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. 3341 _____ Ягудин Д. Р.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург

2023

Цель работы

Целью лабораторной работы было изучение основных управляющих конструкций python и библиотеки numpy, а также их применение в ходе решения практической задачи.

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).

Формальная постановка задачи

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

На вход функции подаются: координаты дакибота *robot* и координаты точек, описывающих перекресток: *point1, point2, point3, point4*. Точка -- это кортеж из двух целых чисел (x, y).

Функция должна возвращать True, если дакибот на перекрестке, и False, если дакибот вне перекрестка.

Примеры входных аргументов и результатов работы функции:

1. Входные аргументы: (9, 3) (14, 13) (26, 13) (26, 23) (14, 23)

Результат: False

2. Входные аргументы: (5, 8) (0, 3) (12, 3) (12, 16) (0, 16)

Результат: True

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).

Формальная постановка задачи

Оформите решение в виде отдельной функции *check_collision()*. На вход функции подается матрица ndarray Nx3 (N -- количество ботов, может быть разным в разных тестах) коэффициентов уравнений траекторий *coefficients*. Функция возвращает список пар -- номера столкнувшихся ботов (если никто из ботов не столкнулся, возвращается пустой список).

Пример входного аргумента ndarray 4x3 :

[[-1 -4 0]

 [-7 -5 5]

 [1 4 2]

 [-5 2 2]]

Пример выходных данных:

[(0, 1), (0, 3), (1, 0), (1, 2), (1, 3), (2, 1), (2, 3), (3, 0), (3, 1), (3, 2)]

Первая пара в этом списке (0, 1) означает, что столкнулись 0-й и 1-й боты (то есть их траектории имеют общую точку).

В списке отсутствует пара (0, 2), можно сделать вывод, это боты 0-й и 2-й не сталкивались (их траектории НЕ имеют общей точки).

Примечание: помните про ранг матрицы и как от него зависит существование решения системы уравнений. В случае, если ни одного решение не было найдено (например, из-за линейно зависимых векторов), функция должна вернуть пустой список *[]*.

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.

Формальная постановка задачи

Оформите задачу как отдельную функцию *check_path*, на вход которой передается последовательность (список) двумерных точек (пар) *points_list*. Функция должна возвращать число -- длину пройденного дакиботом пути (выполните округление до 2 знака с помощью *round(value, 2))*.

Пример входных данных:

[(1.0, 2.0), (2.0, 3.0)]

Пример выходных данных:

1.41

Пример входных данных:

[(2.0, 3.0), (4.0, 5.0)]

Пример выходных данных:

2.83

Выполнение работы

Программа, написанная на языке Python, состоит из трех функций

check_crossroad(robot, point1, point2, point3, point4) — принимает на вход координаты улов перекрестка в формате кортежей. После чего обрабатывает их и возвращает True или False

check_collision(coefficients) — принимает на вход ndarray с коэффициентами функции движения дакиботов. Обрабатывает их и возвращает list кортежей с номерами столкнувшихся ботов (боты нумеруются с 0)

check_path(point_list) — принимает на вход list с координатами точек расположения дакибота. Обрабатывает их и возвращает длину пройденного ботом пути

Переменные, используемые в этой программе:

- points — лист координат точек углов перекрестка
- robot — координаты нахождения дакибота
- collisions — список номеров столкнувшихся дакиботов
- coefficient_i — коэффициенты уравнения движения первого дакибота
- coefficient_j — коэффициенты уравнения движения второго дакибота
- vector_i — вектор направления первого дакибота
- vector_j — вектор направления второго дакибота
- matrix — матрица векторов направления дакиботов
- matrix_rank — ранг матрицы matrix
- path_lenght — длина пути пройденого дакиботом
- start_point — координаты начального положения дакибота
- end_point — координаты конечного положения дакибота

Функции, используемые в этой программе:

- check_crossroad(robot, point1, point2, point3, point4): Проверяет, находится ли точка robot внутри заданного прямоугольника.
- check_collision(coefficients): Проверяет наличие пересечений между прямыми, заданными коэффициентами, и возвращает список пересекающихся пар.
- check_path(points_list): Вычисляет общую длину пути, проходящего через заданные точки points_list.

Эта программа демонстрирует управляющие конструкции языка Python, функции и модули, а также основы работы с библиотекой NumPy.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1

Табл. 1 — Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	(9, 3) (14, 13) (26, 13) (26, 23) (14, 23)	False	функция check_crossroad
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)]	функция check_collision
3.	[(1.0, 2.0), (2.0, 3.0)]	1.41	функция check_path
4.	[(2.0, 3.0), (4.0, 5.0)]	2.83	функция check_path

Выводы

Были изучены и применены основные управляющие конструкции языка python, а также функции и методы библиотеки numpy

Так же была написана программа для решения задач, связанных с определением положения, направления движения, и нахождения длины пройденного пути дакибота

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_lb1.py

```
import numpy as np
import math as m

def check_crossroad(robot, point1, point2, point3, point4):
     points = [point1,point2,point3,point4]
     points = sorted(points)

     if (((robot[0] >= points[0][0]) and (robot[1] >= points[0][1])) and
((robot[0] <= points[3][0])     and (robot[1] <= points[3][1]))):
          return True
     else:
          return False

def check_collision(coefficients):
     collisions = []

     for i in range(len(coefficients)):
          for j in range(len(coefficients)):
               if (i != j):
                    coefficients_i = coefficients[i]
                    coefficients_j = coefficients[j]

                    vector_i  =  np.array(  [1,  (-coefficients_i[0]  -
coefficients_i[2]) /                       coefficients_i[1]]     )     -
np.array([0, -coefficients_i[2] / coefficients_i[1]])
                    vector_j  =  np.array(  [1,  (-coefficients_j[0]  -
coefficients_j[2]) /                       coefficients_j[1]]     )     -
np.array([0, -coefficients_j[2] / coefficients_j[1]])

                    matrix = np.array([vector_i, vector_j])
                    matrix_rank = np.linalg.matrix_rank(matrix)

                    if (matrix_rank == 2):
                         collisions.append((i, j))

     return collisions

def check_path(points_list):
     path_length = 0

     for i in range(len(points_list) - 1):
          start_point = points_list[i]
          end_point = points_list[i + 1]
          path_length += m.sqrt(((end_point[0] - start_point[0]) ** 2) +
((end_point[1] -                start_point[1]) ** 2))

     path_length = round(path_length, 2)
     return path_length
```