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 _____ Лапшов К.Н.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Целью работы является освоение работы с функциями в языке python и с библиотекой numpy.

Задание

Вариант 2.

Задача 1.

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.

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

Данная программа написана на языке Python с использованием библиотеки numpy. Она состоит из 3-функций, которые вызываются сразу на сайте https://e.moevm.info.

Первая функция check_crossroad. Функция возвращает True, если дакибот на перекрестке, и False, если дакибот вне перекрестка. Перекресток определяется 4 данными на входе точками. Для выполнения это функции необходимо сравнить координаты робота и координаты точек перекрестка, находя его границы.

Вторая функция check_collision. Функция возвращает список пар номеров столкнувшихся ботов в виде кортежей. Если никто из ботов не столкнулся, возвращается пустой список. Для реализации этой функции были использованы два цикла, где переменные-итераторы являются индексами строк матрицы с коэффициентами линейных уравнений. Внутри этих циклов создаются массивы, в которые записываются коэффициенты соответствующих строк матрицы. Затем создается матрица, содержащая эти два массива. С помощью функции linalg.matrix_rank из модуля numpy вычисляется ранг матрицы, который определяет, есть ли пересечения у двух линейных функций. Если пересечения есть, значит роботы столкнулись, и соответствующие индексы строк с коэффициентами записываются в массив result. По окончании всех итераций функция возвращает массив result.

Третья функция check_path. Функция принимает список точек points_list и вычисляет длину пути, проходящего через эти точки. Для этого используется функция модуля numpy - linalg.norm которая высчитывает длину вектора. Результат вычислений округляется до двух знаков после запятой и возвращается в виде числа с плавающей точкой.

Переменные, используемые в программе:

- x_line_min, x_line_max, – ограничение перекрестка по ширине.
- y_line_min, y_line_max, – ограничение перекрестка по высоте.
- index – переменная, используемая в первом цикле функции check_collision.
-index_interior – переменная, используемая во внутреннем цикле функции check_collision.
- result – список из картежей с номерами столкнувшихся дакиботов.
- distance – сумма длин путей дакибота.

Функции, используемые в этой программе:

-numpy.array возвращает массив типа numpy.ndarray.
-numpy.linalg.matrix_rank возвращает ранг матрицы.
-numpy.linalg.norm возвращает длину вектора.
-round возвращает округленное число до выбранного значения.

Данная программа демонстрирует использование функций библиотеки numpy и работу функций на языке Python для выполнения различных математический операций.

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.	(15, 16), (5, 10) ,(20, 10), (20, 20), (5, 20)	True	
2.	[[-1 5 7] [1 -5 7] [9 3 32]]	[(0, 1), (0, 2), (1, 0), (1, 2), (2, 0), (2, 1)]	
3.	[(2.0, 2.0), (7.0, 8.0)]	7.81	

Выводы

Исследованы принципы работы с функциями в языке Python и использование библиотеки NumPy.

Разработаны функции, которые вычисляют решения конкретных математических задач.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import numpy as np


def check_crossroad(robot, point1, point2, point3, point4):
    x_line_min, y_line_min = point1
    x_line_max, y_line_max = point3

      if x_line_min <= robot[0] <= x_line_max and y_line_min <=
robot[1] <= y_line_max:
        return True
    else:
        return False


def check_collision(coefficients):
    result = []

    for index in range(len(coefficients)):
        numpay_arr = coefficients[index][0:2]

        for index_interior in range(len(coefficients)):
            numpay_arr_insert = coefficients[index_interior][0:2]
            matrix = np.array([numpay_arr, numpay_arr_insert])

            rank_of_matrix = np.linalg.matrix_rank(matrix)

            if rank_of_matrix == len(numpay_arr):
                result.append((index, index_interior))

    return result


def check_path(points_list):
    distance = 0

    for index in range(len(points_list) - 1):
        if index + 1 == len(points_list):
            break
        else:
            point_a = np.array(points_list[index])
            point_b = np.array(points_list[index + 1])
            distance += np.linalg.norm(point_a - point_b)

    return round(distance, 2)
```