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 Хайруллов Д.Л.

Преподаватель Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Основной целью работы является знакомство с основными управляющими конструкциями языка Python, освоение работы с модулем numpy.

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). Оформите задачу как отдельную функцию: *def check_crossroad(robot, point1, point2, point3, point4)*

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

Решение каждого задания выделено в отдельную функцию.

В функцию check_crossroad подаются координаты робота и координаты четырех точек, ограничивающих прямоугольную область(перекресток). В массив coords_list записываются координаты данных точек. Создаются два массива (coords_x, coords_y), в которые отдельно записываются координаты данных точек по абсциссе и ординате с помощью цикла for. Создается булевая переменная bool_sen, в которой записано выражение, обращающееся в истину при условии нахождения робота в заданной зоне. В зависимости от значения bool_sen выводится True или False.

В функцию check_collision подается массив с коэффициентами уравнения движения для каждого из роботов. В двойном цикле for перебираются всевозможные комбинации уравнений движений роботов. На основе двух строк с коэффициентами составляется матрица, с помощью функции linalg.matrix_rang вычисляется ранг матрицы, которое сравнивается со значением количества строк в матрице. Если ранг матрицы не меньше количества строк в матрице, то соответствующая пара индексов роботов записывается в ранее созданный массив res. Фунцкия выводит res.

В ввод функции check_path подается массив points_list с координатами точек, в которых был робот. В цикле for вычисляются разности соседних в массиве координат по осям абсцисс и ординат, записываемые в переменные x_diff, y_diff. Создается массив с координатами, равными x_diff и y_diff. С помощью функции linalg.norm вычисляется значение модуля данного вектора, которое прибавляется к значению переменной path. После окогнчания работы цикла функция возвращает значение path, округленное 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.	(26, 33), (19, 24), (19, 45), (34, 24), (34, 45) [[-1, -4, 0], [-7, -5, 5], [1, 4, 2], [-5, 2, 2]] [(5.0, 6.0), (8.0, 9.0), (14.0, 15.0)]	True [(0, 1), (0, 3), (1, 0), (1, 2), (1, 3), (2, 1), (2, 3), (3, 0), (3, 1), (3, 2)] 12.73	
2.	(-4, 24), (5, 8), (5, 45), (18, 8), (18, 45) [[5, -3, 0], [13, 2, 0], [-45, 2, 1]] [(5.0, 6.0), (25.0, 13.0)]	False [(0, 1), (0, 2), (1, 0), (1, 2), (2, 0), (2, 1)] 21.19	

Выводы

Были изучены основные управляющие конструкции языка Python, основы работы с библиотекой numpy, ее объектами и функциями. Для ккаждой из поставленных задач были написаны соответствующие функции.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import numpy as np

def check_crossroad(robot, point1, point2, point3, point4):
    coords_x = []
    coords_y = []
    coords_list = [point1, point2, point3, point4]
    for couple in coords_list:
        coords_x.append(couple[0])
        coords_y.append(couple[1])
    bool_sen = min(coords_x) <= robot[0] and robot[0] <= max(coords_x)
and min(coords_y) <= robot[1] and robot[1] <= max(coords_y)
    if bool_sen:
        return True
    else:
        return False


def check_collision(coefficients):
    res = []
    for i1 in range(0,len(coefficients)):
        for i2 in range(0,len(coefficients)):
            check_matrix                                          =
np.array((coefficients[i1][0:2],coefficients[i2][0:2]))
            if                  len(check_matrix)                 <=
np.linalg.matrix_rank(check_matrix) :
                couple = (i1,i2)
                res.append(couple)
    return res


def check_path(points_list):
    path = 0
    for i in range(1,len(points_list)):
        x_diff = points_list[i][0] - points_list[i-1][0]
        y_diff = points_list[i][1] - points_list[i-1][1]
        vec = np.array((x_diff,y_diff))
        length = np.linalg.norm(vec)
        path += length
    return round(path,2)
```