МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе №1

Элементы объектно-ориентированного программирования в языке Python.

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

Выполнила студентка группы ИВТ-б-о-20-1

Новикова В.С. « »_____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 3x.

Ход работы:

Ссылка на репозиторий: https://github.com/Valentina1502/OOP_1

Пример 1. (рис. 1).

Рациональная (несократимая) дробь представляется парой целых чисел (a, b), где a — числитель, b — знаменатель. Создать класс Rational для работы с рациональными дробями. Обязательно должны быть реализованы операции

Код:

```
#!/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()
```

```
C:\Users\Valentina\AppData\Local\Programs\Python\Python38\
3/4
Введите обыкновенную дробь: 5/6
5/6
19/12
1/12
5/8
10/9

Process finished with exit code 0
```

Рисунок 1 – Пример 1

Задание 1 (рис. 2):

Линейное уравнение y = Ax + B. Поле first — дробное число, коэффициент A; поле second — дробное число, коэффициент B. Реализовать метод root() — вычисление корня линейного уравнения. Метод должен проверять неравенство коэффициента B нулю.

Код:

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


'''
Линейное уравнение y = Ax + B. Поле first — дробное число, коэффициент A;
поле second — дробное число, коэффициент B. Реализовать метод
root() — вычисление корня линейного уравнения. Метод должен
```

```
проверять неравенство коэффициента B нулю.
'''


class Korny:

    def __init__(self):
        self.first = 0
        self.second = 0
        self.third = 0

    def root(self):
        x = (float(self.third) - float(self.second)) / float(self.first)
        print(f"   Корень  =  {x:.2f}")

    def read(self):
        self.first, self.second, self.third = \
            input("Введите коэффициенты A B и Y ").split()
        if float(self.first) != 0 and float(self.second) != 0:
            y.root()
        else:
            print("Коэффициент не может быть равен 0")


if __name__ == '__main__':
    y = Korny()
    y.read()
```

```
C:\Users\Valentina\AppData\Local\Programs\Python\
Введите коэффициенты A B и Y 5.2 10.1 36.1
   Корень  =  5.00
```

Рисунок 2 – Задание 1

Задание 2. Вариант 11 (рис. 3):

Реализовать класс Bankomat, моделирующий работу банкомата. В классе должны содержаться поля для хранения идентификационного номера банкомата, информации о текущей сумме денег, оставшейся в банкомате, минимальной и максимальной суммах, которые позволяется снять клиенту в один день. Реализовать метод инициализации банкомата, метод загрузки купюр в банкомат и метод снятия определенной суммы денег. Метод снятия денег должен выполнять проверку на корректность снимаемой суммы: она не должна быть меньше минимального значения и не должна превышать максимальное значение.

Код:

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


'''
Реализовать класс Bankomat, моделирующий работу банкомата. В классе должны
содержаться поля для хранения идентификационного номера банкомата, информации
о
текущей сумме денег, оставшейся в банкомате, минимальной и максимальной
суммах,
которые позволяется снять клиенту в один день.
```

```
Реализовать метод инициализации банкомата, метод
загрузки купюр в банкомат и метод снятия определенной суммы денег. Метод
снятия денег
должен выполнять проверку на корректность снимаемой суммы: она не должна быть
меньше минимального значения и не должна превышать максимальное значение
'''


class Bankomat:

    def __init__(self):
        self.id_bankomata = 0
        self.ostatok_deneg = 0
        self.min = 0
        self.max = 0

    def add_bankomat(self):
        self.id_bankomata = input("ID банкамата: ")
        self.ostatok_deneg = input("Денег в банкомате: ")
        self.min = input("Минимальная сумма для снятия: ")
        self.max = input("Максимальная сумма для снятия: ")
        if self.max >= self.ostatok_deneg:
            self.max = self.ostatok_deneg
        with open(f"{self.id_bankomata}.txt", "w") as file:
            file.write(
                "ID банкамата: " + self.id_bankomata + "\n"
            )
        with open(f"{self.id_bankomata}.txt", "a") as file:
            file.write(
                "Остаток денег: \n" + self.ostatok_deneg + "\n"
            )
        with open(f"{self.id_bankomata}.txt", "a") as file:
            file.write(
                "Минимальная сумма для снятия: \n" + self.min + "\n"
            )
        with open(f"{self.id_bankomata}.txt", "a") as file:
            file.write(
                "Максимальная сумма для снятия: \n" + self.max + "\n"
            )

    def add_money(self, id_b):
        self.id_bankomata = id_b
        add_m = int(input("Внесите сумму денег для зачисления: "))
        with open(f"{self.id_bankomata}.txt", "r") as file:
            l_str = file.readlines()
        with open(f"{self.id_bankomata}.txt", "w") as file:
            ost = int(l_str[2])
            l_str[2] = add_m + ost
            l_str[2] = str(l_str[2]) + '\n'
            file.writelines(l_str)
            print("Деньги внесены на счет ")

    def see_bankomat(self, id_b):
        self.id_bankomata = id_b
        with open(f"{self.id_bankomata}.txt", "r") as file:
            for i in file:
                print(i)

    def withdraw_money(self, id_b):
        self.id_bankomata = id_b
        take_m = int(input("Введите сумму, которую хотите снять: "))
        if take_m % 100:
            print("Такую сумму снять невозможно")
```

```
        else:
            with open(f"{self.id_bankomata}.txt", "r") as file:
                l_str = file.readlines()
            with open(f"{self.id_bankomata}.txt", "r+") as file:
                min = int(l_str[4])
                max = int(l_str[6])
            if take_m < min or take_m > max:
                print("Такую сумму снять невозможно")
            else:
                with open(f"{self.id_bankomata}.txt", "r") as file:
                    l_str = file.readlines()
                with open(f"{self.id_bankomata}.txt", "r+") as file:
                    ost = int(l_str[2]) - take_m
                    l_str[2] = str(ost) + '\n'
                    if int(l_str[6]) > int(ost):
                        l_str[6] = l_str[2]
                    file.writelines(l_str)
                    print("Деньги сняты со счета ")


def choice():
    while True:
        command = input("___: ").lower()
        if command == 'exit':
            break
        elif command == 'add_b':
            y.add_bankomat()
        elif command.startswith('see_b '):
            parts = command.split(' ', maxsplit=1)
            id_bankomata = (parts[1])
            y.see_bankomat(id_bankomata)
        elif command.startswith('add_m '):
            parts = command.split(' ', maxsplit=1)
            id_bankomata = (parts[1])
            y.add_money(id_bankomata)
        elif command.startswith('withdraw_m '):
            parts = command.split(' ', maxsplit=1)
            id_bankomata = (parts[1])
            y.withdraw_money(id_bankomata)
        elif command == 'help':
            print("Список команд:\n")
            print("add_b - добавить банкомат;")
            print("see_b `ID банкомата` - посмотреть банкомат;")
            print("add_m `ID банкомата` - внести денги на счет;")
            print("withdraw_m `ID банкомата`- снять деньги со счета;")
            print("help - отобразить справку;")
            print("exit - завершить работу.")
        else:
            print(f"Введена неизвестная команда {command}")


if __name__ == '__main__':
    y = Bankomat()
    choice()
```

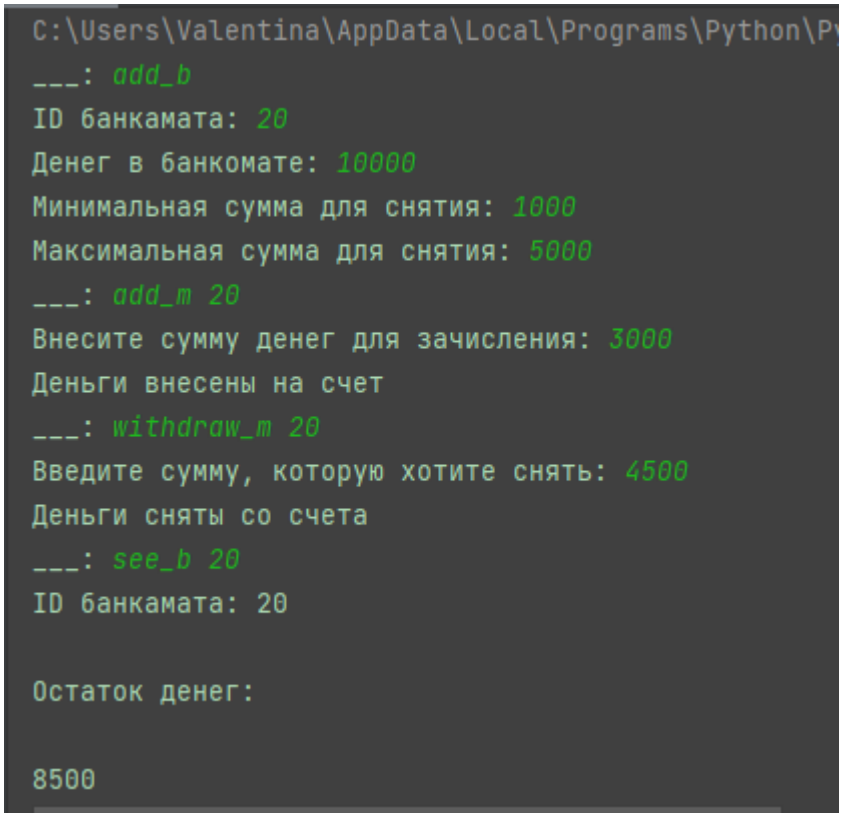

Рисунок 3 – Задание 2

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__ позволяет принимать аргументы для вашего класса. Что еще более важно, метод __init__ дает возможность назначать начальные значения различным атрибутам экземпляров класса.

5. Каково назначение self ?

Self - это обращение к самому экземпляру класса.

6. Как добавить атрибуты в класс?

“Имя объекта”.”Название атрибута” = “Значение атрибута”

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

Для чтения/изменения какого-то атрибута должны использоваться специальные методы, которые называются getter/setter.

8. Каково назначение функции isinstance ?

Функция isinstance() вернет True, если проверяемый объект object является экземпляром указанного класса (классов) или его подкласса (прямого, косвенного или виртуального).

Если объект object не является экземпляром данного типа, то функция всегда возвращает False.

Вывод: при выполнении практических заданий были приобретены простейшие навыки по работе с классами, экземплярами, методами и свойствами в языке программирования Python.