

**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  
Работа с переменными окружения в Python3  
по дисциплине «Объектно-ориентированное программирование»**

Выполнил студент группы ИВТ-б-о-20-1

Ищенко Т.С. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

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

Проверила Воронкин Р.А. \_\_\_\_\_

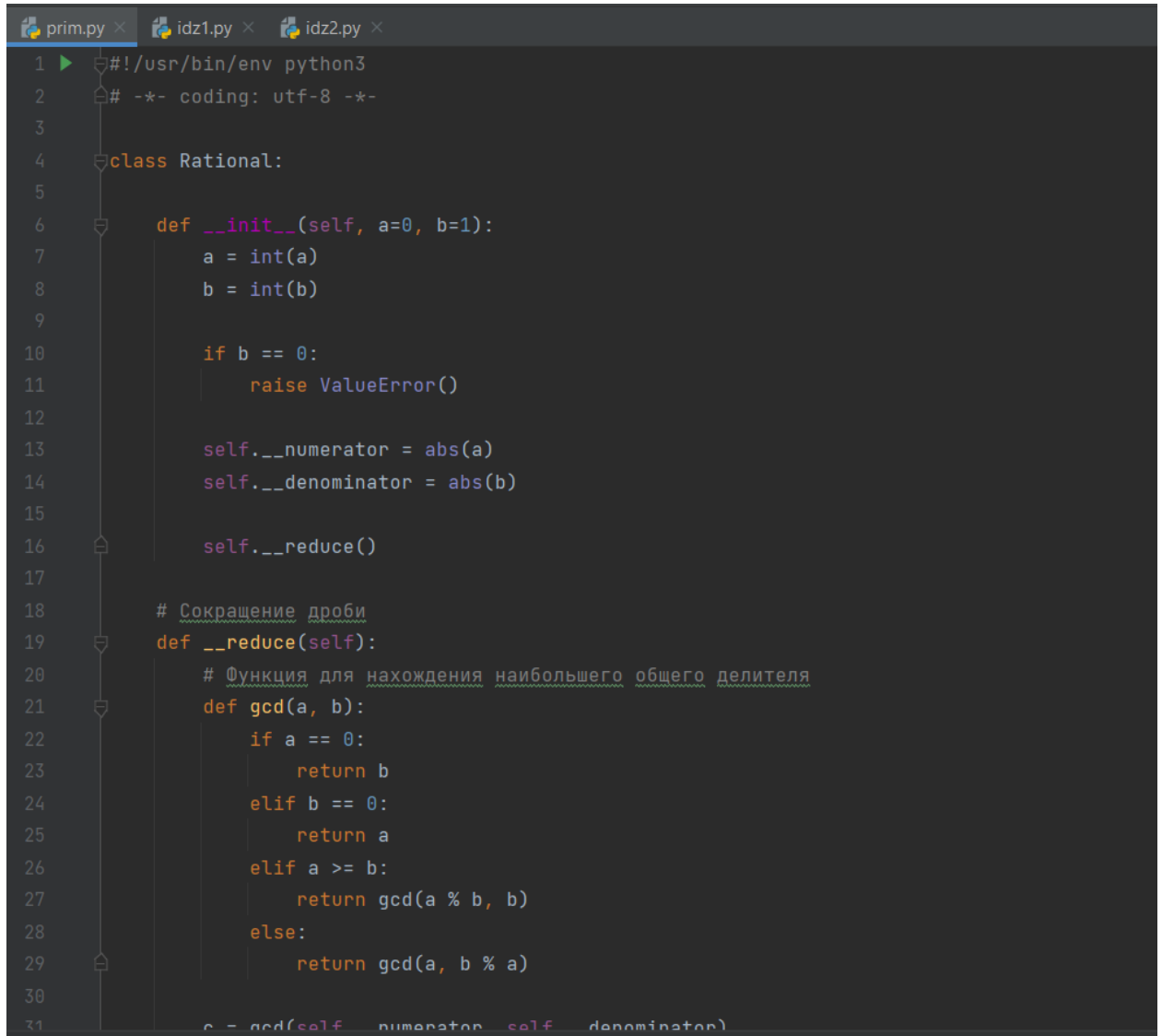
(подпись)

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:

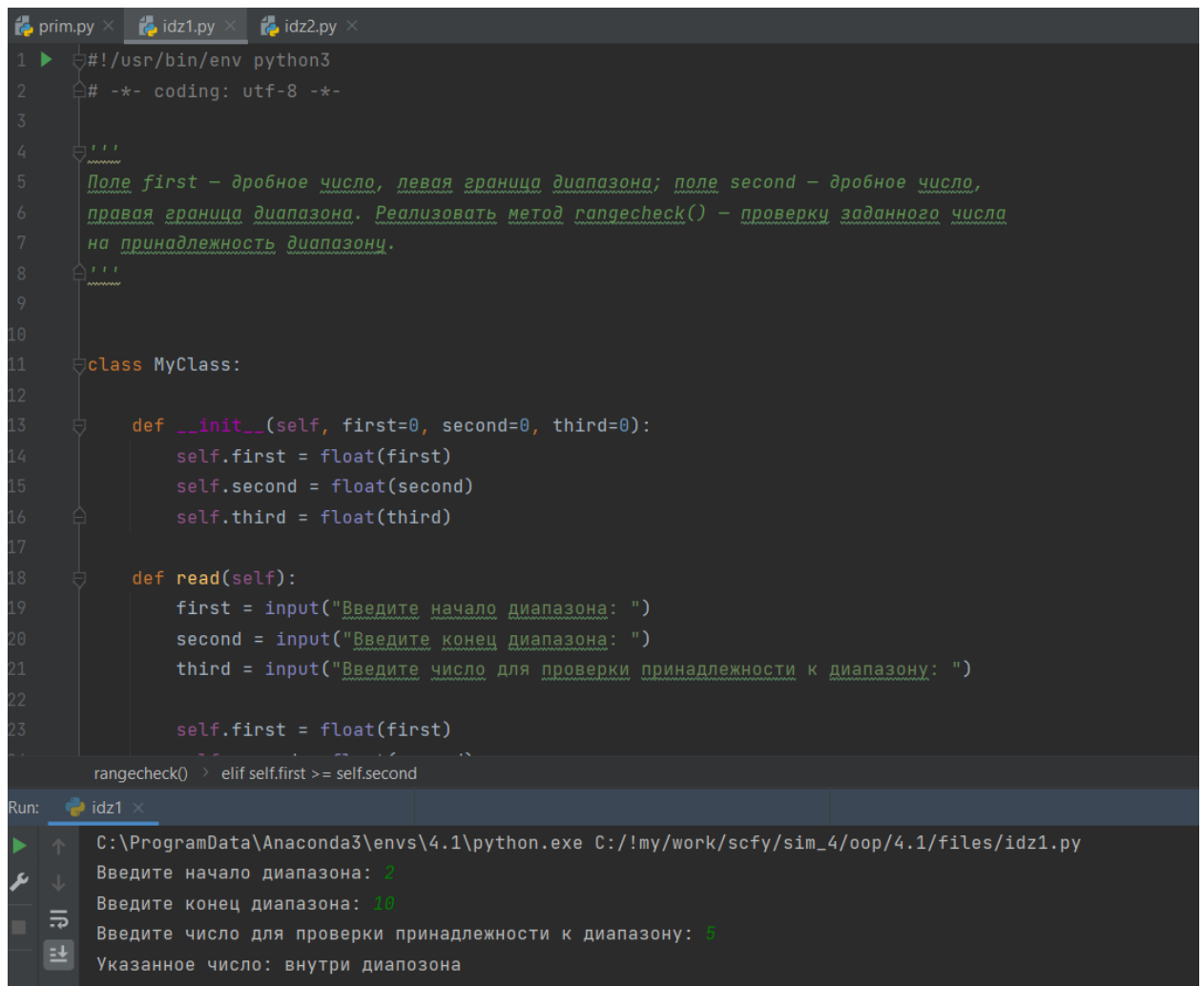
1. Произвёл отработку примера, согласно методическим рекомендациям



```
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
10         if b == 0:
11             raise ValueError()
12
13         self.__numerator = abs(a)
14         self.__denominator = abs(b)
15
16         self.__reduce()
17
18         # Сокращение дроби
19         def __reduce(self):
20             # Функция для нахождения наибольшего общего делителя
21             def gcd(a, b):
22                 if a == 0:
23                     return b
24                 elif b == 0:
25                     return a
26                 elif a >= b:
27                     return gcd(a % b, b)
28                 else:
29                     return gcd(a, b % a)
30
31             c = gcd(self.__numerator, self.__denominator)
```

Рисунок 1 – Результат выполнения примера

2. Произвёл выполнение индивидуального задания



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 '''
5 Поле first – дробное число, левая граница диапазона; поле second – дробное число,
6 правая граница диапазона. Реализовать метод rangecheck() – проверку заданного числа
7 на принадлежность диапазону.
8 '''
9
10
11 class MyClass:
12
13     def __init__(self, first=0, second=0, third=0):
14         self.first = float(first)
15         self.second = float(second)
16         self.third = float(third)
17
18     def read(self):
19         first = input("Введите начало диапазона: ")
20         second = input("Введите конец диапазона: ")
21         third = input("Введите число для проверки принадлежности к диапазону: ")
22
23         self.first = float(first)
```

rangecheck() > elif self.first >= self.second

Run: idz1 ×

C:\ProgramData\Anaconda3\envs\4.1\python.exe C:/!my/work/scfy/sim\_4/oop/4.1/files/idz1.py

Введите начало диапазона: 2

Введите конец диапазона: 10

Введите число для проверки принадлежности к диапазону: 5

Указанное число: внутри диапазона

Рисунок 2 – Результат выполнения индивидуального задания

3. Выполнил задание повышенной сложности, согласно методическим рекомендациям

```
prim.py x idz1.py x idz2.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  """
5  Создать класс Date для работы с датами в формате «год.месяц.день». Дата представляется
6  структурой с тремя полями типа unsigned int: для года, месяца и дня. Класс должен
7  включать не менее трех функций инициализации: числами, строкой вида «год.месяц.день»
8  (например, «2004.08.31») и датой. Обязательными операциями являются: вычисление даты
9  через заданное количество дней, вычитание заданного количества дней из даты,
10 определение високосности года, присвоение и получение отдельных частей (год, месяц,
11 день), сравнение дат (равно, до, после), вычисление количества дней между датами.
12 """
13
14
15 import datetime
16
17
18 class Date:
19
20     def __init__(self, year=None, month=None, day=None, str_date=None, number_of_days=1):
21         # date = datetime.date(year, month, day)
22         # str_date = datetime.datetime.strptime(str_date, "%Y-%m-%d")
23         self.number_of_days = int(number_of_days)
24         self.today_day = datetime.date.today()
25
26         if year == None and str_date != None:
27             self.first_date = datetime.datetime.strptime(str_date, "%Y-%m-%d").day
28         elif year != None and str_date == None:
29             self.first_date = datetime.date(year, month, day)
30
31         # self.second_date = datetime.datetime.strptime(second_date, "%Y-%m-%d")
```

Рисунок 3 – Результат выполнения задания повышенной сложности

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 конкретному классу.

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.

5. Каково назначение `self`? - Аргумент `self` представляет конкретный экземпляр класса и позволяет нам получить доступ к его атрибутам и методам. В примере с `__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.

6. Как добавить атрибуты в класс? - Атрибут класса - это атрибут, общий для всех экземпляров класса. Атрибуты экземпляра - это как раз те, которые мы определяем в методах, поэтому по определению мы можем создавать новые атрибуты внутри наших пользовательских методов.

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`? - В Python есть встроенная функция `instance ()`, которая сравнивает значение с указанным типом. Если данное значение и тип соответствуют, он вернет `true`, иначе `false`. Используя

`isinstance ()`, вы можете проверить строку, число с плавающей точкой, `int`, список, кортеж, `dict`, `set`, `class` и т.д.

Вывод: в ходе выполнения лабораторной работы приобрел навыки по работе с переменными окружения с помощью языка программирования Python версии 3.x.