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

Студент гр. 3341 Гребенюк В.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Целью работы является освоение работы с основными управляющими конструкции языка Python на примере использующей их программы.

Задание

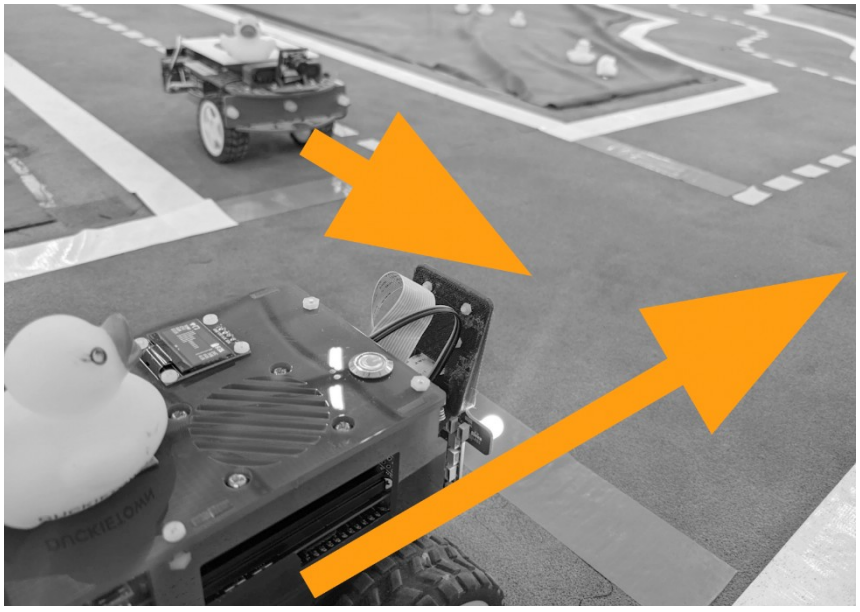
Вариант 1

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.

Пример ситуации:



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)*.

Пример входных данных:

array([-3, -6, 9]), array([8, -7, 0])

Пример возвращаемого результата:

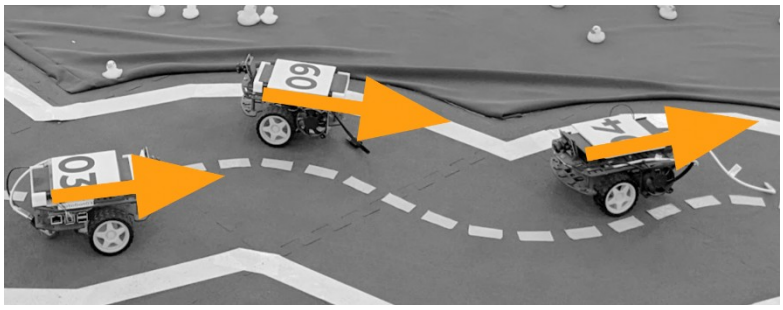
(0.91, 1.04)

Примечание: помните про ранг матрицы и как от него зависит наличие решения системы уравнений. В случае, если решение найти невозможно (например, из-за линейно зависимых векторов), функция должна вернуть ***None***.

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.

Пример ситуации:



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)*.

Например, даны точки: A(1, -6, 1); B(0, -3, 2); C(-3, 0, -1). Подставим их в уравнение плоскости:

$$a \cdot 1 + b(-6) + c = 1$$
$$a \cdot 0 + b(-3) + c = 2$$
$$a(-3) + b \cdot 0 + c = -1$$

Составим матрицу коэффициентов:

$$\begin{pmatrix} 1 & -6 & 1 \\ 0 & -3 & 1 \\ -3 & 0 & 1 \end{pmatrix}$$

Вектор свободных членов:

$$\begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}$$

Для такой системы уравнение плоскости имеет вид: *z = 2x + 1y + 5*

Пример входных данных:

array([1, -6, 1]), array([0, -3, 2]), array([-3, 0, -1])

Возвращаемый результат: [2. 1. 5.]

Примечание: помните про ранг матрицы и как от него зависит существование решения системы уравнений. В случае, если решение найти невозможно (невозможно найти коэффициенты плоскости из-за, например, линейно зависимых векторов), функция должна вернуть ***None***.

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

<Ссылка на википедию с матрицами поворота>

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)*..

Пример входных аргументов:

array([1, -2, 3]), 1.57

Пример возвращаемого результата:

[2. 1. 3.]

Отчет

В отчете обязательно распишите те методы линейной алгебры и модуля numpy, которые вы использовали при решении задач.

Выполнение работы

Функции:

- *check_collision(bot1: np.ndarray, bot2: np.ndarray)*: проверяет столкновение двух роботов и возвращает координаты точки столкновения.
- *check_surface(point1: np.ndarray, point2: np.ndarray, point3: np.ndarray)*: проверяет поверхность, образованную тремя точками, и возвращает координаты точки пересечения поверхности.
- *check_rotation(vec: np.ndarray, rad: float)*: выполняет поворот вектора на заданный угол и возвращает новый вектор.

Импортированные модули:

- *numpy (np)*

В данном коде используются следующие методы линейной алгебры и функции модуля NumPy:

- *np.linalg.solve(A, b):* Этот метод решает систему линейных уравнений Ax = b, где A - матрица коэффициентов, b - вектор правой части, x - вектор неизвестных. В данном коде он используется для решения системы уравнений и нахождения точки пересечения линий или плоскостей.
- *np.array():* Эта функция создает массив *ndarray* из переданных элементов.
- *np.column_stack():* Эта функция объединяет массивы по столбцам.
- *np.dot(A, B):* Эта функция выполняет умножение матрицы A на матрицу B. В коде она используется для поворота вектора *vec* на заданный угол *rad* с помощью матрицы поворота.

- *np.cos(rad), np.sin(rad):* Эти функции вычисляют косинус и синус угла *rad*. В коде они используются для создания матрицы поворота для метода *np.dot()*.
- *np.round(arr, decimals):* Эта функция округляет значения в массиве arr до указанного количества десятичных знаков *decimals*. В коде она используется для округления результатов вычислений.

Выводы

Основные управляющие конструкции языка Python были успешно усвоены. Эти операторы являются ключевыми при написании программ на языке Си**.**

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import numpy as np


def check_collision(bot1: np.ndarray, bot2: np.ndarray):
    return tuple(
        np.linalg.solve(
            np.array([bot1[:2], bot2[:2]]),
            [-bot1[2], -bot2[2]],
        ).round(2)
    )


def check_surface(point1: np.ndarray, point2: np.ndarray, point3:
np.ndarray):
    try:
        return np.linalg.solve(
                        np.column_stack(([point1[:2],  point2[:2],
point3[:2]], [1, 1, 1])),
            np.array([point1[2], point2[2], point3[2]]),
        ).round(2)
    except np.linalg.LinAlgError:
         # no solutions linalg.det is 0 or it unsolvable i dunno
why
        return None


def check_rotation(vec: np.ndarray, rad: float):
    return np.dot(
        [
            [np.cos(rad), -np.sin(rad), 0],
            [np.sin(rad), np.cos(rad), 0],
            [0, 0, 1],
        ],
        vec,
    ).round(2)
```