МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Информатика»

Тема: «Основные управляющие конструкции языка Python»

Студент гр. 3342 _____ Белаид Фарук

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы.

Научиться работать с основными конструкциями языка Python и библиотекой numpy для решения задач линейной алгебры.

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)).

Выполнение работы.

Задача 1.

Для выполнения задачи реализована функция ***check_crossroad(),*** принимающая в качестве аргументов кортежи координат робота и 4 точек, определяющих прямоугольник. В теле функции созданы следующие переменные: ***points*** – список введенных пользователем четырех координат прямоугольника, ***y_down*** и ***y_up*** – наименьшая и наибольшая y-координаты среди введенных точек, ***x_left*** и ***x_right*** – наименьшая и наибольшая x-координаты. В качестве ответа функция возвращает результат выражения ***(x_left <= robot[0] <= x_right) and (y_down <= robot[1] <= y_up)***, которое является конъюнкцией двух условий: x-координата робота находится между наименьшей и наибольшей x-координатой прямоугольника и y-координата робота находится между наименьшей и наибольшей y-координатой прямоугольника. Этих двух условий достаточно для того, чтобы утверждать, находится ли робот внутри прямоугольника или нет.

Задача 2.

Для выполнения задачи реализована функция ***check_collision(),*** принимащая на вход матрицу коэффициентов уравнений траекторий движения дакиботов. В теле функции создается пустой список ***result***, в который будут добавляться пары столкнувшихся ботов. Для нахождения этих пар используются вложенные циклы, перебирающие все возможные пары ботов. Внутри второго цикла объявляются следующие переменные: ***traect1*** и ***traect2*** – *np.array* массивы коэффициентов уравнения траектории движения выбранных ботов (без свободного коэффициента), ***matrix*** – матрица, составленная из ***traect1*** и ***traect2*** с помощью метода библиотеки *numpy* – *numpy.vstack()*. Условие ***numpy.linalg.matrix_rank(matrix) == 2*** определяет, являются ли две строки этой матрицы линейно независимыми, из чего напрямую следует, пересекаются ли траектории движения соответствующих

ботов или нет. Внутри блока *if* в список ***result*** добавляются найденные пары ботов в виде кортежей. Функция возвращает список ***result.***

Задача 3.

Для выполнения задачи реализована функция ***check_path(),*** принимащая на вход список координат точек. В теле функции объявляется переменная ***result***, в которой будет считаться итоговая длина пути. С помощью цикла, который проходится по каждой паре подряд идущих точек из списка, программа прибавляет к переменной ***result*** расстояние между точками, рассчитываемое по теореме Пифагора (корень из суммы квадратов разностей координат). Функция возвращает ***result***.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

Функция	№ п/п	Входные данные	Выходные данные	Комментарии
Check_crossroad	1.	(3, 4), (2, 3), (2, 6), (5, 6), (5, 3)	True	Ответ верный
	2.	(6, 7), (2, 3), (2, 6), (5, 6), (5, 3)	False	Ответ верный
Check_collision	3.	[[-3 -2 1] [-2 -3 5] [3 2 6] [-1 3 4]]	[(0, 1), (0, 3), (1, 0), (1, 2), (1, 3), (2, 1), (2, 3), (3, 0), (3, 1), (3, 2)]	Ответ верный

Check_path	4.	[(1.0, 2.0), (2.0, 3.0)]	1.41	Ответ верный
	5.	[(2.0, 3.0), (4.0, 5.0)]	2.83	Ответ верный

Вывод.

В ходе выполнения лабораторной работы были изучены управляющие конструкции языка Python, а также библиотека NumPy, предназначенная для работы с математическими структурами и операциями.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
import numpy as np
from math import sqrt

def check_crossroad(robot, point1, point2, point3, point4):
    points = [point1, point2, point3, point4]
    y_down = min(points, key=lambda x: x[1])[1]
    y_up = max(points, key=lambda x: x[1])[1]
    x_left = min(points, key=lambda x: x[0])[0]
    x_right = max(points, key=lambda x: x[0])[0]
    return ((x_left <= robot[0] <= x_right) and (y_down <=
robot[1] <= y_up))


def check_collision(coefficients):
    result = []
    for i in range(len(coefficients)):
        for j in range(len(coefficients)):
            traect1 = np.array(list(coefficients[i])[:2])
            traect2 = np.array(list(coefficients[j])[:2])
            matrix = np.vstack((traect1, traect2))
            if np.linalg.matrix_rank(matrix) == 2:
                result.append((i, j))
    return result


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