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. 3341 _____ Анисимов Д.А

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2023

Цель работы

Цель данной работы состоит в изучении основных управляющих структур в языке Python, а также в овладении методами работы с модулем numpy и их применении на практике.

Задание

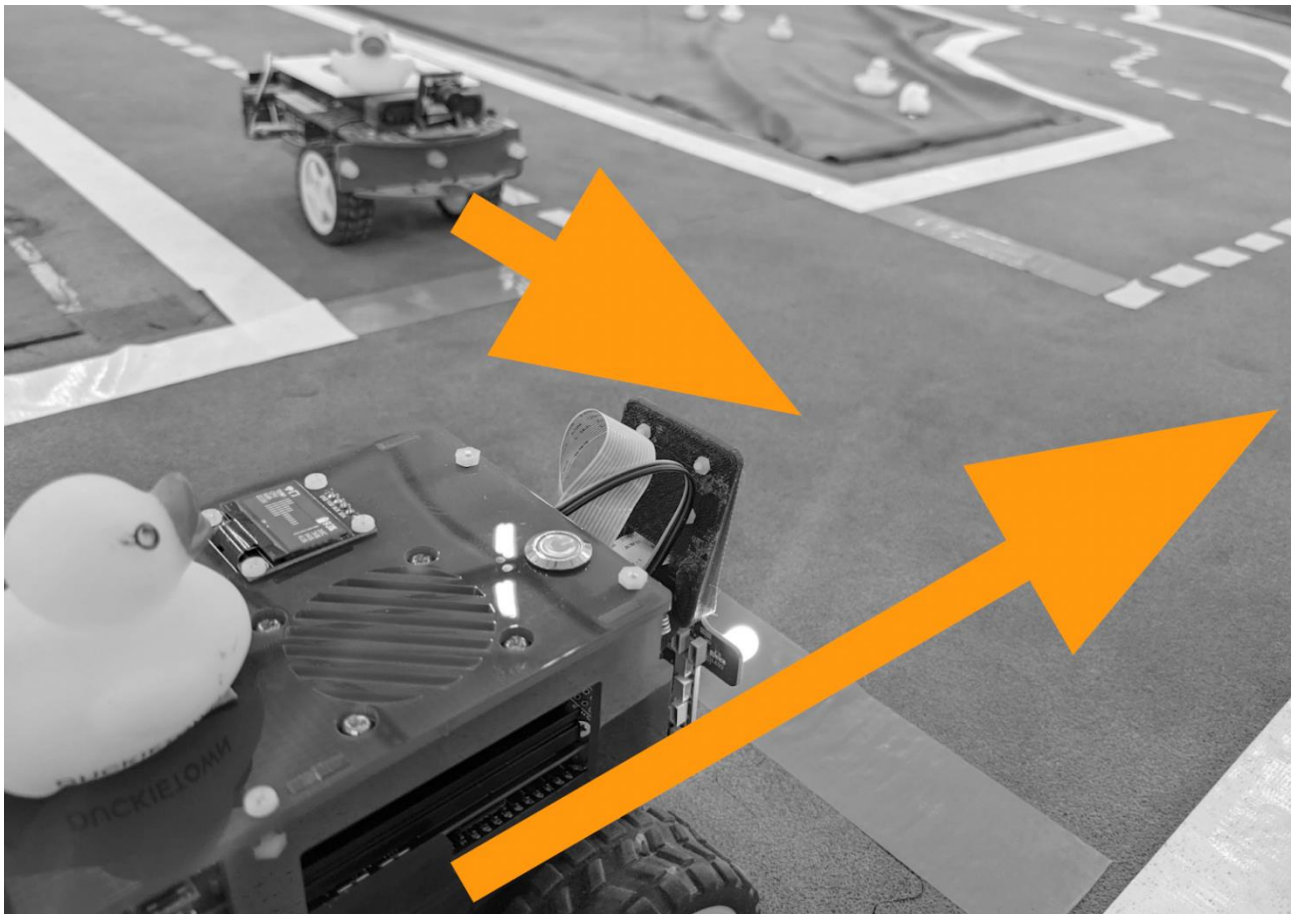
Вариант 1

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:

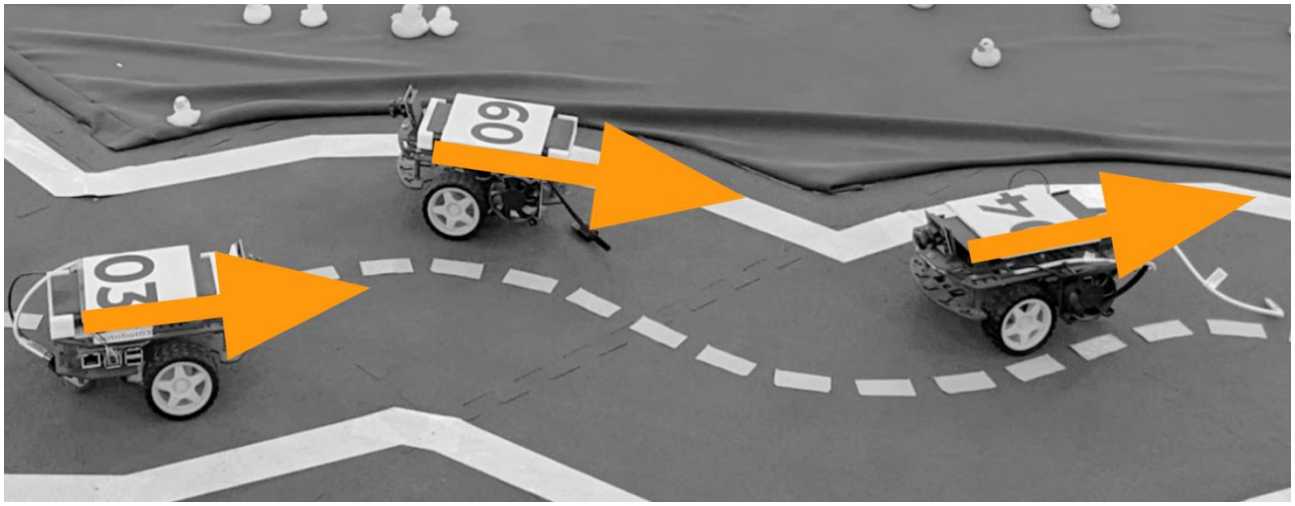
(0.91, 1.04

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 местности, 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

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.

Основные теоретические положения

Для выполнения лабораторной работы использовались библиотека *numpy*, которая предназначена для выполнения математических вычислений в Python, включая базовые функции и линейную алгебру. Применение *numpy* включает в себя использование методов линейной алгебры и работы с матрицами.

numpy.ndarray представляет собой массив или матрицу для работы с большим количеством чисел.

numpy.array(a) преобразует список a в тип *ndarray*.

numpy.vstack и *numpy.hstack* это методы для объединения матриц по вертикали и горизонтали.

Вычисления ранга матрицы *numpy.linalg.matrix_rank*.

Решения систем линейных уравнений *numpy.linalg.solve*.

Вычисление синуса *numpy.sin* и косинуса *numpy.cos* угла в радианах.

Умножение матриц *numpy.dot*.

Для округления чисел используется встроенный метод *round(a, b)*, который возвращает значение a, округленное до b знаков после запятой.

Выполнение работы

Для решения поставленных задач используется модуль *numpy,* который подключается с помощью выражения "*import numpy as np*". После подключения модуля по имени *np* можно обращаться к его функциям. Для каждой из поставленных задач написана соответствующая функция.

1. Функция *check_collision(bot1, bot2)* - принимает два аргумента бота (*bot1* и *bot2*), как списки из трех элементов (a, b, c). Для каждого бота извлекаются значения a, b, c, и они используются для создания матрицы a с помощью *numpy*. Затем функция проверяет ранг матрицы *a* с помощью *np.linalg.matrix_rank(a)*. Если ранг меньше 2, то функция возвращает *None*. В противном случае, создается вектор *b* из значений *-c1* и *-c2*, и с помощью *np.linalg.solve(a, b)* находится точка пересечения этих двух ботов. Затем точка округляется до двух знаков после запятой с помощью *np.round()*, и возвращается в виде кортежа.

2. Функция *check_surface(point1, point2, point3)* - принимает три точки *(point1, point2, point3)* в виде списков из трех элементов (x, y, z). Сначала создается матрица *cef_mat*, которая объединяет точки в столбцы и добавляет дополнительный элемент 1 к каждой точке в виде *np.hstack()* и *np.vstack()*. Затем создается матрица *stb_mat*, которая содержит только значения z-координаты для каждой точки. Затем ранг *cef_mat* проверяется с помощью *np.linalg.matrix_rank(cef_mat)*. Если ранг больше или равен 3, то с помощью *np.linalg.solve(cef_mat, stb_mat)* находится решение для матрицы *cef_mat*. Результат округляется до двух знаков после запятой, и возвращается в виде массива.

3. Функция *check_rotation(coordinates, angle)* - принимает массив координат *(coordinates)* и угол поворота *(angle)*. Создается матрица *rotation_matrix*, используя функции *np.cos()* и *np.sin()* для вычисления значений элементов матрицы. Затем координаты умножаются на матрицу поворота с помощью *np.dot()*, и результат округляется до двух знаков после запятой с помощью *np.round()*. Возвращается округленная матрица координат.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	*np.array*([-3, -6, 9]), *np.array*([8, -7, 0])	(0.91, 1.04)
2.	*np.array*([1, 2, 3]), *np.array*([4, 5, 6])	(0.55, -0.1)
3.	*np.array*([1, -6, 1]), *np.array*([0, -3, 2]), *np.array*([-3, 0, -1])	[2. 1. 5.]
4.	*np.array*([1, -2, 3]), 1.57	[2. 1. 3.]
5.	np.array([1, 2, 3]), np.array([4, 5, 6]), np.array([-7, 8, 9])	None

Выводы

В процессе выполнения работы были достигнуты следующие цели:

1. Освоение основных управляющих конструкций в языке Python, таких как условные операторы if-elif-else. Были изучены основные принципы и синтаксис работы с этими конструкциями.

2. Изучение основных методов работы с библиотекой numpy.

3. Практическое применение полученных навыков для закрепления результатов.

Таким образом, цель работы была успешно достигнута. Овладение основными управляющими конструкциями в языке Python и их использование на практике позволило улучшить навыки программирования и использования различных инструментов для обработки информации.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import numpy as np
def check_collision(bot1, bot2):
    a1, b1, c1 = bot1
    a2, b2, c2 = bot2
    a = np.array([[a1, b1], [a2, b2]])
    if np.linalg.matrix_rank(a) < 2:
        return None
    b = np.array([-c1, -c2])
    point = np.linalg.solve(a, b)
    rounded_point = np.round(point, 2)
    return tuple(rounded_point)
def check_surface(point1, point2, point3):
    cef_mat=          np.vstack((np.hstack((point1[:2],          1)),
np.hstack((point2[:2], 1)), np.hstack((point3[:2], 1))))
    stb_mat = np.vstack((point1[2], point2[2], point3[2]))
    if np.linalg.matrix_rank(cef_mat) >= 3:
        result = np.linalg.solve(cef_mat, stb_mat)
        return np.array([round(result[0][0], 2), round(result[1][0],
2), round(result[2][0], 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)
    rounded_rotated_coordinates = np.round(rotated_coordinates, 2)
    return rounded_rotated_coordinates
```