Домашнее задание.

Маратулы 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!**

Написать программу, которая вычисляет периметр и площадь разных геометрических фигур (круг, квадрат, равносторонний треугольник). Программа должна реализовать следующие пункты меню:

1. Выбрать фигуру;
2. Ввести размер фигуры (радиус, сторону квадрата или треугольника)
3. Вывести периметр фигуры
4. Вывести площадь фигуры

В программе, по мере необходимости и возможности использовать классы, наследование, ссылки на функции, список функций.

import math

class CorrectFigure:

    def \_\_init\_\_(self,side):

        self.\_\_side = side

    def get\_side(self):

        return self.\_\_side

    def set\_side(self,side):

        self.\_\_side = side

class Circle(CorrectFigure):

    \_\_PERIMETER\_TERMINAL = 2

    def \_\_init\_\_(self,side):

        CorrectFigure.\_\_init\_\_(self,side)

    def get\_perimeter(self):

        return self.\_\_PERIMETER\_TERMINAL \* math.pi \* super().get\_side()

    def get\_area(self):

        return math.pi \* super().get\_side() \* super().get\_side()

    def \_\_str\_\_(self):

        return 'Радиус Круга: {} см. Периметр Круга: {} см. \

Площадь Круга: {}см^2\

'.format(super().get\_side(),self.get\_perimeter(),self.get\_area())

class Square(CorrectFigure):

    \_\_PERIMETER\_TERMINAL = 4

    def \_\_init\_\_(self, side):

        super().\_\_init\_\_(side)

    def get\_perimeter(self):

        return self.\_\_PERIMETER\_TERMINAL \* super().get\_side()

    def get\_area(self):

        return super().get\_side() \* super().get\_side()

    def \_\_str\_\_(self):

        return 'Сторона Квадрата: {} см. Периметр Квадрата: {} см. \

Площадь Квадрата: {} см^2\

'.format(super().get\_side(),self.get\_perimeter(),self.get\_area())

class EquilateralTriangle(CorrectFigure):

    \_\_PERIMETER\_TERMINAL = 3

    \_\_AREA\_TERMINAL = 4

    def \_\_init\_\_(self, side):

        super().\_\_init\_\_(side)

    def get\_perimeter(self):

        return self.\_\_PERIMETER\_TERMINAL \* super().get\_side()

    def get\_area(self):

        return (super().get\_side()\*super().get\_side() \* math.sqrt(self.\_\_PERIMETER\_TERMINAL))/self.\_\_AREA\_TERMINAL

    def \_\_str\_\_(self):

        return 'Сторона Треугольника: {} см. Периметр: {} см. \

Площадь: {} см^2\

'.format(super().get\_side(),self.get\_perimeter(),self.get\_area())

ONE\_STEP = 1

class View:

    def input\_square\_side(self):

        return float(input('Введите сторону Квадрата: '))

    def input\_circle\_side(self):

        return float(input('Введите сторону Круга: '))

    def input\_triangle\_side(self):

        return float(input('Введите длину стороны Треугольника: '))

    def input\_user\_choice(self):

        return input('Ваш выбор: ')

    def view\_all\_options(self,options):

        current\_option = 1

        for option in options:

            print(current\_option,option)

            current\_option += ONE\_STEP

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

        print('Извините, Но вы не создали еще фигуру!')

    def view\_wrong\_option\_message(self):

        print('Извините, такой опции нет! Повторите попытку')

    def show\_buy\_message(self):

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

    def input\_figure\_shape(self):

        return input('Введите тип геометрической фигуры: ')

    def view\_figure\_options(self,figure\_variants):

        current\_variant = 1

        for variant in figure\_variants:

            print(current\_variant,variant)

            current\_variant += ONE\_STEP

    def show\_non\_existed\_shape\_message(self):

        print('Извините, но такой фигуры не существует!')

    def show\_successfull\_creation\_shape\_message(self):

        print('Ваша геометрическая фигура успешно создана!')

    def show\_figure\_area(self,area):

        print('Площадь вашей геометрической фигуры составляет: {} см^2'.format(area))

    def show\_figure\_perimeter(self,perimetr):

        print('Периметр вашей геометрической фигуры составляет: {} см'.format(perimetr))

CREATE\_FIGURE\_OPTION = '1'

INPUT\_FIGURE\_DATA\_OPTION = '2'

SHOW\_FIGURE\_PERIMETER\_OPTION = '3'

SHOW\_FIGURE\_AREA\_OPTION = '4'

EXIT\_OPTION = '5'

CIRCLE\_OPTION = '1'

SQUARE\_OPTION = '2'

TRIANGLE\_OPTION = '3'

figure\_shape\_options = (CIRCLE\_OPTION,SQUARE\_OPTION,TRIANGLE\_OPTION)

menu\_options = (CREATE\_FIGURE\_OPTION,INPUT\_FIGURE\_DATA\_OPTION,SHOW\_FIGURE\_PERIMETER\_OPTION,SHOW\_FIGURE\_AREA\_OPTION,EXIT\_OPTION)

class Controller:

    view = View()

    menu\_options = ['Выбрать фигуру',

                    'Ввести/Изменить размер фигуры',

                    'Вывести периметр фигуры',

                    'Вывести площадь фигуры',

                    'Выйти из программы']

    figure\_variants = ['Круг','Квадрат','Правильный треугольник']

    figure = None

    def \_\_is\_figure\_created(self):

        if(self.figure == None):

            return False

        return True

    def create\_circle(self):

        circle\_radius = self.view.input\_circle\_side()

        self.figure = Circle(circle\_radius)

        self.view.show\_successfull\_creation\_shape\_message()

    def create\_square(self):

        square\_side = self.view.input\_square\_side()

        self.figure = Square(square\_side)

        self.view.show\_successfull\_creation\_shape\_message()

    def create\_triangle(self):

        triangle\_side = self.view.input\_triangle\_side()

        self.figure = EquilateralTriangle(triangle\_side)

        self.view.show\_successfull\_creation\_shape\_message()

    shape\_creation\_options = [create\_circle,create\_square,create\_triangle]

    def start\_first\_option(self):

        self.view.view\_figure\_options(self.figure\_variants)

        user\_choice = self.view.input\_figure\_shape()

        if(user\_choice not in figure\_shape\_options):

            self.view.show\_non\_existed\_shape\_message()

        else:

            self.shape\_creation\_options[int(user\_choice) - ONE\_STEP](self)

    def start\_second\_option(self):

        if(self.\_\_is\_figure\_created()):

            if(type(self.figure) == Circle):

                self.figure.set\_side(self.view.input\_circle\_side())

            elif(type(self.figure) == Square):

                self.figure.set\_side(self.view.input\_square\_side())

            elif(type(self.figure) == EquilateralTriangle):

                self.figure.set\_side(self.view.input\_triangle\_side())

        else:

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

    def start\_third\_option(self):

        if(self.\_\_is\_figure\_created()):

            self.view.show\_figure\_perimeter(self.figure.get\_perimeter())

        else:

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

    def start\_fourth\_option(self):

        if(self.\_\_is\_figure\_created()):

            self.view.show\_figure\_area(self.figure.get\_area())

        else:

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

    functions = [start\_first\_option,start\_second\_option,start\_third\_option,start\_fourth\_option]

    def main\_menu(self):

        while True:

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

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

            if(user\_choice not in menu\_options):

                self.view.view\_wrong\_option\_message()

                continue

            elif(user\_choice == EXIT\_OPTION):

                self.view.show\_buy\_message()

                break

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

controller = Controller()

controller.main\_menu()









