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 управляющее конструкции языка Python

Студентка гр. 3342 _____ Епонишникова А.И

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы.

На практике изучить основные управляющие конструкции языка Python. Рассмотреть математический модуль NumPy и методы линейной алгебры.

Задание.

1. 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).
2. 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).
3. 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.

Выполнение работы.

Для начала подключим библиотеку NumPy и присвоим ей для краткости имя np.

1. Функция check_crossroad(robot, point1, point2, point3, point4): на вход функции подаются координаты дакибота (robot) и координаты точек(point1, point2, point3, point4). Они представляют собой кортеж из двух чисел (x,y). Сравниваем координату **x** дакибота с координатой **x** point1 и point3, так как тут минимальное и максимальное значение перекрестка по **x**. Далее аналогично сравниваем координату **y** дакибота с координатой **y** point1 и point3. Если дакибот располагается на перекрестке, то возвращаем True. Если нет, то False.
2. Функция check_collision(coefficients): на вход функции подается матрица ndarray Nx3 коэффициентов уравнений траекторий coefficients. В пустой список answer добавляются номера дакиботов, которое столкнулись. Для проверки этого запускаем два вложенных цикла. В matrix_i добавляем первые два элемента, создаем коэффициенты для линейного уравнения ax+by+c=0. Аналогично для matrix_j. Создаем матрицу(matrix) при помощи модуля NumPy np.vstack((coefficients[i][:2], coefficients[j][:2])). np.vstack() — это метод, который позволяет дописать матрицы последовательно друг к другу, как бы кладёт одну матрицу на другую и создаёт третью матрицу — результат сложения двух матриц друг на друга. Далее проверяем совпадает ли ранг матрицы(matrix) с количеством коэффициентов matrix_i при помощи np.linalg.matrix_rank. Данный метод позволяет посчитать ранг квадратной матрицы, ранг – это количество линейно-независимых строк матрицы или столбцов квадратной матрицы. Если совпадает, то значит добавляем кортеж из номеров дакиботов, которые столкнулись, в answer. Функция возвращает answer.

3. Функция check_path(points_list): на вход функция получает список двумерных точек points_list. В переменную path будет добавляться путь, который прошел дакибот. Запускаем цикл, в переменной vector1 создаем матрицу из points_list[i], в переменной vector2 создаем матрицу из points_list[i+1]. Далее при помощи метода np.linalg.norm считаем длину вектора, который прошел дакибот от i точки до i+1 точки. Добавляем полученный вектор в path. Полученный путь вернем из функции, округлив при помощи round() до двух знаков после запятой.

   Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Номер функции и входные данные через пробел	Выходные данные
1.	(9, 3) (14, 13) (26, 13) (26, 23) (14, 23)	False
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), (2.0, 3.0)]	1.41

Выводы.

На практике научились работать с основными управляющими конструкциями языка Python. Был рассмотрен математический пакет NumPy и методы линейной алгебры, встроенные в него. Также была разработана программа, выполняющая поставленные подзадачи.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab_1_cp.py

```
import numpy as np


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


def check_collision(coefficients):
    answer = []
    for i in range(len(coefficients)):
        for j in range(len(coefficients)):
            matrix_i = coefficients[i][:2]
            matrix_j = coefficients[j][:2]
                        matrix  =  np.vstack((coefficients[i][:2],
coefficients[j][:2]))
            if (np.linalg.matrix_rank(matrix) == len(matrix_i)):
                answer.append((i,j))
    return answer

def check_path(points_list):
    path = 0
    for i in range(len(points_list) - 1):
        vector1 = np.array(points_list[i])
        vector2 = np.array(points_list[i+1])
        path += np.linalg.norm(vector2 - vector1)
    return round(path,2)
```