Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1
дисциплины
«Объектно-ориентированное программирование»
Вариант 15

Выполнил:
Степанов Леонид Викторович
3 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Богданов Сергей Сергеевич,
Ассистент департамента цифровых, робототехнических систем и электроники

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Элементы объектно-ориентированного программирования в языке Python

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

Порядок выполнения работы:

Создал файл (primer.py) в котором выполнил пример практической работы, в котором необходимо: создать класс Rational (рис.1) для работы с рациональными дробями. Обязательно должны быть реализованы операции: сложения add (рис.2), вычитания sub (рис.3), умножения mul (рис.4), деления div (рис.5), сравнения equal, greate, less, приватная функция сокращения дроби reduce (рис.6).

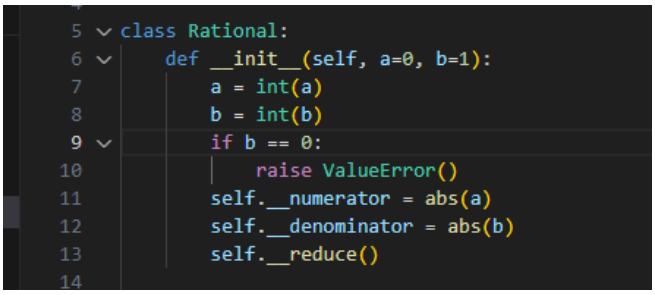

Рисунок 1 –Класс Rational

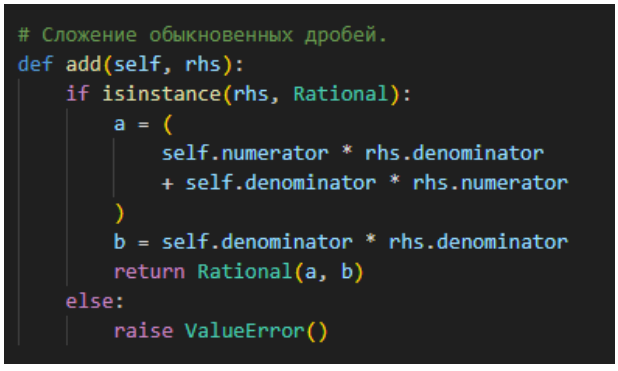

Рисунок 2 – Функция сложения

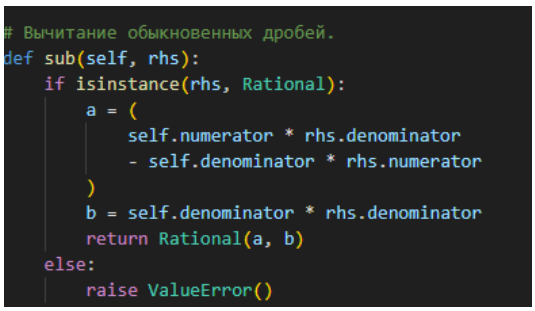

Рисунок 3 – Функция вычитания

```python
# Умножение обыкновенных дробей.
def mul(self, rhs):
    if isinstance(rhs, Rational):
        a = self.numerator * rhs.numerator
        b = self.denominator * rhs.denominator
        return Rational(a, b)
    else:
        raise ValueError()
```

Рисунок 4 – Функция умножения

```python
# Деление обыкновенных дробей.
def div(self, rhs):
    if isinstance(rhs, Rational):
        a = self.numerator * rhs.denominator
        b = self.denominator * rhs.numerator
        return Rational(a, b)
    else:
        raise ValueError()
```

Рисунок 5 – Функция деления

```python
# Отношение обыкновенных дробей.
def equals(self, rhs):
    if isinstance(rhs, Rational):
        return (self.numerator == rhs.numerator) an
            self.denominator == rhs.denominator
        )
    else:
        return False

def greater(self, rhs):
    if isinstance(rhs, Rational):
        v1 = self.numerator / self.denominator
        v2 = rhs.numerator / rhs.denominator
        return v1 > v2
    else:
        return False

def less(self, rhs):
    if isinstance(rhs, Rational):
        v1 = self.numerator / self.denominator
        v2 = rhs.numerator / rhs.denominator
        return v1 < v2
    else:
        return False
```

Рисунок 6 – Функции сравнения

```python
# 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)
        else:
            return gcd(a, b % a)

    c = gcd(self.__numerator, self.__denominator)
    self.__numerator //= c
    self.__denominator //= c
```

Рисунок 7 – Функция сокращения

Пример работы класса отображен на рис. 8

```python
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()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

2 files reformatted, 1 file left unchanged.

(poetry) D:\Python-labs\lab4.1>isort .
Skipped 1 files

(poetry) D:\Python-labs\lab4.1>python .\prog\primer.py
3/4
Введите обыкновенную дробь: 40/10
4/1
19/4
13/4
3/1
16/3
```

Рисунок 8 – Результат работы программы

Индивидуальное задание (Вариант 15):

Задание 1

Ответы на контрольные вопросы:

Парой называется класс с двумя полями, которые обычно име ют имена first и second. Требуется реализовать тип данных с помощью такого класса. Во всех заданиях обязательно должны присутствовать:

1) метод инициализации __init__ ; метод должен контролировать значения аргументов на корректность;

2) ввод с клавиатуры read ;

3) вывод на экран display .

Реализовать внешнюю функцию с именем make_тип() , где тип — тип реализуемой структуры. Функция должна получать в качестве аргументов значения для полей структуры и возвращать структуру требуемого типа. При передаче ошибочных параметров следует выводить сообщение и заканчивать работу.

Вариант 15. Поле first — целое положительное число, продолжительность телефонного разговора в минутах; поле second — дробное положительное число, стоимость одной минуты в рублях. Реализовать метод cost() — вычисление общей стоимости разговора.

Был реализован класс Conversation, в котором проверяется на корректность аргументы (рис.9), реализовано: ввод с клавиатуры, вывод данных на экран, вычисление общей стоимости на рис. 10

```
class Conversation:
    def __init__(self, first, second):
        """
        Инициализатор класса Conversation.
        first: продолжительность телефонного разговора в минутах (целое положительное число)
        second: стоимость одной минуты в рублях (дробное положительное число)
        """
        if not isinstance(first, int) or first <= 0:
            raise ValueError("Поле 'first' должно быть целым положительным числом.")
        if not isinstance(second, (int, float)) or second <= 0:
            raise ValueError("Поле 'second' должно быть положительным числом.")

        self.first = first
        self.second = second
```

Рисунок 9 – Класс Conversation

```
def cost(self):
    """Вычисление общей стоимости разговора"""
    return self.first * self.second

def display(self):
    """Вывод данных на экран"""
    print(f"Продолжительность разговора: {self.first} минут")
    print(f"Стоимость одной минуты: {self.second} рубля")
    print(f"Общая стоимость разговора: {self.cost()} рублей")

def read(self):
    """Ввод данных с клавиатуры"""
    try:
        self.first = int(input("Введите продолжительность разговора в минутах (целое положительное число): "))
        self.second = float(input("Введите стоимость одной минуты в рублях (положительное число): "))
    except ValueError as e:
        print(f"Ошибка ввода: {e}")
        return None
```

Рисунок 10 – Методы объекта

Также была реализована внешняя функция, которая make_conversation для создания объекта (рис. 11), а на рис. 12 представлен пример работы.

```python
def make_Conversation(first, second):
    """
    Внешняя функция для создания объекта Conversation.
    Проверяет корректность переданных параметров.
    """
    try:
        return Conversation(first, second)
    except ValueError as e:
        print(f"Ошибка создания объекта: {e}")
        return None
```

Рисунок 11 – Создание объекта функцией

```python
# Пример использования
if __name__ == "__main__":
    # Создание объекта через функцию make_Conversation
    conv = make_Conversation(10, 2.5)
    if conv:
        conv.display()
        conv.read()
        conv.display()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

(poetry) D:\Python-labs\lab4.1>python .\prog\idz1.py
Продолжительность разговора: 10 минут
Стоимость одной минуты: 2.5 рубля
Общая стоимость разговора: 25.0 рублей
Введите продолжительность разговора в минутах (целое положительное число): 25
Введите стоимость одной минуты в рублях (положительное число): 6
Продолжительность разговора: 25 минут
Стоимость одной минуты: 6.0 рубля
Общая стоимость разговора: 150.0 рублей
150.0

(poetry) D:\Python-labs\lab4.1>
```

Рисунок 12 – Результат выполнения работы

2 Задание.

Реализовать класс Cursor. Полями являются координаты курсора по горизонтали и вертикали — целые положительные числа, вид курсора — горизонтальный или вертикальный, размер курсора — целое от 1 до 15 Реализовать методы изменения координат курсора, изменения вида курсора, изменения размера курсора, метод гашения и восстановления курсора.

Класс Cursor и его метод инициализации:

```python
class Cursor:
    def __init__(self, x=1, y=1, view="horizontal", size=1):
        if not (1 <= x <= 100) or not (1 <= y <= 100):
            raise ValueError("Координаты - целыми числами от 1 до 100.")
        if view not in ["horizontal", "vertical"]:
            raise ValueError(
                "Вид курсора должен быть 'horizontal' или 'vertical'."
            )
        if not (1 <= size <= 15):
            raise ValueError(
                "Размер курсора должен быть целым числом от 1 до 15."
            )

        self.x = x  # Координата по горизонтали
        self.y = y  # Координата по вертикали
        self.view = view  # Вид курсора
        self.size = size  # Размер курсора
        self.visible = True  # Флаг видимости курсора
```

Рисунок 13 – Класс Cursor

Методы Cursor

```python
    def read(self):
        self.x = int(input("Введите координату по горизонтали (1-100): "))
        self.y = int(input("Введите координату по вертикали (1-100): "))
        self.view = input(
            "Введите вид курсора ('horizontal' или 'vertical'): "
        )
        self.size = int(input("Введите размер курсора (1-15): "))

    def display(self):
        if self.visible:
            print(
                f"Курсор на координатах ({self.x}, {self.y}), "
                f"вид: {self.view}, размер: {self.size}."
            )
        else:
            print("Курсор скрыт.")

    def move(self, delta_x, delta_y):
        self.x += delta_x
        self.y += delta_y
        print(f"Курсор перемещён на ({delta_x}, {delta_y}).")

    def change_view(self, new_view):
        if new_view in ["horizontal", "vertical"]:
            self.view = new_view
            print(f"Вид курсора изменён на: {self.view}.")
        else:
            print("Некорректный вид курсора.")

    def change_size(self, new_size):
        if 1 <= new_size <= 15:
            self.size = new_size
            print(f"Размер курсора изменён на: {self.size}.")
        else:
            print("Некорректный размер курсора.")

    def hide(self):
        self.visible = False
        print("Курсор скрыт.")

    def show(self):
        self.visible = True
        print("Курсор восстановлен.")
```

Рисунок 14 – Методы

Пример работы курсора:

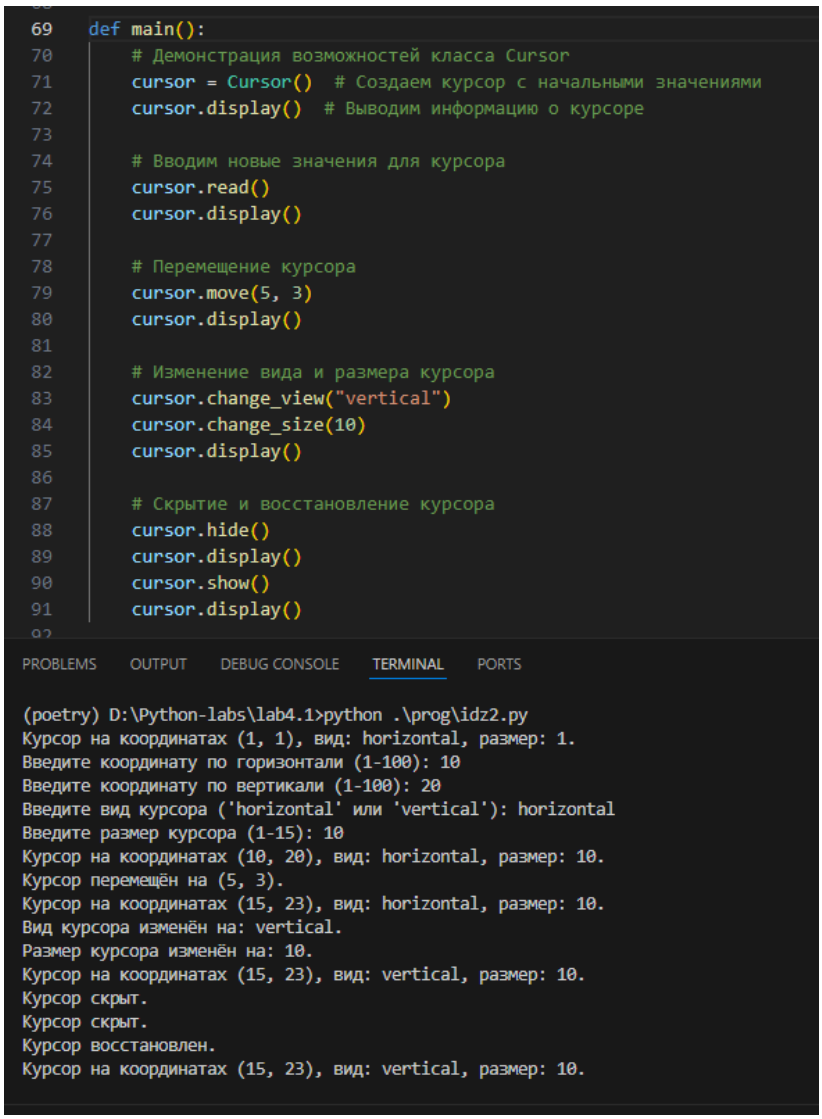

Рисунок 9 – Результат работы программы

Ссылка на gihub лабораторной работы: https://github.com/FiaLDI/lab4.1

Контрольные вопросы

1. Как осуществляется объявление класса в языке Python?

Объявление класса в Python осуществляется с помощью ключевого слова class, за которым следует имя класса и двоеточие.

2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибуты класса принадлежат самому классу и общие для всех его экземпляров, тогда как атрибуты экземпляра уникальны для каждого объекта (экземпляра) класса.

3. Каково назначение методов класса?

Методы класса предназначены для выполнения операций, связанных с классом или его экземплярами. Они могут изменять состояние объекта, выполнять вычисления или взаимодействовать с другими объектами.

4. Для чего предназначен метод __init__() класса?

Метод __init__() является конструктором класса и вызывается при создании нового экземпляра. Он используется для инициализации атрибутов объекта и выполнения любых необходимых начальных настроек.

5. Каково назначение self?

self — это ссылка на текущий экземпляр класса. Она используется для доступа к атрибутам и методам объекта внутри класса. Это позволяет различать атрибуты экземпляра от локальных переменных.

6. Как добавить атрибуты в класс?

Атрибуты можно добавлять в класс как в теле класса (атрибуты класса), так и в методе __init__() (атрибуты экземпляра).

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

В Python управление доступом осуществляется с помощью соглашений об именах. Атрибуты и методы, начинающиеся с одного подчеркивания (`_`), считаются защищенными, а начинающиеся с двух подчеркиваний (`__`) — приватными. Однако это не является строгим ограничением, а лишь соглашением.

8. Каково назначение функции isinstance?

Функция isinstance() используется для проверки, является ли объект экземпляром определенного класса или его подкласса. Это позволяет безопасно проверять типы объектов перед выполнением операций с ними. Например:

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