**5МИНОБРНАУКИ 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. 3343 |  | Жучков О.Д. |
| --- | --- | --- |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

## Изучить и научиться применять основные управляющие конструкции языка Python и библиотеку NumPy.

## 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.

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)*.

***Примечание***: помните про ранг матрицы и как от него зависит наличие решения системы уравнений. В случае, если решение найти невозможно (например, из-за линейно зависимых векторов), функция должна вернуть ***None***.

#### 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)*.

**Примечание**: помните про ранг матрицы и как от него зависит существование решения системы уравнений. В случае, если решение найти невозможно (невозможно найти коэффициенты плоскости из-за, например, линейно зависимых векторов), функция должна вернуть ***None***.

#### 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

Для выполнения задания написана программа на языке Python, в которой описываются функции для решения трёх задач.

Функция check\_collision выполняет первую задачу. На вход подаются два набора коэффициентов траекторий в виде ndarray, функция возвращает точку пересечения траекторий в виде кортежа или None, если траектории не пересекаются.

Вторая задача выполняется функцией check\_surface, на вход которой подаются координаты 3 точек в виде ndarray. На выходе пользователь получает коэффициенты уравнения плоскости в виде ndarray. Если у системы уравнений нет решения, то функция возвращает None*.*

Для решения третьей задачи написана функция check\_rotation. На вход функция принимает ndarray с координатами и угол поворота в радианах. Функция возвращает повернутые координаты в форме ndarray.

При выполнении задач используются такие функции из модуля NumPy, как:

1. *np.array()* и *np.append()* для создания матрицы и добавления элементов.
2. *np.linalg.matrix\_rank()* для вычисления ранга матрицы.
3. *np.linalg.solve()* решает систему линейных уравнений, в матричной форме.
4. *np.vstack() –* функция, вертикально объединяющая ndarray.
5. *np.dot()* для умножения матриц.

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 |
| --- | --- | --- | --- |
|  | [-3,-6,9] [8,-7,0] | (0.91, 1.04) | Первая функция работает прваильно |
|  | [1,-6,1] [0,-3,2] [-3,0,-1] | [2. 1. 5.] | Вторая функция работает прваильно |
|  | [1,-2,3] 1.57 | [2. 1. 3.] | Третья функция работает прваильно |

## Выводы

## В ходе лабораторной работы были изучены основные управляющие конструкции языка Python и модуль NumPy, написана программа на данном языке, выполняющая три различные задачи. Для решения математических задач использована библиотека NumPy, позволяющая эффективно работать с линейной алгеброй.

# Приложение А Исходный код программы

Название файла: main.py

import numpy as np

from math import sin, cos

def check\_collision(b1, b2):

A, B = np.vstack((b1[:-1], b2[:-1])), np.array([-b1[-1], -b2[-1]])

if np.linalg.matrix\_rank(np.vstack((b1, b2))) != 2 or np.linalg.matrix\_rank(A) != 2:

return None

collision\_point = np.linalg.solve(A, B)

return tuple(round(i, 2) for i in collision\_point)

def check\_surface(p1, p2, p3):

a1, a2, a3 = p1[:-1], p2[:-1], p3[:-1]

a1 = np.append(a1, 1)

a2 = np.append(a2, 1)

a3 = np.append(a3, 1)

A, B = np.vstack((a1, a2, a3)), np.array([p1[-1], p2[-1], p3[-1]])

if any((np.linalg.matrix\_rank(np.vstack((np.append(a1, B[0]),

np.append(a2, B[1]),

np.append(a3, B[2])))) != 3,

np.linalg.matrix\_rank(A) != 3)):

return None

surface = np.linalg.solve(A, B)

return np.array([round(i, 2) for i in surface])

def check\_rotation(coords, angle):

rotation\_matrix = np.array([[cos(angle), -sin(angle), 0],

[sin(angle), cos(angle), 0],

[0, 0, 1]])

result\_coords = np.dot(rotation\_matrix, coords)

return np.array([round(i, 2) for i in result\_coords])