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 _____ Мавликаев И.А.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2022

Цель работы.

Цель работы — научиться пользоваться основными управляющими конструкциями языка Python, в частности пакетом numpy.linalg в модуле numpy через написание программы.

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.

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.

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).

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)).

Основные теоретические положения.

numpy.linalg.det()
Функция linalg.det() вычисляет определитель (детерминант) матрицы.
Параметры:

a — массив NumPy или подобный массиву объект.

Это может быть толко "квадратный" двумерный массив, т.е. квадратная матрица. Если это многомерный массив, то две его последние оси должны быть равны, в этом случае он рассматривается как массив матриц и детерминант рассчитывается отдельно для каждой из них.
Возвращает:

результат — массив NumPy или число

В случае если *a* это многомерный массив, то возвращается массив определителей каждой подматрицы исходного массива. Если на вход подана одна квадратная матрица, то ее определитель возвращается в виде одного числа.

Выполнение работы.

Функция *check_crossroad(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).

С помощью конструкции *if* проверяется, находятся ли координаты дакибота внутри прямоугольника, если он находятся на перекрёстке, возвращается True, иначе возвращается False.

Функция *check_collision(coefficients)*

На вход функции подается матрица ndarray Nx3 (N — количество ботов, может быть разным в разных тестах) коэффициентов уравнений траекторий *coefficients*. Создаётся пустой массив *answer*, в который будет записан ответ. В двойном цикле перебираются строки матрицы *coefficients*.

Для того, чтобы понять, параллельны ли два вектора, коэффициенты которых указаны в текущих i, j строках, необходимо посчитать определитель матрицы коэффициентов уравнений траекторий векторов. Создаётся матрица Matrix вида 2 на 2 элемента из срезов (первых двух элементов) i и j строк. С помощью функции *numpy.linalg.det(Matrix)* вычисляется определитель матрицы, и в случае, когда он не равен нулю, векторы не параллельны и где-то пересекутся, поэтому номера векторов (строк матрицы *coefficients* номер i и j) записываются кортежем в массив *answer*.

Функция возвращает массив *answer*.

Функция *check_path(points_list)*

На вход функции передается последовательность (список) двумерных точек (пар) *points_list*.

Создаётся переменная *answer = 0*, в которую будет записываться ответ.

В цикле *for* по теореме Пифагора высчитывается расстояние от координаты i-ой точки до координаты i-1-ой точки, которое прибавляется к *answer*. После выхода из цикла переменная *answer* округляется до двух чисел после запятой с помощью функции *round()*.

Функция возвращает переменную *answer.*

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	Дакибот вне перекрёстка Не столкнулись только боты 0 и 2. Путь найден.
2.	(5, 8) (0, 3) (12, 3) (12, 16) (0, 16) [[1 4 -3] [2 3 -8]] [(2.0, 3.0), (4.0, 5.0)]	True [(0, 1), (1, 0)] 2.83	Дакибот на перекрёстке Боты столкнулись. Путь найден.
3.	(2, 2), (1, 1), (3, 1), (3, 3), (1, 3) [[1 4 3] [1 4 -8]] [(0, 0), (4, 3)]	True [] 5.0	Дакибот на перекрёстке Траектория ботов параллельна. Путь найден.
4.	(5, 2), (1, 1), (3, 1), (3, 3), (1, 3) [[1 3 3] [1 4 -8]] [(0, 0), (4, 3), (5, 4)]	False [(0, 1), (1, 0)] 6.41	Дакибот вне перекрёстка Боты столкнулись. Путь найден.

Выводы.

Были изучены основные управляющие конструкции языка Python, в частности модуль numpy и функции в numpy.linalg. Разработаны три функции, первая из которых определяет, попадают ли координаты точки в заданный прямоугольник, вторая — параллельны ли заданные векторы между собой, третья — путь по координатам. Для решения поставленных задач использовались условные операторы *if-else*. Для нахождения определителя матрицы использовалась функция numpy.linalg.det().

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1.py

```
import numpy as np


def check_crossroad(robot, point1, point2, point3, point4):
     if robot[0] >= point1[0] and robot[0] <= point2[0] and robot[1]
>= point1[1] and robot[1] <= point4[1]:
          return True
     return False


def check_collision(coefficients):
     answer = []
     for i in range (len(coefficients)):
          for j in range (len(coefficients)):
               Matrix         =         np.array([coefficients[i][:2],
coefficients[j][:2]])
               if (np.linalg.det(Matrix) != 0):
                    answer.append((i, j))
     return answer


def check_path(points_list):
     answer = 0
     for i in range (1, len(points_list)):
          answer += ((points_list[i][0] - points_list[i - 1][0])**2 +
abs(points_list[i][1] - points_list[i - 1][1])**2)**0.5
     answer = round(answer, 2)
     return answer
```