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 _____ Пушко К.Д.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Целью работы является освоение работы с функциями и библиотекой numpy.

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

Написанная программа написана на языке Python с использованием библиотеки numpy. Она состоит из 3-функций, которые вызываются сразу на сайте https://e.moevm.info.

Первая функция check_collision возвращает решение системы линейных уравнений. Для её реализации было необходимо создать матрицу с коэффициентами, а также матрицу со свободными членами. Далее с по мощью библиотеки numpy выполняются математические операции с матрицами. Далее с по мощью библиотеки numpy выполняются математические операции с матрицами, проверив допускает ли ранг матрицы решение.

Вторая функция check_surface возвращает решение уравнений плоскости вида ax+by+c=z. Для её реализации было необходимо создать матрицу с коэффициентами, а также матрицу со свободными членами. В матрицу коэффициентов так же было необходимо было добавить столбец единиц для корректного решения. Далее с по мощью библиотеки numpy выполняются математические операции с матрицами, проверив допускает ли ранг матрицы решение.

Третья функция check_rotation возвращает повернутую на определенный градус матрицу. Сначала было необходимо вычислить матрицу поворота вокруг оси z. Далее необходимо было умножить изначальную матрицу на матрицу поворота.

Переменные, используемые в программе:
- coefficient_matrix матрица коэффициентов
- c_vector вектор свободных членов
- result неокругленный результат вычисления функции
- rotation_matrix матрица вращения

Функции, используемые в этой программе:

-numpy.array возвращает массив типа numpy.ndarray.

-numpy.linalg.solve возвращает решение системы линейных уравнений.

-round возвращает округленное число до выбранного значения.

-numpy.linalg.matrix_rang возвращает ранг матрицы.

-numpy.dot возвращает результат перемножения двух матриц.

-numpy.cos возвращает косинус значения.

-numpy.sin возвращает синус значения.

Данная программа демонстрирует использование функций библиотеки numpy и работу функций на языке Python для выполнения различных математический операций

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

Были изучены правила работы с функциями, а также работа с библиотекой numpy.

Разработаны функции, возвращающие решения определенных математических заданий.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import numpy as np


def check_collision(bot1, bot2):
    coefficient_matrix = np.array([coefficient for coefficient in
[bot1[:-1], bot2[:-1]]],dtype = int)
    c _vector = np.array([[-float(bot1[-1])],[-float(bot2[-
1])]],dtype = float)
    if np.linalg.matrix_rank(coefficient_matrix) < 2:
        return
    result = list((np.linalg.solve(coefficient_matrix, c _vector)))

    return (round(result[0][0],2) ,round(result[1][0],2))


def check_surface(point1, point2, point3):
    c _vector = np.array([[point1[-1]],[point2[-1]],[point3[-
1]]],dtype = float)
    point1[-1] = 1
    point2[-1] = 1
    point3[-1] = 1
    coefficient_matrix = np.array([coefficient for coefficient in
[point1, point2, point3]],dtype = float)
    if np.linalg.matrix_rank(coefficient_matrix) < 3:
        return
    result = (np.round(np.linalg.solve(coefficient_matrix, c
_vector),2))
    return str(np.rot90(result))[1:-1]

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