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 №4.1 по дисциплине:

Основы программной инженерии

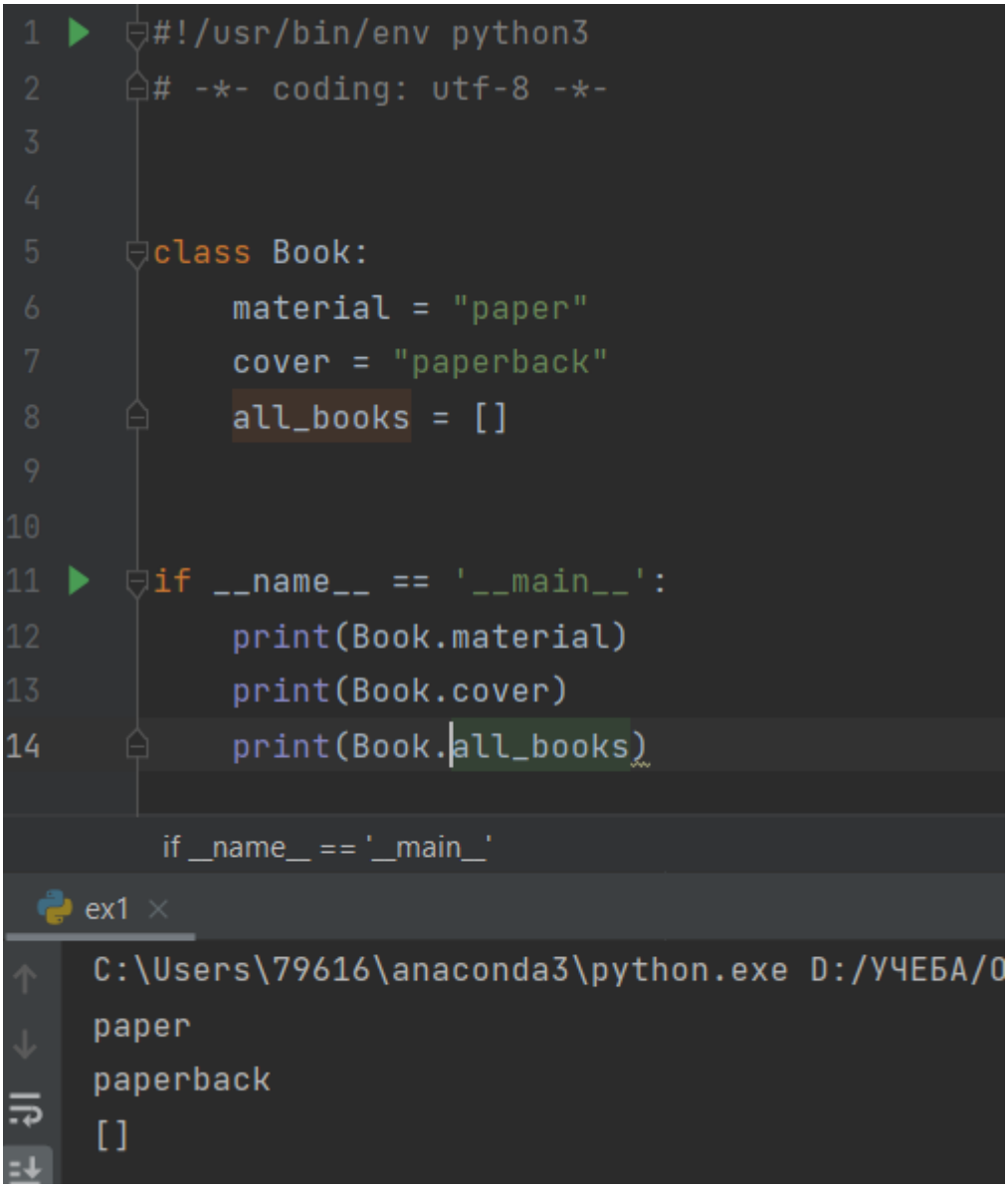
Выполнила:
студент группы ПИЖ-б-о-20-1
Лазарева Дарья Олеговна

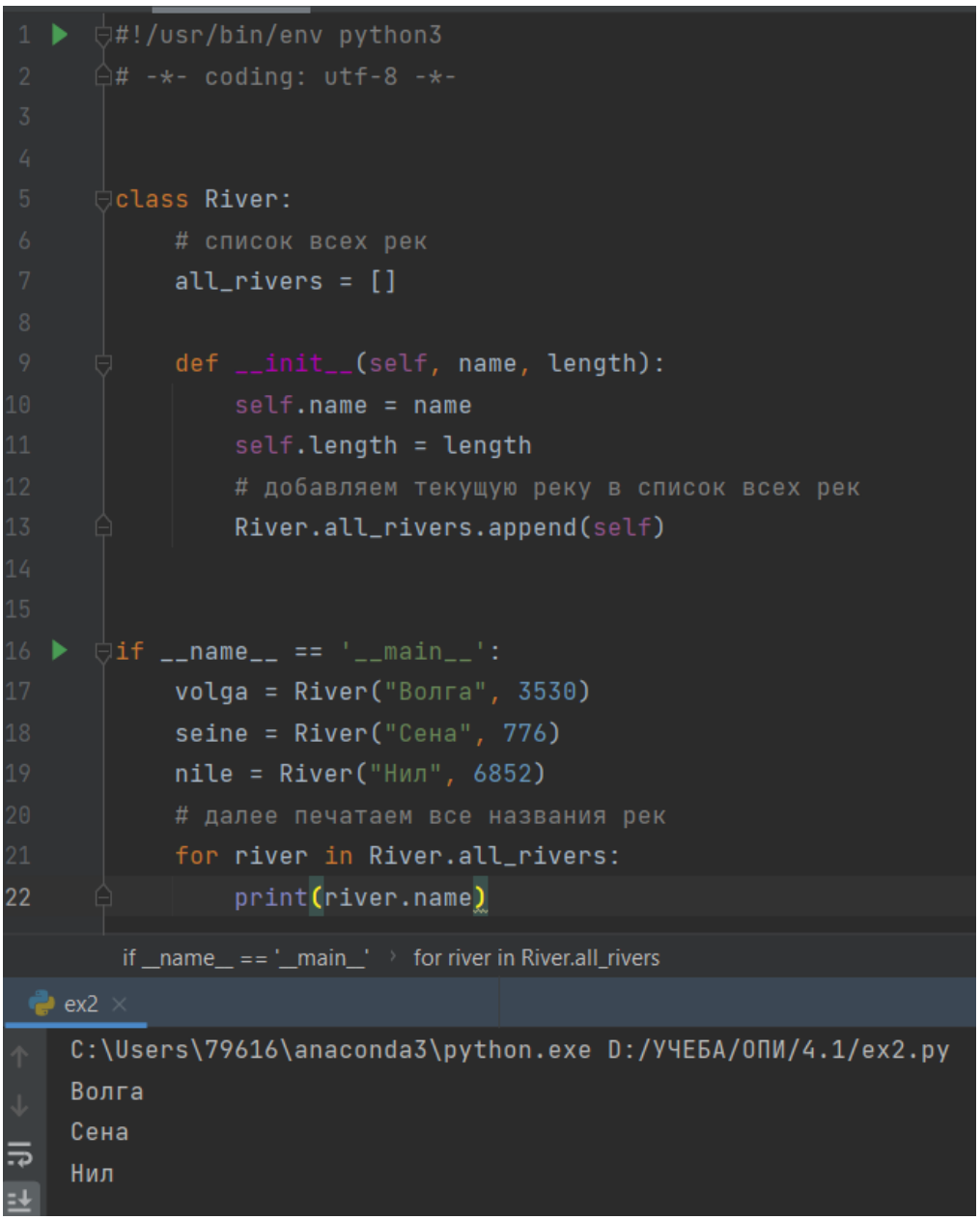
Проверил:
доцент кафедры инфокоммуникаций
Романкин Р.А.

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ:

1. Примеры из методических указаний



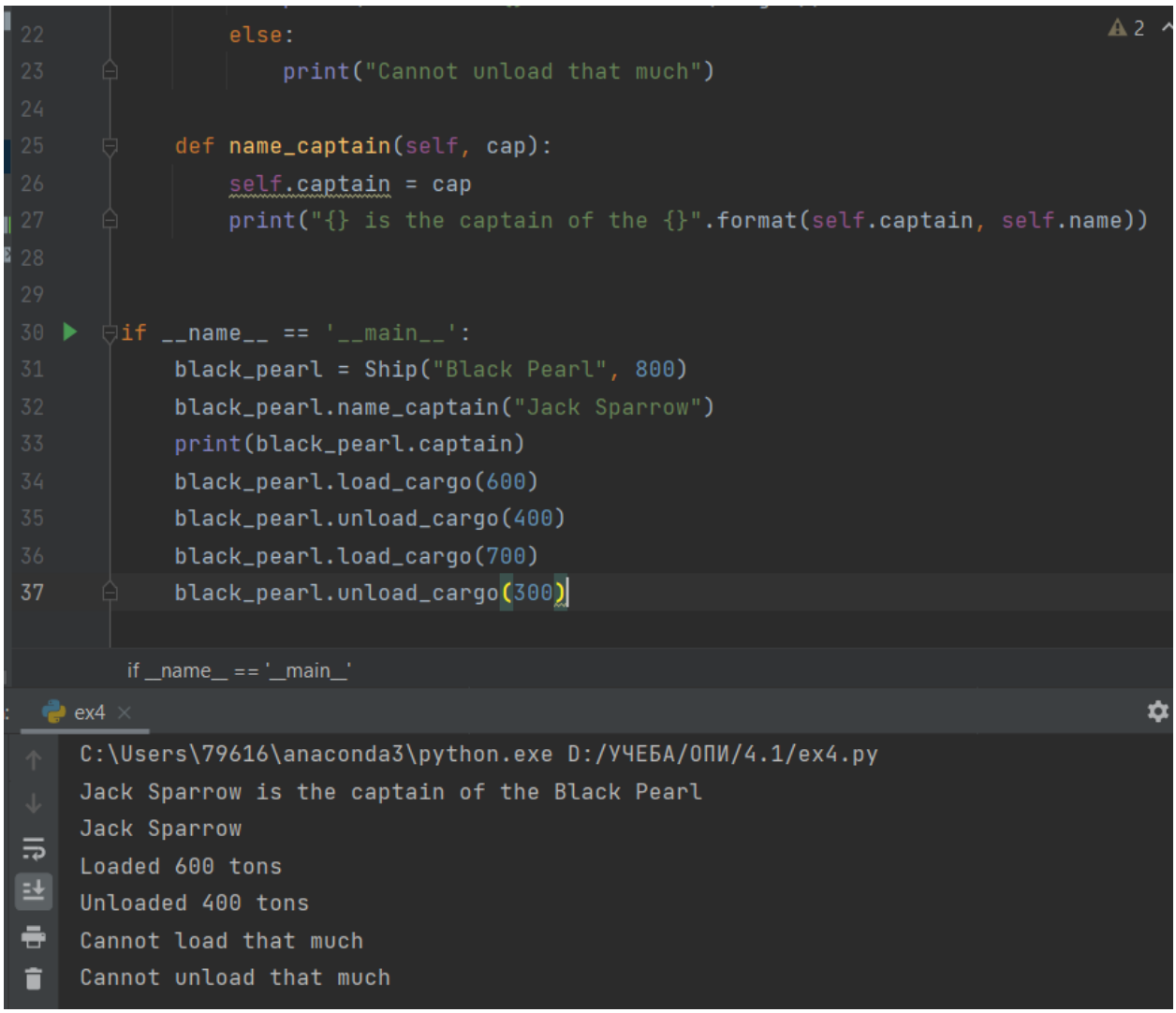

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
class River:
# список всех рек
all_rivers = []
def __init__(self, name, length):
self.name = name
self.length = length
# добавляем текущую реку в список всех рек
River.all_rivers.append(self)
if __name__ == '__main__':
volga = River("Волга", 3530)
seine = River("Сена", 776)
nile = River("Нил", 6852)
# далее печатаем все названия рек
for river in River.all_rivers:
print(river.name)
if __name__ == '__main__' › for river in River.all_rivers
ex2
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/4.1/ex2.py
Волга
Сена
Нил

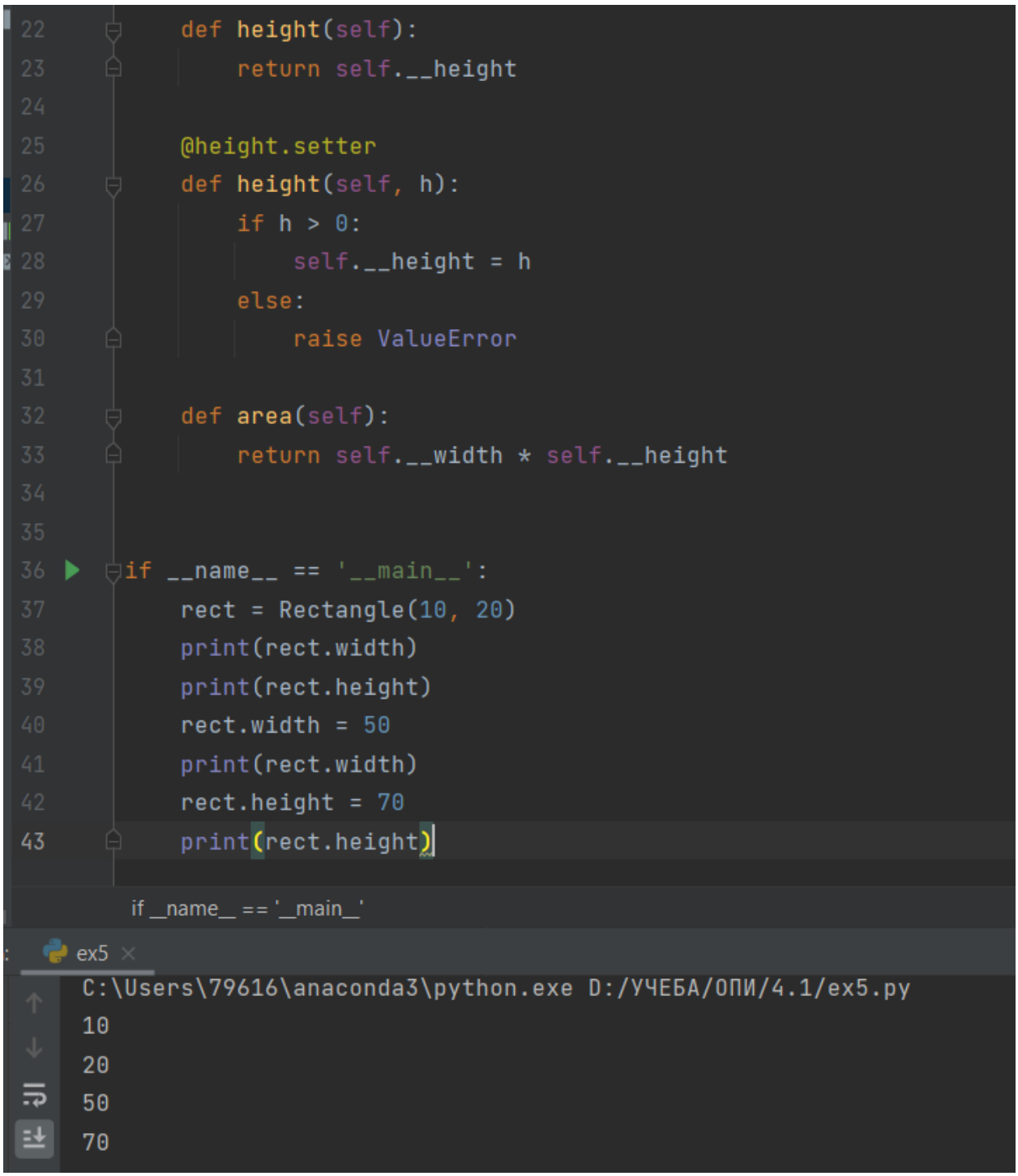
```python
class River:
    all_rivers = []

    def __init__(self, name, length):
        self.name = name
        self.length = length
        River.all_rivers.append(self)

    def get_info(self):
        print("Длина {0} равна {1} км".format(self.name, self.length))


if __name__ == '__main__':
    volga = River("Волга", 3530)
    seine = River("Сена", 776)
    nile = River("Нил", 6852)
    volga.get_info()
    seine.get_info()
    nile.get_info()
```

if __name__ == '__main__'

ex3

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/4.1/ex3.py
Длина Волга равна 3530 км
Длина Сена равна 776 км
Длина Нил равна 6852 км
```


2
else:
print("Cannot unload that much")
def name_captain(self, cap):
self.captain = cap
print("{} is the captain of the {}".format(self.captain, self.name))
if __name__ == '__main__':
black_pearl = Ship("Black Pearl", 800)
black_pearl.name_captain("Jack Sparrow")
print(black_pearl.captain)
black_pearl.load_cargo(600)
black_pearl.unload_cargo(400)
black_pearl.load_cargo(700)
black_pearl.unload_cargo(300)
if __name__ == '__main__'
ex4
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/4.1/ex4.py
Jack Sparrow is the captain of the Black Pearl
Jack Sparrow
Loaded 600 tons
Unloaded 400 tons
Cannot load that much
Cannot unload that much



```
    def height(self):
        return self.__height

    @height.setter
    def height(self, h):
        if h > 0:
            self.__height = h
        else:
            raise ValueError

    def area(self):
        return self.__width * self.__height


if __name__ == '__main__':
    rect = Rectangle(10, 20)
    print(rect.width)
    print(rect.height)
    rect.width = 50
    print(rect.width)
    rect.height = 70
    print(rect.height)
```

```
if __name__ == '__main__'
ex5
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/4.1/ex5.py
10
20
50
70
```

Индивидуальное задание №1

Поле first — дробное число; поле second — дробное число, показатель степени. Реализовать метод power() — возведение числа first в с тепень second. Метод должен правильно работать при любых допустимых значениях first и second.

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


class Num:
    def __init__(self, first=0, second=0):
        self.first = first
        self.second = second

    def read(self):
        self.first = float(input("Введите дробное число: "))
        self.second = float(input("Введите еще одно дробное число: "))

    def display(self):
        print(f"Результат: ", power(self.first, self.second))


def power(first, second):
    if first == 0:
        raise ValueError
    else:
        return first ** second


if __name__ == "__main__":
    newNumber = Num(3.1,4.6)
    newNumber.display()
    newNumber.read()
    newNumber.display()
```

```
Введите дробное число: 3.8
Введите еще одно дробное число: 2.7
Результат:  36.76340349722826
```

Индивидуальное задание №2

Создать класс Vector3D, задаваемый тройкой координат. Обязательно должны быть реализованы: сложение и вычитание векторов, скалярное произведение векторов, умножение на скаляр, сравнение векторов, вычисление длины вектора, сравнение длины векторов.

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

import math


class Vector3D:
    def __init__(self, x=0, y=0, z=0):
        self.x = x
        self.y = y
        self.z = z

    def read(self, prompt=None):
        line = input() if prompt is None else input(prompt)
```

```
        parts = list(map(int, line.split(' ', maxsplit=2)))
        if parts[2] == 0:
            raise ValueError()

        self.x = parts[0]
        self.y = parts[1]
        self.z = parts[2]

    def display(self):
        print(f"Координаты:  {self.x}, {self.y}, {self.z}")

    #Сложение
    def sum(self, rhs):
        if isinstance(rhs, Vector3D):
            return Vector3D((self.x + rhs.x), (self.y + rhs.y), (self.z +
rhs.z))
        else:
            raise ValueError

    #Вычитание
    def sub(self, rhs):
        if isinstance(rhs, Vector3D):
            return Vector3D((rhs.x - self.x), (rhs.y - self.y), (rhs.z -
self.z))
        else:
            raise ValueError

    # Скалярное произведение
    def dot(self, rhs):
        if isinstance(rhs, Vector3D):
            return self.x * rhs.x + self.y * rhs.y + self.z * rhs.z
        else:
            raise ValueError

    # Сравнение векторов
    def com(self, rhs):
        if isinstance(rhs, Vector3D):
            return (self.x == rhs.x) and (self.y == rhs.y) and (self.z ==
rhs.z)
        else:
            return False

    def greater(self, rhs):
        if isinstance(rhs, Vector3D):
            vector1 = (self.x, self.y, self.z)
            vector2 = (rhs.x, rhs.y, rhs.z)
            return vector1 > vector2
        else:
            return False

    def less(self, rhs):
        if isinstance(rhs, Vector3D):
            vector1 = (self.x, self.y, self.z)
            vector2 = (rhs.x, rhs.y, rhs.z)
            return vector1 < vector2
        else:
            return False

    # Вычисление длины векторов
    def length(self, rhs):
        if isinstance(rhs, Vector3D):
            vector1 = math.sqrt(pow(self.x, 2) + pow(self.y, 2) + pow(self.z,
2))
            vector2 = math.sqrt(pow(rhs.x, 2) + pow(rhs.y, 2) + pow(rhs.z,
```

```
2))
            return vector1 + vector2
        else:
            raise ValueError

    # Сравнение длины векторов
    def equal(self, rhs):
        if isinstance(rhs, Vector3D):
            vector1 = math.sqrt(pow(self.x, 2) + pow(self.y, 2) + pow(self.z,
2))
            vector2 = math.sqrt(pow(rhs.x, 2) + pow(rhs.y, 2) + pow(rhs.z,
2))
            return vector1 > vector2 or vector1 < vector2
        else:
            raise ValueError


if __name__ == "__main__":
    v1 = Vector3D(7, 3, 4)
    v1.display()

    v2 = Vector3D()
    v2.read("Введите координаты: ")
    v2.display()

    v3 = v2.sum(v1)
    v3.display()

    v4 = v2.sub(v1)
    v4.display()

    print(f"Скалярное произведение векторов: {v2.dot(v1)}")
```

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/4.1/ind2.py
Координаты:  7, 3, 4
Введите координаты: 1 6 9
Координаты:  1, 6, 9
Координаты:  8, 9, 13
Координаты:  6, -3, -5
Скалярное произведение векторов: 61
```

ВОПРОСЫ

1. Как осуществляется объявление класса в языке Python?

Классы объявляются с помощью ключевого слова class и имени класса:class MyClass:

var = ... # некоторая переменнаяdef do_smt(self):

какой-то метод

2. Чем атрибуты класса отличаются от атрибутов экземпляра? Атрибут класса - это атрибут, общий для всех экземпляров класса.

Атрибуты класса определены внутри класса, но вне каких-либо методов. Ихзначения одинаковы для всех экземпляров этого класса. Так что вы можете рассматривать их как тип значений по умолчанию для всех наших объектов.

Атрибуты экземпляра определяются в методах и хранят информацию,специфичную для экземпляра.

3. Каково назначение методов класса?

Методы определяют функциональность объектов, принадлежащихконкретному классу.

4. Для чего предназначен метод__init__() класса?

Метод___init__является конструктором. Конструкторы - это концепцияобъектно- ориентированного программирования. Класс может иметь один и только один конструктор. Если__init__определен внутри класса, он автоматически вызывается при создании нового экземпляра класса. Метод
_init_указывает, какие атрибуты будут у экземпляров нашего класса.

5. Каково назначение self ?

Аргумент self представляет конкретный экземпляр класса и позволяет нам получить доступ к его атрибутам и методам. В примере __init_мы создаем атрибуты для конкретного экземпляра и присваиваем им значения аргументов метода. Важно использовать параметр self внутри метода, если мы хотим сохранить значения экземпляра для последующего использования.

В большинстве случаев нам также необходимо использовать параметр self в других методах, потому что при вызове метода первым аргументом, который ему передается, является сам объект.

6. Как добавить атрибуты в класс?

Новый атрибут класса указывается через точку после названия класса,затем ему присваивается определенное значение.

7. Как осуществляется управление доступом к методам и атрибутам вязыке Python?

Хорошим тоном считается, что для чтения/изменения какого-то

атрибута должны использоваться специальные методы, которые называютсяgetter/setter, их можно реализовать, но ничего не помешает изменить атрибутнапрямую. При этом есть соглашение, что метод или атрибут, который начинается с нижнего подчеркивания, является скрытым, и снаружи класса трогать его не нужно (хотя сделать это можно).

Если же атрибут или метод начинается с двух подчеркиваний, то тутнапрямую вы к нему уже не обратитесь (простым образом).

8. Каково назначение функции isinstance ?

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

```python
class Rational:
    def __init__(self, a=0, b=1):
        a = int(a)
        b = int(b)
        if b == 0:
            raise ValueError()
        self.__numerator = abs(a)
        self.__denominator = abs(b)
        self.__reduce()
        # Сокращение дроби

    def __reduce(self):
        # Функция для нахождения наибольшего общего делителя
        def gcd(a, b):
            if a == 0:
                return b
            elif b == 0:
                return a
            elif a >= b:
                return gcd(a % b, b)
```

if __name__ == '__main__'

ex6

```
Введите обыкновенную дробь: 5/7
5/7
41/28
1/28
15/28
20/21
```