МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчёт по лабораторной работе №4.3 по дисциплине:

Основы программной инженерии

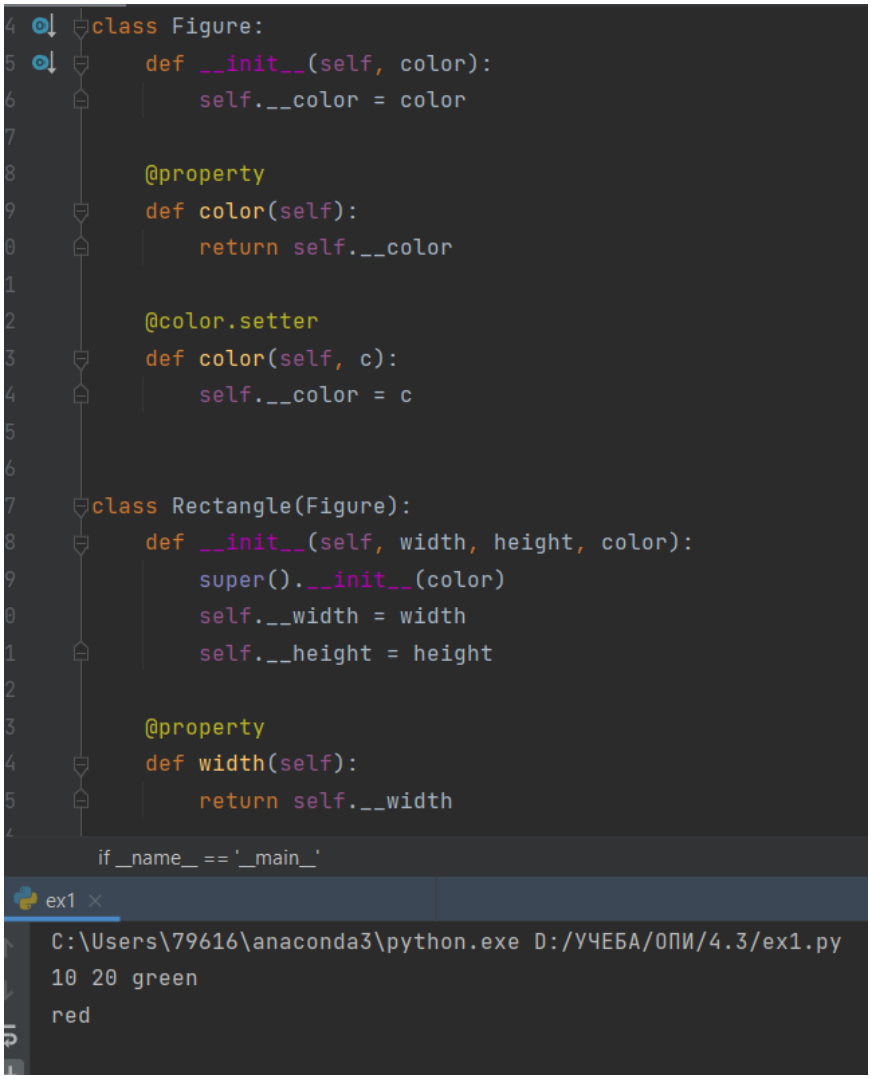
Выполнила:
студент группы ПИЖ-б-о-20-1
Лазарева Дарья Олеговна

Проверил:
доцент кафедры инфокоммуникаций
Романкин Р.А.

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ:

1. Примеры из методических указаний

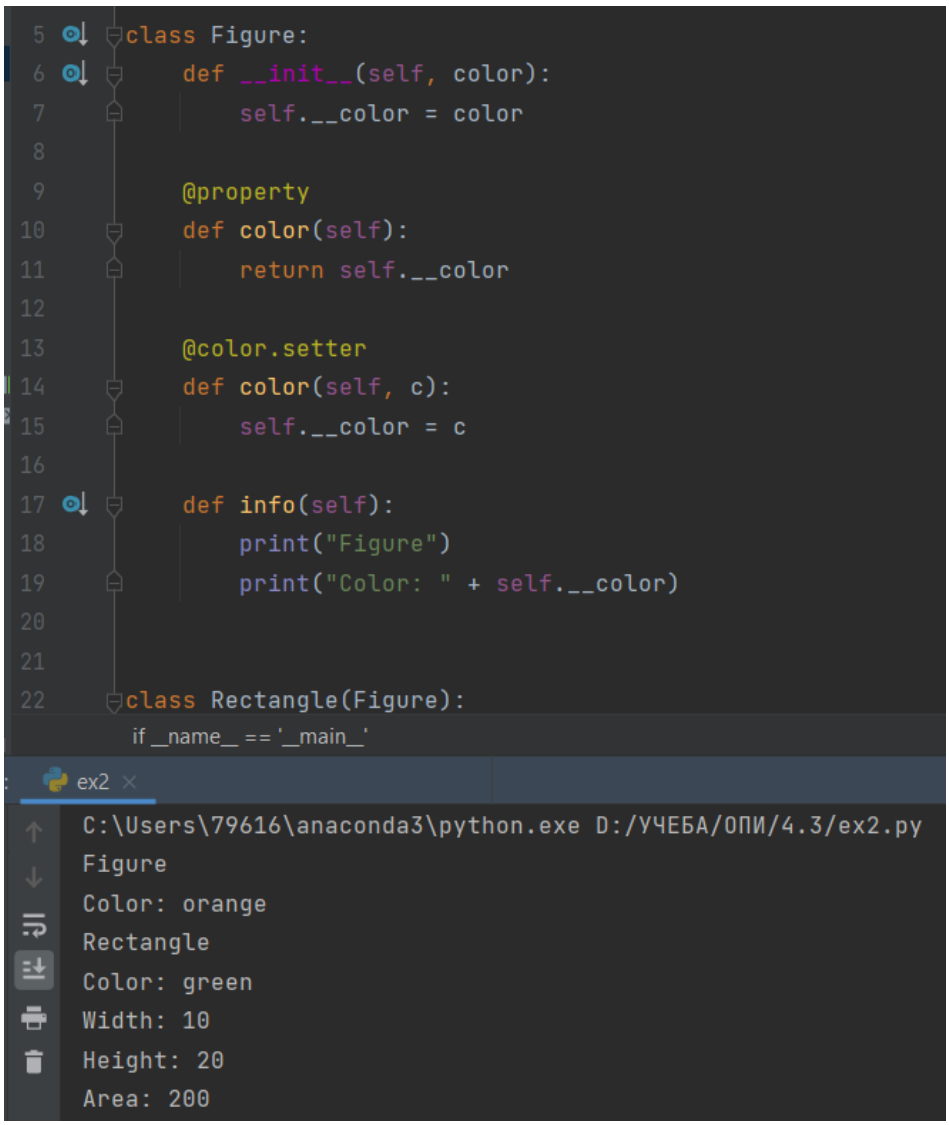


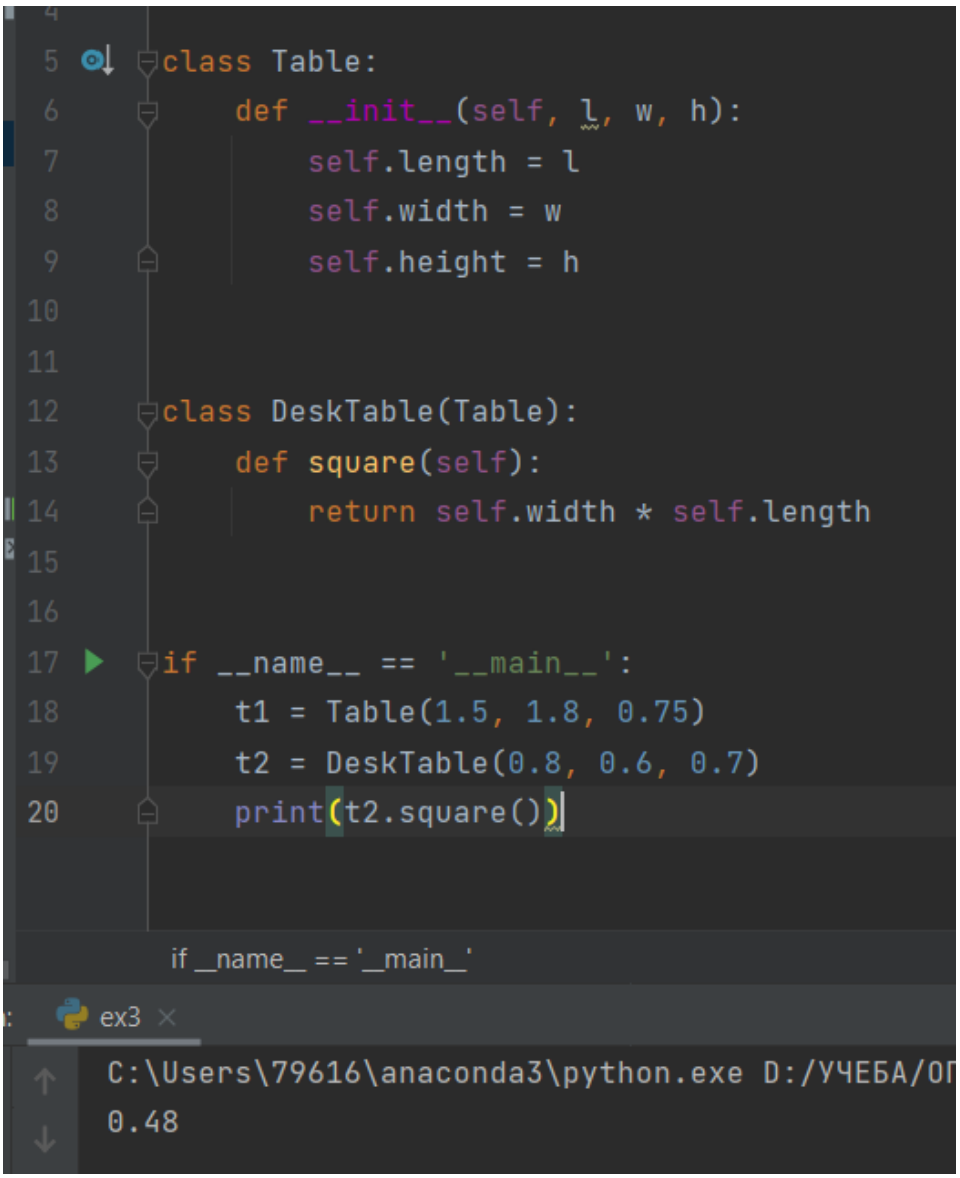
```
class Figure:
    def __init__(self, color):
        self.__color = color

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

    @color.setter
    def color(self, c):
        self.__color = c


class Rectangle(Figure):
    def __init__(self, width, height, color):
        super().__init__(color)
        self.__width = width
        self.__height = height

    @property
    def width(self):
        return self.__width
```

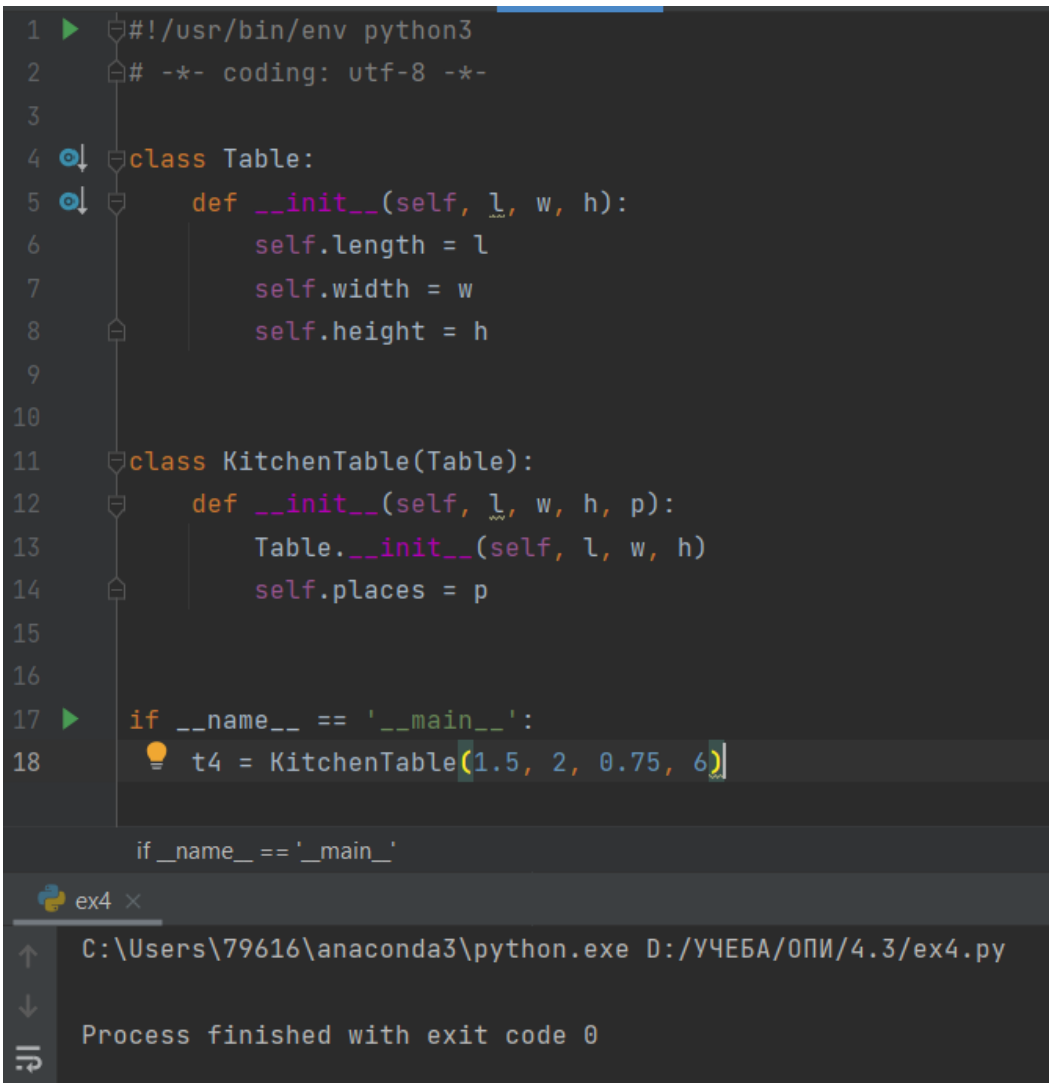
```
if __name__ == '__main__'
```

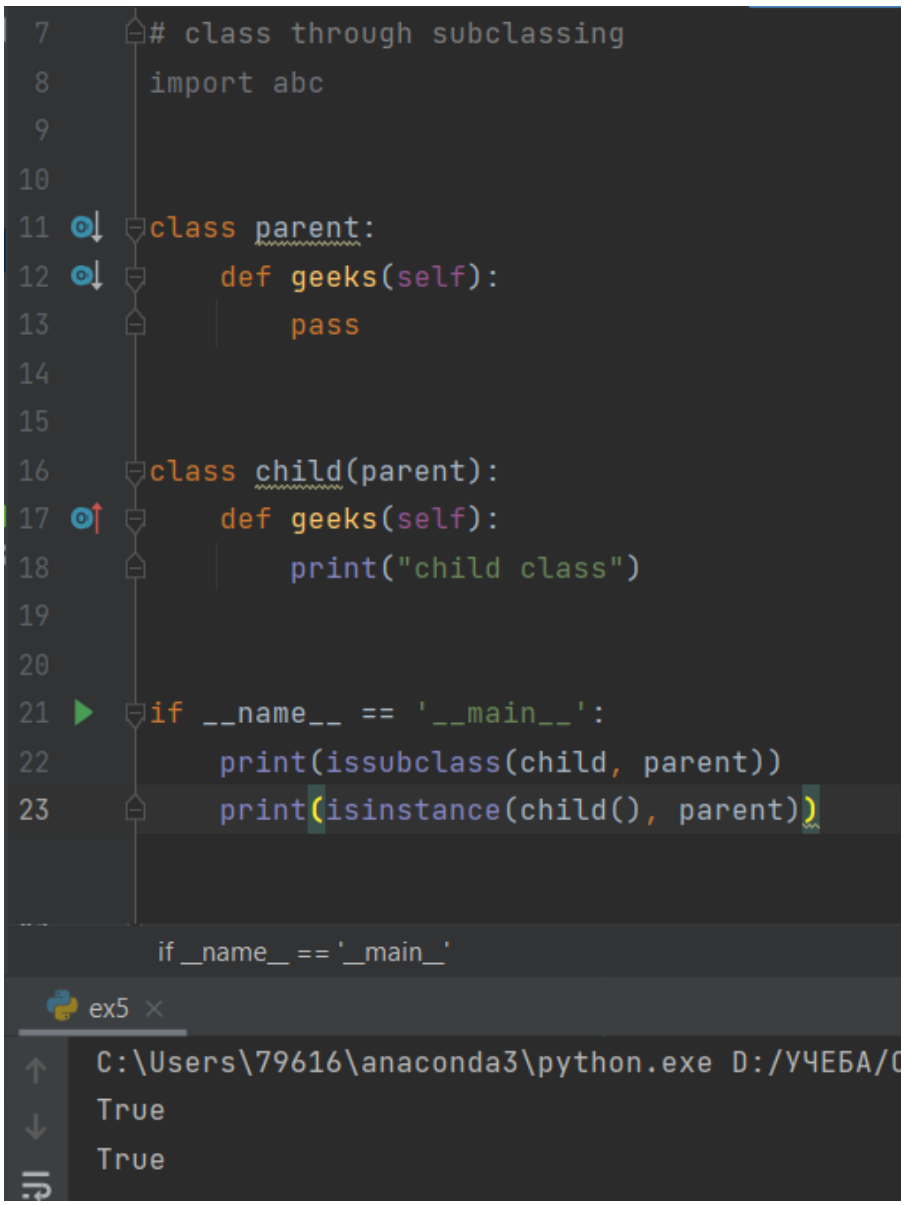
ex1

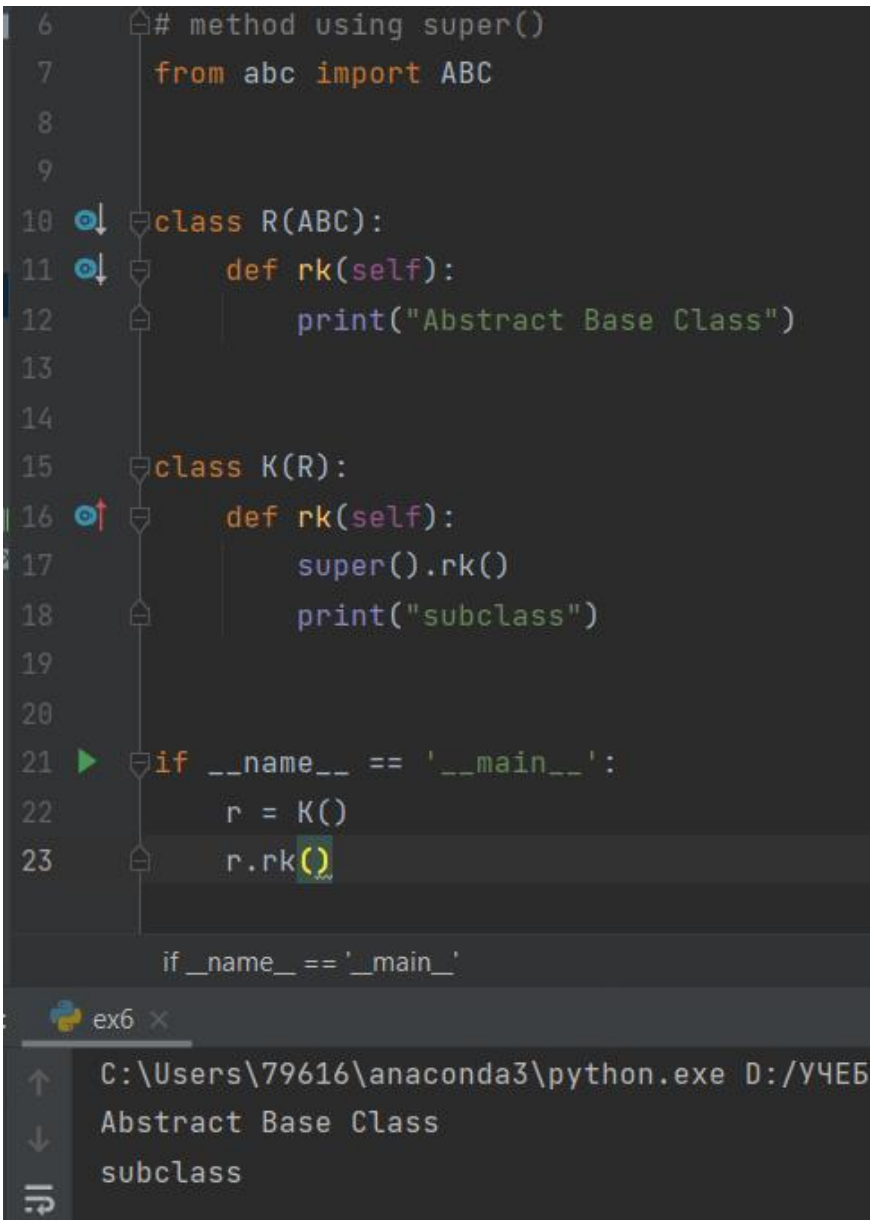
```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/4.3/ex1.py
10 20 green
red
```

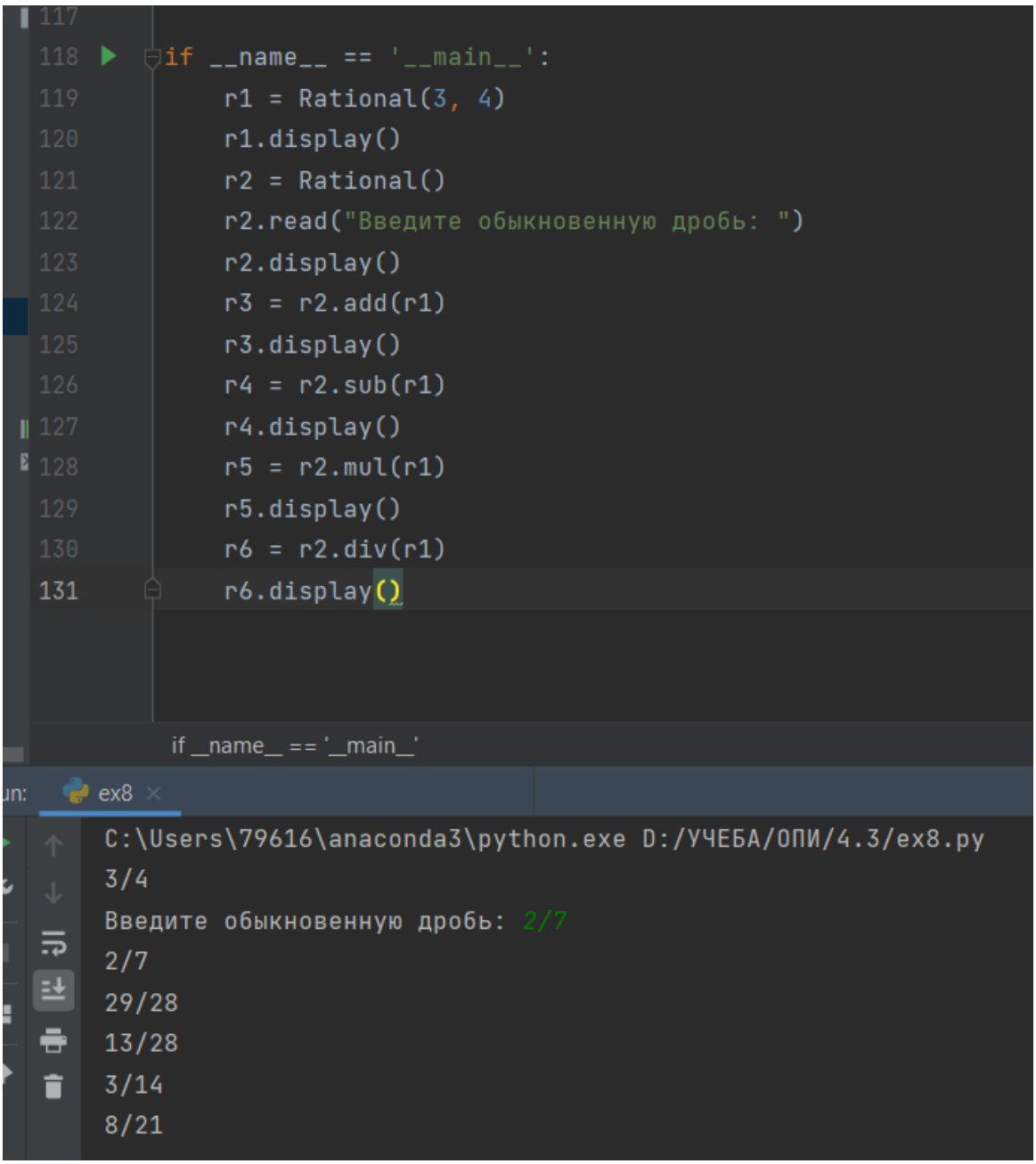

class Figure:
def __init__(self, color):
self.__color = color
@property
def color(self):
return self.__color
@color.setter
def color(self, c):
self.__color = c
def info(self):
print("Figure")
print("Color: " + self.__color)
class Rectangle(Figure):
if __name__ == '__main__'
ex2
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/4.3/ex2.py
Figure
Color: orange
Rectangle
Color: green
Width: 10
Height: 20
Area: 200


class Table:
def __init__(self, l, w, h):
self.length = l
self.width = w
self.height = h
class DeskTable(Table):
def square(self):
return self.width * self.length
if __name__ == '__main__':
t1 = Table(1.5, 1.8, 0.75)
t2 = DeskTable(0.8, 0.6, 0.7)
print(t2.square())
if __name__ == '__main__'
ex3
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОП
0.48


#!/usr/bin/env python3
# -*- coding: utf-8 -*-
class Table:
def __init__(self, l, w, h):
self.length = l
self.width = w
self.height = h
class KitchenTable(Table):
def __init__(self, l, w, h, p):
Table.__init__(self, l, w, h)
self.places = p
if __name__ == '__main__':
t4 = KitchenTable(1.5, 2, 0.75, 6)
if __name__ == '__main__'
ex4
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/4.3/ex4.py
Process finished with exit code 0


7 # class through subclassing
8 import abc
9
10
11 class parent:
12 def geeks(self):
13 pass
14
15
16 class child(parent):
17 def geeks(self):
18 print("child class")
19
20
21 if __name__ == '__main__':
22 print(issubclass(child, parent))
23 print(isinstance(child(), parent))
if __name__ == '__main__'
ex5
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/С
True
True

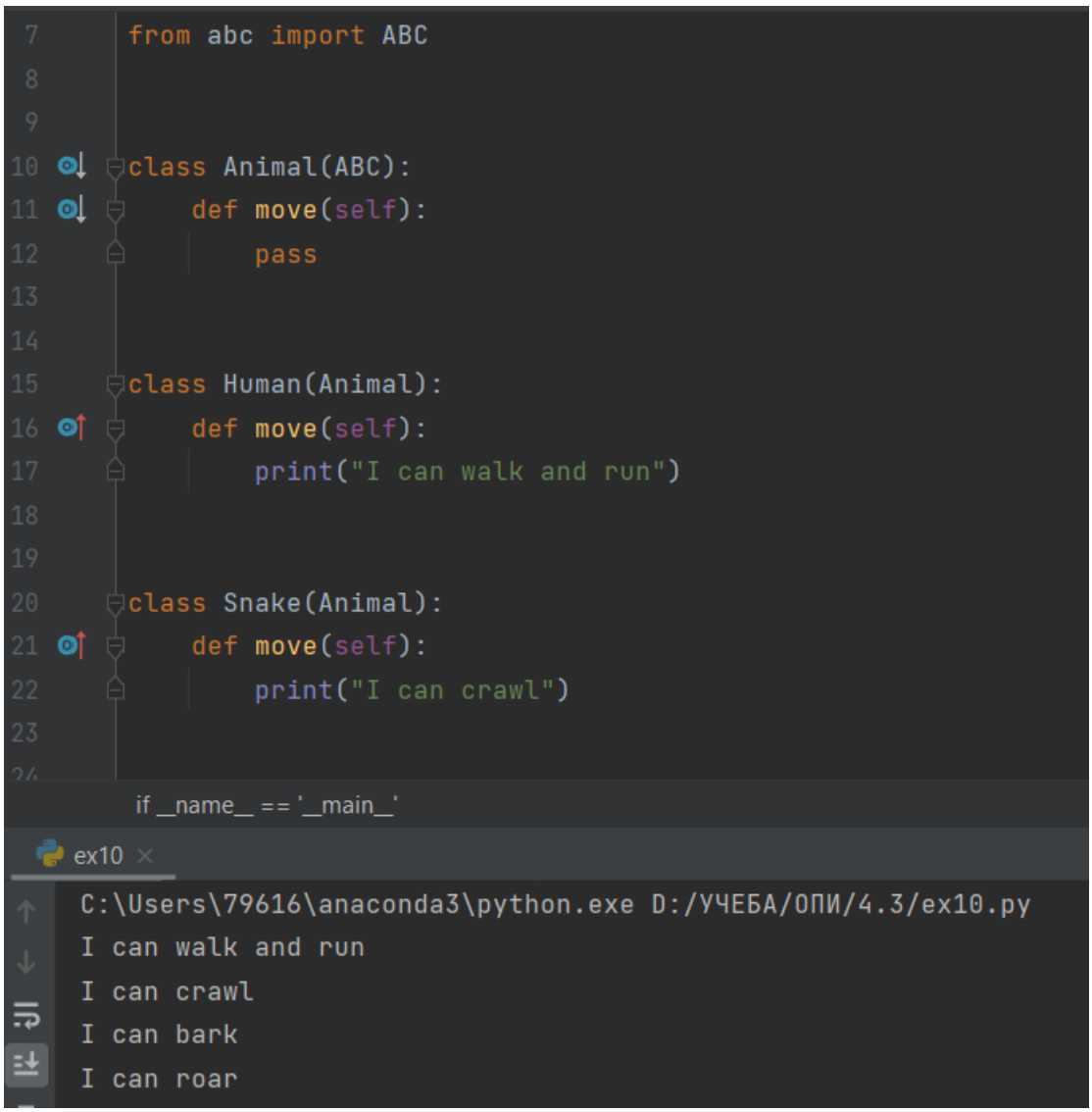

6 # method using super()
7 from abc import ABC
8
9
10 class R(ABC):
11 def rk(self):
12 print("Abstract Base Class")
13
14
15 class K(R):
16 def rk(self):
17 super().rk()
18 print("subclass")
19
20
21 if __name__ == '__main__':
22 r = K()
23 r.rk()
if __name__ == '__main__'
ex6
C:\Users\79616\anaconda3\python.exe D:/УЧЕБ
Abstract Base Class
subclass


if __name__ == '__main__':
r1 = Rational(3, 4)
r1.display()
r2 = Rational()
r2.read("Введите обыкновенную дробь: ")
r2.display()
r3 = r2.add(r1)
r3.display()
r4 = r2.sub(r1)
r4.display()
r5 = r2.mul(r1)
r5.display()
r6 = r2.div(r1)
r6.display()
if __name__ == '__main__'
ex8
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/4.3/ex8.py
3/4
Введите обыкновенную дробь: 2/7
2/7
29/28
13/28
3/14
8/21

```python
class Polygon(ABC):
    @abstractmethod
    def noofsides(self):
        pass


class Triangle(Polygon):
    # overriding abstract method
    def noofsides(self):
        print("I have 3 sides")


class Pentagon(Polygon):
    # overriding abstract method
    def noofsides(self):
        print("I have 5 sides")
```

ex9

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/4.3/ex9.py
I have 3 sides
I have 4 sides
I have 5 sides
I have 6 sides
```



Индивидуальное задание 1.

1. Создать базовый класс Car (машина), характеризуемый торговой маркой (строка), числом цилиндров, мощностью. Определить методы переназначения и изменения мощности. Создать производный класс Lorry (грузовик), характеризуемый также грузоподъемностью кузова. Определить функции переназначения марки и изменения грузоподъемности.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


class Car:
    def __init__(self, mark, cylinders, power):
        self.__mark = mark
        self.__cylinders = cylinders
        self.__power = power

    @property
    def mark(self):
        return self.__mark
```

```
    @mark.setter
    def mark(self, inp):
        self.__mark = inp

    @property
    def cylinders(self):
        return self.__cylinders

    @cylinders.setter
    def cylinders(self, inp):
        self.__cylinders = inp

    @property
    def power(self):
        return self.__power

    @power.setter
    def power(self, inp):
        self.__power = inp


class Lorry(Car):
    def __init__(self, mark, cylinders, power, capacity):
        super().__init__(mark, cylinders, power)
        self.__capacity = capacity

    @property
    def capacity(self):
        return self.__capacity

    @capacity.setter
    def capacity(self, inp):
        self.__capacity = inp


def main():
    truck = Lorry("Газель", 4, 119, 2500)
    passenger = Car("Лада", 8, 105)
    print(f"Марка {truck.mark}, количество цилиндров: {truck.cylinders}, "
          f"мощность: {truck.power} л.с, грузоподъемность: {truck.capacity}
т")
    print(f"Марка {passenger.mark}, количество цилиндров:
{passenger.cylinders}")



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

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/4.3/ind1.py
Марка Газель, количество цилиндров: 3, мощность: 150 л.с, грузоподъемность: 2500 т
Марка Лада, количество цилиндров: 2
```

Индивидуальное задание №2

1. Создать абстрактный базовый класс Figure с абстрактными методами вычисления площади и периметра. Создать производные классы: Rectangle (прямоугольник), Circle (круг), Trapezium (трапеция) со своими функциями

площади и периметра. Самостоятельно определить, какие поля необходимы, какие из них можно задать в базовом классе, а какие — в производных. Площадь трапеции:

$$S = (a + b) \times h/2.$$

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


from abc import ABC, abstractmethod


class Figure(ABC):

    @abstractmethod
    def square(self):
        pass

    @abstractmethod
    def perimeter(self):
        pass


class Rectangle(Figure):
    def square(self, a, b):
        print("Площадь прямоугольника: ", a * b)

    def perimeter(self, a, b):
        print("Периметр прямоугольника: ", 2 * (a + b))


class Circle(Figure):
    def square(self, b, h):
        print("Площадь треугольника: ", 0.5 * b * h)

    def perimeter(self, a, b, c):
        print("Периметр треугольника: ", a + b + c)


class Trapezium(Figure):
    def square(self, a, b, h):
        print("Площадь трапеции: ", (a + b) * h / 2)

    def perimeter(self, a, b, c, d):
        print("Периметр трапеции: ", a + b + c + d)


def main():
    r1 = Rectangle()
    r1.square(8, 10)
    r1.perimeter(4, 5)

    c1 = Circle()
    c1.square(3, 6)
    c1.perimeter(4, 8, 6)
```

```
    t1 = Trapezium()
    t1.square(1, 3, 8)
    t1.perimeter(6, 4, 7, 2)


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

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/4.3/ind2.py
Площадь прямоугольника:  80
Периметр прямоугольника:  18
Площадь треугольника:  9.0
Периметр треугольника:  18
Площадь трапеции:  16.0
Периметр трапеции:  19
```

ВОПРОСЫ

1. Что такое наследование как оно реализовано в языке Python?

Синтаксически создание класса с указанием его родителя выглядит так:
class имя_класса(имя_родителя1, [имя_родителя2,..., имя_родителя_n])
super – это ключевое слово, которое используется для обращения к родительскому классу.

2. Что такое полиморфизм и как он реализован в языке Python?

Полиморфизм, как правило, используется с позиции переопределения методов базового класса в классе наследнике. Переопределение прописывается в классе-наследнике.

3. Что такое "утиная" типизация в языке программирования Python?

Утиная типизация – это концепция, характерная для языков программирования с динамической типизацией, согласно которой конкретный тип или класс объекта не важен, а важны лишь свойства и методы, которыми этот объект обладает. Другими словами, при работе с объектом его тип не проверяется, вместо этого проверяются свойства и методы этого объекта. Такой подход добавляет гибкости коду, позволяет полиморфно работать с объектами, которые никак не связаны друг с

другоми могут быть объектами разных классов. Единственное условие, чтобы все эти объекты поддерживали необходимый набор свойств и методов.

4. Каково назначение модуля abc языка программирования Python?

По умолчанию Python не предоставляет абстрактных классов. Python поставляется с модулем, который обеспечивает основу для определения абстрактных базовых классов (ABC), и имя этого модуля - ABC. ABC работает, декорируя методы базового класса как абстрактные, а затем регистрируя конкретные классы как реализации абстрактной базы.

5. Как сделать некоторый метод класса абстрактным?

Метод становится абстрактным, если он украшен ключевым словом @abstractmethod.

6. Как сделать некоторое свойство класса абстрактным?

Абстрактные классы включают в себя атрибуты в дополнение к методам, вы можете потребовать атрибуты в конкретных классах, определивих с помощью @abstractproperty.

7. Каково назначение функции isinstance ?

Встроенная функция isinstance(obj, Cls) , используемая при реализации методов арифметических операций и операций отношения, позволяет узнатьчто некоторый объект obj является либо экземпляром класса Cls либо экземпляром одного из потомков класса Cls.