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

Студентка гр. 3342 _____ Смирнова Е.С.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Изучить основные управляющие конструкции языка Python, изучить библиотеку numpy и использовать полученные знания на примере практического задания.

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.

Формальная постановка задачи

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 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 карт местности необходимо знать уравнение плоскости. Ваша задача – помочь дакиботам найти уравнение плоскости, в которой они двигались.

Формальная постановка задачи

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.

Формальная постановка задачи

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

Для решения поставленных задач были написаны 3 функции с использованием подключаемого модуля *numpy*, с псевдонимом *np*.

Функция *check_collision* принимает на вход два аргумента *bot1* и *bot2* типа *ndarray*. По условию данные аргументы принимают коэффициенты уравнения прямых.

С помощью *np.array* создается матрица *mat ([[bot1[0],bot1[1]],[bot2[0],bot2[1]])* и вектор *vec (-bot1[2], -bot2[2])*. С помощью *np.linalg.matrix_rank* проверяется ранг матрицы, если он равен 2, то функция через метод *np.linalg.solve* возвращает решение линейной системы уравнений, округленных до двух знаков после запятой с помощью *np.round* . Если же ранг матрицы не равен 2, то функция возвращает *None*.

Функция *check_surface* принимает на вход три аргумента *point1, point2* и *point3.* По условию данные аргументы принимают значения координат трех точек.

С помощью *np.array* создается матрица *mat ([[point1[0],point1[1],1],[point2[0],point2[1],1],[point3[0],point3[1],1]])* и вектор *vec (point1[2],point2[2],point3[2]).* С помощью *np.linalg.matrix_rank* проверяется ранг матрицы, если он равен 3, то функция через метод *np.linalg.solve* возвращает решение линейной системы уравнений, округленное до двух знаков после запятой с помощью *np.round* . Если же ранг матрицы не равен 3, то функция возвращает *None*.

Функция *check_rotation* принимает на вход 2 аргумента *mat* и *rad.* По условию данные аргументы принимают значения 3-х координат дакибота и угол поворота.

С помощью *np.array* создается матрица *mat ([[np.cos(rad), -np.sin(rad), 0],[np.sin(rad), np.cos(rad), 0],[0, 0, 1]]).* С помощью *mat.dot* выполняется умножение матрицы *mat* на переменную типа *ndarray*. Функция возвращает значение повернутых координат матрицы, округленных до двух знаков после запятой с помощью *np.round*.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	[-8 -4 7] [8 1 8]	(-1.62, 5.0)	Функция *check_collision*
2.	[-8 -2 3] [-5 6 0]	(0.31, 0.26)	Функция *check_collision*
3.	[8 1 7] [3 8 1]	(-0.9, 0.21)	Функция *check_collision*
4.	[1 -6 1] [0 -3 2] [-3 0 -1]	[2. 1. 5.]	Функция *check_surface*
5.	[1 -2 3] [2 -3 4] [3 -4 5]	None	Функция *check_surface*
6.	[1 -3 -1] [-2 7 2] [3 2 -4]	[-1.29 -0.09 0.03]	Функция *check_surface*
7.	[1 -2 3] 0.87	[2.17 -0.53 3.]	Функция *check_rotation*
8.	[2 -2 -1] 1.4	[2.31 1.63 -1.]	Функция *check_rotation*
9.	[5 -6 1] 1.57	[6. 5. 1.]	Функция *check_rotation*

Выводы

Были изучены управляющие конструкции языка *Python*, встроенный модуль *numpy* и метод *numpy.linalg*. С помощью полученных знаний были составлены функции для решения практических задач: вычисление точки пересечения траекторий, нахождение уравнения плоскости и нахождения координат при повороте.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1.py

```
import numpy as np

def check_collision(bot1, bot2):
    mat=np.array([[bot1[0],bot1[1]],[bot2[0],bot2[1]]])
    vec=np.array([-bot1[2],-bot2[2]])
    if np.linalg.matrix_rank(mat)==2:
        rez= np.linalg.solve(mat,vec)
        return (round(rez[0],2),round(rez[1],2))
    else:
        return None

def check_surface(point1, point2, point3):

mat=np.array([[point1[0],point1[1],1],[point2[0],point2[1],1],[point3[
0],point3[1],1]])
    vec=np.array([point1[2],point2[2],point3[2]])
    if np.linalg.matrix_rank(mat)==3:
        return (np.round(np.linalg.solve(mat,vec),2))
    else:
        return None

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