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 Python

Студент гр. 3344 _____ Бирюков Л.И

Преподаватель _____ Иванов Д. В

Санкт-Петербург

2023

Цель работы.

Научиться использовать основные управляющие конструкции языка Python, функции из библиотеки “numpy” для решения простых задач 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 вызыватьне 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.

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.

Задача 3. Содержательная часть задачи Дакибот выехал на перекресток и готовится к выполнению 3 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 дляпродолжения движения.

Ход работы.

Импортируем библиотеку numpy как np, а также библиотеку math.

Первая функция - check_collision(bot1, bot2). Из коэффициентов двух поступивших уравнений создаём матрицу 2x2 ‘x’, содержащую коэффициенты a1,b1, a2,b2 и матрицу ‘y’ в которой хранятся коэффициенты - c1, и -c2 (так как изначально уравнение : ax+by+c=0, c мы переносим в правую сторону с противоположным знаком). Затем проверяем ранг матрицы (if np.linalg.matrix_rank(x)==2), если условие выполняется, находим координаты точки пресечения: “intersection = np.linalg.solve(x,y)”. Округляем их до 2 знаков после запятой и возвращаем: ‘’return (round(intersection[0], 2), round(intersection[1], 2))’’

Вторая функция - def check_surface(point1, point2, point3). Исходное уравнение ax + by + c = z. Создаются матрицы “y” и “x”. Матрица “y” содержит значения z, матрица “x” содержит значения x, y и 1. Функция возвращает округленное решение в виде массива. Решение находится при помощи функции “solve” из пакета “linalg”.

Третья функция - check_rotation(v, r). На вход поступают три координаты и угол. В матрицу ‘matrix’ записываем значения координат после поворота, используя для этого формулу x’ = x*cos(a) – y*sin(a) и y’ = x*sin(a) + y*cos(a). Поворот вокруг оси z, поэтому она не меняется. При генерации матрицы округляем все значения до двух знаков после запятой. Функция возвращает ‘matrix’

Тестирование.

Результаты тестирования представлены в табл. 2.

Таблица 2 – Результаты тестирования

Входные данные	**Вывод**	**Комментарий**
check_collision np.array([-3, -6, 9]), np.array([8, -7, 0])	(0.91, 1.04)	Код работает верно
def check_surface np.array([1, -6, 1]), np.array([0, -3, 2]), np.array([-3, 0, -1])	[2. 1. 5.]	Код работает верно
check_rotation np.array([1, -2, 3]), 1.57	[2. 1. 3.]	Код работает верно

Вывод.

Были изучены и применены на практике функции библиотеки numpy: np.array(), np.linalg.matrix_rank(a), np.linalg.solve(a,b), np.cos(a), np.sin(a), round(a,x), а также основные управляющие конструкции.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import numpy as np
import math
from math import sin,cos

def check_collision(bot1,bot2):
      x = np.array([[bot1[0], bot1[1]], [bot2[0],bot2[1]]])
      y = np.array([-bot1[2], -bot2[2]])
      if np.linalg.matrix_rank(x)==2:
            intersection = np.linalg.solve(x,y)
            return (round(intersection[0], 2), round(intersection[1], 2))
      else:
            return None

def check_surface(point1, point2,point3):
      x = np.array([[point1[0], point1[1], 1], [point2[0], point2[1], 1],
[point3[0], point3[1], 1]])
      y = np.array([point1[2],point2[2],point3[2]])
      if np.linalg.matrix_rank(x)==3:
            k = np.linalg.solve(x,y)
            return np.array([round(k[0],2),round (k[1],2), round(k[2],2)])
      else:
            return None

def check_rotation(v, r):
      try:
            matrix = np.array([round(v[0]*np.cos(r)-v[1]*np.sin(r),2),
            round(v[0]*np.sin(r)+v[1]*np.cos(r),2), round(v[2],2)])
            return matrix
      except:
            pass
```