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: Управляющие конструкции
языка Python

Студент гр. 3344 _____ Анахин Е.Д.

Преподаватель _____ Иванов Д.В.

Санкт-Петербур
г 2023

Цель работы

Освоение работы с основными управляющими конструкциями в языке программирования Python, а также изучение модуля numpy для работы с массивами и алгебраическими функциями.

Задание.

2 вариант. 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).

Задача 2: Содержательная постановка 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).

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.

Выполнение работы

Задача 1: Функция *check_crossroad* получает на вход координаты робота *robot* и координаты 4 точек перекрестка - *point1-4*, а возвращает *true* или *false* в зависимости от того, находится робот на перекрестке или нет. Переменные *x* и *y* - координаты робота. *x_coords* и *y_coords* - переменные, которые хранят максимальные и минимальные значения, которые являются допустимыми для того, чтобы быть на перекресте. Далее проводилось сравнение координат бота и координат перекрестка. В зависимости от результата выполнения неравенств и возвращалось значение.

Задача 2: Функция *check_collision* получает на вход список из уравнений, а возвращает список из кортежей со всеми парами номеров роботов, которые столкнулись. Переменная *ANSWER* - список, который выводится после его заполнения. *coeffs1* и *coeffs2* - переменные, которые хранят коэффициенты a и b для дальнейшего их сравнения. Если *coeffs1[0] / coeffs1[1] == coeffs2[0] / coeffs[1]*, то тогда такие роботы не столкнутся, а значит, их пара не будет добавлена в список *answer*.

Задача 3: Функция *check_path* получает на вход *points_list* - *numpy.array*, а возвращает число - длину пройденного роботом пути. Переменная *length* хранит в себе длину пройденного пути. *x0 y0* и *x1 y1* - переменные, которые хранят в себе координаты, между двумя парама точек последовательности *points_list*. Между двумя ближайшими точками расстояние находится и прибавляются к переменной length по формуле √ (x1 - x0) ** 2 + (y1- y0) ** 2.

Разработанный код смотреть 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) [[-1 -4 0] [-7 -5 5] [1 4 2][-5 2 2]]; [(1.0, 2.0), (2.0, 3.0)]	False [(0, 1), (0, 3), (1, 0), (1, 2), (1, 3), (2, 1), (2, 3), (3, 0), (3, 1), (3, 2)] 1.41	-

Выводы

Была освоена работа с управляющими конструкциями на языке Python, а также были изучены некоторые методы из модуля numpy и применены методы линейной алгебры. Были задействованы циклы for, списки и массивы из модуля numpy и булевые операции.

Исходный код программы

Приложение А

Название файла: Anakhin_Egor_lb1.py

```
import numpy as np


def check_crossroad(robot, point1, point2, point3, point4) -> bool:
    [x, y] = robot
    x_coords = [point1[0], point3[0]]
    y_coords = [point1[1], point3[1]]
     return (x_coords[0] <= x <= x_coords[1]) and (y_coords[0] <= y <=
y_coords[1])


def check_collision(coefficients:np.array) -> list:
    answer = []
    for i in range(len(coefficients)):
        for j in range(len(coefficients)):
            if i != j:
                coeffs1 = coefficients[i]
                coeffs2 = coefficients[j]
                if coeffs1[0] / coeffs1[1] != coeffs2[0] / coeffs2[1]:
                    answer.append((i, j))
    return answer


def check_path(points_list: np.array) -> float:
    length = 0
    for i in range(1, len(points_list)):
        x0, y0 = points_list[i - 1]
        x1, y1 = points_list[i]
        length += ((x1 - x0) ** 2 + (y1 - y0) ** 2) ** 0.5
    return round(length, 2)
```