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. 2381 Комосский Е.А.

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2022

Цель работы.

Изучить основные управляющие конструкции языка Python.

Задание.

Вариант 2.

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 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).

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.

Подключаем библиотеку numpy с псевдонимом np, подмодуль linalg из numpy с псевдонимом LA.

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

Функция проверяет, находится ли робот в пределах заданных координат.

Функция получает на вход координаты робота и точек в виде кортежей. Далее она проверяет, находится ли робот внутри заданного поля, с помощью условия: координата X робота находится в промежутке от минимальной до максимальной координат X точек, далее проверяет то же условие для Y. Если робот внутри, то выводится True, в обратном случае – False.

Функция check_collision(coefficients).

Функция проверяет матрицу коэффициентов уравнений траекторий движений роботов и находит те, которые пересекаются.

Функция получает на вход матрицу ndarray размера Nx3, где N – число ботов. Создаётся массив d, в который будут записываться пары номеров ботов, траектории которых пересекаются. С помощью двойного цикла for идёт перебор пар уравнений. С помощью конструкции try-except мы пытаемся выявить, пересеклись ли боты с помощью функции solve() из подмодуля numpy.linalg, которая считает решение системы линейных уравнений . Если функция нашла корни уравнения, то мы записываем в массив d два кортежа с прямым и обратным порядком номеров ботов. В случае выведения функцией ошибки мы переходим к следующей итерации. По завершению циклов функция возвращает список кортежей с номерами пересекающихся ботов.

Функция check_path(points_list).

Функция находит расстояние, пройденное ботом с помощью теорему Пифагора.

Функция получается на вход кортежей с координатами перемещений робота. Создаётся целочисленная переменная path, в которую будет записываться длина пройденного пути. В цикле for мы проходимся по списку координат и с помощью теоремы Пифагора находим расстояние между ними, используя возведение в степень и функцию sqrt() из numpy для нахождения корня, добавляя

его к path. В конце функция возвращает длину пути, пройденного роботом, округлённую до двух знаков после запятой с помощью функции round().

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	Данные подавались для первой функции, ответ верный, робот находится вне заданного поля.
2.	[[-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)]	Данные подавались для второй функции, на выходе получили массив столкнувшихся роботов.
3.	[(1.0, 2.0), (2.0, 3.0)]	1.41	Данные подавались для третьей функции, на выходе получили верный ответ, округлённый до двух знаков.

Выводы.

Были изучены основные управляющие конструкции Python, функции из библиотеки numpy. Знания были применены на практике для решения задач.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_lb1.py

```
import numpy as np
from numpy import linalg as LA

def check_crossroad(robot, point1, point2, point3, point4):
    return min(point1[0], point2[0])<=robot[0]<=max(point1[0], point2[0]) and min(point2[1], point3[1])<=robot[1]<=(max(point2[1],point3[1]))


def check_collision(coefficients):
    d=[]
    for i in range(len(coefficients)):
        for j in range(i+1, len(coefficients)):
            try:
                LA.solve(np.vstack([coefficients[i][:2], coefficients[j][:2]]), np.hstack([coefficients[i][2], coefficients[j][2]]))
                d.append((i, j))
                d.append((j, i))
            except:
                pass
    return sorted(d)



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