их траектории движения, которые задаются линейными уравнениями вида: ax + by + c = 0, ваша задача вывести список номеров ботов, которые столкнулись с друг другом. Функция возвращает список пар столкнувшихся ботов (если никто из ботов не столкнулся, возвращается пустой список).

Задача 3

Дакиботу известна последовательность своих координат (x, y) по которым он проехал, ваша задача помочь дакиботу посчитать длину пути. функция должна возвращать длину пройденного дакиботом пути (выполните округление до 2 знака с помощью *round(value, 2))*.

Основные теоретические положения

Для решения задач использовалась библиотека NumPy которая дает возможность поддержки многомерных массивов и матриц.

Выполнение работы

Задача 1

В функции check_crossroad задаются переменные (robot, point1, point2, point3, point4) содержащие начальные координаты робота и за координаты точек границ перекрестка. По координатам точек границ перекрестка создается массив crossroad, а по координатам робота массив robotcoord. Далее выполняется проверка условия с помощью if, если координаты робота находятся внутри координат точек границ перекрестка, то функция выводит True, в другом случае выводится False.

Задача 2

В функции check_collision задается переменная coefficients, которая содержит в себе матрицу состоящую из коэффициентов a b c для каждого дакибота, эта матрица имеет форму Nx3, где N это количество ботов. Создается пустой список collision_pairs в который потом будут добавляться столкнувшиеся пары. Далее запускаются два цикла for (bot1, bot2), которые будут отвечать за перебор всех пар дакиботов, если bot1 не равен bot2 (для того чтобы не сравнивать одинаковых дакиботов) будет выполнено присвоение переменным a1 b1 c1 коэффициенты бота под номером bot1, а переменным a2 b2 c2 коэффициента бота под номером bot2. С помощью if задаем условие, если a1 * b2 не равно a2 * b1, то уравнения этих ботов не линейно зависимы и у них есть точка пересечения, в collision_pairs будут добавлены номера этих ботов. В конце выводится список collision_pairs.

Задача 3

В функции check_path задается переменная points_list, которая содержит список координат перемещения дакибота. Задается переменная total_distance в которую будет записано расстояние пройденное ботом. Запускается цикл for который перебирает значения от 1 до значения длины списка points_list, далее список points_list конвертируется в массив и переменным point1 и point2 присваиваются значения точек координат под номерами n и n+1, а переменной distance присваивается значение расстояния между этими точками (point1 – point2) найденное с помощью теоремы Евклида (np.linalg.norm). В конце расстояние

distance прибавляется к общему total_distance после чего округляется до 2 знака после запятой с помощью round и выводится в консоль.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	(4, 3) (16, 12) (28, 12) (28, 23) (16, 23)	False
2.	(4, 9) (0, 3) (12, 3) (12, 16) (0, 16)	True
3.	(8, 8) (2, 1) (13, 1) (13, 16) (2, 16)	True

Выводы

Была изучена библиотека NumPy

Разработана программа состоящая из 3 частей, которые используют различные функции и команды из библиотеки NumPy.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_lb1

```
import numpy as np


def check_crossroad(robot, point1, point2, point3, point4):
    crossroad = np.array([point1, point2, point3, point4])
    robotcoord = np.array(robot)
    if (robotcoord[0] >= np.min(crossroad[:, 0]) and robotcoord[0] <=
np.max(crossroad[:, 0]) and
        robotcoord[1] >= np.min(crossroad[:, 1]) and robotcoord[1] <=
np.max(crossroad[:, 1])):
        return True
    else:
        return False

def check_collision(coefficients):
    N = coefficients.shape[0]
    collision_pairs = []
    for bot1 in range(N):
        for bot2 in range(N):
            if bot1 != bot2:
                a1, b1, c1 = coefficients[bot1]
                a2, b2, c2 = coefficients[bot2]
                if a1 * b2 != a2 * b1:
                    collision_pairs.append((bot1, bot2))
    return collision_pairs

def check_path(points_list):
    total_distance = 0.0
    for i in range(1, len(points_list)):
        point1 = np.array(points_list[i-1])
        point2 = np.array(points_list[i])
        distance = np.linalg.norm(point2 - point1)
        total_distance += distance
    return round(total_distance, 2)
```