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

Студентка гр. 3341 Яковлева А.А.

Преподаватель Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Изучение основных управляющих конструкций языка Python, модуля *numpy*, в частности пакета *numpy.linalg.*

Задание

Вариант 1

Вариант лабораторной работы состоит из 3 задач, оформите каждую задачу в виде отдельной функции согласно условиям задач. Приветствуется использование модуля *numpy*, в частности пакета *numpy.linalg*. Вы можете реализовывать вспомогательные функции, главное - использовать те же названия основных функций, что требуются в задании. Сами функции вызывать не надо, это делает за вас проверяющая система.

Задача 1. Содержательная постановка задачи

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.

Формальная постановка задачи

Оформите 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. Содержательная часть задачи

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 местностности, по которой они двигались. Для построения карты местности необходимо знать уравнение плоскости. Ваша задача - помочь дакиботам найти уравнение плоскости, в которой они двигались.

Формальная постановка задачи

Оформите 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. Содержательная часть задачи

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.

Формальная постановка задачи

Оформите 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

Функции:

- *check_collision* принимает на вход два *ndarray* коэффициенты *bot1*, *bot2* уравнений прямых *bot1 = (a1, b1, c1), bot2 = (a2, b2, c2)*, возвращает точку пересечения траекторий (кортеж из 2 значений) или *None* если решение невозможно.
- *check_surface* принимает на вход координаты 3 точек (3 *ndarray* 1 на 3): *point1, point2, point3*, возвращает коэффициенты *a, b, c* в виде *ndarray* для уравнения плоскости вида *ax+by+c=z* или *None* если решение невозможно.
- *check_rotation* принимает на вход *ndarray* 3-х координат *vec* дакибота и угол поворота *rad*, возвращает повернутые *ndarray* координаты.

В функции *check_collision:*

a1x+b1y+c1=0 ⇔ *a1x+b1y = -c1,* где *a1 = bot1[0], b1 = bot1[1], c1 = bot1[2]*

a2x+b2y+c2=0 ⇔ *a2x+b2y = -c2,* где *a2 = bot2[0], b2 = bot2[1], c2 = bot2[2]*

Обозначим *a = [[bot1[0], bot1[1]], [bot2[0], bot2[1]]], b = [[-bot1[2]], [-bot2[2]]].* Тогда *a* - матрица коэффициентов, *b* - вектор свободных членов. Уравнение *ax = b* имеет решение, которое можно найти с помощью *np.linalg.solve(a, b),* если ранг *a = 2.*

В функции *check_surface:*

ax1+by1+c=z1, где *x1 = point1[0], y1 = point1[1], z1 = point1[2]*

ax2+by2+c=z2, где x*2 = point2[0], y2 = point2[1], z2 = point2[2]*

ax3+by3+c=z3, где x3 *= point3[0], y3 = point3[1], z3 = point3[2]*

Обозначим *a = [[point1[0], point1[1], 1], [point2[0], point2[1], 1], [point3[0], point3[1],1]], b = [[point1[2]], [point2[2]], [point3[2]]].* Тогда *a* - матрица коэффициентов, *b* - вектор свободных членов. Уравнение *ax = b* имеет решение, которое можно найти с помощью *np.linalg.solve(a, b),* если ранг *a = 3.*

В функции *check_rotation:*

Поворот выполняется путём умножения матрицы поворота на вектор-столбец, описывающий вращаемую точку:

Вращение вокруг оси z:

$$M_z\ (rad) = \begin{matrix} [\ [\cos(rad), & -\sin(rad), & 0], \\ [\sin(rad), & \cos(rad), & 0], \\ [0, & 0, & 1]\ ] \end{matrix}$$

*x=x1*cos(rad)-y1*sin(rad)*

*y=x1*sin(rad)+y1*cos(rad)*

z=z1, где x1 = vec[0], y1 = vec[1], z1 = vec[2]

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.	*check_collision* input: [-3, 5, 1], [-10, 4, 2] *check_surface* input: [1, -6, 1], [0, -3, 2], [-3, 0, -1] *check_rotation* input: [1, -2, 3], 0.52	(0.16, -0.11) [2. 1. 5.] [1.86 -1.24 3.]	
2.	*check_collision* input: [-7, 4, 5], [-8, -3, 9] *check_surface* input: [1, -2, 3], [2, -3, 4], [3, -4, 5] *check_rotation* input: [2, -2, -1], 0.26	(0.96, 0.43) None [2.45 -1.42 -1.]	*check_surface* output: None, так как ранг матрицы 2 < 3
3.	*check_collision* input: [6, 2, 9], [8, 5, 3] *check_surface* input: [1, -3, -1], [-2, 7, 2], [3, 2, -4] *check_rotation* input: [5, -6, 1], 0.26	(-2.79, 3.86) [-1.29 -0.09 0.03] [6.37 -4.51 1.]	

Выводы

Были изучены основные управляющие конструкции языка Python, модуль *numpy*, пакет *numpy.linalg.*

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import numpy as np


def check_collision(bot1, bot2):
    a = [[bot1[0], bot1[1]], [bot2[0], bot2[1]]]
    b = [[-bot1[2]], [-bot2[2]]]
    if np.linalg.matrix_rank(a) < 2:
        return(None)
    else:
        x = np.linalg.solve(a, b)
        return(tuple((round(x[0][0], 2), (round(x[1][0], 2)))))


def check_surface(point1, point2, point3):
    a = [[point1[0], point1[1], 1], [point2[0], point2[1], 1],
[point3[0], point3[1],1]]
    b = [[point1[2]], [point2[2]], [point3[2]]]
    if np.linalg.matrix_rank(a) < 3:
        return(None)
    else:
        x = np.linalg.solve(a, b)
        return(np.array([round(x[i][0], 2) for i in range(0,3)]))


def check_rotation(vec, rad):
    return(np.array([round(vec[0]*np.cos(rad)
vec[1]*np.sin(rad),2), round(vec[0]*np.sin(rad) + vec[1]*np.cos(rad),2),
round(vec[2], 2)]))
```