

**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**»

на тему: Классы на языке Python

|  |  |  |
| --- | --- | --- |
| **Студент** группа ИДБ–19–07 |  | **Клишина А. В.** |
|  | подпись |  |
| **Руководитель**  старший преподаватель |  | **Адамова Ю. С.** |
|  | подпись |  |

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc86752811)

[ОСНОВНАЯ ЧАСТЬ 4](#_Toc86752812)

[Задача №1 4](#_Toc86752813)

[Задача №2 6](#_Toc86752814)

[ЗАКЛЮЧЕНИЕ 9](#_Toc86752815)

# **ВВЕДЕНИЕ**

При выполнении данной лабораторной работы необходимо реализовать классы на языке программирования Python.

# **ОСНОВНАЯ ЧАСТЬ**

## **Задача №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 программы:

class Sphere(object):

p=3.14159265359

def \_\_init\_\_(self, radius=1, x=0, y=0, z=0):

self.radius=radius; self.x=x; self.y=y; self.z=z

def get\_radius(self):

return self.radius

def get\_volume(self):

return(4.0/3.0\*(self.p\*(self.radius\*\*3)))

def get\_square(self):

return(4\*(self.p\*(self.radius\*\*2)))

def get\_center(self):

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

def set\_radius(self, r):

self.radius=r

def set\_center(self,x1,y1,z1):

self.x=x1; self.y=y1; self.z=z1

def is\_point\_inside(self,x,y,z):

if ((self.x-x)\*\*2+(self.y-y)\*\*2+(self.z-z)\*\*2)<=self.radius\*\*2:

return True

else:

return False

Тестовые данные:

s0 = Sphere (0.5) # test sphere creation with radius and default center

print (s0.get\_center ())

print (s0.get\_volume ())

print (s0.is\_point\_inside (0 , -1.5, 0))

s0.set\_radius (1.6)

print (s0.is\_point\_inside (0, -1.5, 0))

print (s0.get\_radius ())

На рисунке 1 продемонстрирована работа программы, используя тестовые данные.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 1. Работа программы с тестовыми данными

## **Задача №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 программы:

class Matrix:

def \_\_init\_\_(self, \*mater):

if len(mater) == 0:

mater = [[1, 2],

[3, 4]]

self.mater = list(mater)

def \_\_getitem\_\_(self, item):

if isinstance(item, int):

return self.mater[item]

else:

i, j = item

return self.mater[i][j]

def \_\_setitem\_\_(self, key, value):

if isinstance(key, int):

if not isinstance(value, int):

self.mater[key] = value

elif isinstance(value, int) and not isinstance(key, int):

i, j = key

self.mater[i][j] = value

def det(self):

mater = self.mater

return mater[0][0] \* mater[1][1] - mater[0][1] \* mater[1][0]

def \_\_eq\_\_(self, other):

return self.det() == other.det()

def \_\_ne\_\_(self, other):

return self.det() != other.det()

def \_\_lt\_\_(self, other):

return self.det() < other.det()

def \_\_gt\_\_(self, other):

return self.det() > other.det()

def \_\_le\_\_(self, other):

return self.det() <= other.det()

def \_\_ge\_\_(self, other):

return self.det() >= other.det()

def \_\_neg\_\_(self):

mater = self.copy()

for row in range(len(mater)):

for it in range(len(mater[row])):

mater[row, it] \*= -1

return mater

def \_\_str\_\_(self):

mater = ''

for row in self.mater:

mater += str(row) + '\n'

return mater

def \_\_len\_\_(self):

return len(self.mater)

def copy(self):

mater = []

for item in self.mater:

mater.append(item.copy())

return Matrix(\*mater)

def \_\_add\_\_(self, other):

mater = self.copy()

for row in range(len(self)):

for it in range(len(self[row])):

mater[row, it] += other[row][it]

return mater

def \_\_sub\_\_(self, other):

mater = self.copy()

for row in range(len(self)):

for it in range(len(self[row])):

mater[row, it] -= other[row][it]

return mater

def \_\_mul\_\_(self, other):

mater = []

for i in range(len(self)):

mater.append([])

for j in range(len(other[i])):

mater[i].append(0)

mater = Matrix(\*mater)

for mainRow in range(len(mater)):

for mainIndex in range(len(mater[mainRow])):

for index in range(len(mater[mainRow])):

mater.mater[mainRow][mainIndex] += (self[mainRow][index] \* other[index][mainIndex])

return mater

Комментарии к коду:

getitem - позволяет получить элемент с помощью квадратных скобок (print(m[0,0]))

setitem - позволяет изменить элемент с помощью квадратных скобок(m[0,0] = 5)

neg - переворачивает знак элементов матрицы (+ на - и наоборот)

str – позволяет переменные Matrix вставлять в print

len - количество строк в матрице

add - возвращает сумму матриц(print(m+m2))

sub- возвращает разницу матриц(print(m-m2))

mul- возвращает произведение матриц(print(m\*m2))

Тестовые данные:

m = Matrix([1, 2],

[3, 4])

m2 = Matrix([1, 2],

[5, 3])

print(-m)

print(m \* m2)

print(m2 \* m)

print(m + m2)

print(m - m2)

print(f"{m.det()} > {m2.det()} is {m > m2}")

На рисунке 2 продемонстрирована работа программы, используя тестовые данные.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 2. Работа программы с тестовыми данными

# **ЗАКЛЮЧЕНИЕ**

Выполняя данную лабораторную работу, были написаны классы на языке программирования Python.

Результат лабораторной работы выгружен на Github (<https://github.com/AV-Klishina/LAB_3-web> ).