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. 3344 Коняева М.В.

Преподаватель Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Целью работы является освоение работы с управляющими конструкциями языка Python и одного из его модулей numpy, который позволяет работать с массивами и решать алгебраические задачи.

Задание

Вариант 1. 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. 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).

Задача 2. 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).

Задача 3. 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.

Выполнение работы

Подключается библиотека numpy и math для дальнейшего использования их математических функций.

Первая функция *def check_crossroad(robot, point1, point2, point3, point4)*, которая принимает на вход пять кортежей из двух целых чисел. С помощью функции if проверяем, находится ли координаты робота между пограничными линиями (вертикальной и горизонтальной). При выполнении условия функция возвращает *True*, а при невыполнении - *False*.

Вторая функция *def check_collision(coefficients)* принимает на вход многомерный массив, состоящий из трех столбцов, которые отражают три коэффициента a, b, c. Все роботы движутся по траектории ax+by+c=0. Изначально в функцию вводится переменная string, которая считается с помощью функции *coefficients.shape[0]*, в результате мы получаем количество строк в многомерном массиве *coefficients*. Также инициализируется массив *arr*, который и будет результатом. Далее с помощью цикла *for i in range(string)* и вложенного цикла *for j in range(string)*, где i и j означают номера роботов. Проверяем в цикле разницу между i и j, чтобы не было работы с одинаковыми роботами. Если проверка на это условие пройдено, мы создаем матрицу из двух строк. Далее с помощью функции *np.linalg.matrix_rank(L[:,:2])* проверяем ранг матрицы, если он больше или равен двум то система из двух линейных уравнений(строк) имеет решение, и следовательно дакиботы пересекутся. После проверки условия создаем кортеж duo, состоящий из номеров строк, и добавляем его в конечный массив *arr*. После окончания работы циклов *for* возвращаем *arr*.

Третья функция *def check_path(points_list)* получает на вход массив, состоящий из кортежей, в котором записаны координаты перемещения x и y. Введем переменную res равную нулю. Далее с помощью цикла *for i in range(len(points_list)-1)*, где диапазон – это длина входящего массива. Прибавляем каждый раз к *res* корень из суммы квадратов двух выражений. Первое из которых это разность из двух координат по горизонтали, а второе выражение – по вертикали. Возвращаем переменную *res*.

Разработанный программный код см. в приложении А. Результаты тестирования см. в приложении Б.

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) [[-1 -4 0] [-7 -5 5] [1 4 2] [-2 -8 10]] [(1.0, 2.0), (2.0, 3.0)] [(2.0, 3.0), (4.0, 5.0)]	False [(0, 1), (1, 0), (1, 2), (1, 3), (2, 1), (3, 1)] 2.83	Данные обработаны корректно
2.	(5, 8) (0, 3) (12, 3) (12, 16) (0, 16) [[2 3 4] [4 6 1] [1 2 3] [-2 -8 10]] [(1.0, 2.0), (3.0, 5.0),(4.0, 6.0)]	True [(0, 2), (0, 3), (1, 2), (1, 3), (2, 0), (2, 1), (2, 3), (3, 0), (3, 1), (3, 2)] 5.02	Данные обработаны корректно

Выводы

Были изучены основные управляющие конструкции языка Python, а также была освоена библиотека numpy, с помощью которой были решены задачи линейной алгебры. Разработана программа, выполняющая три различные функции. Задействованы такие функции языка, как циклы *for*, *if*, а также функции из модуля numpy и math.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1.py

```
import numpy as np
import math

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

def check_collision(coefficients):
    string = coefficients.shape[0]
    arr = []
    for i in range(string):
        for j in range(string):
            if (abs(i - j) >= 1):
                L = np.array((coefficients[i],coefficients[j]))
                if np.linalg.matrix_rank(L[:,:2]) >= 2:
                    duo = (i,j)
                    arr.append(duo)
    return arr

def check_path(points_list):
    res = 0
    for i in range(len(points_list)-1):
        res += math.sqrt((points_list[i][0] - points_list[i+1][0])**2 +
(points_list[i][1] - points_list[i+1][1])**2)
    return round(res,2)
```