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. 3341 _____ Трофимов В.О.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Целью данной лабораторной работы является освоение работы с управляющими конструкциями языка Python, такими как условные операторы (if-else), циклы (for, while) и операторы прерывания циклов (break, continue), а также приобретение навыков работы с модулем NumPy для эффективной обработки многомерных массивов и выполнения математических операций над ними.

Задание

Вариант 2.

Задача 1.

Оформите задачу как отдельную функцию: def check_rectangle(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.

Примеры входных аргументов и результатов работы функции:

1. Входные аргументы: (9, 3) (14, 13) (26, 13) (26, 23) (14, 23)

Резлультат: False

2. Входные аргументы: (5, 8) (0, 3) (12, 3) (12, 16) (0, 16)

Результат: True

Задача 2.

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).

Пример входного аргумента ndarray 4x3 :

```
[[-1 -4  0]
 [-7 -5  5]
 [ 1  4  2]
 [-5  2  2]]
```

Пример выходных данных:

[(0, 1), (0, 3), (1, 0), (1, 2), (1, 3), (2, 1), (2, 3), (3, 0), (3, 1), (3, 2)]

Первая пара в этом списке (0, 1) означает, что столкнулись 0-й и 1-й боты (то есть их траектории имеют общую точку).

В списке отсутствует пара (0, 2), можно сделать вывод, это боты 0-й и 2-й не сталкивались (их траектории НЕ имеют общей точки).

Примечание: помните про ранг матрицы и как от него зависит существование решения системы уравнений. В случае, если ни одного решение не было найдено (например, из-за линейно зависимых векторов), функция должна вернуть пустой список [].

Задача 3.

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)).

Пример входных данных:

[(1.0, 2.0), (2.0, 3.0)]

Пример выходных данных:

1.41

Пример входных данных:

[(2.0, 3.0), (4.0, 5.0)]

Пример выходных данных:

2.83

Основные теоретические положения

Для решения задачи понадобилось следующее:

1) Импортировать библиотеку NumPy (import numpy as np). В программе были использованы функции этой библиотеки: np.array() – функция для создания массива (одномерного, двумерного, трёхмерного), в которую передаётся ,например, список или кортеж элементов; np linalg.matrix_rank() – функция для вычисления ранга матрицы.

2) Генераторы списка. В программе использованы генераторы с циклом for, генераторы с условием.

3) Встроенная функция round() – функция для округления числа с плавающей точкой до той цифры, которую задает пользователь

Выполнение работы

Определяются следующие функции для решения каждой из задач:

1. def check_crossroad(robot, point1, point2, point3, point4):

2. def check_collision(coefficients):

3. def check_path(points_list):

Для решения задачи №1 была определена функция def check_rectangle(robot, point1, point2, point3, point4).

Объявляется переменная rectangle_length с помощью генератора цикла for. В переменной хранятся значения длины прямоугольника в диапазоне от point1[0] до point2[0] + 1(‘+1’ для включения значения point2[0]).Происходит обращение к кортежу значений точки по индексу(где 0 – x, 1- y точки).

Объявляется переменная rectangle_width с помощью генератора цикла for. В переменной хранятся значения ширины прямоугольника в диапазоне от point1[1] до point4[1] + 1(‘+1’ для включения значения point4[1]).Происходит обращение к кортежу значений точки по индексу(где 0 – x, 1- y точки).

Дальше, если robot[0](координата x робота) in rectangle_length и robot[1](координата y робота) in rectangle_width возвращает значение True , иначе False. То есть происходит проверка на совпадение координат робота(x, y)с координатами перекрёстка, в случае совпадения вернётся значение True, в ином другом случае False.

Для решения задачи №2 была определена функция def check_collision(coefficients):

Объявляется переменная matrix, к которой присваивается пустой массив ([]), дальше переменная будет хранить в себе значения вектора для каждого робота.

Объявляется переменная result, к которой присваивается пустой массив ([]), дальше переменная будет хранить в себе результат.

Нужно подобрать две любые точки для каждой из функции вида ax + by + c = 0(равносильная запись y = -ax/b -c/b). Объявляются переменные x1, x2 и им произвольно присвоены значения 0,1. Нужно получить для каждого x значение

y и после из полученных точек составить координаты вектора vector = [x2-x1, y2-y1]. В matrix добавляется значение переменной vector с помощью функции append(). Создаётся матрица 2x2 для каждой пары векторов с помощью двойного цикла for в диапазоне до len(matrix), поэтому объявляется переменная matrix_vectors, к которой присваивается значение функции np.array([matrix[j],matrix[k]]), где matrix[j],matrix[k] хранят в себе вектор из переменной matrix. Дальше, если функция np.linalg.matrix_rank(matrix_vectors) не вернёт 1, то добавляется к переменной result кортеж из номеров роботов(j,k), иначе функция возвращает ранг матрицы равный 1 , а это означает, что траектории роботов не пересекутся, поэтому возвращаем переменную result без изменений.

Для решения задачи №3 была определена функция def check_path(points_list):

Определяется длина пути по формуле, как длина вектора в линейной алгебре. Начальная длина пути равняется 0, поэтому объявляется переменная d, к которой присваивается значение 0. Расчёт пути происходит, так что берутся точки(xi,yi) и (xi+1,yi+1), для этого нужно преобразовать кортеж входных данных в список с помощью генератора условия. Цикл for используется для прохода по индексам списка. Объявляются переменные x1,y1 и x2,y2, к которым присвоены значения координаты точки i и координаты точки i + 1.После к переменной d прибавляется и присваивается значение длины вектора. В итоге, переменная d будет хранить в себе значение равное длине пути, пройденного роботом. По условию, нужно округлить путь до 2 знаков после запятой, применяется при возвращении переменной d функция round(d,2)

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	(4, 11) (7, 13) (19, 13) (19, 25) (7, 25) [[-8 8 2] [-5 -6 6] [-8 -2 10] [-5 -8 5] [6 -3 2] [-2 4 1] [9 3 8]] [(1.0, 2.0), (2.0, 3.0)]	False [(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (1, 0), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 0), (2, 1), (2, 3), (2, 4), (2, 5), (2, 6), (3, 0), (3, 1), (3, 2), (3, 4), (3, 5), (3, 6), (4, 0), (4, 1), (4, 2), (4, 3), (4, 5), (4, 6), (5, 0), (5, 1), (5, 2), (5, 3), (5, 4), (5, 6), (6, 0), (6, 1), (6, 2), (6, 3), (6, 4), (6, 5)] 1.41	
2.	(8, 0) (6, 11) (21, 11) (21, 22) (6, 22) [[-6 5 0] [4 9 7] [2 3 7] [-2 -10 7] [-2 8 2]] [(2.0, 3.0), (4.0, 5.0)]	False [(0, 1), (0, 2), (0, 3), (0, 4), (1, 0), (1, 2), (1, 3), (1, 4), (2, 0), (2, 1), (2, 3), (2, 4), (3, 0), (3, 1), (3, 2), (3, 4), (4, 0), (4, 1), (4, 2), (4, 3)] 2.83	

Выводы

По достижении цели работы можно сделать следующие выводы:

1. Освоение работы с модулем NumPy позволяет эффективно оперировать многомерными массивами и проводить различные математические операции над ними. Это особенно полезно при работе с большими объёмами данных.

2. Управляющие конструкции на языке Python, такие как условные операторы if-else, позволяют программе принимать решения на основе заданных условий. Это позволяет написать более гибкий и управляемый код.

3. Циклы for и while позволяют повторять определённые действия несколько раз. Цикл for особенно полезен для работы с элементами массивов и коллекций данных. Цикл while позволяет выполнять действия до тех пор, пока заданное условие остаётся истинным.

В целом, освоение работы с модулем NumPy и основными управляющими конструкциями на языке Python позволяет существенно расширить возможности программирования и упростить обработку и анализ данных.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab1_cs.py

```
import numpy as np

def check_crossroad(robot, point1, point2, point3, point4):
    rectangle_length = [i for i in range(point1[0], point2[0] + 1)]
    rectangle_width = [i for i in range(point1[1], point4[1] + 1)]
    if robot[0] in rectangle_length and robot[1] in rectangle_width:
        return True
    else:
        return False

def check_collision(coefficients):
    matrix = []
    result = []
    x1,x2 = 0,1
    for i in range(len(coefficients)):
        y1     =     -coefficients[i][0]*x1/coefficients[i][1]     -
coefficients[i][2]/coefficients[i][1]
        y2     =     -coefficients[i][0]*x2/coefficients[i][1]     -
coefficients[i][2]/coefficients[i][1]
        vector = [x2-x1, y2-y1]
        matrix.append(vector)
    for j in range(len(matrix)):
        for k in range(len(matrix)):
            matrix_vectors = np.array([matrix[j], matrix[k]])
            if np.linalg.matrix_rank(matrix_vectors) != 1:
                result.append(tuple([j,k]))
    return result

def check_path(points_list):
    d = 0
    list_points_list = [list(arr) for arr in points_list]
    for i in range(len(list_points_list) - 1):
        x1,y1 = list_points_list[i][0], list_points_list[i][1]
        x2,y2  =  list_points_list[i  +  1][0],  list_points_list[i  +
1][1]
        d += ((x2 - x1)**2 + (y2 - y1)**2) ** 0.5
    return round(d,2)
```