МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Информатика»
Тема: Основные управляющие конструкции языка Python

Студентка гр. 2384 _____ Валеева А.А.

Преподаватель _____ Шевская Н. В.

Санкт-Петербург
2022

Цель работы.

Научиться использовать основные управляющие конструкции, а также функции библиотеки NumPy, для реализации простых задач на языке Python

Задание.

Вариант 2.

Вариант состоит из 3 задач, оформите каждую задачу в виде отдельной функции согласно условиям задач.

Задача 1. Содержательная постановка задачи

Дакибот приближается к перекрестку. Он знает 4 координаты, соответствующие координатам углов перекрестка (координаты образуют прямоугольник), и свои координаты. По правилам движения дакибот должен остановиться сразу, как только оказывается на перекрестке. Ваша задача -- помочь дакиботу понять, находится ли он на перекрестке (внутри прямоугольника).

Задача 2. Содержательная часть задачи

Несколько дакиботов прибыли на базу, но их корпуса оказались поврежденными. В логах ботов программисты нашли сведения про их траектории движения, которые задаются линейными уравнениями вида: ax+by+c=0. В логах хранятся коэффициенты этих уравнений a, b, c.

Ваша задача - вывести список номеров ботов (кортежи), которые столкнулись с друг другом (боты нумеруются с нуля, порядок следования коэффициентов уравнений соответствует порядку ботов).

Задача 3. Содержательная часть задачи

При перемещении по дакитауну дакибот должен регулярно отправлять на базу сведения, среди которых есть длина пройденного пути. Дакиботу известна последовательность своих координат (x, y), по которым он проехал. Ваша задача - помочь дакиботу посчитать длину пути.

Ход работы.

Импортируется библиотека *NumPy* как *np*.

Задание 1: Выполняется при помощи функции *check_crossroad*. На вход функции подается кортеж с переменными, содержащими координаты дакибота и координаты точек, описывающих перекресток. Делаем проверку: если дакиборт находится по оси *X* между координатой *x* первой точки и координатой *x* второй точки, а по оси *Y* между координатами *y* первой точки и четвертой точек, то в таком случае функция принимает значение *True*, а иначе *False*.

Задание 2: Выполняется при помощи функции *check_collisio.* Создаем пустой массив *result*, куда будем добавлять номера пар столкнувшихся ботов. Для решения системы применяются два цикла *for*, с помощью которых перебираются индексы элементов входящего картежа. С помощью модуля *array* создаем две матрицы — *s1* и *s2*. В матрицу *s1* мы добавляем переменные *a* и *b* квадратного уравнения (с 0 по 2 индекс, не включая 2). А в матрицу *s2* добавляем свободный член квадратного

уравнения — c, так как исходное уравнение $ax+by+c=0$ можно преобразовать в $ax+by=-c$. Далее, если возможно посчитать, используем модуль *linalg.solve* для решения. Добавляем пару в массив *result.*

Задание 3: Выполняется при помощи функции *check_path*. На вход которой поступает картеж неизвестной длины. В функции применим цикл, проходящий по всем элементам картежа (перебор по индексам), который будет высчитывать длину вектора, используя 4 переменные: координаты x, y

одной точки и следующей после нее. С помощью одного из модулей NumPy - *array* создаем две матрицы, а для нахождения длины вектора используем модуль *linalg.norm*. В переменной *way* будет подсчитываться сумма длин всех векторов. В конце с помощью функции *round* округлим значение w*ay* до двух знаков после запятой.

Обработка результатов эксперимента.

Ввод	**Вывод**	**Комментарий**
1. *check_crossroad* (5, 8) (0, 3) (12, 3) (12, 16) (0, 16)	True	верно
2. *check_collisio* [[-1 -4 0] [-7 -5 5] [1 4 2] [-5 2 2]]	[(0, 1), (0, 3), (1, 0), (1, 2), (1, 3), (2, 1), (2, 3), (3, 0), (3, 1), (3, 2)]	верно
3. *check_path* [(1.0, 2.0), (2.0, 3.0)]	1.41	верно

Выводы.

Для решения поставленной задачи были использованы полученные знания и навыки работы с основными управляющими конструкциями и библиотекой NumPy.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Valeeva_Alina_lb1

```
import numpy as np
from math import *
def check_crossroad(robot, point1, point2, point3, point4):
        if point1 [0] <= robot[0] <= point2[0]:
                if point1 [1] <= robot[1] <= point4[1]:
                        return True
                else:
                        return False
        else:
                return False
def check_collision(coefficients):
        result = []
        for i in range(len(coefficients)):
                for j in range(len(coefficients)):
                        if i != j:
                                s1 = np.array([coefficients[i][:2], coefficients[j][:2]])
                                s2 = np.array([-coefficients[i][-1], -coefficients[j][-1]])
                                try:
                                        np.linalg.solve(s1, s2)
                                        result.append((i, j))
                                except:
                                        pass
   return result
def check_path(points_list):
        way = 0
        for x in range(len(points_list)-1):
        coordinate1_1 = points_list[x][0]
        coordinate1_2 = points_list[x][1]
        coordinate2_1 = points_list[x+1][0]
```

```
        coordinate2_2 = points_list[x+1][1]
        array1 = np.array((coordinate1_1, coordinate1_2))
        array2 = np.array((coordinate2_1, coordinate2_2))
        distance = np.linalg.norm(array1 - array2)
        way += distance
result = round(way,2)
return result
```