МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет по лабораторной работе №13

на тему: «Элементы объектно-ориентированного программирования в языке Python»

Дисциплина «Введение в системы искусственного интеллекта»

Выполнил: студент группы ИВТ-б-о-18-1 (1)
Болотов А.В.
_____(подпись)
Проверил: доцент кафедры инфокоммуникаций
Воронкин Роман Александрович
_____(подпись)

Ставрополь, 2022 г.

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

Ход работы

Таблица 1 – Исходные данные

Номер варианта	1

Задание 1:

Линейное уравнение $y = Ax + B$. Поле first — дробное число, коэффициент A; поле second — дробное число, коэффициент B. Реализовать метод `function()` — вычисление для заданного x значения функции y.

Решение:

Ввод [1]:
```
# Класс, содержащий метод function и
# вспомогательные методы для взаимодействия
# с пользователем.

class Task:

    # Метод для инициализации класса
    # Присваивает полям класса значения по умолчанию
    def __init__(self, first=0, second=0, x=None):

        self.first = first
        self.second = second
        self.x = x

    # Метод для чтения параметров функции из консоли
    def read(self):

        print('Введите коэффициенты уравнения y=Ax+B:\n')
        self.first = float(input('Введите коэффициент A: '))
        self.second = float(input('Введите коэффициент B: '))

        self.x = float(input('Введите значение x: '))

    # Метод, вычисляющий значение функции
    # y = Ax + B в точке x
    def function(self):

        if self.x is None:
            print('Значение x не задано. Воспользуйтесь методом read')
        else:
            y = self.first * self.x + self.second
            print(y)


if __name__ == '__main__':

    # Создаем экземпляр класса
    t = Task()
    # Считываем параметры функции
    t.read()
    # Вычисляем значение функции
    t.function()
```

```
Введите коэффициенты уравнения y=Ax+B:

Введите коэффициент A: 5
Введите коэффициент B: 9
Введите значение x: 4
29.0
```

Рисунок 1 – Решение задачи 1 с результатом

Задание 2:

Номиналы российских рублей могут принимать значения 1, 2, 5, 10, 50, 100, 500, 1000, 5000. Копейки представить как 0.01 (1 копейка), 0.05 (5 копеек), 0.1 (10 копеек), 0.5 (50 копеек). Создать класс Money для работы с денежными суммами. Сумма должна быть представлена полями-номиналами, значениями которых должно быть количество купюр данного достоинства. Реализовать сложение сумм, вычитание сумм, деление сумм, деление суммы на дробное число, умножение на дробное число и операции сравнения. Дробная часть (копейки) при выводе на экран должны быть отделена от целой части запятой.

Решение:

Ввод [2]:

```
# Класс для работы с денежными суммами
class Money:

    # Инициализируем начальную сумму.
    # kx - копейки, rx - рубли
    def __init__(self,
                 k1=0, k5=0, k10=0, k50=0,
                 r1=0, r2=0, r5=0, r10=0,
                 r50=0, r100=0, r500=0,
                 r1000=0, r5000=0):

        # Инициализируем копейки
        self.k1 = k1
        self.k5 = k5
        self.k10 = k10
        self.k50 = k50

        # Инициализируем рубли
        self.r1 = r1
        self.r2 = r2
        self.r5 = r5
        self.r10 = r10
        self.r50 = r50
        self.r100 = r100
        self.r500 = r500
        self.r1000 = r1000
        self.r5000 = r5000

        # Кортеж, в котором каждому номиналу
        # соответствует его эквивалент в копейках
        self.coefs = (('k1', 1), ('k5', 5), ('k10', 10), ('k50', 50),
                      ('r1', 100), ('r2', 200), ('r5', 500), ('r10', 1000),
                      ('r50', 5000), ('r100', 10000), ('r500', 50000),
                      ('r1000', 100000), ('r5000', 500000))

    # Получить количество денег номинала name
    def get_amount_nominal(self, name):
        return getattr(self, name)

    # Сделать количество денег номинала name равным amount
    def set_amount_nominal(self, name, amount):
        setattr(self, name, amount)

    # Вычислить значение текущей суммы в копейках
    def money_to_k(self):

        amount = 0
        # Для каждого номинала умножаем количество
        # банкнот на их эквивалент в копейках
        for k, v in self.coefs:
            amount += self.get_amount_nominal(k) * v
```

Рисунок 2 – Решение задачи 2 Ч.1

```
        return amount

    # Перевести копейки в номиналы, с учётом того,
    # какой самый старший номинал есть в наличии.
    # По умолчанию в наличии все номиналы
    def k_to_money(self, amount, greatest='r5000'):

        # Пока сумма не равна нулю,
        # для каждого номинала, начиная с самого старшего
        # конвертируем наибольшее возможное число копеек в этот номинал.

        flag = False
        for name, v in reversed(self.coefs):
            if flag or (name == greatest):
                flag = True
            if amount > 0 and flag:
                self.set_amount_nominal(name, amount // v)
                amount %= v

    # Вывод сумму на экран.
    def display(self):

        amount = self.money_to_k()
        print(f'Сумма: {amount // 100} руб, {amount % 100} коп')


# Функция сравнения сумм
def compare_amounts(m1: Money, m2: Money, report=True):

    # Переводим обе суммы в копейки
    amount_m1 = m1.money_to_k()
    amount_m2 = m2.money_to_k()

    # Возвращаем значение True если
    # первая сумма меньше второй

    if report:
        if amount_m1 < amount_m2:
            print('Сумма 1 меньше суммы 2')
            return True
        if amount_m1 == amount_m2:
            print('Сумма 1 равна сумме 2')
            return False
        print('Сумма 1 больше суммы 2')
        return False

    return amount_m1 < amount_m2
```

Рисунок 3 – Решение задачи 2 Ч.2

```python
# Функция сложения сумм
def sum_amounts(m1: Money, m2: Money):

    # Инициализируем новый класс суммы
    new_m = Money()
    # Для каждого номинала сложим соответствующие
    # значения сумм и присвоим результат новой сумме
    for name, _ in new_m.coefs:
        m1_v = m1.get_amount_nominal(name)
        m2_v = m2.get_amount_nominal(name)
        v = m1_v + m2_v
        new_m.set_amount_nominal(name, v)

    # Вернем полученную сумму
    return new_m


# Вычитание из суммы m1 суммы m2
def subtract_amounts(m1: Money, m2: Money):

    if compare_amounts(m1, m2, report=False):

        print('Сумма 1 меньше Суммы 2. Невозможно выполнить вычитание.')
        return

    # Инициализируем новую сумму
    new_m = Money()
    # Текущая сумма в копейках
    s = 0
    # Необходимая сумма в копейках
    debt = m2.money_to_k()
    # Для каждого номинала выполняем вычитание
    for i, (name, v) in enumerate(new_m.coefs):
        m1_v = m1.get_amount_nominal(name)
        s += m1_v * v
        # Если текущей суммы достаточно, чтобы выполнить вычитание,
        # то сделаем это и завершим операцию
        if s > debt:
            # Вспомогательный класс - текущая сумма
            cur_s = Money()
            cur_s.k_to_money(s, greatest=name)
            # Вычтем из первой суммы текущую
            new_m = subtract_amounts(m1, cur_s)
            # Вспомогательный класс - сдача
            rem_m = Money()
            rem_m.k_to_money(s - debt, greatest=name)
            # Добавим к новой сумме сдачу
            new_m = sum_amounts(new_m, rem_m)

            # Вернем новую сумму
            return new_m
```

Рисунок 4 – Решение задачи 2 Ч.3

```
        # Если текущая сумма в точности равна требуемой,
        # то обнулим все номиналы вплоть до текущего номинала nate
        if s == debt:
            new_m = sum_amounts(new_m, m1)
            for nname, _ in new_m.coefs:
                new_m.set_amount_nominal(nname, 0)
                if nname == name:
                    return new_m


# Деление сумм(отношение)
def ratio_amounts(m1: Money, m2: Money, report=True):

    # Переводим обе суммы в копейки
    amount_m1 = m1.money_to_k()
    amount_m2 = m2.money_to_k()

    if amount_m2 == 0:
        print('Сумма 2 равна нулю. Невозможно выполнить деление')
        return

    if report:
        print(f'Отношение Суммы 1 к Сумме 2 равно: {amount_m1 / amount_m2:.5f}')
    return amount_m1 / amount_m2


# Умножение на число k
def multiply_amount(m: Money, k: float):

    new_m = Money()
    amount = m.money_to_k()
    new_m.k_to_money(int(round(amount * k)))

    return new_m


# Деление на число k
def divide_amount(m: Money, k: float):

    if k == 0.0:

        print('Число k равно нулю. Невозможно выполнить деление')
        return

    return multiply_amount(m, 1 / k)
```

Рисунок 5 – Решение задачи 2 Ч.4

```python
# Запуск программы и тестирование
if __name__ == '__main__':

    m1 = Money(k1=0, k5=0, k10=0, k50=0,
               r1=0, r2=0, r5=0, r10=0,
               r50=5, r100=6, r500=1,
               r1000=0, r5000=0)

    m2 = Money(k1=0, k5=0, k10=0, k50=0,
               r1=10, r2=20, r5=30, r10=0,
               r50=0, r100=0, r500=0,
               r1000=1, r5000=2)

    m1.display()
    m2.display()

    sum_amounts(m1, m2).display()
    subtract_amounts(m1, m2)
    subtract_amounts(m2, m1).display()
    compare_amounts(m1, m2)
    compare_amounts(m2, m1)
    ratio_amounts(m1, m2)
    ratio_amounts(m2, m1)
    multiply_amount(m1, 1.5).display()
    divide_amount(m2, 3).display()
```

```
Сумма: 1350 руб, 0 коп
Сумма: 11200 руб, 0 коп
Сумма: 12550 руб, 0 коп
Сумма 1 меньше Суммы 2. Невозможно выполнить вычитание.
Сумма: 9850 руб, 0 коп
Сумма 1 меньше суммы 2
Сумма 1 больше суммы 2
Отношение Суммы 1 к Сумме 2 равно: 0.12054
Отношение Суммы 1 к Сумме 2 равно: 8.29630
Сумма: 2025 руб, 0 коп
Сумма: 3733 руб, 33 коп
```

Рисунок 6 – Решение задачи 2 Ч.5 с ответом

Вывод: были получены навыки по работе с классами и объектами при написании программ с помощью языка программирования Python версии 3.x.