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 _____ Роднов И.С.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Целью работы является освоение работы с функциями в языке python и 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

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. По входным данным(4 точки) функция возвращает True, если дакибот в пределах перекрестка, и False, если дакибот вне пределов перекрестка. Для выполнения функции необходимо сравнить координаты робота и координаты границ перекрестка.

Вторая функция check_collision. Функция возвращает список номеров(парами) столкнувшихся ботов в виде кортежей или пустой список если никто не столкнулся. Для реализации 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, благодаря которому определяется, есть ли пересечения у двух линейных функций. Если пересечения есть, значит роботы столкнулись, и соответствующие индексы строк с коэффициентами записываются в массив answer. По окончании всех итераций функция возвращает массив answer.

Третья функция check_path. Функция принимает список точек points_list и вычисляет длину пути, проходящего через эти точки. Для этого используется формула для вычисления расстояния (в данном случае Теорема Пифагора). Результат вычислений округляется до двух знаков после запятой и возвращается в виде числа с плавающей точкой.

Переменные, используемые в программе:

- answer – список из картежей с номерами столкнувшихся дакиботов.
- a – 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.

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.	(4, 4), (2, 2) ,(8, 2), (8, 6), (2, 6)	True	
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)]	
3.	[(1.0, 2.0), (4.0, 5.0)]	4.24	

Вывод

Изучены принципы работы с функциями в языке Python и применение библиотеки numpy.

Разработаны функции, которые решают конкретные математические задачи.

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] <= point3[0] and point2[1] <=
robot[1] <= point4[1]


def check_collision(coefficients):
    answer = []
    for i in range(coefficients.shape[0]):
        for j in range(coefficients.shape[0]):
            if i != j:
                arri = coefficients[i][0:2]
                arrj = coefficients[j][0:2]
                Mat = np.array([arri, arrj])
                if np.linalg.matrix_rank(Mat) == 2:
                    answer.append((i, j))
    return answer


def check_path(points_list):
    a = 0
    for i in range(len(points_list)-1):
        a += np.sqrt(((points_list[i+1][0]-points_list[i][0])**2) +
((points_list[i+1][1]-points_list[i][1])**2))
    return np.around(a, 2)
```