

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

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

Курс «Парадигмы и конструкции языков  
программирования» Отчёт по лабораторной  
работе №2

Выполнил:  
студент группы ИУ5-35Б  
Александров Георгий

Проверил: преподаватель каф. ИУ5  
Нардид Анатолий Николаевич

Москва, 2025 г.

## Описание задания

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 "get", 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](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

### main.py

```
import sys
import math

def get_valid_float(prompt):
    while True:
        try:
            value = float(input(prompt))
            return value
        except ValueError:
            print("Ошибка! Введите число.")

def solve_biquadratic(a, b, c):
    if a == 0:
        print("Коэффициент А не может быть равен 0!")
        return []

    print(f"\nУравнение: {a}*x^4 + {b}*x^2 + {c} = 0")

    D = b**2 - 4*a*c
    print(f"Дискриминант D = {D}")

    roots = []

    if D < 0:
        print("Дискриминант отрицательный. Действительных корней нет.")
    elif D == 0:
        t = -b / (2*a)
        if t > 0:
            x1 = math.sqrt(t)
            x2 = -math.sqrt(t)
            roots = [x1, x2]
            print(f"Корни уравнения: x1 = {x1:.4f}, x2 = {x2:.4f}")
        elif t == 0:
            roots = [0]
            print("Корень уравнения: x = 0")
        else:
            print("Нет действительных корней.")
    else:
        t1 = (-b + math.sqrt(D)) / (2*a)
        t2 = (-b - math.sqrt(D)) / (2*a)

        if t1 > 0:
```

```

    x1 = math.sqrt(t1)
    x2 = -math.sqrt(t1)
    roots.extend([x1, x2])
elif t1 == 0:
    roots.append(0)

if t2 > 0:
    x3 = math.sqrt(t2)
    x4 = -math.sqrt(t2)
    if x3 not in roots and -x3 not in roots:
        roots.extend([x3, x4])
elif t2 == 0 and 0 not in roots:
    roots.append(0)

if roots:
    roots = list(set(roots))
    roots.sort()
    print(f"Найдено {len(roots)} действительных корней:")
    for i, root in enumerate(roots, 1):
        print(f"  x {i} = {root:.4f}")
else:
    print("Действительных корней нет.")

```

```

return roots

```

```

def main():
    print("=" * 60)
    print("РЕШЕНИЕ БИКВАДРАТНОГО УРАВНЕНИЯ:  $A \cdot x^4 + B \cdot x^2 + C = 0$ ")
    print("=" * 60)

```

```

a = b = c = None

```

```

if len(sys.argv) >= 4:
    try:
        a = float(sys.argv[1])
        b = float(sys.argv[2])
        c = float(sys.argv[3])
        print(f"Коэффициенты из командной строки: A={a}, B={b}, C={c}")
    except ValueError:
        print("Ошибка в аргументах. Будет выполнен ввод с клавиатуры.")
        a = b = c = None

```

```

if a is None:
    print("\nВведите коэффициенты биквадратного уравнения:")

```

```

a = get_valid_float("Коэффициент А (не равен 0): ")
while a == 0:
    print("Коэффициент А не может быть равен 0!")
    a = get_valid_float("Коэффициент А (не равен 0): ")

```

```

if b is None:

```

```

    b = get_valid_float("Коэффициент В: ")

```

```

if c is None:

```

```

    c = get_valid_float("Коэффициент С: ")

```

```

solve_biquadratic(a, b, c)

```

```

if __name__ == "__main__":
    main()

```

## **В папке lab\_python\_oop** **init.py**

## **В папке lab\_python\_oop** **circle.py**

```

from lab_python_oop.figure import Figure
from lab_python_oop.color import Color
import math

```

```

class Circle(Figure):

```

```

    figure_type = "Круг"

```

```

    def __init__(self, radius, color):

```

```

        self.radius = radius

```

```

        self.color = Color(color)

```

```

    def area(self):

```

```

        return math.pi * self.radius ** 2

```

```

    def __repr__(self):

```

```

        return '{} {} цвета радиусом {} площадью {:.2f}'.format(
            self.get_figure_type(),
            self.color.color,
            self.radius,
            self.area()

```

```

        )

```

## **В папке lab\_python\_oop**

### **color.py**

```
class Color:
    def __init__(self, color):
        self._color = color

    @property
    def color(self):
        return self._color

    @color.setter
    def color(self, value):
        self._color = value
```

## **В папке lab\_python\_oop**

### **figure.py**

```
from abc import ABC, abstractmethod

class Figure(ABC):
    figure_type = "Фигура"

    @abstractmethod
    def area(self):
        """Абстрактный метод для вычисления площади"""
        pass

    @classmethod
    def get_figure_type(cls):
        return cls.figure_type
```

## **В папке lab\_python\_oop**

### **rectangle.py**

```
from lab_python_oop.figure import Figure
from lab_python_oop.color import Color

class Rectangle(Figure):
    figure_type = "Прямоугольник"

    def __init__(self, width, height, color):
```

```

        self.width = width
        self.height = height
        self.color = Color(color)

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

    def __repr__(self):
        return '{} {} цвета шириной {} и высотой {} площадью {}'.format(
            self.get_figure_type(),
            self.color.color,
            self.width,
            self.height,
            self.area()
        )

```

## **В папке lab\_python\_oop**

### **square.py**

```

from lab_python_oop.rectangle import Rectangle

```

```

class Square(Rectangle):
    figure_type = "Квадрат"

    def __init__(self, side, color):
        super().__init__(side, side, color)

    def __repr__(self):
        return '{} {} цвета со стороной {} площадью {}'.format(
            self.get_figure_type(),
            self.color.color,
            self.width,
            self.area()
        )

```

## Листинг кода

```
• (venv) PS C:\Users\MSI-Crosshair16\Documents\лабы\lab_python_fp\2> python main.py
Лабораторная работа №2. Объектно-ориентированные возможности Python
=====
Прямоугольник синего цвета шириной 5 и высотой 5 площадью 25.
Круг зеленого цвета радиусом 5 площадью 78.54.
Квадрат красного цвета со стороной 5 площадью 25.

=====
Демонстрация работы внешнего пакета (numpy):
Создан массив numpy: [1 2 3 4 5]
Среднее значение: 3.0
Стандартное отклонение: 1.41
○ (venv) PS C:\Users\MSI-Crosshair16\Documents\лабы\lab_python_fp\2> █
```