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

Студентка гр. 3341 _____ Кузнецова С.Е.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2023

Цель работы

Целью работы является освоение работы с управляющими конструкциями и модулем NumPy на языке Python на примере использующей их программы.

Для достижения поставленной цели требуется решить три задачи, реализовав для решения каждой функции с использованием модуля NumPy и, в частности, пакета NumPy.linalg.

Задание

Вариант 2

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).

Формальная постановка задачи

Оформите задачу как отдельную функцию: def check_crossroad(robot, point1, point2, point3, point4)

На вход функции подаются: координаты дакибота robot и координаты точек, описывающих перекресток: point1, point2, point3, point4. Точка — это кортеж из двух целых чисел (x, y).

Функция должна возвращать True, если дакибот на перекрестке, и False, если дакибот вне перекрестка.

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).

Формальная постановка задачи

Оформите решение в виде отдельной функции check_collision(). 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 (если никто из ботов не столкнулся, возвращается пустой список).

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.

Формальная постановка задачи

Оформите задачу как отдельную функцию check_path, на вход которой передается последовательность (список) двумерных точек (пар) points_list. Функция должна возвращать число -- длину пройденного дакиботом пути (выполните округление до 2 знака с помощью round(value, 2)).

Основные теоретические положения

Используемые библиотеки

NumPy — библиотека с открытым исходным кодом для языка программирования Python с поддержкой многомерных массивов (включая матрицы) и высокоуровневых математических функций, предназначенных для работы с многомерными массивами. Пакет NumPy.linalg организует функции линейной алгебры.

Math – встроенная библиотека в Python, которая предоставляет набор функций для выполнения математических, тригонометрических и логарифмических операций.

Использованные функции:

1. np.ndarray.shape – кортеж - определение размера массива.
2. np.linalg.solve – решение системы линейных уравнений.
3. math.sqrt – нахождение корня числа.

Кроме того, были использованы такие конструкции, как:

1. Функция - объект, принимающий аргументы и возвращающий значение. Обычно функция определяется с помощью инструкции def. Значение возвращается с помощью инструкции return.
2. Цикл for – в цикле указывается переменная и множество значений, по которому будет пробегать переменная. Множество значений может быть задано списком, кортежем, строкой или диапазоном.
3. Try-except – конструкция для обработки исключений. В блоке try мы выполняем инструкцию, которая может породить исключение, а в блоке except мы перехватываем их.

Выполнение работы

Импортируем модули NumPy и Math.

1 задача:

Объявляем функцию def check_crossroad(robot, point1, point2, point3, point4). Функция принимает на вход четыре кортежа со значениями – координатами точек. Кортеж robot содержит координаты дакибота, кортежи point1 – point4 содержат координаты границ перекрестка.

Для того, чтобы узнать, попадет ли робот в область перекрестка по оси x, проверим, больше или равна координата робота по оси x чем координата крайней левой точки по оси x и меньше или равна координата робота по оси x чем координата крайней правой точки по оси x.

Аналогично делаем для координаты робота по оси y, сравнивая ее с координатами верхней и нижней точек по оси y.

Возвращаем значение, получаемое при конъюнкции обоих результатов.

2 задача:

Объявляем функцию def check_collision(coefficients). Функция принимает на вход матрицу размера n*3. В каждом массиве матрицы находится три числа – коэффициенты уравнения движения каждого из n роботов. Чтобы определить, какие роботы столкнутся, используем функцию np.linalg.solve. Если функция не выдаст ошибку, то это значит, что решение уравнения есть, то есть роботы столкнутся. В этом случае в пустой массив, который будет являться ответом, запишем кортеж с номерами двух роботов, которые столкнутся. Если же функция выдаст ошибку, мы пропустим этот шаг, перейдя к следующей итерации цикла с помощью конструкции except: pass.

Возвращаем массив, содержащий кортежи с номерами столкнувшихся роботов.

3 задача:

Объявляем функцию def check_path(points_list). Функиця принимает на вход массив, состоящий из кортежей, в каждом из которых записаны координаты робота. По координатам необходимо узнать, какой путь преодолеет робот. Переберем все кортежи последовательно и в переменную ох будем записывать квадрат расстояния между двумя соседними точками по оси x, в переменную оу – по оси y. К переменной r, изначально равной нулю, будем прибавлять значение расстояния между точками на плоскости (т.е. корень из суммы квадратов изменения расстояния по оси x и y).

Возвращаем полученное значение пути r, округленное до сотых.

См. приложение А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

п/п	Входные данные	Выходные данные	Комментарии
	(9, 3) (14, 13) (26, 13) (26, 23) (14, 23)	False	№1
	(5, 8) (0, 3) (12, 3) (12, 16) (0, 16)	True	№1
	[[-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)]	№2
	[(1.0, 2.0), (2.0, 3.0)]	1.41	№3
	[(2.0, 3.0), (4.0, 5.0)]	2.83	№3

Выводы

Изучены основные управляющие конструкции языка Python и модуль NumPy, создана программа с использованием данных управляющих конструкций и модуля NumPy.

Реализована программа, состоящая из трех задач, под каждую из которых выделена отдельная функция.

1 задача: определение, находится ли дакибот на перекрестке или вне его по координатам границ перекрестка и координатам дакибота.

2 задача: определение, какие дакиботы столкнутся при движении, зная коэффициенты уравнений их движения.

3 задача: определение пути, пройденного дакиботом, зная его координаты.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_lb1.py

```
import numpy as np
import math

def check_crossroad(robot, point1, point2, point3, point4):
    ox=(robot[0]>=point1[0] and robot[0]<=point3[0])
    oy=(robot[1]>=point1[1] and robot[1]<=point3[1])
    return (ox and oy)

def check_collision(coefficients):
    ans=[]
    length=coefficients.shape[0]
    for i in range(length):
        for j in range(length):
            if i!=j:

array_first=[[coefficients[i][0],coefficients[j][0]],[coefficients[i][
1],coefficients[j][1]]]
                array_second=[coefficients[i][2],coefficients[j][2]]
                try:
                    point=np.linalg.solve(array_first,array_second)
                    ans+=[(i,j)]
                except:
                    pass
    return ans

def check_path(points_list):
    r=0
    for i in range(len(points_list)-1):
        ox=(points_list[i][0]-points_list[i+1][0])**2
        oy=(points_list[i][1]-points_list[i+1][1])**2
        r+=math.sqrt(ox+oy)
    return round(r,2)
```