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: Управляющие конструкции языка Python

Студент гр. 3344 Гусева Е.А.

Преподаватель Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Изучить управляющие конструкции языка Python, такие какввод/вывод, списки, циклы, словари, функции и модули, а также работу с модулем numpy.

Задание.

Вариант 2. 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. 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

Были импортирована библиотека numpy. Функция check_crossroad принимает в себя 5 значений: robot, point1, point2, point3, point4. Первый аргумент - координаты дакиробота, далее - границы перекрестка. При помощи переменных rx и ry записываются соответствующие координаты дакибота. Далее в переменных x и y сортируется список координат перекрестка. В условии сравниваются координаты дакибота с максимальными и минимальными координатами перекрестка при помощи неравенства, в зависимости от того, попадает ли дакироборот на перекресток или нет, функция возвращает True или False. Функция check_collision принимает список из уравнений coefficients. Создается список crashed, в который будут записываться столкнувшиеся дакиботы. При помощи двух циклов сравниваются траектории движения

дакиботов, их координаты записываются в переменные, далее при помощи функции Np.vstack () — это библиотечная функция numpy в Python, которая возвращает вертикальное размещение массива сздаются вертикальные массивы, которые встают рядом и образуют матрицу, с помощью функции np.linalg.matrix_rank(), в которой использован метод matrix_rank(), возвращается значение ранга матрицы, если ранг матрицы будет >= 2, это будет значить, что дакиботы столкнуться. Функция возращает список из кортежей всех столкнувшихся дакиботов. Функция check_path принимает список двумерных пар point_list. Инициализируется переменная length для определения длины пройденного пути. При помощи цикла, в length записываются разницы двух координат, которые прошел дакибот. В переменную length добавляется длина между этими координатами, которая высчитывается при помощи np.sqrt, которое возвращает неотрицательный квадратный корень из массива по элементам. Функция возращает length, округленное до 2 чисел после запятой при помощи round.

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	
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)]	
3.	[(1.0, 2.0), (2.0, 3.0)]	1.41	

Выводы

Создан код на Python, в котором применены фундаментальные языковые конструкции, а также приобретены навыки работы с библиотекой NumPy. В ходе разработки были освоены методы вычисления ранга матриц, а также использования функции округления и других функций.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла `Guseva_Elema_cs_lb1.c`

```
import numpy as np


      def check_crossroad(robot,
 point1, point2, point3, point4):
               rx, ry=robot
            x=sorted([point1[0],
point2[0], point3[0], point4[0]])
            y=sorted([point1[1],
point2[1], point3[1], point4[1]])
           if x[0]<=rx<=x[3] and
         y[0]<=ry<=y[3]:
                  return True
                   else:
                 return False


   defcheck_collision(coefficients):
              crashed = []
       num_bots = len(coefficients)

         for i in range(num_bots):
           for j in range(num_bots):

                    a1, b1, c1 =
         coefficients[i]
                    a2, b2, c2 =
         coefficients[j]
                      matrix =
  np.vstack(([a1, b1],[a2, b2]))

                          if
np.linalg.matrix_rank(matrix) >= 2:

      crashed.append((i, j))

              return crashed



     def check_path(points_list):
                length = 0
                 for i in
    range(len(points_list)-1):
                  length +=
  np.sqrt((points_list[i+1][0] -
     points_list[i][0])**2 +
      (points_list[i+1][1] -
     points_list[i][1])**2)
```

```
    return round(length, 2)
```