МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Информатика»

Тема: Основные управляющие конструкции языка Python

Студентка гр. 2381 _____ Аникина Д.Р.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2022

Цель работы.

Изучить основные управляющие конструкции языка Python, а также подключаемый модуль 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.

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 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.

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.

Функция *check_collision(bot1, bot2)*

На вход объявляются два аргумента bot1 и bot2 типа ndarray. По условию данные аргументы принимают коэффициенты уравнения прямых.

С помощью np.array создается матрица a ([bot1[0], bot1[1], [bot2[0], bot2[1]) и вектор vec (-bot1[2], -bot2[2]).

Через np.linalg.matrix_rank проверяется ранг матрицы, если он равен 2, то функция возвращает решение линейной системы уравнений, найденной через метод np.linalg.solve. При этом используется встроенная функция tuple(), которая меняет тип ndarray на кортеж. Если же ранг матрицы не равен двум, то функция возвращает None.

Функция check_surface(point1, point2, point3)

На вход функции подаются подаются значения координат трех точек типа ndarray.

Через np.vstack((point1, point2, point3)) создается матрица 3 на 3. После значения последнего столба заменяются на единицы. Создается вектор b через np.array

Матрица проверяется на ранг. Если ее ранг равен 3, то функция находит значение системы линейных уравнений через np.linalg.solve(a, b) (это записывается в переменную rez). Далеев переменной p сохраняется значение переменной rez, округленное до двух значений после запятой с помощью round(). Функция возвращает p.

В противном случае, если ранг матрицы не равен 3, то функция возвращает None.

Функция check_rotation(vec, rad)

На вход функции подаются координаты дакибота типа ndarray и угол поворота типа float. С помощью np.array создается матрица a. С помощью метода dot матрица умножается на переменную типа ndarray. Функция возвращает значение повернутых координат матрицы, округленных до двух знаков после запятой с помощью round.

Разработанный программный год см. в приложении А.

Тестирование.

Результаты тестирования представлены в таблице 1.

Таблица 1 - Результаты тестирования.

№ п/п	Входные данные	Выходные данные	Комментарий
1.	array([-3, -6, 9]), array([8, -7, 0])	(0.91, 1.04)	Верный ответ
2.	array([1, -6, 1]), array([0, -3, 2]), array([-3, 0, -1])	[2. 1. 5.]	Верный ответ
3.	array([1, -2, 3]), 1.57	[2. 1. 3.]	Верный ответ

Вывод.

Были изучены управляющие конструкции языка Python и встроенный модуль numpy. 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

```
import numpy as np
def check_collision(bot1, bot2):
    a = np.array([
        [bot1[0], bot1[1]],
        [bot2[0], bot2[1]]]
    )
    if np.linalg.matrix_rank(a) == 2:
        vec = np.array([-bot1[2], -bot2[2]])
        return tuple(np.linalg.solve(a, vec).round(2))
    else:
        return None
def check_surface(point1, point2, point3):
    a = np.vstack((point1, point2, point3))
    a[0][2] = 1
    a[1][2] = 1
    a[2][2] = 1
    b = np.array([point1[2], point2[2], point3[2]])
    if np.linalg.matrix_rank(a) == 3:
        rez = np.linalg.solve(a, b)
```

```
        p = np.round(rez, 2)
        return p
    else:
        return None
def check_rotation(vec, rad):
    a = np.array([
        [np.cos(rad), -np.sin(rad), 0],
        [np.sin(rad), np.cos(rad), 0],
        [0, 0, 1],
    ])
    return np.round(a.dot(vec), 2)
```