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 Гончаров С. А.

Преподаватель Иванов Д. В.

Санкт-Петербург

2023

Цель работы

Изучить основные управляющие конструкции языка Python. Написать программу, состоящую из трёх задач. Каждую задачу вывести в отдельную функцию.

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, 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 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).

Выполнение работы

Для работы были использованы библиотеки numpy и math.

1. Оформить решение в виде отдельной функции check_collision(bot1, bot2). На вход функции подаются два ndarray – коэффициенты bot1, bot2 уравнений прямых bot1 = (a1, b1, c1) bot2 = (a2, b2, c2). Функция возвращает точку пересечений траекторий, округлив координаты до 2 знаков после запятой с помощью round(value, 2).
2. Решение оформлено как отдельная функция check_surface(point1, point2, point3). На вход функции передаются координаты 3 точек (point1, point2, point3). Функция возвращает коэффициенты a, b, c в виде массива для уравнения плоскости вида ax + by + c = z. Результат так же нужно округлить до 2 знаков после запятой.
3. Решение третьей задачи оформлено в отдельной функции check_rotation(coordinates, radians). На вход функции подается массив координат и угол поворота в радианах. Нужно вывести повёрнутые координаты, каждая из которых округлена до 2 знаков после запятой. Поворот выполняется относительно оси z, используя (рис. 1) подставляем значения получаем координаты поворота.

• Вращение вокруг оси z:

$$M_z(\varphi) = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

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)	Верный вывод
2.	check_surface(([1, -6, 1]), ([0, -3, 2]), ([-3, 0, -1]))	[2. 1. 5.]	Верный вывод
3.	check_rotation([1, -2, 3], 1.57)	[2. 1. 3.]	Верный вывод

Выводы

Был создан файл main.py с решением поставленных задач. Я научился использовать основные управляющие конструкции языка python.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import numpy as np
from math import *

def check_collision(bot1, bot2):    #print(check_collision(([-3, -6, 9]),
([8, -7, 0])))
    a1, a2, a3 = map(int, bot1)
    b1, b2, b3 = map(int, bot2)
    array1 = np.array([[a1, a2], [b1, b2]])
    array2 = np.array([-a3, -b3])
    if np.linalg.matrix_rank(array1) < 2:
        return None

    answer = np.linalg.solve(array1, array2)
    return (round(answer[0], 2), round(answer[1], 2))

def check_surface(point1, point2, point3):  #print(check_surface(([1, -
6,  1]), ([0, -3,  2]), ([-3,  0, -1])))
    a1, b1, c1 = map(int, point1)
    a2, b2, c2 = map(int, point2)
    a3, b3, c3 = map(int, point3)
    array1 = ([[a1, b1, 1], [a2, b2, 1], [a3, b3, 1]])
    array2 = ([c1, c2, c3])
    if np.linalg.matrix_rank(array1) < 3:
        return None

    answer = np.linalg.solve(array1, array2)
    return (np.array([round(answer[0], 2), round(answer[1], 2), round(an-
swer[2], 2)]))

def check_rotation(coordinates, radiants):  #print(check_rotation([1, -2,
3], 1.57))
    array = np.array([[cos(radiants), -sin(radiants), 0], [sin(radiants),
cos(radiants), 0], [0, 0, 1]])

    answer = np.dot(array, coordinates)
    return (np.array([round(answer[0], 2), round(answer[1], 2), round(an-
swer[2], 2)]))
```