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. 2381 _____ Соколов С.А.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2022

Цель работы.

Изучить основные управляющие конструкции языка Python, модуль *numpy*, в частности пакет *numpy.linalg.*

Задание.

Вариант 1.

Задача 1.

Два дакибота приближаются к перекрестку. Чтобы избежать столкновения, им необходимо знать точку пересечения их траекторий движения. Траектории -- линейные, и дакиботы уже вычислили коэффициенты этих уравнений. Ваша задача -- помочь ботам вычислить точку потенциального столкновения.

Оформить решение в виде отдельной функции *check_collision*. На вход функции подаются два *ndarray* -- коэффициенты *bot1, bot2* уравнений прямых *bot1 = (a1, b1, c1), bot2 = (a2, b2, c2)* (уравнение прямой имеет вид *ax+by+c=0*).

Функция должна возвращать точку пересечения траекторий (кортеж из 2 значений), предварительно округлив координаты до 2 знаков после запятой с помощью *round(value, 2)*.

Задача 2.

Три дакибота начали движение, отъехали от условной точки старта и через некоторое время остановились. Каждый дакибот уже вычислил свою координату относительно точки старта. Дакиботам нужно передать на базу карту местности, по которой они двигались. Для построения карты местности необходимо знать уравнение плоскости. Ваша задача -- помочь дакиботам найти уравнение плоскости, в которой они двигались.

Оформить задачу как отдельную функцию *check_surface*, на вход которой передаются координаты 3 точек (3 *ndarray* 1 на 3): *point1, point2, point3*. Функция должна возвращать коэффициенты a, b, c в виде ndarray для уравнения плоскости вида *ax+by+c=z*. Перед возвращением результата выполнение округление каждого коэффициента до 2 знаков после запятой с помощью *round(value, 2)*.

Задача 3.

Дакибот выехал на перекресток и готовится к выполнению поворота вокруг своей оси (вокруг оси z), чтобы продолжить движение в другом направлении. Он знает свои координаты и знает угол поворота (в радианах). Помогите дакиботу повернуться в нужное направление для продолжения движения.

Оформить решение в виде отдельной функции *check_rotation*. На вход функции подаются *ndarray* 3-х координат дакибота и угол поворота. Функция возвращает повернутые *ndarray* координаты, каждая из которых округлена до 2 знаков после запятой с помощью *round(value, 2).*

Выполнение работы.

Функция *check_collision* принимает два аргумента: коэффициенты уравнений прямых, по которым перемещаются дакиботы. Для нахождения точки пересечения необходимо решить систему из этих уравнений. СЛУ представляется в матричной форме, где *matrix* это матрица системы, а *vec* столбец свободных членов. Если ранг матрицы не равен двум, то система не имеет решений, а функция возвращает *None*. Если ранг матрицы равен двум, то используется функция *solve* из пакета *numpy.linalg*, которая возвращает столбец решений. Значения столбца решений округляются до второго знака после запятой, функция возвращает кортеж из этих значений.

Функция *check_surface* принимает три аргумента – три точки. Для нахождения коэффициентов уравнения плоскости необходимо составить матрицу коэффициентов, и вектор свободных членов. Если ранг матрицы равен трем, то используется функция *solve* из пакета *numpy.linalg*, которая возвращает столбец решений. Функция возвращает значения столбца решений, которое округлено до второго знака после запятой. Если ранг матрицы не равен трем, то система не имеет решений, а функция возвращает *None*.

Функция *check_rotation* принимает два аргумента – координаты дакибота, а также угол поворота. Для нахождения повернутых координат дакибота

используется матрица поворота вокруг оси z. Угол поворота передаётся тригонометрическим функциям матрицы в качестве аргумента. Функция возвращает произведение матрицы поворота *rot* на вектор координат дакибота vec, значения которого округлены до второго знака после запятой. Для получения произведения матриц используется метод *dot*, который вызывается у переменной *rot*.

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.	array([-3, -6, 9]), array([8, -7, 0])	(0.91, 1.04)	Пример данных для первой задачи. Вводятся коэффициенты прямых, возвращаются координаты точек пересечения.
2.	array([1, -6, 1]), array([0, -3, 2]), array([-3, 0, -1])	[2. 1. 5.]	Пример данных для второй задачи. Вводятся координаты трех точек, возвращаются коэффициенты для уравнения плоскости.
3.	array([1, -2, 3]), 1.57	[2. 1. 3.]	Пример данных для третьей задачи. Вводятся координаты точки и угол поворота, возвращаются повернутые координаты.

Выводы.

Были изучены и использованы на практике основные управляющие конструкции языка Python, модуль *numpy*, в частности пакет *numpy.linalg.*

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import numpy as np

def check_collision(bot1, bot2):
    matrix = np.array([
        [bot1[0], bot1[1]],
        [bot2[0], bot2[1]]]
    )
    vec = np.array([-bot1[2], -bot2[2]])
    if np.linalg.matrix_rank(matrix) == 2:
        return tuple(np.linalg.solve(matrix, vec).round(2))
    return None

def check_surface(point1, point2, point3):
    matrix = np.array([
        [point1[0], point1[1], 1],
        [point2[0], point2[1], 1],
        [point3[0], point3[1], 1]
    ])

    vec = np.array([point1[2], point2[2], point3[2]])

    if np.linalg.matrix_rank(matrix) == 3:
        return np.linalg.solve(matrix, vec).round(2)
    return None

def check_rotation(vec, rad):
    rot = np.array([
        [np.cos(rad), -np.sin(rad), 0],
        [np.sin(rad), np.cos(rad), 0],
        [0, 0, 1],
    ])
    return np.round(rot.dot(vec),2)
```