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: Управляющие конструкции языка Python

Студент гр. 3344 _____ Волохов М.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

Цель работы

Освоение работы с управляющими конструкциями на языке Python.

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. Два дакибота приближаются к перекрестку. Чтобы избежать столкновения, им необходимо знать точку пересечения их траекторий движения. Траектории -- линейные, и дакиботы уже вычислили коэффициенты этих уравнений. Ваша задача -- помочь ботам вычислить точку потенциального столкновения. 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. Три дакибота начали движение, отъехали от условной точки старта и через некоторое время остановились. Каждый дакибот уже вычислил свою координату относительно точки старта. Дакиботам нужно передать на базу карту местностности, по которой они двигались. Для построения карты местности необходимо знать уравнение плоскости. Ваша задача -- помочь дакиботам найти уравнение плоскости, в которой они двигались. 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 до 2 знаков после запятой с помощью round(value, 2).

Задача 3. Дакибот выехал на перекресток и готовится к выполнению поворота вокруг своей оси (вокруг оси z), чтобы продолжить движение в другом

направлении. Он знает свои координаты и знает угол поворота (в радианах). Помогите дакиботу повернуться в нужное направление для продолжения движения. 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

Была импортирована библиотека *numpy*.

Функция check_collision(bot1, bot2), принимающая на вход два ndarray – коэффициенты bot1, bot2 уравнений прямых bot1 = (a1, b1, c1), bot2 = (a2, b2, c2), уравнение прямой имеет вид ax+by+c=0. Эта функция проверяет, пересекаются ли два бота на плоскости. Функция создает систему линейных уравнений на основе данных ботов, и, если система уравнений имеет решение (ранг матрицы a равен 2), то возвращает точку пересечения. Если система не имеет решения, функция возвращает None.

Функция def check_surface(point1, point2, point3), принимающей на вход три ndarray – координаты дакиботов, Эта функция проверяет, можно ли по трем точкам на плоскости построить плоскость (прямую в трехмерном пространстве). Функция создает систему линейных уравнений на основе данных точек, и, если система уравнений имеет решение (ранг матрицы A равен 3), то возвращает параметры плоскости в виде вектора [A, B, C], где уравнение плоскости будет иметь вид Ax + By + Cz = D. Если система не имеет решения, функция возвращает

Функция def check_rotation(coordinates, angle), принимает два аргумента: coordinates - массив координат точек, представленных в виде [x, y], и angle - угол в радианах, на который необходимо повернуть точки. Эта функция выполняет поворот точек в двумерном пространстве на заданный угол. Функция создает матрицу поворота и применяет ее к переданным точкам, возвращая новые координаты после поворота, округленные до двух знаков после запятой.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	[-3, -6, 9], [8, -7, 0]	[0.91, 1.04]	-
2.	[1, -6, 1], [0, -3, 2], [-3, 0, -1]	[2. 1. 5.]	-
3.	[1, -2, 3], [1.57]	[2. 1. 3.]	-

Выводы

Была освоена работа с управляющими конструкциями на языке Python. Были получены навыки работы с библиотекой numpy. Были освоены функции решения линейных систем уравнения, умножения матриц, получения ранга матрицы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Volokhov_Mikhail_lb1.py

```
import numpy as np


def check_collision(bot1, bot2):
    a = np.array([[bot1[0], bot1[1]], [bot2[0], bot2[1]]])
    b = np.array([-bot1[2], -bot2[2]])

    if np.linalg.matrix_rank(a) < 2:
        return None
    intersection = np.linalg.solve(a, b)
    intersection = tuple(round(value, 2) for value in intersection)

    return intersection


def check_surface(point1, point2, point3):
    A = np.array([[point1[0], point1[1], 1], [point2[0], point2[1], 1],
[point3[0], point3[1], 1]])
    B = np.array([point1[2], point2[2], point3[2]])

    if np.linalg.matrix_rank(A) == 3:
        solution = np.linalg.solve(A, B)
        return np.round(solution, 2)
    else:
        return None


def check_rotation(coordinates, angle):
    rotation_matrix = np.array([[np.cos(angle), -np.sin(angle), 0],
[np.sin(angle), np.cos(angle), 0], [0, 0, 1]])

    rotated_coordinates = np.dot(rotation_matrix, coordinates)
    rotated_coordinates = np.round(rotated_coordinates, 2)

    return rotated_coordinates
```