

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт**  информационных технологий | **Кафедра**  информационных систем |

**Отчет по лабораторной работе № 3**

по дисциплине «**Веб-программирование**»

|  |  |  |
| --- | --- | --- |
| **Студентка** группа ИДБ–19–05 |  | **Ле Тхи Хань Хуен** |
|  | подпись |  |
| Преподователь |  | **Кайшев Д.А.** |
|  | подпись |  |

Москва 2021 г.

**ЗАДАНИЕ 1**

Написать класс Sphere для представления сферы в трехмерном пространстве.

* конструктор, принимающий 4 действительных числа: радиус, и 3 координаты центра шара. Если конструктор вызывается без аргументов, создать объект сферы с единичным радиусом и центром в начале координат.
* метод get\_volume (), который возвращает действительное число — объем шара, ограниченной текущей сферой.
* метод get\_square\_(), который возвращает действительное число — площадь внешней поверхности сферы.
* метод get\_radius\_(), который возвращает действительное число — радиус сферы.
* метод get\_center\_(), который возвращает тьюпл с 3 действительными числами — координатами центра сферы в том же порядке, в каком они задаются в конструкторе.
* метод set\_radius\_(r), который принимает 1 аргумент — действительное число, и меняет радиус текущей сферы, ничего не возвращая.
* метод set\_center (x, y, z), который принимает 3 аргумента — действительных числа, и меняет координаты центра сферы, ничего не возвращая. Координаты задаются в том же порядке, что и в конструкторе.
* метод is\_point\_inside (x, y, z), который принимает 3 аргумента — действительных числа — координаты некоторой точки в пространстве (в том же порядке, что и в конструкторе), и возвращает логическое значение True или False в зависимости от того, находится эта точка внутри сферы.

**Код:**

import math;

class Sphere:

    def \_\_init\_\_(self,r, x,y,z):

        self.r = r

        self.x = x

        self.y = y

        self.z = z

    def get\_volumn(self):

        v=(4/3)\* (math.pi) \* math.pow(self.r,3)

        return round(v,3)

    def get\_square(self):

        s= 4\*(math.pi)\*math.pow(self.r,2)

        return round(s,3)

    def get\_radius(self):

        return round(self.r,3)

    def get\_center(self):

        return round(self.x,3),round(self.y,3), round(self.z,3)

    def set\_radius(self,rIn):

        self.r=rIn

    def set\_center(self, xIn,yIn,zIn):

        self.x=xIn

        self.y=yIn

        self.z=zIn

    def is\_point\_inside(self, xIn, yIn,zIn):

        distance = math.sqrt(math.pow((xIn-self.x),2)+math.pow((yIn-self.y),2)+math.pow((zIn-self.z),2))

        if distance>self.r:

            return False

        else:

            return True

ob=Sphere(4,5,3,2)

print ('->Volumn of sphere : '+str(ob.get\_volumn()))

print('->Square : ' + str(ob.get\_square()))

print('->radius of sphere : '+str(ob.get\_radius()))

xOut,yOut,zOut = ob.get\_center()

print('->\nX = '+str(xOut)+'\nY = '+str(yOut)+'\nZ = '+str(zOut))

ob.set\_radius(14.6)

print('->Radius after set : '+str(ob.get\_radius()))

ob.set\_center(10,11,12)

xOut,yOut,zOut = ob.get\_center()

print('->After set center\nX = '+str(xOut)+'\nY = '+str(yOut)+'\nZ = '+str(zOut))

xTest = 9

yTest = 11

zTest = 7

print('->Is point ('+str(xTest)+','+str(yTest)+','+str(zTest) +') inside sphere : '+str(ob.is\_point\_inside(xTest,yTest,zTest)))

xTest = 40

yTest = 28

zTest = -6

print('->Is point ('+str(xTest)+','+str(yTest)+','+str(zTest) +') inside sphere : '+str(ob.is\_point\_inside(xTest,yTest,zTest)))

**Результат:**

**Изображение выглядит как текст

Автоматически созданное описание**

**ЗАДАНИЕ 2**

Написать класс Matrix, который будет прообразом математического объекта(для простоты квадратная матрица). Определить для него методы сравнения на основе детерминанта этой матрицы - если детерминант первой матрицы больше, то и матрица больше. Для остальных операций аналогично. Так же реализовать операции сложения, умножения двух матриц.

**Код:**

import copy

class Calculate\_Matrix:

    def \_\_init\_\_(self,m1In,m2In):

        self.m1 = copy.deepcopy(m1In)

        self.m2 = copy.deepcopy(m2In)

        sameLevel = self.check\_matrixs\_same\_level()

        if sameLevel == True:

            sum = self.find\_sum()

            print('Sum 2 matrixs : '+str(sum))

            product = self.find\_product()

            print('Product 2 matrixs : '+ str(product))

        else:

            print('matrix m1, m2 have to be same level')

            exit()

    def find\_sum(self):

        size = len(self.m1)

        result=[[0]\*size, [0]\*size]

        for i in range(0,size):

            for j in range (0,size):

                result[i][j] = self.m1[i][j] + self.m2[i][j]

        return result

    def find\_product(self):

        size = len(self.m1)

        result=[[0]\*size,[0]\*size]

        for i in range(0,size):

            for k in range (0,size):

                result[i][k] =0

                for j in range(0,size):

                    result[i][k] = result[i][k] + self.m1[i][j]\*self.m2[j][k]

        return result

    def check\_matrixs\_same\_level(self):

        m1Width = len(self.m1)

        m2Width = len(self.m2)

        if m1Width == m2Width:

            return True

        else:

            return False

class Matrix:

    def \_\_init\_\_(self, matrixIn):

        self.matrix = matrixIn[:]

        squareMatrix = self.check\_square\_matrix()

        if squareMatrix == False:

            print('Matrix have to be square')

            exit()

    def get\_matrix(self):

        matrixOut = copy.deepcopy(self.matrix)

        return matrixOut

    def find\_det(self):

        matrixNew = copy.deepcopy(self.matrix)

        det=1

        counter=0

        size = len(matrixNew)

        cache=[None]\*size

        cache2=[None]\*size

        kt=0

        for i in range(0,size-1):

            if matrixNew[i][i] == 0:

                kt=0

                for j in range(0, size):

                    if matrixNew[i][j] !=0:

                        for k in range(0,size):

                            cache[k] = matrixNew[k][i]

                            matrixNew[k][i] = matrixNew[k][j]

                            matrixNew[k][j] = cache[k]

                        counter+=1

                        kt+=1

                        break

                if kt==0: return 0

            cache2[i]=matrixNew[i][i]

            for j in range(0,size):

                matrixNew[i][j] = matrixNew[i][j]/cache2[i]

            for j in range(i+1, size):

                h = matrixNew[j][i]

                for k in range(0, size):

                    matrixNew[j][k] = matrixNew[j][k] - h\*matrixNew[i][k]

        #cache2[1]=0

        cache2[size-1]=matrixNew[size-1][size-1]

        for i in range(0, size):

            det = det\*cache2[i]

        if counter%2 == 0:

            return det

        else:

            return -det

    def check\_square\_matrix(self):

        if len(self.matrix) == len(self.matrix[0]):

            return True

        else:

            return False

m1=Matrix([[1,1],[2,2]])

matrix1 = m1.get\_matrix()

print('Matrix m1 : '+str(matrix1))

m2=Matrix([[1,2],[3,4]])

matrix2 = m2.get\_matrix()

print('matrix m2 :' + str(matrix2))

print ('det m1 : '+str(m1.find\_det()))

print ('det m2 : '+str(m2.find\_det()))

if(m1.find\_det()>m2.find\_det()):

    print("m1 > m2")

elif(m1.find\_det()<m2.find\_det()):

    print("m1 < m2")

else:

    print("m1 = m2")

calMatrix = Calculate\_Matrix(matrix1, matrix2)

**Результат:**

Изображение выглядит как текст

Автоматически созданное описание

**ЗАДАНИЕ 3**

Написать класс, который будет является http клиентом. Что подразумевается под клиентом:

1. Используется библиотека requests
2. при инициализации класса создается сессия (https://docs.python requests.org/en/master/user/advanced/, <https://pythonru.com/biblioteki/prodvinutoe-rukovodstvo-po-biblioteke-python-requests>), и передается host для запросов
3. в деструкторе класса, созданная сессия закрывается (метод у сессии close)
4. задаются заголовки для сессии при помощи метода класса, путем изменения поля объекта класса
5. поддерживаются методы get, post - в данные методы передаются путь и query\_params. Внутри этих методов происходит отправка запросов и обработка статусов(с использованием сессии). Если статус отличный от 200 методы возвращают None, если статус 200, возвращается тело ответа

**Код:**

import requests

class Client:

    def \_\_init\_\_(self, URL):

        self.sessionCreated = requests.Session()

        response = self.sessionCreated.get(URL)

    def set\_header(self,key,value):

        self.sessionCreated.headers.update({key:value})

    def get\_response(self, URL):

        response = self.sessionCreated.get(URL)

        print("response status : "+str(response.status\_code))

        if(response is not None):

            print('ok')

        if response.status\_code == 200:

            return response.text

        else:

            return None

    def \_\_del\_\_(self):

        self.sessionCreated.close()

client = Client('https://httpbin.org')

client.set\_header('user-agent','test\_client')

response = client.get\_response('https://httpbin.org')

print(response)

**Результат:**

**Изображение выглядит как текст

Автоматически созданное описание**

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание