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. 3344		**Сьомак Д.А.**
Преподаватель		**Иванов Д.В.**

Санкт-Петербург
2023

Цель работы

Освоение основных управляющих конструкций языка Python. Практика использования методов модуля numpy для решения алгебраических задач.

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. 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)).

Выполнение работы

Задача 1:

Используем оператор if для сравнения координат робота и крайних точек перекрёстка. point2[0] >= robot[0] >= point1[0] - проверяет, что координата x робота не выходит за координаты x крайней левой и правой точек (point1 и point2). point4[1] >= robot[1] >= point1[1] - проверяет, что координата y робота не выходит за координаты y крайней нижней и верхней точек (point1 и point4). В совокупности с оператором and такое сравнение обеспечивает точную проверку наличие дакибота на перекрёстке и возвращает True в случае его наличия, False при его отсутствии.

Задача 2:

Задаём пустой список result , который в будущем будет использован для вывода ответа. Далее используем два цикла for (один внутри другого), что позволит перебрать все существующие комбинации коэффициентов уравнений траекторий для пары дакиботов с номерами i и j. Переменные str1 и str2 хранят уравнения траекторий дакиботов в виде строк матрицы. Используем метод numpy.vstack(). Данный метод позволяет дописать матрицы последовательного друг к другу. Передаём методу кортеж (str1[0:2], str2[0:2]), который состоит из срезов строк матриц до 2 числа, чтобы сделать матрицу квадратной. Используем оператор if и метод numpy.linalg.matrix_rank(). Данный метод из пакета linalg позволяет посчитать ранг квадратной матрицы, ранг — это количество линейно-независимых строк или столбцов квадратной матрицы. Сравниваем возвращаемое методом значение с 2, так как квадратная матрица 2x2 должна иметь как минимум две линейно независимых строки или столбца, чтобы произошло столкновение дакиботов. Если матрица удовлетворяет сравнению, то в список result добавляется кортеж (i, j). Возвращаем список result

Задача 3:

Задаём переменную lenght = 0, которая в будущем будет использована для вывода ответа. Используем цикл for, который будет идти с первого до предпоследнего элемента списка, который передаётся основной функции. Переменные point1 и point2 хранят координаты текущий и последующей точки в виде картежа. Переменные x1, y1 и x2, y2 хранят координаты x и y переменных point1 и point2. Используем метод numpy.array() для создания матрицы, передаём методу список разностей координат текущей и последующей точки [x2-x1, y2-y1]. Переменная vec хранит созданную матрицу. Используем метод numpy.linalg.norm(). Данный метод из пакета linalg позволяет вычислить норму (модуль, длину) вектора (в общем случае — матрицы), переданного на вход. Значение, которое возвращает метод, прибавляется к переменной lenght каждую итерацию цикла. Округляем итоговое значение lenght (после завершения цикла for) до второго знака после запятой с помощью встроенной функции round(). Возвращаем округлённое значение lenght.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	(5, 8) (0, 3) (12, 3) (12, 16) (0, 16)	True	-
	[[-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)]	-
	[(2.0, 3.0), (4.0, 5.0)]	2.83	-

Выводы

Были освоены основные управляющие конструкции языка Python. Для решения алгебраических задач были использованы некоторые методы модуля numpy, преимущественно из пакета linalg. Была освоена работа с созданием матрицы, рангом матрицы, линейными уравнениями, вычислением длинны вектора.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Somak_Demid_lb1.py

```
import numpy as np


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


def check_collision(coefficients):
    result = []
    for i in range(len(coefficients)):
        for j in range(len(coefficients)):
            str1 = coefficients[i]
            str2 = coefficients[j]
            matr = np.vstack((str1[0:2], str2[0:2]))
            if np.linalg.matrix_rank(matr) >= 2:
                result.append((i, j))
    return result


def check_path(points_list):
    lenght = 0
    for i in range(0,len(points_list)-1):
        point1, point2 = points_list[i], points_list[i+1]
        x1, y1 = point1[0], point1[1]
        x2, y2 = point2[0], point2[1]
        vec = np.array([x2-x1, y2-y1])
        lenght += np.linalg.norm(vec)
    lenght = round(lenght, 2)
    return lenght
```