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

Погорелов Д.Н « »_____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.
2. Проработал пример лабораторной работы.

```
# Вывести дробь на экран
def display(self):
    print(f"{self.__numerator}/{self.__denominator}")

# Сложение обыкновенных дробей.
def add(self, rhs):
    if isinstance(rhs, Rational):
        a = self.numerator * rhs.denominator + \
            self.denominator * rhs.numerator
        b = self.denominator * rhs.denominator

        return Rational(a, b)
    else:
        raise ValueError()

# Вычитание обыкновенных дробей.
def sub(self, rhs):
    if isinstance(rhs, Rational):
        a = self.numerator * rhs.denominator - \
            self.denominator * rhs.numerator
        b = self.denominator * rhs.denominator

        return Rational(a, b)
    else:
        raise ValueError()
```

Рисунок 1 – Основной код программы

```
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()
```

Рисунок 2 – Демонстрация возможностей класса

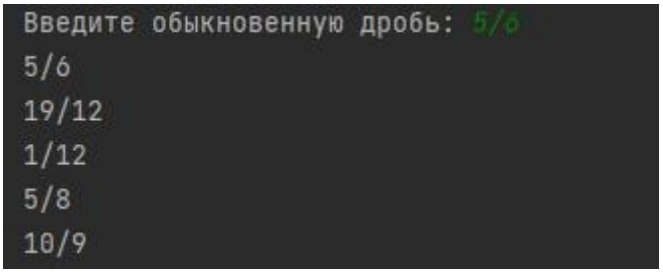

Рисунок 3 – Вывод результатов

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.

```
class Number:

    def __init__(self, first, second):
        self.first = first
        self.second = second


    def read(self):
        self.first = float(input("Введите дробное число >> "))
        self.second = int(input("Введите целое число >> "))


    def display(self):
        print(f"Число возведенное в степень {power(self)}")


def power(self):
    return self.first ** self.second


if __name__ == "__main__":
    newNumber = Number(3, 4)
    newNumber.display()
    newNumber.read()
    newNumber.display()
```

Рисунок 4 – Код для решения задачи первого задания

```
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 5 – Вывод результата

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.

```python
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}")
```

Рисунок 6 – Реализация методов

```python
# Сложение
def add(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 Vector3D((self.x * rhs.x) + (self.y * rhs.y) + (self.z * rhs.z))
    else:
        raise ValueError
```

Рисунок 7 – Операции над векторами

Рисунок 8 – Сравнение векторов

```python
# Сравнение векторов
def equals(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
```

```python
# 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
```

Рисунок 9 – Вычисление длины векторов и их сравнение

2. Затем добавил демонстрацию возможностей созданного класса.

```python
if __name__ == "__main__":
    v1 = Vector3D(4, 1, 2)
    v1.display()

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

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

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

    v5 = v2.dot(v1)
    v5.display()
```

Рисунок 10 – 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 помешает 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.