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. 3342 _____ Шушко Л.Д.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Целью работы является освоение работы с библиотекой numpy на языке Python.

Задание

Вариант 1.

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, 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.

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

Первая функция check_collision находит точку пересечения траекторий двух дакиботов . Для этого она создает матрицы коэффициентов, используя метод numpy.array, проверяет ранк матрицы(numpy.linalg.marix_rank), решает линейное уравнение с помощью метода numpy.linalg.solve и выводит округленное значение(numpy.round) точки пересечения.

Вторая функция check_surface находит коэффициенты для уравнения плоскости, в которой двигались дакиботы. Для этого она создает матрицы коэффициентов используя метод numpy.array, проверяет ранк матрицы(numpy.linalg.matrix_rank), решает уравнение плоскости с помощью метода numpy.linalg.solve и выводит округленные значения(numpy.round) коэффициентов.

Третья функция check_rotation поворачивает дакибота вокруг своей оси. Для этого она создает матрицу поворота в трехмерном пространстве, используя метод numpy.array, умножает матрицу поворота на координаты дакибота с помощью метода np.dot и выводит округленные повернутые координаты(np.round).

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:([-3,-6,9]), [8,-7,0])	(0.91,1.04)	-
2.	check_surface:([1,-6,1],[0,-3,2],[-3,0,-1])	[2. 1. 5.]	-
3.	check_rotation:([1,-2,3],1.57)	[2. 1. 3.]	-

Выводы

Освоена работа с библиотекой на языке на примере использующей ее функций.

Разработаны функции, выполняющие определенные действия над дакиботами.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Shushko_Leya_lb1.py

```
     import numpy as np
     def check_collision(coords1, coords2):
          ndarray1=np.array([coords1[:-1],coords2[:-1]])
          ndarray2=np.array([-coords1[-1],-coords2[-1]])
          if np.linalg.matrix_rank(ndarray1)<2:
                    return None
          else:

intersection_point=np.linalg.solve(ndarray1,ndarray2)

intersection_point=np.round(intersection_point,2)
                    return tuple(intersection_point)
          def check_surface(array1, array2, array3):
                    ndarray_of_constants=np.array([[array1[-1]],[array2[-
1]],[array3[-1]]])
                    array1[-1]=1
                    array2[-1]=1
                    array3[-1]=1
                    ndarray_of_coefficients=np.array([array1,array2,array3])
                    if np.linalg.matrix_rank(ndarray_of_coefficients) < 3:
                              return None
                    else:
                              coefficients_of_surface_equation
np.linalg.solve(ndarray_of_coefficients, ndarray_of_constants)
                              coefficients_of_surface_equation
np.round(coefficients_of_surface_equation, 2)
                    return coefficients_of_surface_equation.flatten()



     def check_rotation(vec, rad):
          rotated_matrix=np.array([[np.cos(rad),-np.sin(rad),0],
[np.sin(rad),np.cos(rad),0],[0,0,1]])
          rotated_ndarray=np.dot(rotated_matrix,vec)
          rounded_rotated_ndarray
np.array([np.round(rotated_ndarray[0],  2),  np.round(rotated_ndarray[1],
2), np.round(rotated_ndarray[2], 2)])
          return rounded_rotated_ndarray
     print(check_collision(coords1,coords2))
```