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. 2384 ___ Кузьминых Е.М.

Преподаватель ___ Шевская Н.В.

Санкт-Петербург

2022

Цель работы.

Научиться использовать основные управляющие конструкции языка программирования Python, а также возможности библиотеки NumPy для написания простых программ.

Задачи.

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).

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).

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.

Выполнение работы.

Задача 1 – Выполняется функцией **check_crossroad.** На вход эта функция принимает 5 кортежей, которые содержат в себе координаты расположения дакибота и крайних точек перекрестка. Для того чтобы проверить, находится ли дакибот внутри перекрестка, нужно проверить, чтобы его значение по ***y*** было больше**,** чем у точки **point2**, но меньше, чем у **point3**. Аналогично со значением по ***x,*** оно должно быть больше, чем у точки **point1**, но меньше, чем у **point2**. Если все эти условия будут выполнены, то функция возвращает **True**, иначе **False**.

Задача 2 – Выполняется функцией **check_collisio**. На вход функция принимает матрицу **ndarray**, которая содержит в себе коэффициенты линейных уравнений. Для решения мы создаем матрицу matrix, и удаляем коэффициент с из каждой матрицы с помощью функции **delete()** из библиотеки **NumPy** (точнее, удаляем последний столбец матрицы). В переменные **rows** и **_** мы сохраняем количество столбцов и строк в матрице, а после двумя циклами проходим по каждому уравнению и каждой его паре. Мы с помощью функции **vstack()** мы соединяем строки двух уравнений в 1 матрицу, после чего проверяем ранг этой матрицы. Если он равен 2, то прямые, заданные этими уравнениями, являются линейно независимыми, а также пересекаются, и мы добавляем номера подходящих пар в массив **answers**, который создали в начале. В конце функции возвращаем массив подходящих пар.

Задача 3 – Выполняется функцией **check_path**. Принимает на вход список из двухмерных координат точек. В начале создадим переменную **answer**, равную 0. После с помощью цикла **for** проходим по каждой i-ой и i+1

точке, и с помощью теоремы Пифагора считаем расстояние между координатами(вычитаем из x2 x1, а из y2 y1 для получения длин), суммируя все значения в answer. В конце функция возвращает **answer**, округленный с помощью round().

Тестирование.

№	Входные данные	Выходные данные	Комментарий
1	1.*Check_crossroad* (5, 8) (3, 3) (9, 3) (9, 16) (3, 16)	True	Ответ верный
2	2.*Check_collisio* [[-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)]	Ответ верный
3	3.Check_path [(1.0,2.0),(2.0,3.0),(3.0,4.0)]	2.83	Ответ верный

Вывод.

В ходе выполнения лабораторной работы были изучены управляющие конструкции языка Python, использована библиотека NumPy, а также решены задачи с учетом разбиения кода на функции.

Приложение 1. Исходный код программы.

```
import numpy as np
import math
from numpy import linalg

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


def check_collision(coefficients):
    answers = []
    rows,_=coefficients.shape
    matrix = np.delete(coefficients,2,1)
    for i in range(0,rows):
        for j in range(0,rows):
            if
np.linalg.matrix_rank(np.vstack([matrix[i],matrix[j]]))==2:
                answers.append((i,j))
    return answers


def check_path(points_list):
    answer=0
    for i in range(len(points_list)-1):
        first=points_list[i]
        second = points_list[i+1]
        answer+=math.sqrt(abs(first[0]-second[0])**2+abs(first[1]-
second[1])**2)
    return round(answer,2)
```