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 _____ Романов Е.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Освоение основных управляющих конструкция языка python, таких как условные выражения и циклы, освоение работы с функциями и модулем numpy.

Задание

Вариант 1

Задание 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).

Задание 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).

Задание 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 состоит из 3 функций, каждая из которых содержит решение каждого задания.

Функция check_collision вычисляет координаты точки пересечения траекторий движения дакиботов. На вход функции подаются два массива ndarray, содержащих коэффициенты уравнений движения ботов. Точку пересечения можно найти, если вычислить решение линейной системы уравнений, состоящей из матрицы коэффициентов, содержащей в себе коэффициенты при x и y, и вектора правой части, содержащего в себе свободные члены уравнений движения дакиботов. Функция проверяет ранг матрицы: не меньше ли он числа уравнений в матрице коэффициентов и, если это так, то находит решения линейной системы уравнений и возвращает полученные значения x и y, которые округляются до двух знаков после запятой, иначе возвращает None.

Функция check_surface находит коэффициенты уравнения плоскости. На вход функции подаются координаты трёх точек (x,y,z). Координаты точек подставляются в матрицу коэффициентов и в вектор свободных членов. Решение линейной системы уравнений будет содержать в себе коэффициенты уравнения плоскости. Функция проверяет ранг матрицы и, если он не меньше количества уравнений в системе, находит решение системы и возвращает массив ndarray содержащий округлённые до двух знаков после запятой коэффициенты уравнения плоскости, иначе возвращает None.

Функция check_rotation находит повёрнутые вокруг оси z координаты дакибота. На вход подаётся массив ndarray с координатоми бота и угол поворота. Поворот выполняется путём умножения матрицы поворота, элементы которой находятся с помощью использования функций cos и sin модуля math, на вектор-столбец, содержащий координаты x и y. Умножения производится с помощью функции dot модуля numpy.

Решения всех линейных систем уравнений осуществляется с помощью функции solve модуля numpy. А нахождения ранга матриц производится с помощью функции matrix_rank.

Переменные, используемые в программе:

- free_coeficients используется как локальная переменная в функциях check_collision и check_surface для хранения коэффициентов свободных членов в виде вектор-столбца.

- elder_coeficients используется для хранения коэффициентов старших членов уравнений движения дакиботов в функции check_collision

- matrix_coeficients используется для хранения коэффициентов старших членов уравнений движения дакиботов в функции check_surface

result хранит результат выполнения функции numpy.linalg.solve в функциях check_collision и check_surface, а также результат выполнения функции numpy.dot в функции check_rotation.

Функции используемые в программе:

-numpy.array возвращает массив ndarray

-numpy.dot возвращает произведение матриц

-numpy.linalg.matrix_rank возвращает ранг матрицы

-numpy.linalg.solve находит решение линейной системы уравнений

-math.cos находит косинус угла, переданного в радианах

-math.sin находит синус угла, переданного в радианах

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Функция	Входные данные	Выходные данные
1.	check_collision	array([2,4,1]) array([-1,8,10])	(1.6, -1.05)
2.	check_collision	np.array([5,7,3]) np.array([3,-8,2])	(-0.62, 0.02)
3.	check_surface	np.array([5,7,3]) np.array([3,-8,2]) np.array([4,31,-9])	[3.24 -0.37 -10.63]
4.	check_surface	np.array([-10,10,1]) np.array([4,-2,2]) np.array([4,1,-9])	[-3.07 -3.67 6.95]
5.	check_rotation	np.array([-10,10,1]) 2.1	[-3.58 -13.68 1.]
6.	check_rotation	np.array([3,7,1]) 1.2	[-5.44 5.33 1.]

Выводы

Были изучены основные управляющие конструкции языка Python: условные операторы if, циклы for и while, а также модуль numpy.

Была разработана программа, состоящая из трёх функций. Программа работает с массивами ndarray и функциями библиотеки numpy для нахождения решений линейных систем уравнений. Результаты выводятся на экран.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import numpy as np
import math

def check_collision(bot1,bot2):
    free_coeficients = np.array([-bot1[2],-bot2[2]])
    elder_coeficients = np.array([[bot1[0],bot1[1]], [bot2[0],bot2[1]]])
    if np.linalg.matrix_rank(elder_coeficients) >=2:
        result = np.linalg.solve(elder_coeficients, free_coeficients)
        return (round(result[0],2), round(result[1],2))
    else: return None


def check_surface(point1,point2,point3):
    matrix_coeficients =
np.array([[point1[0],point1[1],1],[point2[0],point2[1],1],[point3[0],poin
t3[1],1]])
    free_coeficients = np.array([point1[2],point2[2],point3[2]])
    if np.linalg.matrix_rank(matrix_coeficients)>=3:
        result = np.array([round(x,2) for x in
np.linalg.solve(matrix_coeficients,free_coeficients)])
        return result
    else: return None


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