Домашнее задание.

Маратұлы Темирболат

**Замечания по выполнению домашнего задания.**

1. **Вверху этого документа напишите свою Фамилию и Имя.**
2. Исходные коды Ваших программ сохраняйте прямо здесь, в тексте задания, строго под текстом каждого задания.
3. В дополнение к исходным кодам вставляйте в документ скриншот результата выполнения программ.
4. Не надо документ архивировать и вкладывать скриншоты в архив. Скриншоты сразу вставляете в документ, как картинку.
5. **Скриншоты удобно делать программой https://app.prntscr.com/ru/privacy.html**
6. Переменные начинаются с маленькими буквами, слова разделяются символом «\_» нижняя черта.
7. Константы пишутся заглавными буквами, слова разделяются символом «\_» нижняя черта.
8. **Не забывайте КРАСИВО оформлять интерфейс программ, за плохой, ЛЕНИВЫЙ интерфейс я тоже СНИЖАЮ ОЦЕНКУ!**

Создайте класс Circle (окружность). Для данного класса реализуйте ряд перегруженных операторов:

■ Проверка на равенство радиусов двух окружностей (операция = =);

■ Сравнения длин двух окружностей (операции >, <, <=,>=);

■ Пропорциональное изменение размеров окружности, путем изменения ее радиуса (операции + - += -=).

При выводе данных на экран используйте декораторы.

import math

ONE\_STEP = 1

class Circle:

    \_\_LENGTH\_TERMINAL = 2

    def \_\_radius\_validation(self,radius):

        if(radius < 0):

            raise ValueError("Радиус не может быть отрицательным!")

        self.\_\_radius = radius

    def \_\_init\_\_(self, radius):

        self.\_\_radius\_validation(radius)

    def get\_radius(self):

        return self.\_\_radius

    def set\_radius(self,radius):

        self.\_\_radius\_validation(radius)

    def \_\_str\_\_(self):

        return 'Радиус окружности: {} см'.format(self.\_\_radius)

    def get\_circle\_length(self):

        return self.\_\_LENGTH\_TERMINAL \* math.pi \* self.\_\_radius

    def \_\_add\_\_(self,object):

        if not isinstance(object,Circle):

            raise TypeError("Правый операнд должен быть типом 'Circle'")

        return self.\_\_radius + object.get\_radius()

    def \_\_radd\_\_(self,object):

        if not isinstance(object,Circle):

            raise TypeError("Левый операнд должен быть по типу 'Circle'")

        return object.get\_radius() - self.get\_radius()

    def \_\_sub\_\_(self,object):

        if not isinstance(object,Circle):

            raise ArithmeticError("Правый операнд должен быть типом 'Circle'")

        return self.\_\_radius - object.get\_radius()

    def \_\_rsub\_\_(self,object):

        if not isinstance(object,Circle):

            raise TypeError("Левый операнд должен быть по типу 'Circle'")

        return object.get\_radius() - self.\_\_radius

    def \_\_iadd\_\_(self,object):

        if not isinstance(object,Circle):

            raise TypeError("Все операнды должны быть по типу 'Circle'")

        self.\_\_radius += object.get\_radius()

        return self

    def \_\_isub\_\_(self,object):

        if not isinstance(object,Circle):

            raise TypeError("Все операнды должны быть по типу 'Circle'")

        self.\_\_radius -= object.get\_radius()

        return self

    def \_\_eq\_\_(self,object):

        if not isinstance(object,Circle):

            raise TypeError("Все операнды должны быть по типу 'Circle'")

        return self.\_\_radius == object.get\_radius()

    def \_\_ne\_\_(self,object):

        return not self.\_\_eq\_\_(object)

    def \_\_lt\_\_(self,object):

        if not isinstance(object,Circle):

            raise TypeError("Все операнды должны быть по типу 'Circle'")

        return self.get\_circle\_length() < object.get\_circle\_length()

    def \_\_gt\_\_(self,object):

        if not isinstance(object,Circle):

            raise TypeError("Все операнды должны быть по типу 'Circle'")

        return self.get\_circle\_length() > object.get\_circle\_length()

    def \_\_le\_\_(self,object):

        return self.\_\_lt\_\_(object) or self.get\_circle\_length() == object.get\_circle\_length()

    def \_\_ge\_\_(self,object):

        return  self.\_\_gt\_\_(object) or self.get\_circle\_length() == object.get\_circle\_length()

class View:

    def \_\_view\_dollars\_controller(function):

        def add\_dollars(self):

            print('\n$$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$')

            print()

            function(self)

            print('\n$$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$ $$')

        return add\_dollars

    def \_\_view\_mines\_controller(function):

        def add\_mines(self):

            print('\n-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --')

            print()

            function(self)

            print('\n-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --')

        return add\_mines

    def input\_radius(self):

        return float(input('Пожалуйста, введите радиус окружности: '))

    @\_\_view\_mines\_controller

    def view\_non\_created\_circle\_error(self):

        print('Извините, но вы еще не создали обе окружности, что выполнить что то')

    def show\_stars(self):

        print('\n\*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\*')

    def input\_user\_choice(self):

        return int(input('Ваш выбор: '))

    def view\_options(self,options):

        print()

        current\_option = 1

        for option in options:

            print(current\_option,option)

            current\_option += ONE\_STEP

    @\_\_view\_mines\_controller

    def view\_non\_available\_option\_message(self):

        print('Извините, но такой опции нет.')

    @\_\_view\_dollars\_controller

    def view\_buy\_message(self):

        print('Спасибо за использование программы! До свидания!')

    @\_\_view\_dollars\_controller

    def view\_successful\_circle\_creation\_message(self):

        print('Ваша окружность успешно создана')

    def view\_sum\_radius\_result(self,result):

        print('Результат суммы радиусов двух окружностей:',result)

    def view\_subtraction\_radius\_result(self,circle\_a,circle\_b,result):

        print('{} - {} = {} '.format(circle\_a,circle\_b,result))

    def view\_circle\_radius(self,circle,result):

        print('\nРадиус {} окружности составляeт: {} см'.format(circle,result))

    @\_\_view\_mines\_controller

    def view\_non\_availabe\_radius(self):

        print('Длина радиуса не может быть отрицательной!')

    @\_\_view\_dollars\_controller

    def show\_equal\_radiuses\_message(self):

        print('Радиусы ваших окружностей равны друг другу!')

    @\_\_view\_mines\_controller

    def show\_non\_equal\_radiuses\_message(self):

        print('Радиусы ваших окружностей НЕ равны друг другу!')

    @\_\_view\_dollars\_controller

    def show\_less\_left\_length\_circle\_message(self):

        print('Длина окружности слева МЕНЬШЕ длины окружности справа')

    @\_\_view\_dollars\_controller

    def show\_higher\_rigth\_length\_circle\_message(self):

        print('Длина окружности слева БОЛЬШе длины окружности справа')

    @\_\_view\_dollars\_controller

    def show\_less\_equal\_left\_length\_circle\_message(self):

        print('Длина окружности слева МЕНЬШЕ или РАВНА длине окружности справа')

    @\_\_view\_dollars\_controller

    def show\_higher\_equal\_left\_length\_circle\_message(self):

        print('Длина окружности слева БОЛЬШЕ или РАВНА длине окружности справа')

    def view\_both\_circles\_info(self,circle\_a,circle\_b):

        print('Ваши Созданные Окружности: ')

        print('\t Первая:',circle\_a)

        print('\t Вторая:',circle\_b,end = "")

    def view\_error\_message(self,error):

        print(error)

LESS\_OPTION = 2

MORE\_OPTION = 3

LESS\_EQUAL\_OPTION = 4

MORE\_EQUAL\_OPTION = 5

class Controller:

    menu\_options = ['Создать Первую Окружность',

                    'Создать Вторую Окружность',

                    'Операция ( Первая окружность + Вторая окружность )',

                    'Операция ( Первая окружность - Вторая окружность )',

                    'Операция ( Вторая окружность - Первая окружность )',

                    'Операция ( Увеличить Первую окружность значением Второй окружности)',

                    'Операция ( Увеличить Вторую окружность значением Первой окружности)',

                    'Операция ( Уменьшить Первую окружность значением Второй окружности)',

                    'Операция ( Уменьшить Вторую окружность значением Первой окружности)',

                    'Сранивить Окружности оператором сравнения',

                    'Выйти из программы']

    condition\_options = ['Радиус Первой Окружности == Радиус Второй Окружности',

                        'Длина Первой Окружности < Длина Второй Окружности',

                        'Длина Первой Окружности > Длина Второй Окружности',

                        'Длина Первой Окружности <= Длина Второй Окружности',

                        'Длина Первой Окружности >= Длина Второй Окружности']

    view = View()

    circle\_a = None

    circle\_b = None

    def \_\_create\_circle(self):

        self.view.show\_stars()

        radius = self.view.input\_radius()

        self.view.show\_stars()

        circle = Circle(radius)

        self.view.view\_successful\_circle\_creation\_message()

        return circle

    def create\_first\_circle(self):

        self.circle\_a = self.\_\_create\_circle()

    def create\_second\_circle(self):

        self.circle\_b = self.\_\_create\_circle()

    def make\_plus\_operation(self):

        if(self.circle\_a is None or self.circle\_b is None):

            self.view.view\_non\_created\_circle\_error()

        else:

            sum\_result = self.circle\_a + self.circle\_b

            self.view.show\_stars()

            self.view.view\_sum\_radius\_result(sum\_result)

            self.view.show\_stars()

    def make\_subtraction\_first\_second\_circles(self):

        if(self.circle\_a is None or self.circle\_b is None):

            self.view.view\_non\_created\_circle\_error()

        else:

            subtraction\_result = self.circle\_a - self.circle\_b

            self.view.show\_stars()

            self.view.view\_subtraction\_radius\_result('Радиус Первой окружности','Радиус Второй окружности',subtraction\_result)

            self.view.show\_stars()

    def make\_subtraction\_second\_first\_circles(self):

        if(self.circle\_a is None or self.circle\_b is None):

            self.view.view\_non\_created\_circle\_error()

        else:

            subtraction\_result = self.circle\_b - self.circle\_a

            self.view.show\_stars()

            self.view.view\_subtraction\_radius\_result('Радиус Второй окружности','Радиус Первой окружности',subtraction\_result)

            self.view.show\_stars()

    def increase\_first\_circle\_radius(self):

        if(self.circle\_a is None or self.circle\_b is None):

            self.view.view\_non\_created\_circle\_error()

        else:

            self.circle\_a += self.circle\_b

            self.view.show\_stars()

            self.view.view\_circle\_radius('Первой',self.circle\_a.get\_radius())

            self.view.show\_stars()

    def increase\_second\_circle\_radius(self):

        if(self.circle\_a is None or self.circle\_b is None):

            self.view.view\_non\_created\_circle\_error()

        else:

            self.circle\_b += self.circle\_a

            self.view.show\_stars()

            self.view.view\_circle\_radius('Второй',self.circle\_b.get\_radius())

            self.view.show\_stars()

    def decrease\_first\_circle\_radius(self):

        if(self.circle\_a is None or self.circle\_b is None):

            self.view.view\_non\_created\_circle\_error()

        else:

            if(self.circle\_a - self.circle\_b < 0):

                self.view.view\_non\_availabe\_radius()

            else:

                self.circle\_a -= self.circle\_b

                self.view.show\_stars()

                self.view.view\_circle\_radius('Первой',self.circle\_a.get\_radius())

                self.view.show\_stars()

    def decrease\_second\_circle\_radius(self):

        if(self.circle\_a is None or self.circle\_b is None):

            self.view.view\_non\_created\_circle\_error()

        else:

            if(self.circle\_b - self.circle\_a < 0):

                self.view.view\_non\_availabe\_radius()

            else:

                self.circle\_b -= self.circle\_a

                self.view.show\_stars()

                self.view.view\_circle\_radius('Второй',self.circle\_b.get\_radius())

                self.view.show\_stars()

    def compare\_radiuses(self):

        if(self.circle\_a == self.circle\_b):

            self.view.show\_equal\_radiuses\_message()

        else:

            self.view.show\_non\_equal\_radiuses\_message()

    def compare\_circle\_lengths(self,option):

        if(option == LESS\_OPTION):

            if(self.circle\_a < self.circle\_b):

                self.view.show\_less\_left\_length\_circle\_message()

            else:

                self.view.show\_higher\_rigth\_length\_circle\_message()

        elif(option == MORE\_OPTION):

            if(self.circle\_a > self.circle\_b):

                self.view.show\_higher\_rigth\_length\_circle\_message()

            else:

                self.view.show\_less\_left\_length\_circle\_message()

        elif(option == LESS\_EQUAL\_OPTION):

            if(self.circle\_a <= self.circle\_b):

                self.view.show\_less\_equal\_left\_length\_circle\_message()

            else:

                self.view.show\_higher\_equal\_left\_length\_circle\_message()

        elif(option == MORE\_EQUAL\_OPTION):

            if(self.circle\_a >= self.circle\_b):

                self.view.show\_higher\_equal\_left\_length\_circle\_message()

            else:

                self.view.show\_less\_equal\_left\_length\_circle\_message()

    def compare\_two\_circles(self):

        if(self.circle\_a is not None and self.circle\_b is not None):

            self.view.view\_options(self.condition\_options)

            user\_choice = self.view.input\_user\_choice()

            if(user\_choice == 1):

                self.compare\_radiuses()

            else:

                self.compare\_circle\_lengths(user\_choice)

        else:

            self.view.view\_non\_created\_circle\_error()

    functions = [create\_first\_circle,create\_second\_circle,make\_plus\_operation,

                make\_subtraction\_first\_second\_circles,make\_subtraction\_second\_first\_circles,

                increase\_first\_circle\_radius,increase\_second\_circle\_radius,

                decrease\_first\_circle\_radius,decrease\_second\_circle\_radius,compare\_two\_circles]

    def start\_menu(self):

        while True:

            try:

                if(self.circle\_a is not None and self.circle\_b is not None):

                    self.view.view\_both\_circles\_info(self.circle\_a,self.circle\_b)

                self.view.view\_options(self.menu\_options)

                user\_choice = self.view.input\_user\_choice()

                if(user\_choice < 1 or user\_choice > len(self.menu\_options)):

                    self.view.view\_non\_available\_option\_message()

                    continue

                elif(user\_choice == len(self.menu\_options)):

                    self.view.view\_buy\_message()

                    break

                self.functions[user\_choice - ONE\_STEP](self)

            except TypeError as te:

                print('asd')

                self.view.view\_error\_message(te)

            except ValueError as ve:

                self.view.view\_error\_message(ve)

            except Exception as ex:

                self.view.view\_error\_message(ex)

controller = Controller()

controller.start\_menu()





























