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. 2381 _____ Зазуля И.А.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2022

Цель работы.

Изучить основные управляющие конструкции языка *Python*, библиотеку *numpy* и применить полученные знания на практике.

Задание.

Задача 1. 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.

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. 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.

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. 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.

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.

Функция *check_collison*. На вход функция принимает два аргумента: коэффициенты уравнений прямых, по которым перемещаются дакиботы. Чтобы найти точку пересечения дакиботов необходимо решить систему из этих уравнений. Посредством функции *numpy.concatenate* мы объединяем входные массивы в один двумерный массив(матрицу). Следующим шагом создаём столбец со значениями свободных членов. Следом проверяем систему на наличие решений, то есть проверяем, что ранг матрицы без свободных членов и ранг расширенной матрицы равен двум. Если условие выполняется то посредством функции *solve* из пакета *numpy.linalg* находим решение системы, которое записывается в кортеж. В итоге функция вернёт этот кортеж. Если же условие не выполняется, то есть система не имеет решений, то функция возвращает *None*.

Функция *check_surface*. На вход функция принимает три аргумента: три точки для каждого дакибота. Чтобы найти коэффициенты уравнения плоскости, необходимо решить матрицу коэффициентов для каждого дакибота и вектора свободных членов. Для этого объединяем точки дакиботов в матрицу, после в получившуюся матрицу добавляем столбец свободных членов. Следующим шагом проверяем систему на наличие решений, то есть проверяем, что ранг матрицы без свободных членов и ранг расширенной матрицы равен трём. Если

условие выполняется, то посредством функции *solve* из пакета *numpy.linalg* находим решение системы, которое возвращается функцией с округлением до 2-ого знака после запятой. Если же условие не выполняется, то есть система не имеет решений, то функция возвращает *None*.

Функция *check_rotation*. На вход функция принимает два аргумента: координаты дакибота, угол его поворота. С помощью переменных обозначим координаты дакиботы по x и y. Далее с помощью формул определим новые координаты дакибота после поворота. Мы не используем координату z, так как поворачиваемся вокруг этой оси. Функция возвращает столбец координат после поворота дакибота.

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.	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.

Выводы.

Были изучены и применены на практике основные управляющие конструкции языка *Python*, а также модуль *numpy*.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: `main.py`

```
import numpy
from math import *

def check_collision(bot1, bot2):
     b = numpy.concatenate(([bot1],[bot2]), axis=0)
     c = (-1) * b[:,2]
     # Проверяем наличие решения.
     if (numpy.linalg.matrix_rank(b) == 2 and
numpy.linalg.matrix_rank(b[:,0:2]) == 2):
          result = tuple(round(x,2) for x in
numpy.linalg.solve(b[:,:2],c))
          return result
     else:
          return None


def check_surface(point1, point2, point3):
     # Create a matrix
     a = numpy.concatenate(([point1],[point2],[point3]), axis=0)
     a = numpy.insert(a, 2, 1, axis=1)
     # Checking the availability of solutions
     if (numpy.linalg.matrix_rank(a) == 3 and
numpy.linalg.matrix_rank(a[:,0:3]) == 3):
          result = numpy.array([round(x,2) for x in
numpy.linalg.solve(a[:,0:3],a[:, 3])])
          return result
     else: return None


def check_rotation(vec, rad):
     vec = numpy.array([float(x) for x in vec])
     a, b = vec[0], vec[1]
     vec[0] = a*cos(rad)-b*sin(rad)
     vec[1] = b*cos(rad)+a*sin(rad)
     return numpy.array([round(x,2) for x in vec])
```