

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Объектно-ориентированное программирование

Отчет по лабораторной работе 4.1

**Элементы объектно-ориентированного
программирования**

Элементы объектно-ориентированного программирования в языке Python.

Выполнил студент группы

ИВТ-б-о-21-1

Богдашов Артём . «»_____20__г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил доцент

Кафедры инфокоммуникаций, старший
преподаватель

Воронкин Р.А.

(подпись)

Ставрополь 2023

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

Проработка примера.

Код программы:

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

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)
            else:
                return gcd(a, b % a)

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

        self.__numerator //= c
        self.__denominator //= c

    @property
    def numerator(self):
        return self.__numerator

    @property
    def denominator(self):
        return self.__denominator

    # прочитать значение дроби с клавиатуры. дробь вводится как a/b
    def read(self, prompt=None):
        line = input() if prompt is None else input(prompt)
        parts = list(map(int, line.split('/', maxsplit=1)))
        if parts[1] == 0:
            raise ValueError()
        self.__numerator = abs(parts[0])
        self.__denominator = abs(parts[1])
        self.__reduce()

    # 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()

# 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()

# 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()

# Отношение обыкновенных дробей.
def equals(self, rhs):
    if isinstance(rhs, Rational):
        return (self.numerator == rhs.numerator) and \
            (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

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()
```

Результат выполнения программы:

```
3/4
Введите обыкновенную дробь: 6/8
3/4
3/2
0/1
9/16
1/1
```

Рисунок 1. Результат работы программы

Выполнение задания.

Код программы:

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

class Pair:
    def __init__(self, first, second):
        if not isinstance(first, int) or not isinstance(second, int):
            raise TypeError("Оба поля (first и second) должны быть целыми числами")

        if second == 0:
            raise ValueError("Знаменатель (second) не может быть равен нулю")

        self.first = first
        self.second = second

    def ipart(self):
        """
        Возвращает целую часть от деления числителя на знаменатель.
        """
        if self.second != 0:
            return self.first // self.second
        else:
            raise ValueError("Невозможно вычислить целую часть при нулевом знаменателе")

    def display(self):
        print(f"({self.first}/{self.second})")

    @classmethod
    def read(cls):
        a = int(input("Введите числитель: "))
        b = int(input("Введите знаменатель: "))
```

```
        return cls(a, b)

def make_pair(first, second):
    try:
        return Pair(first, second)
    except (TypeError, ValueError) as e:
        print(f"Ошибка создания объекта Pair: {e}")
        exit()

if __name__ == "__main__":
    pair = Pair.read()
    pair.display()
    print("Целая часть:", pair.ipart())
```

Результат выполнения программы:

```
Введите числитель: 672  
Введите знаменатель: 2  
(672/2)  
Целая часть: 336  
  
Process finished with exit code 0
```

Рисунок 2. Результат выполнения

Второе задание.

Код программы:

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

class Money:
    def __init__(self, rubles=0, kopecks=0):
        self.rubles, self.kopecks = rubles, kopecks

    def read(self):
        self.rubles, self.kopecks = map(int, input("Введите количество рублей и копеек через пробел: ").split())

    def display(self):
        print(f"{self.rubles} руб. {int(self.kopecks):02d} коп.")

    def add(self, other):
        result = Money()
        total_kopecks = self.rubles * 100 + self.kopecks + other.rubles * 100 + other.kopecks
        result.rubles, result.kopecks = divmod(total_kopecks, 100)
        return result

    def subtract(self, other):
        result = Money()
        total_kopecks = self.rubles * 100 + self.kopecks - (other.rubles * 100 + other.kopecks)
        result.rubles, result.kopecks = divmod(total_kopecks, 100)
        return result

    def divide_sum(self, num):
        result = Money()
        total_kopecks = self.rubles * 100 + self.kopecks
        result_kopecks = total_kopecks / num
        result.rubles, result.kopecks = divmod(int(result_kopecks), 100)
        return result

    def divide_by_number(self, num):
        result = Money()
        total_kopecks = self.rubles * 100 + self.kopecks
        result_kopecks = total_kopecks / num
        result.rubles, result.kopecks = divmod(int(result_kopecks), 100)
        return result

    def multiply_by_number(self, num):
        result = Money()
        total_kopecks = self.rubles * 100 + self.kopecks
        result_kopecks = total_kopecks * num
        result.rubles, result.kopecks = divmod(result_kopecks, 100)
        return result

    def compare(self, other):
        return (self.rubles == other.rubles) and (self.kopecks == other.kopecks)

    def is_less_than(self, other):
        return (self.rubles * 100 + self.kopecks) < (other.rubles * 100 + other.kopecks)

if __name__ == '__main__':
    money1 = Money()
    money1.read()
    money1.display()
```

```
money2 = Money()
money2.read()
money2.display()

sum_result = money1.add(money2)
print("Сумма:")
sum_result.display()

diff_result = money1.subtract(money2)
print("Разность:")
diff_result.display()

divide_sum_num = float(input("Введите число для деления суммы: "))
div_sum_result = money1.divide_sum(divide_sum_num)
print("Деление суммы на число:")
div_sum_result.display()

divide_by_num = float(input("Введите число для деления: "))
div_result = money1.divide_by_number(divide_by_num)
print("Деление на число:")
div_result.display()

multiply_by_num = float(input("Введите число для умножения: "))
mul_result = money1.multiply_by_number(multiply_by_num)
print("Умножение на число:")
mul_result.display()

comparison_result = money1.compare(money2)
print(f"Сравнение: {comparison_result}")

comparison_result_lt = money1.is_less_than(money2)
print(f"Сравнение меньше: {comparison_result_lt}")
```


Результат выполнения программы:

```
Введите количество рублей и копеек через пробел: 125 05
125 руб. 05 коп.
Введите количество рублей и копеек через пробел: 25 5
25 руб. 05 коп.
Сумма:
150 руб. 10 коп.
Разность:
100 руб. 00 коп.
Введите число для деления суммы: 2
Деление суммы на число:
62 руб. 52 коп.
Введите число для деления: 2
Деление на число:
62 руб. 52 коп.
Введите число для умножения: 2
Умножение на число:
250.0 руб. 10 коп.
Сравнение: False
Сравнение меньше: False
```

Рисунок 3. Результат работы программы

Контрольные вопросы:

1. Как осуществляется объявление класса в языке Python?

Для создания класса в Python используется инструкция `class`. Она сильно похожа на объявление функций `def` и так же, как и `def`, `class` создаёт объект.

Инструкция `class` имеет следующий синтаксис:

```
**class <Name> ([<Superclass1>], [<Superclass2>]):  
  
<name declarations>**
```

2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибуты экземпляра и класса отличаются способом получения доступа к ним. Другими словами, речь идет об использовании названия класса и использовании названия экземпляра. С другой стороны, глобальные и локальные переменные отличаются своими областями видимости, другими словами, местами, где к ним может быть получен доступ.

3. Каково назначение методов класса?

Методы определяют набор действий, которые доступны классу (часто говорят, что они определяют поведение класса). Метод описывается один раз, а может вызываться для различных объектов класса столько раз, сколько необходимо. Общий формат записи методов класса имеет следующий вид:

[атрибуты] [спецификаторы] тип метода имя метода
([параметры]) {тело метода}.

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

Метод `__init__` в определении класса позволяет нам инициализировать атрибуты или переменные экземпляра для всех экземпляров класса. Метод `__init__` вызывается каждый раз, когда создается новый экземпляр класса. Цель наличия нескольких методов `__init__` в классе Python – предоставить несколько конструкторов для создания объектов.

5. Каково назначение `self` ?

Ключевое слово `self` в Python используется для ссылки на

текущий экземпляр объекта класса. Оно обычно используется в методах класса, чтобы обращаться к атрибутам и методам этого объекта. Когда мы вызываем метод объекта класса, Python автоматически передает ссылку на этот объект в качестве первого аргумента метода, который мы обычно называем `self`. Таким образом, мы можем обращаться к атрибутам и методам объекта через `self`, как в примере выше, где мы сохраняем имя объекта в атрибуте `name` и выводим его через метод `say_hello`.

6. Как добавить атрибуты в класс?

Атрибуты могут быть добавлены в класс путем определения их внутри класса.

Например:

```
class MyClass:

    def __init__(self, attribute1, attribute2):

        self.attribute1 = attribute1

        self.attribute2 = attribute2
```

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

Управление доступом к методам и атрибутам в языке Python осуществляется с помощью модификаторов доступа. В Python есть три уровня доступа: `public`, `protected` и `private`.

8. Каково назначение функции `isinstance`?

Функция `isinstance` в языке Python используется для проверки принадлежности объекта определенному классу. Она принимает два аргумента: объект и класс, и возвращает `True`, если объект принадлежит к указанному классу или его наследникам, и `False` в противном случае. Функция `isinstance` может быть полезна, например, при проверке типов аргументов функции или при обработке объектов разных классов в цикле

Вывод: в ходе работы были приобретены навыки работы с классами и объектами с помощью языка программирования Python версии 3.x.