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 _____ Кузнецов И.И

Преподаватель _____ Жангиров Т.Р

Санкт-Петербург

2022

Цель работы.

Изучить основные управляющие конструкции языка Python.

Задание.

Задача 1.

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.

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.

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.

Выполнение работы.

Подключается модуль numpy с псевдонимом np.

Функция len_way(a, b).

Функция принимает на вход два кортежа, которые содержат в себе координаты точки. Функция выводит расстояние между этими точками, считаемое с помощью теоремы Пифагора.

Функция check_crossroad(robot, point1, point2, point3, point4).

Функция принимает на вход кортежи содержащие в себе координаты работа и точек, ограничивающих перекресток. Путем проверки условия, что робот находится внутри прямоугольника (его координата по X больше минимальной но меньше максимальной X координат точек, ограничивающих прямоугольник, и координата по y больше минимальной но меньше максимальной Y координат точек, ограничивающих прямоугольник. Функция возвращает True, если условие выполняется и False в противном случае.

Функция check_collision(coefficients).

На вход функции подается матрица ndarray Nx3 (N -- количество ботов, может быть разным в разных тестах) коэффициентов уравнений траекторий coefficients. Функция возвращает список пар -- номера столкнувшихся ботов.
Двумя циклами for перебираются все пары линейных уравнений без повторений.
В matrix записываются np-массив содержащий в себе коэффициенты линейных уравнений. В b записываются свободные члены. В rang1 с помощью функции rank записывается ранг матрицы без свободных членов. С помощью функции hstack() к матрице справа добавляется столбец, содержащий в себе свободные члены. В rang2 записывается ранг матрицы содержащей свободные члены. Если rang1 == rang2, то система уравнений имеет решение, то есть две прямые пересеклись, то есть роботы столкнулись. В таком случае в массив ans добавляется такая пара.

Функция check_path(points_list).

Функция принимает на вход список кортежей, которые содержат в себе координаты точек, через которые прошел робот. Вызывая функция len_way(),

мы находим расстояние между точками через которые прошел робот и прибавляем его к общему расстоянию. С помощью функции round() округляем ответ до 2 знаков после запятой.

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.	(9, 3) (14, 13) (26, 13) (26, 23) (14, 23)	False	Результатом работы первой функции является False, то есть робот находится вне перекрестка.
2.	[[-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)]	Результатом работы второй функции является массив, содержащий в себе пары роботов, которые столкнулись.
3.	[(1.0, 2.0), (2.0, 3.0)]	1.41	Результатом работы функции является расстояние, которые прошел робот, округленное до двух знаков после запятой.
...			

Выводы.

Были изучены основные управляющие конструкции языка Python. Изучены методы библиотеки numpy. Полученные знания были использованы для решения задач основанных на практических ситуациях, происходящих в ЛАТС

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_lb1.py

```
import numpy as np


def len_way(a, b):
    delt_x = a[0] - b[0]
    delt_y = a[1] - b[1]
    return ((delt_x ** 2 + delt_y ** 2) ** (0.5))


def check_crossroad(robot, point1, point2, point3, point4):
      if (point3[0] >= robot[0] >= point1[0]) and (point4[1] >=
robot[1] >= point2[1]):
          return True
    return False


def check_collision(coefficients):
    ans = []
    for i in range(len(coefficients)):
        for j in range(i + 1, len(coefficients)):
            matrix = np.array([[coefficients[i][0], coefficients[i]
[1]], [coefficients[j][0], coefficients[j][1]]])
              b = np.array([[coefficients[i][2]], [coefficients[j]
[2]]])
            rang1 = np.linalg.matrix_rank(matrix)
            matrix = np.hstack((matrix, b))
            rang2 = np.linalg.matrix_rank(matrix)
            if rang1 == rang2:
                ans.append((i, j))
                ans.append((j, i))
    return sorted(ans)


def check_path(points_list):
    way = 0
    for i in range(len(points_list) - 1):
```

```
        way += len_way(points_list[i + 1], points_list[i])
    return round(way, 2)
```