**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 |  | Львов А.В. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2023

## Цель работы

Целью работы является изучение основных управляющих конструкций языка Python, ознакомление с библиотекой numpy.

## Задание

Задача 1.

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.

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, с 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.

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

Программа написана на языке Python с использованием библиотек math и numpy.

Первая функция check\_collision принимает коэффициенты bot1 и bot2 уравнений прямых. В ней создаётся матрица с коэффициентами matrix\_1 и матрица с свободными членами matrix\_2. С помощью функции linalg.matrix\_rank() библиотеки numpy происходит проверка, имеет ли система уравнений решения. Если нет – возвращается None, в ином случае возвращаются координаты точки пересечения траекторий (кортеж из 2 значений), округлённые до двух знаков после запятой.  
 Вторая функция check\_surface принимает координаты 3 точек point1, point2, point3. В ней создаётся матрица коэффициентов matrix\_coefs и матрица с свободных членов vec. С помощью функции linalg.matrix\_rank() библиотеки numpy происходит проверка, имеет ли система уравнений решения. Если нет – возвращается None, в ином случае возвращаются коэффициенты a, b, c в виде ndarray для уравнения плоскости вида ax+by+c=z, округлённые до двух знаков после запятой.

Третья функция check\_rotation принимает координаты (vec) и угол поворота (rad). В ней создаётся матрица поворота вокруг оси z (matrix), продемонстрированная на рисунке 1. Функции косинуса и синуса реализованы в библиотеке math (math.cos() и math.sin() соответственно). Затем, с помощью функции dot() модуля numpy, результат умножения матрицы поворота вокруг оси z на матрицу координат записывается в переменную result. Функция возвращает координаты x, y, z (x, y, z = result), округлённые до двух знаков после запятой.

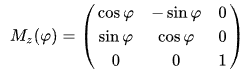


Рисунок 1 - Матрица поворота вокруг оси z

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
| 1. | check\_collision(array([- 3, -6, 9]), array([8, -7,  0])) | (0.91, 1.04) |  |
| 2. | check\_surface(array([ 1  , -6, 1]), array([ 0, -3,  2]), array([-3, 0, -1])) | [2. 1. 5.] |  |
| 3. | check\_rotation(array([ 1  , -2, 3]), 1.57) | [2. 1. 3.] |  |

## Выводы

Была разработана программа на языке Python с использованием библиотек numpy и math, решающая данные в задании задачи, изучены основные управляющие конструкции языка.

# Приложение А Исходный код программы

Название файла: main.py

import numpy as np

import math

def check\_collision(bot1, bot2):

a1, b1, c1 = bot1

a2, b2, c2 = bot2

matrix\_1 = np.array([(a1, b1),

(a2, b2)])

matrix\_2 = np.array([-c1, -c2])

if np.linalg.matrix\_rank(matrix\_1) < 2:

return None

else:

x, y = np.linalg.solve(matrix\_1, matrix\_2)

return round(x, 2), round(y, 2)

def check\_surface(point1, point2, point3):

x1, y1, z1 = point1

x2, y2, z2 = point2

x3, y3, z3 = point3

matrix\_coefs = np.array([(x1, y1, 1),

(x2, y2, 1),

(x3, y3, 1)])

vec = np.array([(z1), (z2), (z3)])

if np.linalg.matrix\_rank(matrix\_coefs) < 3:

return None

else:

result = np.linalg.solve(matrix\_coefs, vec)

a, b, c = result

return np.array([round(a, 2), round(b, 2), round(c,2)])

def check\_rotation(vec, rad):

matrix = np.array([(math.cos(rad), -math.sin(rad), 0),

(math.sin(rad), math.cos(rad), 0),

(0, 0, 1)])

result = np.dot(matrix, vec)

x, y, z = result

return np.array([round(x, 2), round(y, 2), round(z, 2)])