

**Московский государственный технический
университет им. Н.Э. Баумана**

**Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»**

Курс «Базовые компоненты интернет-технологий»

Лабораторная работа №2

Выполнил:
студент группы ИУ5-34Б
Гордеев Никита

Подпись и дата:

Проверил:
Гапанюк Ю. Е.

Подпись и дата:

Москва, 2021 г.

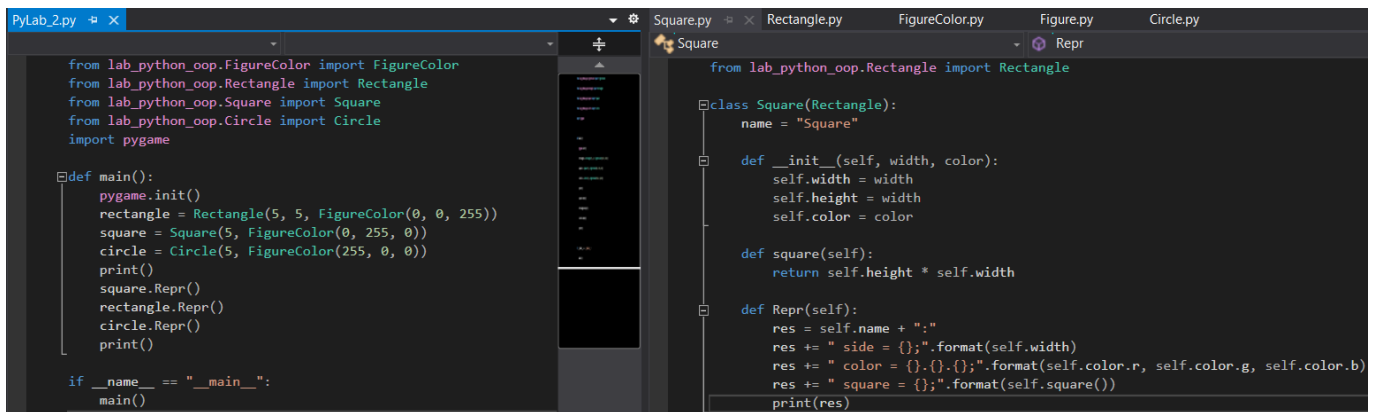
Постановка задачи

Задание:

1. Необходимо создать виртуальное окружение и установить в него хотя бы один внешний пакет с использованием `pip`.
2. Необходимо разработать программу, реализующую работу с классами. Программа должна быть разработана в виде консольного приложения на языке Python 3.
3. Все файлы проекта (кроме основного файла `main.py`) должны располагаться в пакете `lab_python_oop`.
4. Каждый из нижеперечисленных классов должен располагаться в отдельном файле пакета `lab_python_oop`.
5. Абстрактный класс «Геометрическая фигура» содержит абстрактный метод для вычисления площади фигуры. Подробнее про абстрактные классы и методы Вы можете прочитать [здесь](#).
6. Класс «Цвет фигуры» содержит свойство для описания цвета геометрической фигуры. Подробнее про описание свойств Вы можете прочитать [здесь](#).
7. Класс «Прямоугольник» наследуется от класса «Геометрическая фигура». Класс должен содержать конструктор по параметрам «ширина», «высота» и «цвет». В конструкторе создается объект класса «Цвет фигуры» для хранения цвета. Класс должен переопределять метод, вычисляющий площадь фигуры.
8. Класс «Круг» создается аналогично классу «Прямоугольник», задается параметр «радиус». Для вычисления площади используется константа `math.pi` из модуля `math`.
9. Класс «Квадрат» наследуется от класса «Прямоугольник». Класс должен содержать конструктор по длине стороны. Для классов «Прямоугольник», «Квадрат», «Круг»:
 - Определите метод `getr`, который возвращает в виде строки основные параметры фигуры, ее цвет и площадь. Используйте метод `format` - <https://pyformat.info/>
 - Название фигуры («Прямоугольник», «Квадрат», «Круг») должно задаваться в виде поля данных класса и возвращаться методом класса.
10. В корневом каталоге проекта создайте файл `main.py` для тестирования Ваших классов (используйте следующую конструкцию - https://docs.python.org/3/library/__main__.html). Создайте следующие объекты и выведите о них информацию в консоль (N - номер Вашего варианта по списку группы):
 - Прямоугольник синего цвета шириной N и высотой N.
 - Круг зеленого цвета радиусом N.
 - Квадрат красного цвета со стороной N.
 - Также вызовите один из методов внешнего пакета, установленного с использованием `pip`.

Текст программы

Файлы: PyLab_2.py и Square.py



The screenshot shows a code editor with two files open. The left file, `PyLab_2.py`, contains a `main` function that imports `FigureColor`, `Rectangle`, `Square`, and `Circle` from `lab_python_oop`, initializes a pygame window, and creates and prints objects of these classes. The right file, `Square.py`, defines a `Square` class that inherits from `Rectangle`. It includes an `__init__` method, a `square` method to calculate area, and a `Repr` method for string representation.

```
from lab_python_oop.FigureColor import FigureColor
from lab_python_oop.Rectangle import Rectangle
from lab_python_oop.Square import Square
from lab_python_oop.Circle import Circle
import pygame

def main():
    pygame.init()
    rectangle = Rectangle(5, 5, FigureColor(0, 0, 255))
    square = Square(5, FigureColor(0, 255, 0))
    circle = Circle(5, FigureColor(255, 0, 0))
    print()
    square.Repr()
    rectangle.Repr()
    circle.Repr()
    print()

if __name__ == "__main__":
    main()
```

```
from lab_python_oop.Rectangle import Rectangle

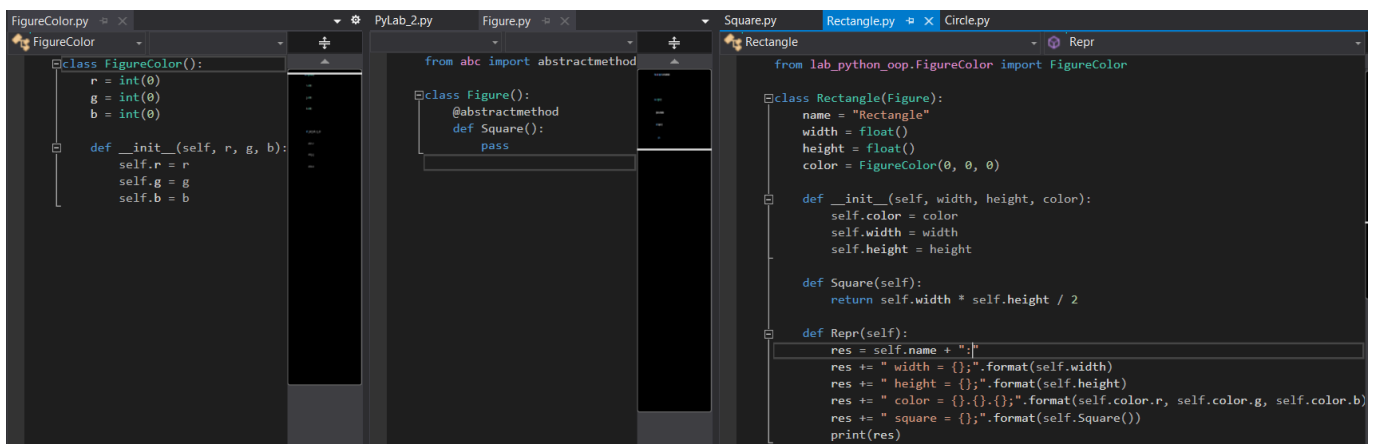
class Square(Rectangle):
    name = "Square"

    def __init__(self, width, color):
        self.width = width
        self.height = width
        self.color = color

    def square(self):
        return self.height * self.width

    def Repr(self):
        res = self.name + ":"
        res += " side = {}:".format(self.width)
        res += " color = {}.{}.{}:".format(self.color.r, self.color.g, self.color.b)
        res += " square = {}:".format(self.square())
        print(res)
```

Файлы: FigureColor.py, Figure.py и Rectangle.py



The screenshot shows a code editor with three files open. The left file, `FigureColor.py`, defines a `FigureColor` class with an `__init__` method that initializes red, green, and blue components. The middle file, `Figure.py`, defines an abstract `Figure` class with an abstract `square` method. The right file, `Rectangle.py`, defines a `Rectangle` class that inherits from `Figure`, implementing the `square` method and adding a `Repr` method for string representation.

```
class FigureColor():
    r = int(0)
    g = int(0)
    b = int(0)

    def __init__(self, r, g, b):
        self.r = r
        self.g = g
        self.b = b
```

```
from abc import abstractmethod

class Figure():
    @abstractmethod
    def square():
        pass
```

```
from lab_python_oop.FigureColor import FigureColor

class Rectangle(Figure):
    name = "Rectangle"
    width = float()
    height = float()
    color = FigureColor(0, 0, 0)

    def __init__(self, width, height, color):
        self.color = color
        self.width = width
        self.height = height

    def square(self):
        return self.width * self.height / 2

    def Repr(self):
        res = self.name + ":"
        res += " width = {}:".format(self.width)
        res += " height = {}:".format(self.height)
        res += " color = {}.{}.{}:".format(self.color.r, self.color.g, self.color.b)
        res += " square = {}:".format(self.square())
        print(res)
```

```
from lab_python_oop.Figure import Figure
from lab_python_oop.FigureColor import FigureColor
from math import pi

class Circle(Figure):
    name = "Circle"
    radius = float()
    color = FigureColor(0, 0, 0)

    def __init__(self, radius, color):
        self.radius = radius
        self.color = color

    def Square(self):
        return pi * self.radius**2

    def Repr(self):
        res = self.name + ":"
        res += " radius = {}:".format(self.radius)
        res += " color = {}.{}.{}:".format(self.color.r, self.color.g, self.color.b)
        res += " square = {}:".format(self.Square())
        print(res)
```

Результат выполнения

```
C:\Users\nagor\AppData\Local\  +  v
pygame 2.1.0 (SDL 2.0.16, Python 3.10.1)
Hello from the pygame community. https://www.pygame.org/contribute.html

Square: side = 5; color = 0.255.0; square = 25;
Rectangle: width = 5; height = 5; color = 0.0.255; square = 12.5;
Circle: radius = 5; color = 255.0.0; square = 78.53981633974483;

Press any key to continue . . . |
```