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. 3344		Тукалкин. В.А.
Преподаватель		Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Ознакомиться с основными управляющими конструкциями языка Python и библиотеки Numpy.

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. Содержательная постановка задачи

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. Содержательная часть задачи

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 каждого коэффициента до 2 знаков после запятой с помощью round(value, 2).

Задача 3. Содержательная часть задачи

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

Выполнение работы будет расписано по шагам:

1) Подключить модуль numpy и сократить до np.

2) Написать функцию check_collision, в которой при помощи np.linalg.matrix_rank проверяется, чтобы суммарный ранг двух поступающий матриц был равен 2. Если ранг равен 2, то матрицы разделяются, чтобы посчитать уравнение прямой вид ax+by+c=0 и считается точка пересечения траекторий и округляется до 2 знаков после запятой. Если ранг не равен 2, то возвращается ‘None’.

3) Написать функцию check_surface, в которую поступают координаты 3 точек, проверяется суммарный ранг 3 матриц, если он равен 3, то считаются коэффициенты a, b, c для уравнения плоскости вида ax+by+c=z. Далее данные округляются до 2 знаков после запятой с помощью round и возвращаются в виде матрицы. Если ранг не равен 3, то возвращается ‘None’.

4) Написать функцию check_rotation, в которую поступают координаты и угол, строится матрица поворота и считаются координаты. Функция возвращает повернутые ndarray координаты, каждая из которых округлена до 2 знаков после запятой с помощью round.

5) Методы, которые использовались для решения:

np.linalg.matrix_rank – определения ранга матрицы

np.array – создание матрицы

np.linalg.inv – вычисление обратной матрицы

np.linalg.solve – решение линейного матричного уравнения или системы линейных скалярных уравнений.

np.round – округление значений матрицы

np.transpose – перемещение значений матрицы

np.hstack – добавить столбец

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	check_collision input: [5 -4 8] [-7 9 2] check_surface input: [1 -6 1] [0 -3 2] [-3 0 -1] check_rotation input: [1 -2 3] 1.57	(-4.71, -3.88) [2. 1. 5.] [2. 1. 3.]	Верный ответ
2.	check_collision input: [-2 7 9] [1 -8 0] check_surface input: [1 -2 3] [2 -3 4] [3 -4 5] check_rotation input: [2 -2 -1] 1.13	(8.0, 1.0) None [2.66 0.96 -1.]	Верный ответ
3.	check_collision input: [-8 -6 0] [-4 -8 1] check_surface input: [1 -3 -1] [-2 7 2] [3 2 -4] check_rotation input: [5 -6 1] 0.87	(-0.15, 0.2) [-1.29 -0.09 0.03] [7.81 -0.05 1.]	Верный ответ

Выводы

Были изучены основные управляющие конструкции языка Python и библиотеки Numpy на примере использующей их программы.

Разработана программа, выполняющая обработку информации, поступающей от дакиботов и возвращающая результат. Обработка информации производилась с помощью модуля numpy и встроенных в него функций.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1.py

```
import numpy as np

def check_collision(bot1, bot2):
    if np.linalg.matrix_rank([bot1,bot2])==2:
        left_side=np.array([bot1[:2],bot2[:2]])
        right_side=np.array([bot1[2:]*-1,bot2[2:]*-1])
        matrix_final=np.linalg.inv(left_side).dot(right_side)
        return round(matrix_final[0][0],2),round(matrix_final[1][0],2)
    else: return 'None'

def check_surface(point1, point2, point3):
    if np.linalg.matrix_rank([point1, point2, point3])==3:
        left_side=np.array([[point1[0],point1[1],1],
[point2[0],point2[1],1], [point3[0],point3[1],1]])
        right_side=np.array([point1[2],point2[2],point3[2]])
        return np.round(np.linalg.solve(left_side,right_side),2)
    else: return 'None'

def check_rotation(vec, rad):
    matrix_rotation=np.transpose(np.array([[np.cos(rad),              -
1*np.sin(rad)], [np.sin(rad), np.cos(rad)]]))

matrix_final=np.round(np.linalg.solve(matrix_rotation,np.array([vec[0]
,vec[1]])),2)
    return np.hstack([matrix_final,vec[2]])
```