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

Студен гр. 3343 _____ Кербель Д. А.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург

2023

Цель работы

Изучить и научиться применять основные управляющие конструкции языка Python и библиотеку NumPy.

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.

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)*.

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

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

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 выполнения поставленных задач, мною была написана программа на языке Python, в которой описываются функции для решения поставленных задач.

Самая первая функция check_collision(bot1, bot2) выполняет первую задачу. Она принимает два аргумента bot1 и bot2, которые представляют собой списки с 3 элементами. Создает матрицу A размером 2x2, где первая строка состоит из элементов bot1, а вторая строка из элементов bot2. Создает вектор b, состоящий из отрицательных значений третьего элемента каждого бота. Далее проверяет ранг матрицы A. Если он меньше 2, функция возвращает None. Иначе, пытается решить уравнение A * x = b методом linalg.solve, где x - искомая точка пересечения. Округляет полученную точку пересечения до двух знаков после запятой. Возвращает кортеж с округленной точкой пересечения, если уравнение было успешно решено, иначе возвращает None.

Вторая задача выполняется функцией check_surface(pt1, pt2, pt3). Функция принимает три аргумента pt1, pt2, pt3, которые представляют собой списки с 3 элементами. Извлекает координаты x, y и z для каждой точки из аргументов. Создает матрицу matrix_coeff размером 3x3, где каждая строка состоит из координат x, y и 1 для соответствующей точки. Создает вектор free_vector, состоящий из координат z для каждой точки. Проверяет значение определителя матрицы matrix_coeff. Если он равен 0.0, функция возвращает None. Иначе, решает уравнение matrix_coeff * x = free_vector методом linalg.solve, где x - искомая точка пересечения поверхности. Округляет полученную точку пересечения до двух знаков после запятой. Возвращает полученную точку пересечения, если уравнение было успешно решено, иначе возвращает None.

Для решения третьей задачи написана функция check_rotation(coordinates, angle). Она принимает два аргумента coordinates и

angle. Создает матрицу rotation_matrix размером 2x2, содержащую значения cos(angle) и -sin(angle) в первой строке и sin(angle) и cos(angle) во второй строке. Применяет матрицу rotation_matrix к первым двум элементам списка coordinates с помощью функции np.dot. Округляет полученные координаты до двух знаков после запятой. Объединяет округленные координаты с остальными элементами из списка coordinates. Возвращает полученный список округленных 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	Комментарии
1.	[-3,-6,9] [8,-7,0]	(0.91, 1.04)	Выходные данные соответствуют ожиданиям.
2.	[1,-6,1] [0,-3,2] [-3,0,-1]	[2. 1. 5.]	Выходные данные соответствуют ожиданиям.
3.	[1,-2,3] 1.57	[2. 1. 3.]	Выходные данные соответствуют ожиданиям.

Выводы

В ходе выполнения лабораторной работы, мною были изучены управляющие конструкции языка Python и модуль NumPy. Была оформлена программа, выполняющая три задачи. Для решения поставленных математических задач была использована библиотека NumPy, позволяющая эффективно работать с линейной алгеброй.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import numpy as np

def check_collision(bot1, bot2):
    A = np.array([[bot1[0], bot1[1]], [bot2[0], bot2[1]]])
    b = np.array([-bot1[2], -bot2[2]])
    if np.linalg.matrix_rank(A) < 2:
        return None
    try:
        intersection = np.linalg.solve(A, b)
        rounded_intersection = np.round(intersection, 2)
        return tuple(rounded_intersection)
    except np.linalg.LinAlgError:
        return None
def check_surface(pt1, pt2, pt3):
    x1, y1, z1 = pt1
    x2, y2, z2 = pt2
    x3, y3, z3 = pt3
    matrix_coeff = np.array([[x1, y1, 1], [x2, y2, 1], [x3, y3, 1]])
    free_vector = np.array([z1, z2, z3])
    if np.linalg.det(matrix_coeff) == 0.0:
        return None
```

```
            else:
                answer = np.round(np.linalg.solve(matrix_coeff, free_vector), 2)
                return answer
        def check_rotation(coordinates, angle):
            rotation_matrix = np.array([[np.cos(angle), -np.sin(angle)], [np.sin(angle),
np.cos(angle)]])
            rotated_coordinates = np.dot(rotation_matrix, coordinates[:2])
            rounded_rotated_coordinates = np.round(rotated_coordinates, 2)
            return np.concatenate((rounded_rotated_coordinates, coordinates[2:]))
```