

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«Северо-Кавказский федеральный университет»**

**Кафедра инфокоммуникаций**

**Отчет по лабораторной работе № 4.1  
«Элементы объектно-ориентированного программирования в языке  
Python»**

**по дисциплине «Объектно-ориентированное программирование»**

Выполнил студент группы ИВТ-б-о-20-1

Бобров Н.В. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

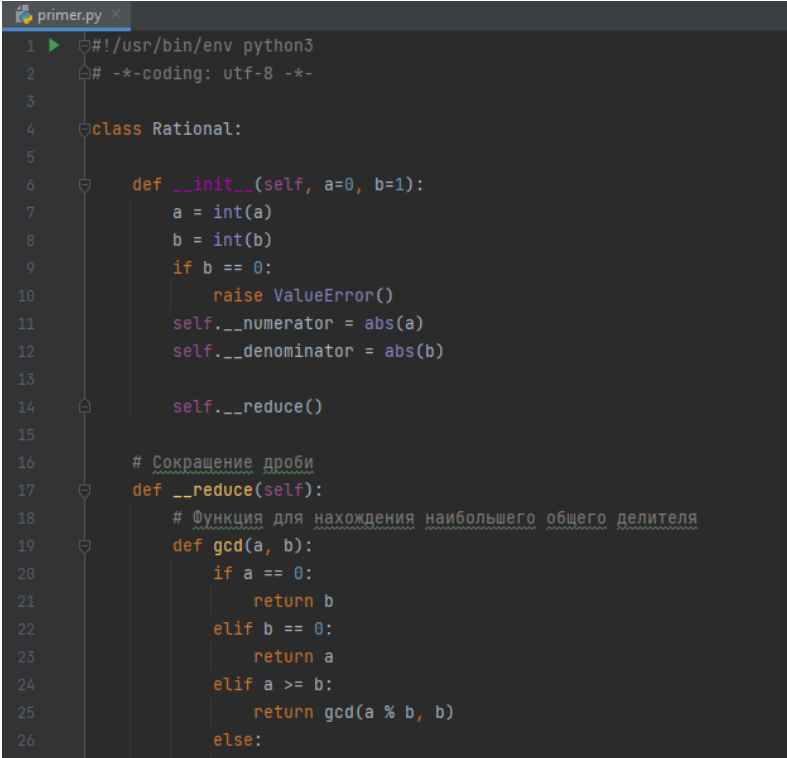
Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2022

**Цель работы:** приобретение навыков по работе с классами и объектами при написании программ с помощью языка программирования Python версии 3.x.

### Ход работы:

1. Создал общедоступный репозиторий на GitHub, клонировал его на локальный сервер.
2. Проработал пример лабораторной работы.



```
1  #!/usr/bin/env python3
2  # -*-coding: utf-8 -*-
3
4  class Rational:
5
6      def __init__(self, a=0, b=1):
7          a = int(a)
8          b = int(b)
9          if b == 0:
10             raise ValueError()
11             self.__numerator = abs(a)
12             self.__denominator = abs(b)
13
14             self.__reduce()
15
16     # Сокращение дроби
17     def __reduce(self):
18         # Функция для нахождения наибольшего общего делителя
19         def gcd(a, b):
20             if a == 0:
21                 return b
22             elif b == 0:
23                 return a
24             elif a >= b:
25                 return gcd(a % b, b)
26             else:
```

Рисунок 1 – Объявление метода `__init__`

```

56 # Вывести дробь на экран
57 def display(self):
58     print(f"{self.__numerator}/{self.__denominator}")
59
60 # Сложение обыкновенных дробей.
61 def add(self, rhs):
62     if isinstance(rhs, Rational):
63         a = self.numerator * rhs.denominator + \
64             self.denominator * rhs.numerator
65         b = self.denominator * rhs.denominator
66
67         return Rational(a, b)
68     else:
69         raise ValueError()
70
71 # Вычитание обыкновенных дробей.
72 def sub(self, rhs):
73     if isinstance(rhs, Rational):
74         a = self.numerator * rhs.denominator - \
75             self.denominator * rhs.numerator
76         b = self.denominator * rhs.denominator
77
78         return Rational(a, b)
79     else:
80         raise ValueError()

```

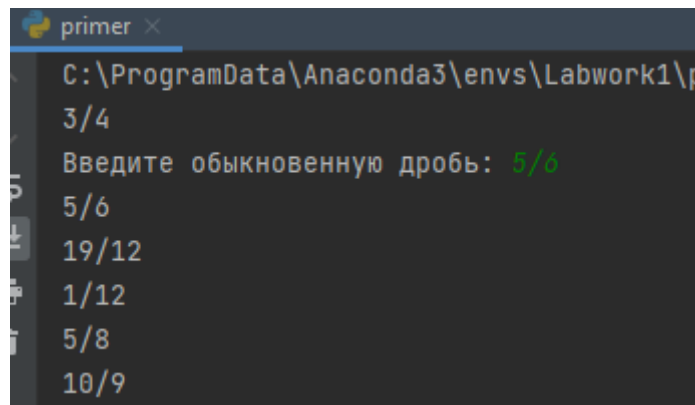
Рисунок 2 – Основной код программы

```

126 > if __name__ == "__main__":
127     r1 = Rational(3, 4)
128     r1.display()
129
130     r2 = Rational()
131     r2.read("Введите обыкновенную дробь: ")
132     r2.display()
133
134     r3 = r2.add(r1)
135     r3.display()
136
137     r4 = r2.sub(r1)
138     r4.display()
139
140     r5 = r2.mul(r1)
141     r5.display()
142
143     r6 = r2.div(r1)
144     r6.display()

```

Рисунок 3 – Демонстрация возможностей класса



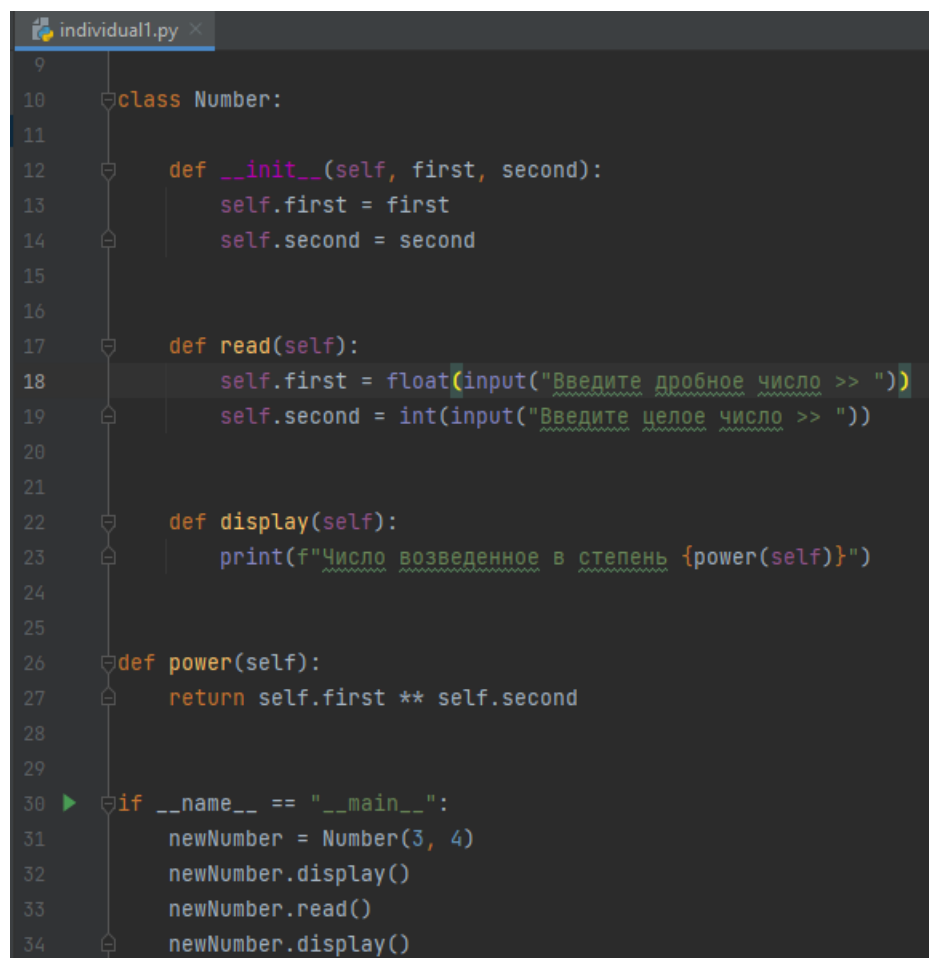
```
primer x
C:\ProgramData\Anaconda3\envs\Labwork1\p
3/4
Введите обыкновенную дробь: 5/6
5/6
19/12
1/12
5/8
10/9
```

Рисунок 4 – Вывод результатов

3. Приступил к выполнению индивидуальных заданий. Вариант 1.

**Задание 1.** Поле `first` — дробное число; поле `second` — целое число, показатель степени. Реализовать метод `power()` — возведение числа `first` в степень `second`. Метод должен правильно работать при любых допустимых значениях `first` и `second`.

1. Написал код для решения задачи.



```
individual1.py x
9
10 class Number:
11
12     def __init__(self, first, second):
13         self.first = first
14         self.second = second
15
16
17     def read(self):
18         self.first = float(input("Введите дробное число >> "))
19         self.second = int(input("Введите целое число >> "))
20
21
22     def display(self):
23         print(f"Число возведенное в степень {power(self)}")
24
25
26     def power(self):
27         return self.first ** self.second
28
29
30 if __name__ == "__main__":
31     newNumber = Number(3, 4)
32     newNumber.display()
33     newNumber.read()
34     newNumber.display()
```

Рисунок 5 – Код для решения задачи первого задания

```
individual1 x
C:\ProgramData\Anaconda3\envs\Labwork1\python.exe
Число возведенное в степень 81
Введите дробное число >> 4.6
Введите целое число >> 3
Число возведенное в степень 97.33599999999998

Process finished with exit code 0
```

Рисунок 6 – Вывод результата

2. Приступил к выполнению второго индивидуального задания.

**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.

1. Написал код для разработанного класса.

```
12 class Vector3D:
13
14     def __init__(self, x=0, y=0, z=0):
15         self.x = x
16         self.y = y
17         self.z = z
18
19
20     def read(self, prompt=None):
21         line = input() if prompt is None else input(prompt)
22         parts = list(map(int, line.split(' ', maxsplit=2)))
23         if parts[2] == 0:
24             raise ValueError()
25
26         self.x = parts[0]
27         self.y = parts[1]
28         self.z = parts[2]
29
30
31     def display(self):
32         print(f"Координаты - {self.x}, {self.y}, {self.z}")
33
```

Рисунок 7 – Реализация методов

```

34
35     # Сложение
36     def add(self, rhs):
37         if isinstance(rhs, Vector3D):
38             return Vector3D((self.x + rhs.x), (self.y + rhs.y), (self.z + rhs.z))
39         else:
40             raise ValueError
41
42
43     # Вычитание
44     def sub(self, rhs):
45         if isinstance(rhs, Vector3D):
46             return Vector3D((rhs.x - self.x), (rhs.y - self.y), (rhs.z - self.z))
47         else:
48             raise ValueError
49
50
51     # Скалярное произведение
52     def dot(self, rhs):
53         if isinstance(rhs, Vector3D):
54             return Vector3D((self.x * rhs.x) + (self.y * rhs.y) + (self.z * rhs.z))
55         else:
56             raise ValueError
57

```

Рисунок 8 – Операции над векторами

```

58
59     # Сравнение векторов
60     def equals(self, rhs):
61         if isinstance(rhs, Vector3D):
62             return (self.x == rhs.x) and (self.y == rhs.y) and (self.z == rhs.z)
63         else:
64             return False
65
66
67     def greater(self, rhs):
68         if isinstance(rhs, Vector3D):
69             vector1 = (self.x, self.y, self.z)
70             vector2 = (rhs.x, rhs.y, rhs.z)
71             return vector1 > vector2
72         else:
73             return False
74
75
76     def less(self, rhs):
77         if isinstance(rhs, Vector3D):
78             vector1 = (self.x, self.y, self.z)
79             vector2 = (rhs.x, rhs.y, rhs.z)
80             return vector1 < vector2
81         else:
82             return False
83

```

Рисунок 9 – Сравнение векторов

```

85     # Вычисление длины векторов
86     def length(self, rhs):
87         if isinstance(rhs, Vector3D):
88             vector1 = math.sqrt(pow(self.x, 2) + pow(self.y, 2) + pow(self.z, 2))
89             vector2 = math.sqrt(pow(rhs.x, 2) + pow(rhs.y, 2) + pow(rhs.z, 2))
90             return vector1 + vector2
91         else:
92             raise ValueError
93
94     # Сравнение длины векторов
95     def equal(self, rhs):
96         if isinstance(rhs, Vector3D):
97             vector1 = math.sqrt(pow(self.x, 2) + pow(self.y, 2) + pow(self.z, 2))
98             vector2 = math.sqrt(pow(rhs.x, 2) + pow(rhs.y, 2) + pow(rhs.z, 2))
99             return vector1 > vector2 or vector1 < vector2
100        else:
101            raise ValueError
102

```

Рисунок 10 – Вычисление длины векторов и их сравнение

2. Затем добавил демонстрацию возможностей созданного класса.

```

105  ▶ if __name__ == "__main__":
106      v1 = Vector3D(4, 1, 2)
107      v1.display()
108
109      v2 = Vector3D()
110      v2.read("Введите координаты: ")
111      v2.display()
112
113      v3 = v2.add(v1)
114      v3.display()
115
116      v4 = v2.sub(v1)
117      v4.display()
118
119      v5 = v2.dot(v1)
120      v5.display()
121

```

Рисунок 11 – Демонстрация возможностей класса Vector3D

### Контрольные вопросы:

1. Как осуществляется объявление класса в языке Python?

Классы объявляются с помощью ключевого слова `class` и имени класса

2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибуты класса определены внутри класса, но вне каких-либо методов. Их значения одинаковы для всех экземпляров этого класса. Так что вы можете рассматривать их как тип значений по умолчанию для всех наших объектов. Что касается переменных экземпляра, они хранят данные, уникальные для каждого объекта класса.

### 3. Каково назначение методов класса?

Методы определяют функциональность объектов, принадлежащих данному классу. Методы не являются независимыми, поскольку они определены внутри класса.

### 4. Для чего предназначен метод `__init__()` класса?

Метод `__init__` является конструктором. Конструкторы - это концепция объектно-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. Важно использовать параметр `self` внутри метода, если мы хотим сохранить значения экземпляра для последующего использования.

### 6. Как добавить атрибуты в класс?

Атрибуты экземпляра - это как раз те, которые мы определяем в методах, поэтому по определению мы можем создавать новые атрибуты внутри наших пользовательских методов.

На атрибуты данных класса могут ссылаться как методы, так и обычные пользователи - "клиенты" объекта.

### 7. Как осуществляется управление доступом к методам и атрибутам в языке Python?

В *Python* таких возможностей нет, и любой может обратиться к атрибутам и методам вашего класса, если возникнет такая необходимость. Это



существенный недостаток этого языка, т.к. нарушается один из ключевых принципов ООП – инкапсуляция.

Хорошим тоном считается, что для чтения/изменения какого-то атрибута должны использоваться специальные методы, которые называются *getter/setter*, их можно реализовать, но ничего не мешает изменить атрибут напрямую. 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).

#### 8. Каково назначение функции `isinstance`?

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

**Вывод:** в ходе выполнения лабораторной работы были приобретены навыки по работе с классами и объектами при написании программ с помощью языка программирования Python версии 3.x.