**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 |  | Колесниченко М.А. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Изучение и освоение работы с функциями и с модулем NumPy в языке Python.

## Задание

Вариант 2.

Задача 1.

Оформите задачу как отдельную функцию: def 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).Функция должна возвращать True, если дакибот на перекрестке, и False, если дакибот вне перекрестка.

Задача 2.

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.

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

Программа написана на языке Python с использованием библиотеки NumPy и содержит 3 функции.

Первая функция check\_crossroad возвращает True, если дакибот на перекрестке, и False, если дакибот вне перекрестка. Внутри функции сравниваются координаты робота и координаты углов перекрестка.

Вторая функция check\_collision. Функция возвращает список пар в виде кортежей - номера столкнувшихся ботов (если никто из ботов не столкнулся, возвращается пустой список). Внутри функции использованы два цикла for для перебора всех траекторий дакиботов. Внутри циклов создается матрица, которая содержит в себе две траектории движения дакиботов. С помощью функции из модуля numpy linalg.matrix\_rank вычисляется ранг матрицы, с помощью которого определяется имеет ли система уравнений решение. Если решение существует – значит роботы столкнулись, и в массив ans записываются соответствующие индексы строк с коэффициентами. После всех итераций функция возвращает массив ans.

Третья функция check\_path принимает список точек points\_list и вычисляет длину пути, проходящего через эти точки. Для этого используется формула расстояния между двумя точками на плоскости. Результат вычислений округляется до двух знаков после запятой и возвращается в виде числа с плавающей точкой.

Переменные, используемые в программе:

- ans – список из кортежей с номерами столкнувшихся дакиботов.

- sum\_path – сумма длин путей дакибота.

Функции, используемые в этой программе:

-numpy.array возвращает массив типа numpy.ndarray.

-numpy.linalg.matrix\_rank возвращает ранг матрицы.

-round возвращает округленное число до выбранного значения.

Программа демонстрирует использование функция в языке Python а также использование функции модуля NumPy для решения поставленных задач.

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 |
|  | (9, 3),(14, 13),(26, 13),(26, 23),(14, 23) | False |  |
|  | [[-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)] |  |
|  | [(1.0, 2.0), (2.0, 3.0)] | 1.41 |  |

## Выводы

Были изучены правила работы с функциями в языке python и работа с библиотекой numpy.

Реализованы функции, возвращающие решения определенных математических заданий.

# Приложение А Исходный код программы

Название файла: main.py

import numpy as np

def check\_crossroad(robot, point1, point2, point3, point4):

return point1[0] <= robot[0] <= point2[0] and \

point1[1] <= robot[1] <= point4[1]

def check\_collision(coefficients):

ans = []

for i in range(len(coefficients)):

for j in range(len(coefficients)):

if i != j:

mat = np.array([coefficients[i][0:2],coefficients[j][0:2]])

if np.linalg.matrix\_rank(mat) == 2: ans.append((i,j))

return ans

def check\_path(points\_list):

sum\_path = 0

for i in range(1,len(points\_list)):

sum\_path += ((points\_list[i][0]-points\_list[i-1][0])\*\*2 + (points\_list[i][1]-points\_list[i-1][1])\*\*2)\*\*0.5

return round(sum\_path,2)