**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. 3341 |  | Бойцов В.А. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Целью работы является освоение управляющих конструкций на языке Python, а также модуля NumPy на примере программы, в которой они применяются.

## 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

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

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

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

В лабораторной работе была применена библиотека NumPy, используемая для разнообразных математических вычислений.

Методы модуля NumPy:

1. *numpy.radians(array)*Данных метод позволяет перевести последовательность значений углов (в геометрическом смысле) *array* (типа *ndarray*) из градусов в радианы. Возвращает новый *ndarray* со значениями тех же углов в радианах.
2. *numpy.ones(shape))*Данный метод позволяет создать матрицу из единиц заданного размера. Размер задается с помощью кортежа *shape*, где через запятую передаются размеры матрицы.
3. *numpy.vstack(arr1,arr2,[arrN])*Данный метод позволяет дописать матрицы последовательного друг к другу.
4. *numpy.linalg.norm(vector)*Данный метод из пакета *linalg* модуля *numpy* позволяет вычислить норму (модуль, длину) вектора *vector*(в общем случае — матрицы), переданного на вход.
5. *numpy.linalg.matrix\_rank(matrix)*Данный метод из пакета *linalg* позволяет посчитать ранг квадратной матрицы matrix.
6. *numpy.linalg.solve(A,v)*Данный метод из пакета *linalg* позволяет найти решение линейной системы уравнений, которая представлена матрицей коэффициентов *A* и вектором свободных членов *v*.

## Выполнение работы

Подключается модуль NumPy: *import numpy as np.*

Далее каждая из 3 подзадач оформлена в виде отдельной функции.

Задача 1. Функция *def check\_crossroad(robot, point1, point2, point3, point4)* получает на вход координаты дакибота robot и координаты точек, описывающих перекрёсток (*point1, point2, point3, point4*), а возвращает *True* или *False*, если дакибот на перекрестке или вне него соответственно. В связи с тем, что расположение точек заведомо известно, для решения задачи достаточно сравнить координаты робота и *point1* и *point3*: *if (point3[0]>=robot[0] and point1[0]<=robot[0] and point3[1]>= robot[1] and point1[1]<=robot[1]).* Если условие выполняется, то функция возвращает *True*, а если нет – то *False*.

Задача 2. Функция *def check\_collision(coefficients)* принимает на вход матрицу (*ndarray*) *coefficients* коэффициентов уравнений траекторий ботов, а возвращает список пар номеров столкнувшихся ботов *collisions*.   
Создаётся пустой список *collisions*, а также двумерный список *matrix=[[y[x] for x in range(2)] for y in coefficients]*, в котором находятся коэффициенты уравнений траекторий, заполненные из *coefficients* при помощи генераторов списков. Далее в двух вложенных циклах for, пробегающих с помощью переменных-итераторов x и y элементы матрицы *coefficients* (индексы элементов получаются с помощью *coefficients.shape[0]*)*,* вычисляется ранг матрицы, состоящей из элементов матрицы *coefficients* c индексами *x* и *y* (дополнительно проверяется, что x не равен y), и если этот ранг равен 2 (необходимое и достаточное условие существования решений), то в *collisions* добавляется пара *(x, y).*

Задача 3. Функция *check\_path* принимает на вход список *points\_list* двумерных точек, а возвращает вещественное число – длину пройденного дакиботом пути. Инициализируется переменная *path\_len=0* – в ней будет храниться длина пройденного пути. Далее создаётся переменная *points\_num*, которой присваивается значение *len(points\_list)* – количество переданных функции точек. Также переменная *points\_list* преобразовывается к *ndarray.* Далее в цикле *for x in range(1, points\_num)* в переменную *vector* записывается разность *x*-го и (*x-1*)-го элемента *points\_list* (*vector=points\_list[x]-points\_list[x-1]),* что задаёт координаты вектора из точки *points\_list[x-1]* в *points\_list[x]*, после чего к значению переменной *path\_len* прибавляется длина этого вектора: *path\_len+=np.linalg.norm(vector).* Функция возвращает значение переменной *path\_len,* округленное до 2 знаков после запятой с помощью функции *round(path\_len, 2).*

## 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 | Задача 1 |
|  | (5, 8) (0, 3) (12, 3) (12, 16) (0, 16) | True | Задача 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)] | Задача 2 |
|  | [(1.0, 2.0), (2.0, 3.0)] | 1.41 | Задача 3 |
|  | [(2.0, 3.0), (4.0, 5.0)] | 2.83 | Задача 3 |

## Выводы

В результате работы были освоены основные управляющие конструкции языка Python, а так же получены практические навыки использования модуля NumPy.

Были разработаны 3 функции, каждая из которых решает свою поставленную задачу. В функциях применялись пакеты модуля NumPy, что значительно облегчило решение поставленных задач.

# Приложение А Исходный код программы

Название файла: main.py

import numpy as np

def check\_crossroad(robot, point1, point2, point3, point4):

if (point3[0]>=robot[0] and point1[0]<=robot[0] and point3[1]>= robot[1] and point1[1]<=robot[1]):

return(True)

else:

return(False)

def check\_collision(coefficients):

collisions=[]

matrix=[[y[x] for x in range(2)] for y in coefficients]

for x in range(coefficients.shape[0]):

for y in range(coefficients.shape[0]):

if(x!=y and np.linalg.matrix\_rank(np.array([matrix[x], matrix[y]]))==2):

collisions.append((x, y))

return(collisions)

def check\_path(points\_list):

path\_len=0

points\_num=len(points\_list)

points\_list=np.array(points\_list)

for x in range(1, points\_num):

vector=points\_list[x]-points\_list[x-1]

path\_len+=np.linalg.norm(vector)

return(round(path\_len, 2))