**Подвиг 9.** Объявите класс с именем Money (деньги), объекты которого создаются командой:

money = Money(value)

где value - любое число (целое или вещественное). Если указывается не числовое значение, то генерируется исключение командой:

raise TypeError('сумма должна быть числом')

В каждом объекте этого класса должен формироваться локальный атрибут \_money с соответствующим значением. Также в классе Money должно быть объект-свойство (property):

**money** - для записи и считывания значения из атрибута \_money.

В связке с классом Money работает еще один класс:

class MoneyOperators:

def \_\_add\_\_(self, other):

if type(other) in (int, float):

return self.\_\_class\_\_(self.money + other)

if type(self) != type(other):

raise TypeError('Разные типы объектов')

return self.\_\_class\_\_(self.money + other.money)

Он определяет работу арифметических операторов. В данном примере описан алгоритм сложения двух объектов класса Money (или объектов его дочерних классов).

Обратите внимание, как реализован метод \_\_add\_\_() в этом классе. Он универсален при работе с любыми объектами класса Money или его дочерних классов. Здесь атрибут \_\_class\_\_ - это ссылка на класс объекта self. С помощью \_\_class\_\_ можно создавать объекты того же класса, что и self.

Вам необходимо добавить в класс **MoneyOperators** аналогичную реализацию оператора вычитания.

На основе двух классов (Money и MoneyOperators) предполагается создавать классы кошельков разных валют. Например, так:

class MoneyR(Money, MoneyOperators):

def \_\_str\_\_(self):

return f"MoneyR: {self.money}"

class MoneyD(Money, MoneyOperators):

def \_\_str\_\_(self):

return f"MoneyD: {self.money}"

И, затем применять их следующим образом (эти строчки в программе писать не нужно):

m1 = MoneyR(1)

m2 = MoneyD(2)

m = m1 + 10

print(m)  # MoneyR: 11

m = m1 - 5.4

m = m1 + m2  # TypeError

P.S. В программе требуется объявить только классы. На экран выводить ничего не нужно.

class MoneyOperators:

def \_\_add\_\_(self, other):

if type(other) in (int, float):

return self.\_\_class\_\_(self.money + other)

if type(self) != type(other):

raise TypeError('Разные типы объектов')

return self.\_\_class\_\_(self.money + other.money)

# здесь объявляйте еще один метод

# здесь объявляйте класс Money

class MoneyR(Money, MoneyOperators):

def \_\_str\_\_(self):

return f"MoneyR: {self.money}"

class MoneyD(Money, MoneyOperators):

def \_\_str\_\_(self):

return f"MoneyD: {self.money}"