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. 3343 _____ Пивоев Н.М.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2023

Цель работы

Создание программы на языке Python с использованием основных управляющих конструкций языка, а также ознакомление с модулем *numpy* и применение его в созданном проекте.

Задание

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.

Рисунок 1 – Задача 1

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.

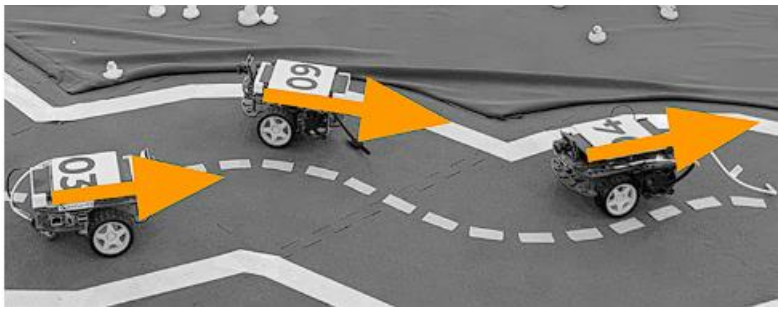

Рисунок 2 – Задача 2

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)*.

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.

Рисунок 3 – Задача 3

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

Созданный проект включает три функции, направленных на управление дакиботами. Каждая функция направлена на выполнение соответственной задачи.

Первая функция – *check_collision*. Она определяет точку пересечения траекторий двух дакиботов, основываясь на двух уравнениях прямых, которые подаются на вход функции. С помощью метода *linalg.solve* модуля *numpy* определяется *crossingPoint* путём решения системы линейных уравнений. Полученные значения округляются до сотых и включаются в кортеж. Функция возвращает данный кортеж.

Вторая функция – *check_surface*. Она находит уравнение плоскости через три точки, которые подаются на вход функции. Создаётся матрица элементов вида $a + b + c = d$, пусть $c = 1$, потому что вариантов уравнений бесконечно много. *a, b, c* хранятся в *index*, а свободные члены *d* в *vector*. Перед расчётом уравнения плоскости идёт проверка на существование решения с помощью метода *linalg.matrix_rank*. Если решения нет, то функция возвращает *None*. Если решение есть, то с помощью метода *linalg.solve*, принимающего матрицу из трёх точек и свободных членов, определяется уравнение плоскости. Коэффициенты уравнения округляются до сотых с помощью метода *round*. Функция возвращает уравнение плоскости.

Третья функция – *check_rotation*. Она находит координаты дакибота при повороте на *rad*, основываясь на начальных координатах. На вход подаётся начальные координаты *vec* и угол поворота в радианах *rad*. Поскольку координата *z* не изменяется, то можно выполнять расчёты для двухмерного пространства. С помощью матрицы поворота можно вычислить необходимые координаты. Создаётся две матрицы: первая состоит из косинусов и синусов данного угла, вторая состоит из координат *x* и *y*. С помощью метода *dot* вычисляется произведение матриц и возвращается матрица *twoDimensions* с координатами *x* и *y* после поворота. Эти координаты округляются до сотых с

помощью метода *round*. С помощью метода *append* модуля *numpy* создаётся новая матрица threeDimensions на основе предыдущей, в которую также добавляется координата *z*. Функция возвращает эту матрицу.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования содержатся в таблице 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	[5, 1, -2] [8, -2, 0]	(0.22, 0.89)	Тестирование функции *check_collision*
2.	[5, -2, 3] [1, 2, 4] [-5, 0, 1]	[0.31 0.56 2.56]	Тестирование функции *check_surface*
3.	[3, 2, 1] 2.31	[-3.5 0.87 1]	Тестирование функции *check_rotation*

Выводы

В результате работы были реализованы функции с использованием основных управляющих конструкций языка и модуля *numpy*, также изучены основные методы этого модуля.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import numpy


def check_collision(bot1, bot2):
    index = [[bot1[0],bot1[1]],[bot2[0],bot2[1]]]
    vector = [-bot1[2],-bot2[2]]
    crossingPoint = numpy.round(numpy.linalg.solve(index,vector),2)
    return tuple(crossingPoint)


def check_surface(point1, point2, point3):
    index1 = [point1[0], point1[1], 1]
    index2 = [point2[0], point2[1], 1]
    index3 = [point3[0], point3[1], 1]
    vector = [point1[2], point2[2], point3[2]]
    if numpy.linalg.matrix_rank([index1, index2, index3]) != 3:
        return None
    equation = numpy.round(numpy.linalg.solve([index1, index2, in-
dex3],vector),2)
    return equation


def check_rotation(vec, rad):
    line1 = [numpy.cos(rad),-numpy.sin(rad)]
    line2 = [numpy.sin(rad),numpy.cos(rad)]
    line = numpy.vstack((line1,line2))
    vector = [vec[0],vec[1]]
    twoDimensions = numpy.round(numpy.dot(line,vector),2)
```

```
threeDimensions = numpy.append(twoDimensions,vec[2])

return threeDimensions
```