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 _____ Кузнецов Р.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Изучение и освоение работы с управляющими конструкциями на языке программирования Python.

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. 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

Были импортированы две библиотеки numpy и math.

Функция check_crossroad принимает в себя 5 значений: robot, point1, point2, point3, point4. Первый аргумент - координаты дакиробота, остальные координаты - границы перекрестка, в котором необходимо находится. При помощи переменных rx и ry записываются соответствующие координаты дакибота. Затем в переменных x и y сортируется список координат перекрестка. В условии сравниваются координаты дакибота с максимальными и минимальными координатами перекрестка при помощи неравенства. Функция возращает True или False взависимости от выполнения условия.

Функция check_collision принимает список из уравнений coefficients. Создается список crashed_robots, в который будут записываться столкнувшиеся дакиботы. При помощи двух циклов сравниваются траектории движения дакиботов, посредством деления коэффицентов уравнений. Если отношения коэффицентов будут равны, то пара дакиботов не столкнется, так как их траектории движения параллельны. В противном случае, пара записывается в список crashed_robots. Функция возращает список из кортежей всех столкнувшихся дакиботов.

Функция check_path принимает список двумерных пар point_list. Инициализируется переменная path_length для определения в дальнейшем длину пройденного пути. При помощи цикла, в переменные point1 и point2 записываются две ближайшие координаты, которые прошел дакибот. В переменную path_length добавляется длина вектора между этими координатами. Функция возращает path_length, округленное до 2 чисел после запятой при помощи round.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	(8, 17) (8, 3) (20, 3) (20, 18) (8, 18)	True	-
2.	[[-8 -9 4] [1 -5 1] [-2 -1 6] [-10 3 2] [-1 -9 7] [8 -7 5] [7 -2 9]]	[(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (1, 0), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 0), (2, 1), (2, 3), (2, 4), (2, 5), (2, 6), (3, 0), (3, 1), (3, 2), (3, 4), (3, 5), (3, 6), (4, 0), (4, 1), (4, 2), (4, 3), (4, 5), (4, 6), (5, 0), (5, 1), (5, 2), (5, 3), (5, 4), (5, 6), (6, 0), (6, 1), (6, 2), (6, 3), (6, 4), (6, 5)]	-
3.	[(3.61, 1.4),(3.63, 1.47), (3.57, 1.57), (3.54, 1.72)]	0.34	-

Выводы

Была изучена и освоена работа с управляющими конструкциями на языке программирования Python. При помощи методов линейной алгебры, были выполнены задания лабораторной.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1.py

```
import numpy as np
import math

def check_crossroad(robot, point1, point2, point3, point4):
    rx, ry = robot
    x = sorted([point1[0], point2[0], point3[0], point4[0]])
    y = sorted([point1[1], point2[1], point3[1], point4[1]])
    return x[0] <= rx <= x[3] and y[0] <= ry <= y[3]



def check_collision(coefficients):
    crashed_robots = []
    for i in range(len(coefficients)):
        for j in range(len(coefficients)):
            if         (coefficients[i][0]/coefficients[i][1])         !=
(coefficients[j][0]/coefficients[j][1]):
                    crashed_robots.append((i, j))
    return crashed_robots


def check_path(points_list):
    path_length= 0
    for i in range(1,len(points_list)):
        point1=points_list[i-1]
        point2=points_list[i]
        path_length += math.sqrt((point1[0]-point2[0]) ** 2 + (point1[1]-
point2[1]) ** 2)
    return round(path_length, 2)
```