

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

Институт цифрового развития
Кафедра инфокоммуникаций

**«Лабораторная работа 4.1 Элементы объектно-ориентированного
программирования в языке Python»**

**ОТЧЕТ
по лабораторной работе №29
дисциплины
«Основы программной инженерии»**

Выполнил:
Луценко Дмитрий Андреевич
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Лабораторная работа 4.1 Элементы объектно-ориентированного программирования в языке Python

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

Ход работы:

Задание 1

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

- метод инициализации `__init__`; метод должен контролировать значения аргументов на корректность;
- ввод с клавиатуры `read`;
- 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.

18. Поле *first* — целое число, целая часть числа; поле *second* — положительное целое число, дробная часть числа. Реализовать метод `multiply()` — умножение на произвольное целое число типа `int`. Метод должен правильно работать при любых допустимых значениях *first* и *second*.

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

class Pair:
    def __init__(self, a, b):
        if not isinstance(a, int) or not isinstance(b, int) or b < 0 or a <
0:
            raise ValueError()

        self.__first = a
        self.__second = b

    @property
    def first(self):
        return self.__first

    @property
    def second(self):
        return self.__second

    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 or parts[0] < 0:
    raise ValueError()

self.__first = parts[0]
self.__second = parts[1]

def multiply(self, number):
    if isinstance(number, int):
        result = (self.first * 100 + self.second) * number
        return Pair(result // 100, result % 100)
    else:
        raise ValueError()

def display(self):
    print(f"first: {self.__first}, second: {self.__second}")

def make_pair(a, b):
    try:
        return Pair(a, b)
    except:
        ValueError()

if __name__ == '__main__':
    pair = make_pair(1, 90)
    pair.display()
    number = int(input("Введите целое число: "))
    result = pair.multiply(number)
    result.display()

```

Листинг 1 – Код программы индивидуального задания

```

first: 1, second: 90
Введите целое число: 6
first: 11, second: 40

```

Рисунок 1 – Результат выполнения программы

Составить программу с использованием классов и объектов для решения задачи. Во всех заданиях, помимо указанных в задании операций, обязательно должны быть реализованы следующие методы:

- метод инициализации `__init__`;
- ввод с клавиатуры `read`;
- вывод на экран `display`.

..

Создать класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа int для рублей и копеек. Дробная часть (копейки) при выводе на экран должна быть отделена от целой части запятой. Реализовать сложение, вычитание, деление сумм, деление суммы на дробное число, умножение на дробное число и операции сравнения.

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

class Money():
    def __init__(self, a=0, b=1):
        a = int(a)
        b = int(b)

        if a < 0 or b < 0 or b > 99:
            raise ValueError()

        self.ruble = a
        self.penny = b

    def display(self):
        print(f"Cash: {self.ruble},{self.penny}P")

    def read(self, prompt=None):
        line = input() if prompt is None else input(prompt)
        parts = list(map(int, line.split('.', maxsplit=1)))

        if parts[0] < 0 or parts[1] < 0:
            raise ValueError()
        if len(str(parts[1])) == 1:
            parts[1] = int(str(parts[1]) + "0")

        self.ruble = int(parts[0])
        self.penny = int(parts[1])

    def add(self, rhs):
        if isinstance(rhs, Money):
            b = self.ruble * 100 + self.penny + rhs.ruble * 100 + rhs.penny
            a = b // 100
            b %= 100
            return Money(a, b)
        else:
            raise ValueError()

    def sub(self, rhs):
        if isinstance(rhs, Money):
            a = self.ruble - rhs.ruble
            b = self.penny - rhs.penny
            if b < 0:
                a -= 1
                b += 100
            return Money(a, b)
        else:
            raise ValueError()

    def mul(self, rhs):
        if isinstance(rhs, (int, float)):
            b = self.ruble * 100 + self.penny
            a = b * rhs
            a = int(a) // 100
            b = int(b) % 100
```

```

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

def div(self, rhs):
    if isinstance(rhs, Money):
        return self.div_money(rhs)
    elif isinstance(rhs, (int, float)):
        return self.div_number(rhs)
    else:
        raise ValueError()

def div_money(self, rhs):
    if rhs.ruble == 0 and other.penny == 0:
        raise ValueError()
    a = self.ruble * 100 + self.penny
    b = rhs.ruble * 100 + rhs.penny
    return a / b

def div_number(self, rhs):
    if rhs == 0:
        raise ValueError()
    a = self.ruble * 100 + self.penny
    a /= rhs
    a = round(a, 2)
    b = int(a // 100)
    a %= 100
    return Money(b, a)

def equals(self, rhs):
    if isinstance(rhs, Money):
        return (self.ruble == rhs.ruble) and \
            (self.penny == rhs.penny)
    else:
        return False

def greater(self, rhs):
    if isinstance(rhs, Money):
        v1 = (self.ruble + self.penny) / 100
        v2 = (rhs.ruble + rhs.penny) / 100
        return v1 > v2
    else:
        return False

def less(self, rhs):
    if isinstance(rhs, Money):
        v1 = (self.ruble + self.penny) / 100
        v2 = (rhs.ruble + rhs.penny) / 100
        return v1 < v2
    else:
        return False

if __name__ == '__main__':
    cp = Money()
    cp.read("Введите денежную сумму: ")
    cp.display()

    cp1 = Money(3, 40)
    cp1.display()

    cp2 = cp1.add(cp)
    cp2.display()

```

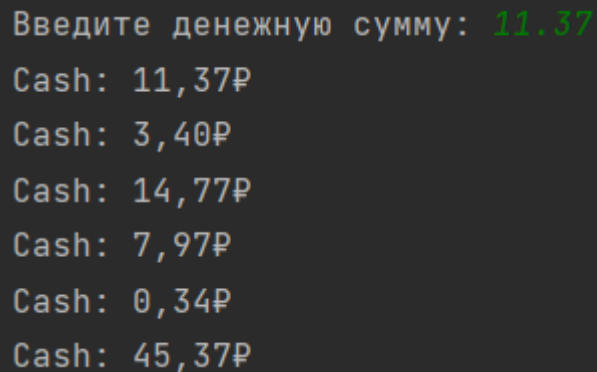
```
cp3 = cp.sub(cp1)
cp3.display()

cp4 = cp.div(33.3)
cp4.display()

cp5 = cp.mul(4)
cp5.display()

print(cp.less(cp1))
```

Листинг 1 – Листинг индивидуального задания 2



Введите денежную сумму: 11.37
Cash: 11,37Р
Cash: 3,40Р
Cash: 14,77Р
Cash: 7,97Р
Cash: 0,34Р
Cash: 45,37Р

Рисунок 2 – Результат выполнения программы

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

Ответы на 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. Например, `class MyClass:`

2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибуты класса принадлежат классу в целом и доступны всем его экземплярам. Атрибуты экземпляра, наоборот, принадлежат конкретному экземпляру класса и могут быть разными у разных объектов.

3. Каково назначение методов класса? Методы класса являются функциями, которые связаны с классом и могут быть вызваны у его экземпляров. Они часто используются для работы с атрибутами экземпляра, их изменения или чтения.

4. Для чего предназначен метод `__init__()` класса? `__init__()` - это метод класса, который вызывается при создании экземпляра объекта. Он используется для установки начальных значений атрибутов объекта.

5. Каково назначение `self`? `self` - это первый параметр всех методов класса в Python. Данный параметр используется для доступа и изменения атрибутов объекта, от которого вызывается метод.

6. Как добавить атрибуты в класс? Для добавления атрибутов в класс нужно просто создать новые атрибуты в методе класса или за его пределами, используя имя класса, например: `MyClass.new_attribute = "value"`.

7. Как осуществляется управление доступом к методам и атрибутам в языке Python? Управление доступом к методам и атрибутам в Python осуществляется с помощью модификаторов доступа: `public`, `private` и `protected`. В Python все атрибуты и методы класса по умолчанию являются публичными, т.е. доступ к ним возможен из любого места программы. Для создания приватных атрибутов и методов используется синтаксис с двойным подчеркиванием (`__`), а для создания защищенных атрибутов и методов одинарным подчеркиванием (`_`).

8. Каково назначение функции `isinstance()`? Функция `isinstance()` возвращает `True`, если переданный объект является экземпляром указанного класса или его потомком. Она используется для проверки типа объекта в Python.