Видео-разбор подвига (решение смотреть только после своей попытки): <https://youtu.be/ArF90ldgm70>

**Подвиг 9 (на повторение).** Необходимо объявить функцию-декоратор class\_log для класса, которая бы создавала логирование вызовов методов класса. Например следующие строчки программы:

vector\_log = []

@class\_log(vector\_log)

class Vector:

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

self.\_\_coords = list(args)

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

return self.\_\_coords[item]

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

self.\_\_coords[key] = value

декорируют класс Vector и в список vector\_log добавляются имена методов, которые были вызваны при использовании этого класса. В частности, после выполнения команд:

v = Vector(1, 2, 3)

v[0] = 10

в списке vector\_log должны быть два метода:

['\_\_init\_\_', '\_\_setitem\_\_']

Ваша задача реализовать декоратор с именем class\_log.

**Напоминание.** Ранее вы уже создавали функцию-декоратор для класса следующим образом:

def integer\_params(cls):

methods = {k: v for k, v in cls.\_\_dict\_\_.items() if callable(v)}

for k, v in methods.items():

setattr(cls, k, integer\_params\_decorated(v))

return cls

Используйте этот принцип для успешного прохождения подвига.

P.S. В программе нужно объявить только класс и необходимые функции. На экран выводить ничего не нужно.

# здесь объявляйте декоратор и все что с ним связано

vector\_log = [] # наименование (vector\_log) в программе не менять!

@class\_log(vector\_log)

class Vector:

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

self.\_\_coords = list(args)

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

return self.\_\_coords[item]

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

self.\_\_coords[key] = value